temenos

Manager

JOURNEY MANAGER

VERSION 24.10

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, for any purpose, without the express written permission of TEMENOS HEADQUARTERS SA. © 2021 Temenos Headquarters SA - all rights reserved.

TOC

Journey Manager Overview **46**

Security 48

Additional Data Protection 49

Why TLS is Sufficient for Most Use Cases 49

Services 49

Journey Manager Features **51**

Main Features 51

Features by Release 52

Data Protection **70**

Additional Data Protection 70

Why TLS is Sufficient for Most Use Cases 70

Unified Application Data Document Model **71**

When to use Unified App Data 72

When to use Transaction Properties 73

Save Challenge **74**

Temenos Journey Manager Integration **75**

Integration Areas 75

Authenticated and Unauthenticated Transactions 78

Extension Points 80

Calling External Services and API 83

Web Site Integration and Single Signon 84

Form Level Integration 85

Integrate Manager with SharePoint **86**

Five Temenos Journey Manager Integration Areas **88**

Delivery 88

Form Discovery 92
3 Pre-Population - Accelerating Data Entry 93
Streamlining Data Entry 93
Execution 95
The Economics of Integration 97
Google Analytics Virtual Page Integration 100
Cloud-hosted Temenos Journey Manager VDC & Enterprise Editions 102
Test Specification 106
Unit Tests 106
Performance Tests 106
Automated Tests 106
Performance Specification 107
Scenario 1 107
Scenario 2 114
Resource Limitations 116
JDBC Connections 116
Java Threads 116
Transactions 116
Services 116
Forms 116
User Accounts 116
Journey Manager Home Dashboard 117
Screen Header 118
Menu Bar 119
Recent Forms Card 119

Recent Services Card 119
Latest Transactions Card 119
System Issues Card 120
Password Expiry Notification Card 120
Journey Manager Modules 121
Journey Manager Third-Party Libraries 124
Licenses 124
Java 125
Build Automation 125
Libraries 126
End-of life Releases 136
Installation Overview 137
Download the JM Installation Guide 137
Plan the Installation 138
Organizations Overview 140
View Organizations 142
Create an Organization 144
Copy an Organization 146
Remove an Organization 148
Export an Organization 149
Import an Organization 150
Import Using UI 150
Import Using SDK 153
Import Using API 153
Migrate an Organization 155

Configure an Organization **156**

Assign Form Spaces to an Organization **158**

Configure an Organization Payment Gateway **160**

Configure Organization Properties **162**

Configure Organization Metadata Tags **165**

Configure Organization Reference Data **166**

Test Organization Reference Data **170**

Query Syntax 172

Configure Organization Form Categories **174**

Create an Application Package **176**

View Application Packages **177**

Configure an Application Package **180**

Export an Application Package **182**

Import an Application Package **184**

Import Using UI 184

Import Using SDK 187

Import Using API 187

Configure Organization Security **189**

Form Spaces Overview **191**

Structure of Form Space WAR File 192

Default Form Spaces 193

Pre-configured Form Spaces 194

View Form Spaces **195**

Create a Form Space **197**

Configure a Form Space **199**

Configure a Form Space Status **204**

Assign Organizations to a Form Space **207**

Assign Form Spaces to a Form **209**

Remove a Form Space **211**

Export Form Spaces **212**

Import Form Spaces **213**

Import Using UI 213

Import Using SDK 215

Import Using API 215

Migrate Form Spaces **217**

Export a Form Space as a WAR Archive **219**

Import a Form Space as a WAR Archive **220**

Standard Form Space Properties **222**

Cache Headers 222

CORS Headers 222

Security Headers 224

X-Frame-Options 226

Create Form Space Properties **227**

Configure Form Space Properties **229**

Configure Workspaces Properties **236**

Configure a Form Space Email Template **239**

Configure Form Space Pages **241**

View Form Space Pages **243**

Create a Form Space Resource **246**

Create Using UI 246

Create Using Fluent API 247
Create Using REST API 248
Configure a Form Space Resource 250
Update Using UI 250
View Form Space Resources 253
View Forms Hosted on a Form Space 256
Form Space Hot Deployment 258
Web Plug-In Form Space 259
Host an Embedded Form 262
How does it work? 262
Step-by-step guide 263
Explanation of Form Page Content 264
Map a Template to a Model 266
How to Unlock a Transaction 269
Upgrade a Form Space 270
Content Only Spaces 270
Custom Java Spaces 271
Styles and Mark-up 271
Journey Space 273
Journey Space Login 275
Journey Space Change Password 278
Journey Space 2 Factor Authentication 280
Journey Space Home 282
Journey Space Forms 284
Journey Space Tasks 285

Journey Space History **286**

Journey Space Reviews **287**

Journey Space Help Desk **289**

Anonymous Users Only 289

Anonymous and Authenticated Users 290

Journey Space Account **292**

Configure Journey Space Properties **296**

Configure Journey Space Page Markup **298**

Forms Overview **302**

Global Forms 302

Form Dashboard **305**

Form Details 305

Form Versions 306

Form Spaces 307

Latest Transactions 308

Deploy Forms **309**

View Forms **310**

Configure a Form **313**

Configure Form Details **314**

Configure Form Abandonment **319**

Configure Form Email Verification **322**

Configure a Form Flow **324**

Configure the User Flow options for the form 325

Submission Confirmation Options 326

Custom Page Flow Options 327

Configure Form Group Access **329**

Configure Form Page Tracking **331**

Configure Using UI 331

Configure Using SDK 333

Configure Using API 333

Configure Form Promotion **335**

Schedule a Form Deployment **338**

Configure Form Test Mode **341**

Render a Form **344**

Test a Form **345**

Import Forms **346**

Import Using UI 346

Import Using SDK 352

Import Using API 353

Import a Form Application **354**

Export Forms **355**

Remove a Form **356**

Make a Form Inactive 356

Remove a Form 356

Migrate Forms **357**

Rollback Forms **358**

Save Challenge **360**

Maestro Form Static Content Externalization **361**

Server Monitor Test Form **362**

Integrate Forms to Existing Websites **363**

iFrame Embedding 363
<DIV> Embedding 363
Launch a new tab or window 364
Configure the Form Display Mode 364
Form URLs 364
Maestro Data Model 366
Data Mapper 369
Component XML Name and Entity XML Path 371
ID without an Entity 371
ID with an Entity 372
Form XML Data Payload 373
Data Extract Elements 377
$formDataMap 377
$job 377
$jobAction 378
$func 378
Submission Data Extract Overview 379
View Form Submission Data Extract 381
Form Data Overview 383
View Form Data Configuration 388
Form Configuration Mapping 390
View Form XML Data 393
View Form XML Data in a Browser 394
Form Seed 395
Form Data Extract Mappings 397

Collaboration Job Contex 400
AvailableRoutes 401
ReferenceNumber 402
StepName 402
RouteName 403
Test a Form Data Extract Mapping 404
Form Prefill Options 405
Property Prefill Mapping 405
Input XML Prefill Mapping 405
Request Parameter Prefill Mapping 406
Form Input XML Prefill Mapping 407
Form Input XML Prefill Mapping Using Drag and Drop 410
Form Input XML Prefill Mapping Using XPath 412
Form Property Prefill Mapping 414
Form Property Prefill Mapping Using Drag and Drop 418
Form Property Prefill Mapping Using XPath 420
Form Request Parameter Prefill Mapping 422
GET vs POST for Request Parameter Prefill Mapping 424
POST Requests 425
Prefilling Customer Information 425
Add Repeating Content from XML Prefill Mapping Dynamically 427
Form Prefill for Anonymous Users 430
URL Parameter passing on a GET request 430
HTTP request headers on a POST request 431
Encrypted Cookies 432

Short life tokens (SLT) 433
Encrypted String 435
Launch a Form From Another Form 438
Technique 438
Main form 439
Alternatives 439
Restrictions 439
Prefill a Cascading Dropdown in Composer 440
Form Versions Overview 444
View Form Versions 446
Configure a Form Version 448
Create a Form Version 455
Make Current Form Version 457
Configure a Form Version Metadata 459
Form Version Functions 460
Create a Trigger 461
Form Version Services 463
Form Version Properties 466
View Form Version Properties 466
Create a Form Version Property 467
Edit a Form Version Property 468
Override a Form Version Property 470
Form Categories 474
Form Design Info 476
Job Properties of a Form Version 478

Job Information of a Form Version **480**

Delete a Form Version **481**

Delivery Channels Overview **482**

Create a Delivery Channel **486**

Create a Delivery Process Channel **488**

Create an Email Delivery Channel **491**

Create an Email Secure Delivery Channel **495**

Create a Web Service Delivery Channel **497**

Create a REST Delivery Channel **498**

View Delivery Channels **499**

Assign a Delivery Channel to a Form **501**

View Forms Associated with a Delivery Channel 503

Remove a Delivery Channel **504**

View Transactions Delivery **505**

Service Connection Overview **507**

Service Connection Types **509**

View Service Connections **511**

Create a Service Connection **513**

Configure a Service Connection **515**

Remove a Service Connection **517**

Migrate Service Connections **518**

Export Service Connections **519**

Import Service Connections **520**

Import Using UI 520

Import Using SDK 522

Import Using API 522

AWS KMS Service Connection 524

AWS S3 Service Connection 526

AWS SQS Service Connection 528

AWS STS Service Connection 530

Azure Service Connection 532

ClamAV Virus Scan Service Connection 534

Symantec Virus Scan Service Connection 536

HTTP Endpoint Service Connection 538

DocuSign Service Connection 540

DocuSign OAuth Service Connection 544

Integration Gateway Service Connection 547

Kafka Service Connection 548

Kafka Producer Configuration 550

Salesforce Service Connection 551

Services Overview 553

Service Types 557

View Services 569

Create a Service 572

Configure a Service 574

Configure Service Parameters 576

Configure Service Parameters Edit 579

Assign Organizations to a Service 581

Service Usage 583

Copy a Service 584

Remove a Service **586**

Deactivate a Service 586

Remove a Service 586

Migrate Services **588**

Export Services **589**

Import Services **590**

Import Using UI 590

Import Using SDK 593

Import Using API 593

Groovy Console **594**

Services Troubleshooting **599**

Service Configuration 599

Service Errors 600

Service Timeout 600

Core Global Services Overview **601**

View Core Global Services **602**

A/B Testing Form Version Selector Service **604**

Database Submission Data Storage Service **607**

File System Submission Data Storage Service **610**

Amazon Submission Data Storage Service **613**

Azure Submission Data Storage Service **617**

System Event Publisher Service **620**

AWS SQS System Event Publisher Service **626**

Azure System Event Publisher Service **627**

Chromium Puppeteer Receipt Service **628**

Composer Render HTML Form Service **633**

Current Version Selector Service **635**

Database Integrity Checker Service **636**

Data Retention Management Service **638**

Data Retention Management v2 Service **644**

Dynamic PDF Receipt Service **650**

Dynamic PDF Receipt 2 Service **653**

Event Configuration Storage Service **656**

Event Outbox Processor Service **658**

Form Submission Access Controller Service **661**

Jakarta Transformer Service **664**

Jakarta Transformer Scheduled Service **666**

Payment Reminder Service **668**

Random Tracking Number Service **669**

Status Management Service **670**

Submission Receipt Render Service **672**

Transaction Processor Service **674**

Attachment Virus Scanning Service **679**

ClamAV Attachment Virus Scanning Service **681**

Symantec Attachment Virus Scanning Service **683**

Fluent Attachment Virus Scanning Service **685**

Dynamic Data Service **687**

Submission Completed Processor Service **691**

Task Cancellation Processor Service 691

Job Services Overview **696**

Job Definitions **697**

jobDetails 697

jobGroups 698

steps 699

actions 701

properties 702

routes 705

View Job Services **707**

Create a Job Service **709**

Job Service Templates **711**

Job Definitions **713**

jobDetails 713

jobGroups 714

steps 715

actions 717

properties 718

routes 721

Configure Job Definition **723**

View Job Info **725**

Create a Job Controller Service **726**

View Job Controller Services **729**

Job Action Service **730**

Job Action Wait Service **735**

Job Delivery Service **740**

Job Task Wait Action Service **743**

Open the TaskUserWebService in SOAP UI **748**

Scheduled Jobs Overview **753**

Trigger Types 753

Standard Scheduled Jobs **755**

Manage Scheduled Jobs **758**

Create a Scheduled Job **762**

Configure a Scheduled Job **764**

Configure Using UI 764

Configure Using a Configuration Service 765

Export Scheduled Jobs **767**

Import Scheduled Jobs **769**

Import Using UI 769

Import Using SDK 770

Import Using API 770

View Scheduled Job History **772**

Collaboration Job Controller **774**

Database Integrity Checker Job **776**

Event Outbox Processor Job **779**

Receipts Overview **780**

Configure Receipt Numbers and Tracking Codes **783**

Configure Receipts Delivery Mode **785**

Configure a Receipt Server Node **786**

Receipt Headers and Footers Overview **787**

Header and Footer Templates 788

Custom Headers and Footers In a Form 788

Receipt Headers and Footers Customization **791**

Enable a Render Receipt Service 791

Update Receipt Properties 791

Customize Receipts with JavaScript 794

Default Receipt Properties 800

Using Header and Footer Templates **802**

Field Overlay Receipts Using AcroForms **803**

Getting Started 803

Step 1: Add Data Extracts to form fields 804

Step 2: Export the form from Maestro 805

Step 3: Publish the form to Manager 805

Step 4: Create a fillable PDF with Adobe Acrobat Pro DC 806

Step 5: Upload the fillable PDF to Manager 810

Step 6: Configure the Receipt Render Service 811

Step 7: Verify the results 812

Test a Receipt with Receipt Test Harness **814**

PDF Receipt Options **817**

Manager PDF Receipts 817

XFA Receipts 818

Field Overlay Receipts 819

Field Overlay Receipts Using Adobe Acrobat 820

Field Overlay Receipts Using Adobe LiveCycle Designer 821

Field Overlay Receipts Using Texcel 821

Custom Receipt Generation 822

Generate Multiple PDF Receipts from a Single Submission **823**

Storing multiple PDF receipts 825
Custom Rendering Service 826
Import an Acrobat PDF Form as a Receipt Template (AcroForms) 829
What is an AcroForm? 829
High Level Overview of steps required 829
Step 2: Creating AcroForms 831
Step 3: Correcting automatic conversion issues 834
Step 5: Manager Configuration 843
Extending this solution 844
Insert a Watermark in a PDF Receipt 845
Receipt Generator Service 847
Generate Using Fluent API 847
Generate Using API 848
Regenerate a Receipt 850
Collaboration Jobs Overview 852
Collaboration Job Modes 856
Standard Mode 856
Process Immediate Mode 856
Job Services Overview 858
Job Definitions 859
jobDetails 859
jobGroups 860
steps 861
actions 863
properties 864

routes 867

View Job Services 869

Create a Job Service 871

Job Service Templates 873

Job Definitions 875

jobDetails 875

jobGroups 876

steps 877

actions 879

properties 880

routes 883

Configure Job Definition 885

View Job Info 887

Create a Job Controller Service 888

View Job Controller Services 891

Job Action Service 892

Job Action Wait Service 897

Job Delivery Service 902

Job Task Wait Action Service 905

Open the TaskUserWebService in SOAP UI 910

Steps 915

Actions 923

Simple Actions 924

Auto Retry Actions 924

Wait Actions 925

Example 1 - Task Step - Standard Mode Processing 925
Example 2 - Multiple Tasks Step - Standard Mode Processing 926
Example 3 - Form Bundle - Process Immediate Mode 927
Action Properties 930
Task Types 930
Job Functions 935
Collaboration Job Basics 936
View Collaboration Jobs 938
Collaboration Job Details 944
Job Data Purging 947
View Collaboration Job Steps 948
View Collaboration Job Actions 950
View Collaboration Job Submissions 954
View Collaboration Job Events 956
View Collaboration Job Errors 958
View Collaboration Job Status History 960
View Collaboration Job History 962
Collaboration Job Examples 965
Configure Forms for Use with Collaboration Jobs 971
Configure Maestro Forms for Use with Collaboration Jobs 978
Data Extracts 978
Collaboration Job Components 979
Form Rules 982
Using a Radio Button Group for the Reviewer and Manager Decision 986
Configure Composer Forms for Use with Collaboration Jobs 994

Data Extracts 994
Form Data Mappings 995
Collaboration Widgets and Components 996
Collaboration Reference Number and Collaboration Step Name 996
Collaboration Routes Dropdown List 997
Collaboration Submit Button 998
Collaboration Test Block 999
Routing Using the Standard Radio Button Group 1000
Visibility and Editability Rules 1003
Attachments 1005
Configure Routes with Business Rules 1007
Customer Onboarding Collaboration Jobs 1008
The Workflow 1008
Dynamic Customer Onboarding Collaboration Jobs 1018
Maestro Customer Onboarding Collaboration Jobs 1019
Multi-Step Group Review Collaboration Jobs 1020
Configure a Form 1020
Configure Manager 1022
Configure Users and Groups 1029
Route Options 1029
Form - Anne 1031
Initial Review - Roger 1033
Applicant Update - Anne 1037
Initial Review - Roger 1039
Additional Review - Mike 1041

One Step Review Collaboration Job **1044**

Configure a Form 1044

Configure Manager 1046

Configure Users and Groups 1049

Form - Anne 1050

Manager - Administrator 1052

Form Space - Anne 1053

Form Space - Roger 1054

Manager - Administrator 1056

Form Space - Anne 1057

One Step Anonymous Review Collaboration Jobs **1059**

Cancel an Anonymous Task in Maestro Forms 1063

Step Expiry **1068**

Configure a Form 1069

Configure a Job Controller 1069

Configure a Job Expiry Service 1071

Form - Start 1075

Manager - Action Review Expiry 1075

Form Space - Application Escalated 1076

Task Assign Repeat Action **1078**

Assignee 1078

AssignRepeatIndex 1079

AssignRepeatItem 1079

Collaboration Jobs Reuse **1081**

Collaboration Job Developers Documentation **1082**

Services guide 1082
Core API 1082
Form Bundles Overview 1084
Sharing Data Between Tasks 1088
Show Previous Forms 1089
Forms in the Same Step Editable 1089
Preconditions 1089
Dynamic Preconditions 1090
Dynamic Precondition and Action Order 1090
User Authentication 1091
Form Bundles Context 1093
Configure Form Bundles 1097
Step Attributes For Form Bundles with More Than Two Tasks 1097
Step Attributes For Form Bundles with One or More Tasks 1097
Step Properties 1097
Action Properties 1098
Anonymous Form Bundles 1099
Configure a Form 1099
Job Definition - Additional Products 1100
Example 1 1102
Example 2 1103
Authenticated Form Bundles 1104
User Task 1104
Non Claimable Tasks 1104
Claimable Group Tasks 1104

Job Definition - Additional Products 1105
Example 1 1107
Example 2 1107
Review Task 1107
Example 3 1107
Form Bundle Flow Example 1109
The Flow 1109
Form Chaining Overview 1120
Change of User 1122
Legacy Mode 1123
Transact Functions Overview 1124
How It Works 1125
View Transact Functions 1128
Create a Transact Function 1130
Copy a Transact Function 1132
Import a Transact Function 1133
Export a Transact Function 1136
Backup a Transact Function 1137
Rollback a Transact Function 1138
Remove a Transact Function 1139
Run a Transact Function Unit Test 1141
Configure a Transact Function Service Definition 1144
Edit a Transact Function Groovy Script 1147
Configure Transact Function Parameters 1150
Configure Transact Function Parameters Edit 1153

Configure a Transact Function Unit Test **1161**

Configure Transact Function Unit Test Parameters **1164**

Configure Transact Function Triggers **1166**

Transact Function Help Doc **1168**

Server-Side Persistence Pattern **1170**

Storing Data 1170

Retrieving Data 1170

Handling Form Function Exceptions in Maestro Applications **1172**

Journey Manager - the Server Side 1172

Maestro Application - the Client Browser Side 1173

Code Examples 1173

Success 1178

DataError 1179

SystemError 1179

Exception 1179

Manager Error 1179

Conclusion 1180

Validate Form Data Against a Schema **1181**

User Builder API Overview **1182**

Create New User 1182

Update an Existing User 1183

Use the Groovy Console 1183

Conclusion 1183

Access Form Properties with a Groovy Service **1184**

Access Service Parameters of Another Groovy Service **1185**

Create Anonymous Tasks with a Groovy Service **1186**

Create Form Tasks with a Groovy Delivery Service **1190**

Mix Standard Delivery Functions with Custom Groovy Processing **1195**

Invoke Dynamic Data Service from a Groovy Service **1199**

Use Case 1199

Solution 1199

Dynamic Data Lookup Service **1201**

Introduction to Dynamic Data 1201

Handling Data 1201

Building the Composer Form 1201

Building the Transaction Manager Dynamic Data Service. 1203

A Final Word - Testing Testing Testing 1206

Bonus Points - Repeating blocks of results 1207

Conclusion 1207

Check a Form Request IP Address **1208**

Groovy Service Tips and Tricks **1210**

Get the version of Groovy 1210

Get Date widget into a W3C dateTime 1210

Manager included libraries 1211

Groovy script body building 1212

Capturing Groovy Script Output 1212

Use a Builder and a Map to Conveniently Return Dynamic Data 1213

Mocking up Web Service SOAP Return Data 1213

Typed Objects 1216

JSON 1216

Collections and JSON 1217

Create an Object Based on a Collection 1218

HTTP Session Object Pattern 1219

Invoke Core Services Pattern 1220

Step 1 1220

Step 2 1221

Step 3 1222

Step 4: 1223

Step 5 1225

Step 6 1226

Benefits 1227

Long Running Service Pattern 1228

Progress Polling 1229

Blind State Machine 1232

Enterprise Service Bus 1234

Conclusion 1234

Multi-Step Delivery Process 1235

Basic Groovy Delivery Process 1235

Dual Delivery 1237

Checkpoints 1240

Restore Undeliverable Transactions 1244

Reload Form Submission Data Extracts from the Groovy Console 1246

Server-Side Persistence Pattern 1249

Storing Data 1249

Retrieving Data 1249

Tasks Overview **1251**

View Tasks **1252**

Create a Task **1255**

Configure a Task **1261**

View Task Transaction Details **1263**

Abandon a Task **1264**

Event-Based Architecture Overview **1265**

Publishing Flow 1267

Subscribing Flow 1268

Event Definition **1269**

Event Configuration Storage Service **1271**

Event Outbox Processor Service **1273**

Event Inbox Processor Service **1276**

Event Listener Configuration Service **1279**

View Event Listeners **1282**

System Event Publisher Overview **1283**

A Single Queue 1284

Multiple Queues 1284

Configure System Event Publisher **1285**

System Event Publisher Events **1286**

System Event Publisher Audit Log Events **1288**

System Event Publisher Job Events **1289**

System Event Publisher Submission Events **1291**

System Event Publisher System Health Events **1296**

System Event Publisher User Authentication Events **1298**

Security Managers Overview **1300**

SSO Flow 1303

Login Flow 1303

Download the Security Managers PDF 1303

User Accounts Overview **1304**

Authenticated Users 1306

Anonymous Users 1307

View User Accounts **1310**

Create a User Account **1313**

Configure a User Account **1317**

Assign Roles to a User Account **1322**

Assign Organizations to a User Account **1325**

Assign Form Spaces to a User Account **1328**

Provision a User Account **1330**

Provision a User Account for Accessing a Maestro Shared Environment Only 1330

Provision a User Account with the Organization User Manager Role 1331

Restrict User's Access to Forms 1332

Troubleshooting 1332

User Profiles Overview **1335**

Create a User Profile **1336**

Configure a User Profile **1338**

User Property Types **1339**

Remove a User Account **1341**

Deactivate a User Account 1341

Remove a User Account 1342

Bulk Import of User Accounts **1344**

Migrate User Accounts **1347**

View User Login History **1348**

User Visibility Rules for Restricted Content **1351**

User Visibility Rules for Collaboration Jobs **1352**

User Visibility Rules for Transactions **1355**

Roles Overview **1359**

Root Administrator 1359

Developer / System Manager 1359

Operations 1359

Organization User Manager 1359

System Support 1360

Transaction Data Access 1360

Standard Roles **1361**

View Roles **1367**

Create a Role **1370**

Assign Users to a Role **1372**

Configure Role Expiry **1375**

Remove a Role **1377**

Deactivate a Role 1377

Remove a Role 1377

Export Roles **1379**

Import Roles **1380**

Import Using UI 1380

Import Using SDK 1382

Import Using API 1382

Migrate Roles **1383**

Groups Overview **1384**

View Groups **1385**

Alert Groups **1387**

Receive Delivery Escalation Alerts 1387

Receive Outstanding Payment Alerts 1387

Receive Promotion Alerts 1388

Receive System Alerts 1388

Receive Submission Updates 1388

Receive System Alerts - Database Integrity 1388

Receive System Alerts - Delivery 1388

Receive System Alerts - Receipts 1388

Receive User Account Creation Alerts 1389

Custom Alerts 1389

Create a Group **1390**

Remove a Group **1393**

Assign Users to a Group **1394**

Assign Groups to a User **1397**

Assign Groups 1397

Revoke Groups 1398

Configure Group Expiry for a User Account **1399**

View Forms Assigned to a Group **1400**

Migrate Groups **1401**

Export Groups **1402**

Import Groups **1403**

Import Using UI 1403

Import Using SDK 1405

Import Using API 1405

Permissions Overview **1406**

Core Permissions **1407**

Maestro Permissions **1440**

Exchange Permissions **1444**

Journey Analytics Permissions **1445**

Workspaces Permissions **1446**

Assign Permissions to a Role **1449**

View User Login History **1451**

Log in to Manager **1454**

Two-Factor Authentication 1456

Change a Password **1458**

Log out of Manager **1460**

Operations Overview **1462**

Transactions Search 1462

System Check 1463

Logs Check 1464

Data Retention Check 1465

Configurations Check 1465

Reporting 1466

Transactions Overview **1467**

Submission Life Cycle **1468**

User flow - Form Transaction Experiences (FTX) 1469
User flow -Traditional (Non-FTX) 1469
Delivery flow 1469
User Actions 1470
Manager Actions 1472
Submission Categories 1473
Transaction Processing Sequence 1474
Completed Submissions 1477
Collaboration Job Submission 1479
Abandoned Submissions 1482
Authenticated vs Unauthenticated Transactions 1485
Form Pre-fill 1485
Form Save & Resume 1486
Transaction History 1486
Tasks 1486
Best Practices 1486
Configure Form Abandonment 1488
Abandon a Transaction 1491
Transactions Abandonment 1493
Automatic Abandonment 1493
Transaction and Task Purging 1496
Manual Abandonment 1497
Transaction Details 1498
Transaction Status 1502
User Transaction Flow Steps 1508

PDF Receipt Generation 1509
Transaction Data Delivery 1509
Collaboration Job 1510
Transaction Data Purging 1510
Change Transaction Status 1514
Method 1 1514
Method 2 1515
Form Sessions 1516
Transaction History 1518
Transaction History Details 1521
Form XML Data 1523
Form Data Extract 1527
Payment 1529
Attachments 1531
Transaction Groovy Log 1533
Transaction Events 1535
Transaction Properties 1537
Transaction Timeline 1541
Transaction Support Search 1542
Bulk Operations 1546
Track Transactions 1548
View Form Transactions 1549
Form Data Extract 1552
View Abandoned Transactions 1554
View Saved Transactions 1556

View Payment Transactions **1558**

Payment Transaction Details **1561**

View Form Submission Data **1563**

View Form Requests **1565**

View Form Rendering Metrics **1568**

View Transactions History **1570**

View Transaction Service Log **1572**

Debug Maestro Forms **1574**

Compare a Form Version with a Previous Version 1574

Build a Form 1575

Open Chrome DevTools 1575

Inspect the Form Object 1576

Debug JavaScript Rules 1577

Set Breakpoints 1580

Debug Form Styles 1584

Change Data Format in Form XML Data **1586**

Linux 1587

Windows 1587

Operations Portal Overview **1588**

Concepts 1588

UI Tour 1589

Manage Transactions **1592**

Item List 1593

Views 1595

Global Filters 1596

Finding transactions 1596

All Transactions View 1600

Item List 1600

View Filters 1601

Form Transactions View 1603

Item List 1603

Filter 1604

Saved Transactions View 1605

Item List 1605

Filter 1606

Abandoned Transactions View 1608

Item List 1608

Filter 1609

Assigned Tasks View 1611

Item List 1611

Filter 1613

Transaction Actions Overview 1614

Action Availability 1614

Abandon a Transaction 1617

Purge PII Data 1618

Purge a Transaction 1619

Recover a Transaction 1620

Resume a Transaction 1622

View Form XML 1623

View a Transaction Receipt 1624

Transaction Details Screen **1625**

Transaction Details Tab **1628**

Transaction Details 1629

Form Details 1629

User Information 1629

Transaction Status Tab **1631**

User Transaction Flow Steps 1632

User Processing Status Message 1633

PDF Receipt Generation 1633

Transaction Abandonment 1633

Transaction Data Delivery 1634

Transaction Data Purging 1634

Collaboration Job 1634

Task Tab **1635**

Task Details 1636

Task Status 1637

Form Sessions Tab **1638**

Transaction History Tab **1642**

Form Data Extracts Tab **1644**

Transaction Attachments Tab **1646**

Groovy Log Tab **1648**

Transaction Events Tab **1650**

Transaction Errors Tab **1651**

Transaction Properties Tab **1652**

Transaction Notes Tab **1654**

System Logs **1656**

Dos: 1656

Don'ts: 1657

Error Log **1658**

Error Log Trend **1662**

Event Log **1663**

Database Version Log **1667**

Import Log **1669**

Audit Log **1672**

Message Examples 1675

Groovy Service Log **1677**

Groovy Log to Splunk Integration **1683**

Apache Log **1685**

T.Field Device Log **1686**

T.Field Sync Log **1687**

Journey Manager Analytics **1688**

Analyze Transaction Trend **1689**

Transaction Volume 1691

Completion Rate (%) 1691

Conversion Rate (%) 1691

Form Abandonment Rate (%) 1691

Bounce Rate (%) 1692

Median Form Completion Time (mins) 1692

Form A/B Testing Overview **1694**

Configure Form A/B Testing **1696**

Elect Form A/B Testing **1698**

Analyze Form A/B Testing **1701**

Analyze Forms Activity **1703**

Analyze Device Type Submissions **1705**

Analyze Form Submissions **1707**

Analyze Submission Trend **1709**

Analyze Referer Submissions **1710**

Analyze Browser Submissions **1713**

Analyze OS Submissions **1714**

Analyze Reader Versions **1715**

Maestro and Composer Forms A/B Testing **1716**

Maestro 1716

Composer 1717

Manager 1718

Performing the Tests, Reviewing Results 1720

Reports Overview **1722**

Licensing Reports **1723**

Billing Records 1724

Report Configurations **1726**

System Report Schedules **1730**

View Transaction Licensing Report **1733**

View Job Licensing Report **1736**

Create a Custom Report **1739**

Form 1739

Groovy Script 1745

Outgoing Email Service Overview **1753**

Configure a SMTP Server **1755**

Create an Email Queue Item **1757**

Edit an Email Queue Item **1760**

View the Email Queue **1762**

Data Retention Management Overview **1764**

Global Data Retention Configuration **1768**

Transaction Records (max age) 1770

System Logs (max age) 1774

Data Retention Configs 1774

Configure Organization Data Retention **1776**

Configure Form Data Retention **1779**

Data Retention Policy Mode **1782**

Strict vs Relaxed Mode 1785

Data Retention Schedule **1787**

Manual Data Purge **1789**

Data Retention Summary **1791**

Data Retention Summary: Transactions **1792**

Data Retention Summary: Job Transactions **1794**

Data Retention Summary : Collaboration Jobs **1796**

Data Retention Summary: System Logs **1798**

Data Retention Summary: Top Table Sizes **1800**

Server Nodes Overview **1802**

View Server Nodes **1803**

Create a Server Node **1804**

Configure a Server Node **1805**

Monitor Server Node Performance **1807**

Remove a Server Node **1808**

System Info **1809**

Runtime Info 1809

Build Properties 1809

System Properties 1810

System Health **1812**

System User Preferences **1816**

System Licensing **1817**

Deployment Properties **1819**

Default Deployment Properties 1820

Important Deployment Properties 1828

Journey Analytics Deployment Properties 1829

Journey Maestro Deployment Properties 1829

Enable Deprecated Features **1830**

Configuration Service **1832**

Create Entities 1832

Deploy Using UI 1834

Deploy Using SDK 1839

Deploy Using API 1839

View TPacs **1842**

TPac Info 1843

Remove a TPac 1844

Import TPacs **1846**

Import Using UI 1846
Import Using SDK 1848
Import Using API 1848
About 1849
Multi-Environment Support 1850
Multi-Environment Migration 1851
Manager Deployment and Rollback Procedure 1852
Deployment 1852
Security Managers 1852
Organisations 1852
Form Spaces 1852
Organisation Applications 1853
Service Connections 1853
Verification 1853
Rollback 1853
Security Managers 1853
Organisations 1853
Form Spaces 1853
Organisation Applications 1853
Service Connections 1854
Verification 1854
Appendix - SSL Keystore Generation 1854
Appendix - Organizations Environment 1854
Appendix - Form Spaces Environment 1854
Appendix - Organization Applications Environment 1854

Appendix - Service Connections Environment 1854

Appendix - Verification 1854

Rollback Overview 1855

Rollback Configuration 1856

Manager FAQ 1857

General 1857

Journey Manager Overview

Journey Manager is the heart of the Temenos Journey Manager platform. It is a system of engagement, purpose-built to manage customer onboarding[1] and customer journeys, such as account opening and servicing.

Manager productizes the technology needed for a best-in-class digital sales experience in a pre-built platform. This experience enables financial and government institutions, large or small, to create their user journeys without having to engineer complex features such as security, data capture, resiliency, convenience, and integration.

Manager is a complex software system, consisting of many interconnected components. It's designed to:

- Host application forms and associated resources, and provide customers with secure access to them.
- Manage forms' transaction flow.
- Create highly configurable services to implement the desired customer experience[2].
- Enable integration with various FinTech[3] services and systems.
- Analyze actual customer behavior to continuously improve the customer experience.

Manager provides the following core capabilities:

- Form hosting - multiple forms can be hosted centrally and be made accessible to applicants by embedding links to the forms into organization's web pages or by using form search engines. You can also manage forms and process submissions made by users.

---

[1]The steps required to get a new customer integrated into a new program. These steps may vary business to business.

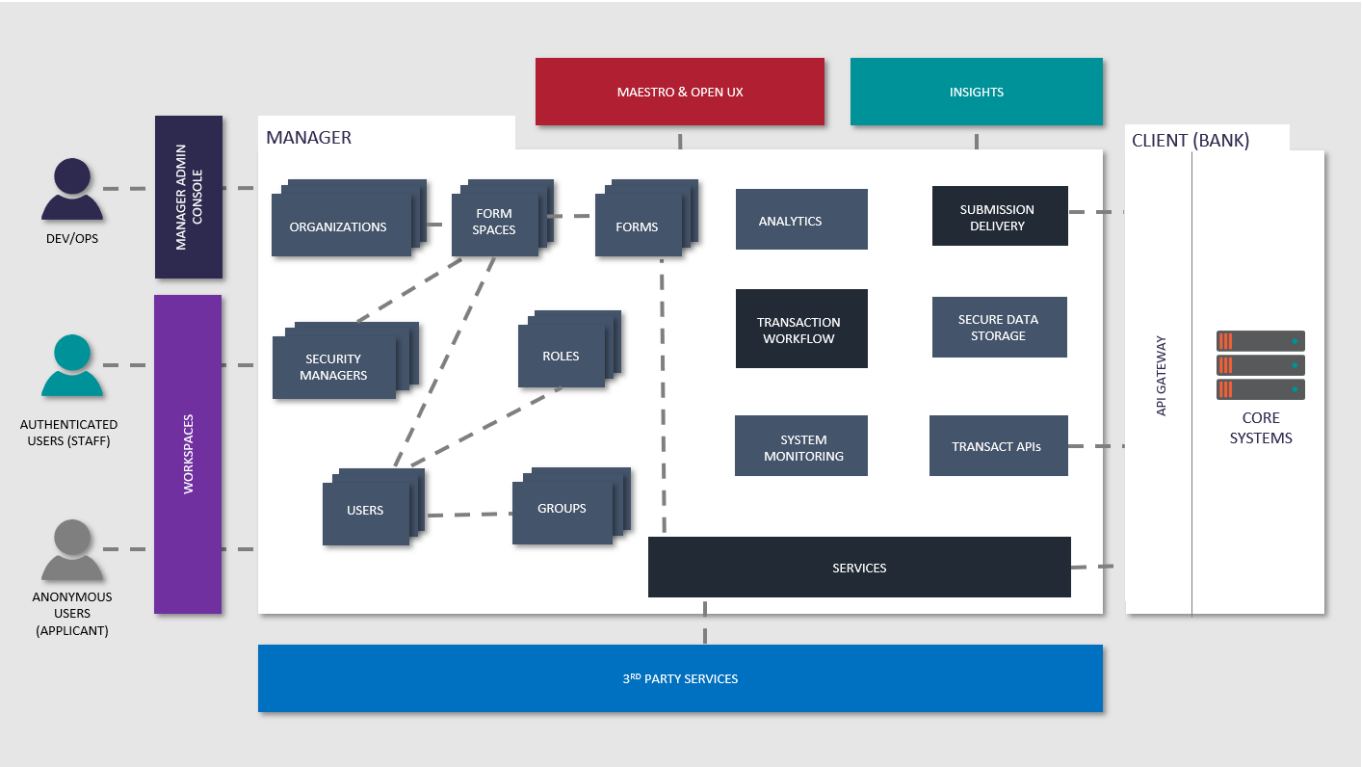
[2]Customer Experience (CX) is the product of an interaction between an organization and a customer over the duration of their relationship. This interaction is made up of three parts: the customer journey, the brand touchpoints the customer interacts with, and the environments the customer experiences (including digital environment) during their experience. A good customer experience means that the individual's experience during all points of contact matches the individual's expectations.

[3]Financial Technology, also known as FinTech, is a new industry that uses new technology and innovation to improve activities in finance. It aims to compete with traditional financial methods in the delivery of financial services to customers by improving customer experience, reducing time-to-market, decreasing cost to mention a few.

- Data pre-population - users can quickly complete forms that are pre-populated with their information, which comes from one or more user's profiles or organization provided data retrieved via web services or APIs.
- Receipts - users are provided with an electronic receipt that includes the submitted form, payment and submission information. Receipts can be viewed online, emailed, or downloaded and then printed.
- Payments - if a submission needs to be accompanied by a payment, Manager can be configured to read the correct payment amount from the form and process the payment using a secure credit card payment process.
- Supporting documentation - submissions can be supported by additional documents that may be uploaded electronically or submitted in paper form. Business logic in the forms can be applied to supporting documents.
- Submission delivery - completed submissions can be delivered by email to one or more user's addresses. The delivery process can be configured separately for each form.
- Form configuration and version control - forms provide fine-grained control over behavior, including form properties, form metadata, and custom rendering orchestrations. Forms also support version control enabling parallel development and multiple versions of the same form, so you can promote forms from development, to test, and finally production on separate servers.
- Task assignment - creates a task for a user or a group of users to work with forms to complete, review or reject then.
- Submission tracking - all submissions can be tracked from form download to submission delivery. Customer support can view forms and attachments to provide user assistance during the submission process.
- Reports - provides multiple charts and standard reports for reporting on submissions. Additional reports can be configured using advanced reporting modules. Reports can be displayed on a screen or can be automatically sent out via email.

For the complete list of all product capabilities, see Manager features.

The diagram below outlines the main components, with the submission delivery, transaction workflow and services being of the high significance.



Manager provides a well-defined structure to enable authenticated (staff) and anonymous (applicants) users' interaction with forms. An organization sits at the top of that structure, followed by form spaces and finally forms. A client can have multiple organizations, which each organization using several form spaces. Each form space can provide access to different forms and a form itself can be associated with several form spaces.

Security

Manager is a secure system that allows you to enable fine grained user access to forms by using preconfigured security managers or and customized groups and roles. The platform developers or system admins can have dedicated access to configure, maintain and manage your Manager environment. Staff and applicants can access forms directly or indirectly through Journey Workspaces.

Journey Manager protects your data as it travels between your device and our servers using industry-standard TLS[1] encryption. This robust encryption ensures that your information remains confidential and secure from unauthorized access. Consequently, data in transit is protected from eavesdropping and tampering.

---

[1]Transport Layer Security (TLS) is a cryptographic protocol designed to provide communications security over a computer network.

Additional Data Protection

For an extra layer of protection, Manager offers the optional Form Data Obfuscation. This feature further scrambles sensitive data, making it even more difficult for potential attackers to decipher. Although the key exchange mechanism for this feature is weaker compared to TLS, it still serves as a valuable component of a defense-in-depth strategy. Given the robust security provided by TLS, this additional layer is typically unnecessary in industrial applications. However, this additional measure can be a valuable part of a comprehensive security strategy.

Why TLS is Sufficient for Most Use Cases

In most cases, as used by millions of people and businesses worldwide, the high level of security provided by TLS is sufficient to protect sensitive data in transit. By following industry best practices, such as using strong encryption algorithms and keeping the system up-to-date, we can ensure that your data remains safe and secure.

Services

Manager provides rich and fine grained framework for developing, testing and deploying services that implement client's business logic, interact with FinTech services and external systems. Manager comes with many preconfigured core services, covering most of common client's requirements, but you can easily create new services using Transact Fluent API and Transact functions as they are based on the standard Groovy framework.

Manager includes pre-built connectors for best in class services such as fraud detection, ID verification, digital signatures, and over 30 different connection partners. The integration has already been done, so each connector can be added to the user experience as needed when designing the application.

Every user interaction with a form is a transaction that is created and managed in Manager using the submission delivery mechanism. You have a wide of range of tools to view and manage these transactions. Manager comes with analysis tools including Insights to drill down to transactions, analyze user tends so you can make an educated decision how to improve a customer journey.

Manager stores all users' transactions, including PII[1] data, in a secure data storage and provides a mechanism to purge that data when it's not required any more. Data purging is an automated process which you can control by defining various data retention policies. Additionally, you can remove any transaction's data from the system manually. Manager lets you define how long to keep data and when to purge on the global (system), organization and form levels that gives you flexibility in overriding important settings for individual organizations. This is important for maintaining optimal performance of your server as well as to ensure that data is stored just long enough for a customer to complete their applications and after that is deleted from the system.

It's also important to remember that Temenos Journey Manager is not intended to be used as a System of Record[2] that stores large amount of data as this can affect system performance.

At any given time, system administrators can check Manager system info and health to see if any fine-tuning or adding a new server node is required, such as

Manager can be deployed in the cloud environment[3], or on-premise depending on customer requirements and available infrastructure.

Manager is implemented using a broad range of open source 3rd party libraries.

Next, learn how to create an organization.

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

[2]System of Record (SOR) is an ISRS (information storage and retrieval system) that is the authoritative source for a particular data element in a system containing multiple sources of the same element. To ensure data integrity, there must be one -- and only one -- system of record for a given piece of information.

[3]Cloud computing is shared pools of configurable computer system resources and higher-level services that can be rapidly provisioned with minimal management effort, often over the Internet. Cloud computing relies on sharing of resources to achieve coherence and economies of scale, similar to a public utility.

Journey Manager Features

Journey Manager comes packed with the wide range of features empowering you to create complex customer onboarding and user journey solutions. Some of the main product features have now become its core functionality. Click on each feature to learn more.

Main Features

- Forms for both anonymous and authenticated users.
- Global forms to enable multi-tenanted solutions.

  > WARNING
  >
  > Global Forms functionality has been temporarily disabled while we are reworking this feature.

- Form prefill with user data.
- Document attachments to facilitate form flows.
- Configurable antivirus scanning services for form attachments and internal configuration files.
- Form save and resume.
- Form delivery and provisioning.
- Receipts and attachment management.
- Services and service connections to 3rd party services.
- Form data access using the Unified Application Data Document Model and transaction properties.
- Digital signature integration.
- Collaboration workflows and product bundling orchestration.
- Form and process analytics to improve customer journey.
- Form testing, including A/B form testing.
- Security management, including Single sign-on[1] and a user identity verification.
- Management of in-flight transactions and transient data.
- Data retention and purging including manual data purging.
- Error handling, monitoring, and logging for auditability and compliance regulations.
- Integration with systems of records.

---

[1]Single sign-on (SSO) is a property of access control of multiple related, yet independent, software systems. With this property, a user logs in with a single ID and password to gain access to any of several related systems.

- Configurable support of server redundancy, quality of service and disaster recovery.
- Encryption of in-flight data for enhanced security.
- Management of complex PII[1] data and encryption keys.
- Email notifications for various form's events.
- System Event Publisher to sent server events to external systems.
- Event-Based Architecture to publish various business events to external providers using Apache Kafka.
- Quick server configuration update or re-configuration with the Configuration Service.

Features by Release

Each release of Manager is shipped with new features and bug fixes to provide you with the best-in-class software and to accelerate business agility and improvements across the board.

The key features for each major Manager release are listed below.

24.10

API Security

It is important that we provide customers with the best tools and options to ensure their solution is secure. We have made changes to Manager API endpoints to allow setting separate security managers from what is set on the Manager module. We have split out the APIs into their own module war. This will allow using different authentication methods for your APIs, such as OAuth or JWTs, then what you have configured for users accessing the Journey ManagerHome Dashboard.

Collaboration Jobs

We've added the ability to start a collaboration job earlier in the initial journey flow. Previously, if a job service was configured in a form, collaboration jobs were created automatically after the initial submission was made by a user. Now, we've now enabled creation of collaboration jobs for submissions in the Open or Saved states any time after the initial form request. Using the new `JobBuilder` Fluent API - `setInitialTxnTrackingCode()` - developers can create a job linked to such submissions, which provides better flexibility in orchestrating user's journeys. For more information, see Job Details.

Fine-grained Billing

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

We've introduced a new feature to allow more customizable tracking of the billable aspects of clients onboarding journey. Through the Fluent API a client can now create a bill record. These records indicate a billable transaction and can created on demand in any Fluent Service. Bill records can be linked together to form a journey and are periodically published out to the Temenos transaction record keeping service.

Orphaned Records Auto-deletion

We've updated the Database Integrity Checker service and the Database Integrity Checker job to auto-purge invalid orphaned database records if the Delete Orphaned Records option is selected. This option is set back to false to prevent records removal on each service run.

Password Expiry

We've enhanced the Local Security Manager with users' password expiration, rotation and notification functionality, which applies to user accounts assigned to form spaces that use the Local Security Manager for access authentication and authorization. This is achieved by introducing 2 new parameters - Number of old passwords and Password Expiry Days - allowing you to configure desired behaviors.

The Security Policy Manager job has an additional task to check user account's password expiration and send a password expiry notification email reminder to the account at a configurable time - Password Expiry Reminder Days - before the password expires.

Data Retention Management

We've added 3 new settings - Max Billing Record Age Days, Max Event Inbox Age Days, and Max Event Outbox Age Days - to the Global Data Retention Configuration and Data Retention Management v2 Service to refine data purging:

We've also added the Reset Defaults button to reset Global Data Retention Configuration with the recommended default values.

Enhanced PII Search

To ensure the optimal performance is achieved in Workspaces, the Enhanced PII Search property is set to `true` by default now.

Forms

Global forms functionality has been temporarily disabled while we are reworking this feature.

Form Data Obfuscation

We've added a new option - Force Obfuscate - to obfuscate the form data regardless of the client agent type. Some clients, such as Postman or embedded browsers, receive the data in clear text format, so when choosing Force Obfuscate, the data will be forced to obfuscate regardless of the client type.

Form Space Resources

We've added the Fluent API and REST API to create and updateform space resources.

Fluent Services

As the range of solutions expand on a given system customers often find that they have repeated services that are utilized by multiple solutions across a number of different organizations. Certain Fluent Service types can now be assigned to multiple organizations, allowing the services to be shared. Reducing the number of individual services that are deployed to a system. This helps keep the system usage down and makes support and maintenance easier.

Delivery

We've added the Not Required and Completed delivery statuses to the Transaction Status UI.

Receipt Generator Service

We've added a new Receipt Generator service to create receipts using only a form design and a form XML data.

Security Manager

We've replaced the single Strict Header Security option with 3 separate options - Strict Header Security By Host, Strict Header Security By IP, and Strict Header Security By User Agent - in the Security Manager configuration and the Form Version configuration. This is to provide a fine-grained control of the headers.

For more information, see Manager 24.10 Release Notes.

24.04

Data Retention Enhancements

We've created a new version of the data retention management v2 service, which provides more efficient purging processes and added resilience to issues caused by bad data. These improvements will help prevent abnormal database growth that often leads to performance degradation. We have also added more visibility through enhanced logging that help narrow down the individual steps during the data purging process.

Configuration Service Extension

The Configuration Service can now create properties and parameters.

We have also extended the coverage of the Configuration Service, which includes setting of form space and module context paths, updating security manager parameters and organization security settings, such as CSP, as well as scheduled job parameters and form page tracking. For more information, see Import Forms, Import Form Spaces, Import Groups, Import an Application Package, Import an Organization, Import Roles, Import a Security

Manager, Import Services, Import Service Connections, Import TPacs, Import Scheduled Jobs.

The Configuration Service now supports pipeline deployments.

Scheduled Jobs

We've added export and import support for scheduled jobs via the Manager UI, SDK and API.

We've also enabled updating of scheduled jobs via the Configuration Service.

Platform Updates

We've updated Manager to Open JDK 21 and Wildlfy 31.0.1. For more information, see 3-rd party library updates.

We've added support for Ubuntu OS, such as Ubuntu 22.04.4 LTS, so this version can now be provisioned in our cloud environments as well as it is available in the Manager 24.04 installer for on premise environments.

> NOTE
>
> The Manager installer versions 23.10, 23.04, 22.10, 22.04, 21.11, 21.05, 20.11, 20.05 have been back-ported to support Ubuntu OS.

Ubuntu is a very secure, widely used open-source operating system with strong community support. It is enterprise-compatible and has proven to be performant and efficient. Ubuntu is the standard Temenos OS for Linux based systems in cloud environments. For more information, see Journey Manager System Requirements.

We've removed support for CentOS 9 and 7, because CentOS 7 will reach end of life (EOL) on June 30, 2024 and we have made a decision to continue with Ubuntu OS. For more information, see What to know about CentOS Linux EOL.

We've added support for Oracle MySQL 8 database service.

We've enabled configuration of a context path of a Manager module during software installation, so the Configuration Service can be run right after that to perform other system configuration tasks. For more information, see Installation > Context Path.

Eventing Enhancements

We've added 2 new types of services - Event Listener Configuration and Inbox Processor - to define services that consume Kafka topics and performs REST API calls specified in the topics. These services are Event Listener Configuration Service and Event Inbox Processor Service.

You can view all configured Event Listener Configurations and stop and start them as needed.

We've introduced 2 new deployment properties - Eventing Error Log Flood Threshold Count and Eventing Error Log Flood Threshold Time - to control the growth of the error log table if the above services fail and thus generate a high number or error entries.

Security

We've added 2 new deployment properties - Allowed File Name Extensions In JM and Disable Render Link for Portal Resources - to prevent users from uploading malicious files on the portal resource page and subsequently rendering them, causing the malicious code to execute.

We've implemented a data type validation for file upload when you create or edit organization reference data object.

We've added a new deployment property - Regex Pattern For Url Fields, which is regular expression used to validate the URL fields of form space properties, module properties, security managers and form flow configuration.

For more information, see Manager 24.04 Release Notes.

23.10

Global Form

We've introduced a feature that allows rendering of forms across multiple organizations. Submissions can be rendered against a global form from other organizations allowing for single deployment of solutions that can be shared. Also, new or existing forms can be configured as global. For more information, see form import, form details and form view.

We've changed the way global services, functions and tasks obtain correct organizations. For more information, see Create a Service, Create a Transact Function and Create a Task.

System Issues

A new card has been added to the Journey Manager Admin Home Dashboard that displays unresolved system problems that are affecting performance and maintenance. This dashboard report is driven by the database integrity checker introduced in 23.04.

Security Enhancements

We've introduced the following:

- Default Admin Password – You can now set the Administrator password for new Journey Manager instances during the installation process. For more information, see the installation via the UI or a script.
- Virus Scanning on Admin UI – Uploads to the Manager UI will now be virus scanned, but this feature can be disabled, if needed.

Enhanced Configuration Service

The Configuration Service has been further extended to cover additional entities in Journey Manager.

Renamed Work Space

The Work Space form space has been renamed to Journey Space, so there is no confusion with the Workspaces portal. The Context Path URL has also been changed from `https://my-domain.com/workspace/` to `https://my-domain.com/journeyspace/`.

For the changes to take effect, you need to have a fresh installation of Manager or the Journey Space form space. If you only upgrade Manager, you still have the old Work Space name with the old Context Path URL as `https://my-domain.com/workspace/`. | 23.10This feature was updated in 23.10.

OS and Database

We've added support for CentOS Linux[1] 9 and Microsoft SQL Server[2] 2022 and 2019. For more information, see Journey Manager System Requirements.

For more information, see Manager 23.10 Release Notes.

23.04

Configuration Service

Introduced a few Configuration Service allowing you to quickly and easily reconfigure a server instance by uploading a configuration file containing all relevant information.

Database Integrity Checker

Added the Database Integrity Checker service and scheduled jobs to validate database integrity periodically. This service will generate a report for any issues in the Manager database that need to be addressed.

New Database Integrity Checker Event Type and Configuration Service Event Type

Introduced 2 new event types in Event Log.

Changes to Jakarta EE

---

[1]CentOS (Community Enterprise Operating System) is a Linux distribution that provides a free, enterprise-class, community-supported computing platform functionally compatible with its upstream source, Red Hat Enterprise Linux (RHEL).

[2]Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

Replaced Java EE with Jakarta EE. This is a requirement for WildFly, and will affect all groovy services within Journey Manager for customers upgrading. For more information, see Migrate to Jakarta EE.

Jakarta Migration Support

In order to support our customers upgrading Journey Manager, we've created a service allowing you to convert Groovy services and security managers to be Jakarta EE 10 compatible. You can download this service (service-jakarata_transformer-v0.1.0.zip) from the download page. We have also enhanced our SDK documentation with strategies for migration of services.

Upgrades & security

Updated 3-rd party libraries with the normal library updates to address security vulnerabilities as well as with some updates to major core technologies, such as Jakarta EE and WildFly.

For more information, see Manager 23.04 Release Notes.

22.10

Event-Based Architecture

Introduced the concept of Event-Based Architecture where you can publish and subscribe to events using Kafka. In this release, you can only publish events.

Apache Kafka

Created a Kafka service connection to establish a connection to Apache Kafka 3.2.3 to publish events.

DocuSign OAuth service connection

Added a new DocuSign OAuth service connection to connect to DocuSign services using OAuth user authentication.

OpenShift support

Added support for Journey Manager deployments using OpenShift[1] containerization.

Billing support

Added support for billing with more fine grain control to allow for setting of a billing flag on transactions. A billable flag on the Submission class is also exposed via Fluent API, so you can access it via `Txn` objects and update it via `TxnUpdater`.

Operations portal

---

[1]OpenShift is a family of containerization software products developed by Red Hat. Its flagship product is the OpenShift Container Platform — a hybrid cloud platform as a service built around Linux containers orchestrated and managed by Kubernetes on a foundation of Red Hat Enterprise Linux.

The new operations portal can be deployed using the Journey Manager installer

Upgrades & security

Updated 3-rd party libraries with the normal library updates to address security vulnerabilities as well as with some updates to major core technologies, such as Wildfly 26.1.2.Final and Open JDK 17.0.4.1+1.

New look and feel

Rebranded the Manager product according to Temenos' vision, mission and strategy.

For more information, see Manager 22.10 Release Notes.

22.04

Status Management

Improved the existing notification messages interface to let you create and schedule status change of various entities, such as form spaces, organizations, forms, and form versions.

Added a new status management service to automatically execute scheduled entity status changes.

Workspaces API updates

Provided a set of new REST API to expose existing Operations as a service. These API will be consumed by new Workspaces functionality.

Added new permissions to the Workspacesform space to provide fine-grained control for new Workspaces Operational Portal REST API, such as `txn/query`, `txn/status`, `txn/task`, and `txn/purge`, as well as the existing `txn/cancel`.

Maven SDK

Included a new version of the Journey SDK, which requires the Journey SDK Maven Plugin to run. We've modified the underlying framework of the integration points between Manager and the SDK to accommodate this change.

Upgrades & Security

Updated 3-rd party libraries

For more information, see Manager 22.04 Release Notes.

21.11

New Statuses added for organizations, form spaces, form versions and forms

Introduced a new set of statuses for organizations, form spaces, forms and form versions, so you have fine control over each of the entities.

New ServiceUser Role

Scheduled Groovy services now run using the ServiceUser role, which defaults to a admin level role. This role can be limited to enhance the security around these services, if required.

New CSRF Protection Option

Added the CSRF Protection option to the Workspaces, Workspace and Web-plug spaces.

For more information, see Manager 21.11 Release Notes.

21.05

New REST service - Proof of Concept

A new service which exposes some of Manager's Fluent API via REST[1] end points.

Fluent API

We've further expanded the Fluent API to expose more features to help reduce the dependency on the deprecated Core Groovy API:

- Form VO - Included the form type through the `getFormType()` method.
- `TxnUpdater`, `JobUpdater` and `TxnQuery` - Included feature to view and manager comments against Transactions and Jobs
- `TxnUrlBuilder` - Included the capability to generate secure URL for forms/transactions through the `setSecured()` accessor method.
- `TxnUpdator` - Included the capability to lock and unlock the save Challenge lock state for a transaction

Manager Installer

We've enhanced the installer so that for AWS installs using MySQL SSL is enabled by default to improve server security. For more information, see the SSL JDBC Connection Enforcement for MySQL section of the installation guide.

Due to components requiring commercial licenses, iText, Oracle and Luna dependencies are not included with installs by default. These will need to be deployed separately. For more information, see the Manual Libraries Deployment section of the installation guide.

Reports

Remove the PDF export option from the Transaction and Job Licensing reports because it uses iText, which requires a commercial license.

For more information, see Manager 21.05 Release Notes.

---

[1]REST or RESTful API design (Representational State Transfer) is designed to take advantage of existing protocols. While REST can be used over nearly any protocol, it usually takes advantage of HTTP when used for Web APIs.

20.11

Core Services

Added support for:

- A new magic byte scanning service
- A new Attachment File Name Regex deployment property

For more information, see Manager 20.11 Release Notes.

20.05

Azure Cloud support

Added support for:

- Encryption key management using Azure Key Vault
- Attachment and Receipt storage using Azure Blob Storage
- System Event Publisher using Azure Queue

Automatic deployment and testing

Added support for:

- The ExampleImportService, which is used to import example artifacts into Manager on initial start-up, now supports all archive types. This is useful to import test artifacts for automation testing.
- The bulk user import spreadsheet is expanded to support Groups, Roles and Organization. This is also incorporated into the ExampleImportService.

For more information, see Manager 20.05 Release Notes.

19.11

General Fluent API enhancements

Added support for:

- `TxnUpdater`, `TxnQuery`, `TxnBuilder` and `Txn VO` to improve the transaction level capability of the Fluent API
- `Emailer` and `UserEmailer`
- Added a new `Job Builder` class

Fluent API receipting enhancements

Added support for:

- Calling a named receipt render service from within a Fluent Receipt Service to enable receipt merging while still generating a receipt for the associated transaction.
- Generating a Web and Email received from a Fluent Receipt Service.
- Externalization of the Core PDFUtils methods to merge receipt in Fluent.

Fluent Security Managers support added

Added support for:

- A Fluent User Authentication Provider, with a template for a skeleton Fluent Groovy User Auth Provider.
- A Fluent Security Manager, which includes a template for a skeleton SSO Security Manager and an out-of-the-box ADFS Security Manager.
- Fluent Security Managers implementation has resulted in a number of new Fluent API classes to support the scripting within a Security Manager.

Form version import enhancements

Form version import options include Form Function now.

Removed Features

Adobe LiveCycle support removed

Card Payment page removed from the Workspace

For more information, see Manager 19.11 Release Notes.

19.05

Release Key Features	Description
Workspace API	Enhanced to support the development of the new help desk persona for the Workspaces, including: • Transaction search using multiple criteria • Additional updates to Txn and Job Queries • Security enhancements
Fluent API	Enhanced to reduce the dependency on the Core API and improve upgrade supportability, including: • New PaymentLog Query class • Improved TxnQuery transaction search capabilities • New feature to support collaboration jobs • Added form expiry information to the Form object

Security enhancements	Improved CORS support for form spaces
Chromium puppeteer receipt service	Added a new Chromium puppeteer receipt service to render forms as PDF receipts
Library upgrades	Upgraded the dependent libraries, including: • Apache Cayenne 4.0 • Spring Security 4 • Wildfly 16
Removed features	Removed: • Birt and PayPal Payment gateway. It is replaced with the Transaction and Job Licensing screens for transaction usage information. • PayPal Payment gateway.

For more information, see Manager 19.05 Release Notes.

18.11

Release Key Features	Description
Workspace API	Introduced new REST API to support the development of rich UI experiences for 'Back of House' staff who support complex workflows and provide help desk functions to end users. The API utilizes Swagger and Spring Security.
Static content externalization	Added a new Maestro form rendering mode which significantly reduces the time it takes to load large forms. This is important as long application loading times can significantly increase the bounce rate with users simply closing the browser window before starting the application.
System Event Publisher	Added a new user authentication event feed to send user login events. Enhanced scheduled purge time and actual purge time for a transaction.
Security hardening	Added support for: • Added Validation of date field to the workspace portal to

	fix internal error when provided date is invalid. • Security around service connection password in the manager UI. Removed confirmation page and receipting from the Server Monitor Form configuration.
Library upgrades	Upgraded the dependent libraries, including: • Open JDK 11 • User Agent Utils 1.21 • Spring Framework 4.3.19 • Spring Security 43.1.7 • Bouncycastle 1.60 • SDK 3.9 • org.apache.xmlgraphics batik 1.10 Removed: • JLine
Platform support enhancements	Added support for: • Microsoft SQL Server 2017 database. • Microsoft Windows Server 2016 operating system.
Fluent API	Added new API.
Behavior changes	Enhanced management of submissions' lifetime.
Removed features	Removed: • FopRenderReceiptService receipt render service and `org.apache.xmlgraphics` library • Use of Flash in the Manager UI • Apache Camel Support for Groovy Services

For more information, see Manager 18.11 Release Notes.

18.05

Release Key Features	Description

System event publisher	Introduced the System event publisher to connect external systems through an AWS SQS queue, in near real time, where the messages can be picked to provide up to date information about the states of form transactions in Journey Manager.
Library upgrades	Upgraded the dependent libraries, including: • Apache Groovy Lang 2.4.15 • Eclipse BIRT Runtime 4.6.0 • JDBC Database Drivers (MySql, Oracle and SQL Server) • JUnit 4.12 • Oracle Java JDK 1.8u172
Platform support enhancements	Added support for new Insights, including: • Centrally stored ECCL Script • Fluent API to send Insights Segments from server side Groovy Scripts
Fluent API	Added new API
Behavior changes	Enhanced the transaction abandonment and data purging to ensure PII data is removed accordigly.

For more information see Manager 18.05 Release Notes.

17.10

Release Key Feature	Description
Transact Functions	Introduced Fluent Functions to allow you to use functional programming within Journey Manager, so you can now write a function once and simply trigger it at whatever points or points you need.
Continuous Integration	Provided improved CI-based testing support. Created a definition of what is included in the application package and versioned in GIT or another SCM tool.
Transact SDK	Combined developer tools into a unified Transact SDK to provide a single set of tools to help developers make the most of integrating with Transact API while using Continuous Integration (CI/CD).

Reference data	Introduced Transact Reference Data to decouple reference data from the form, decrease the form loading time, and enable you to deploy once and use in many forms.
Semantic versioning	Unified versioning across the platform for consistency including full semantic versioning across all forms and services. Semantic Versioning enables two teams to work in parallel on more than one form solution and permits hierarchical management of versions. Additional features include support for up to 3 levels of versions and version branching.
Security hardening	Add support for Strict Security Headers to enforce that each HTTP request within authenticated user session is checked to make sure it is the originated source that created the session. The HTTP request's IP Address and Browser User Agent are checked.
Collaboration Jobs	Enhanced collaboration jobs to support group sharing.

For more information, see Manager 17.10 Release Notes.

5.1

Release Key Features	Description
Data retention policy enhancements	Retention duration time is reduced to make sure Journey Manager is used as a system of engagement and not a system of records: • Transactions - reduced from 180 days to 90 days. • Transaction history - reduced from 2 years to 180 days.
Transaction history data warehouse integration	Added the following publishing services to publish transaction history directly to a data warehouse: • AWS S3 • File system • Fluent Groovy
Automated deployments	Introduced a new script-able install or upgrade mode for deployment automation, which supports Continuous Integration (CI) testing. This mode is beneficial to both hosted services and IT professionals in that it helps to automate provisioning and upgrading, which in turn can improve quality by reducing operator errors.

Fine grained form space authentication policies	Enhanced forms authentication policies, which depend on form space they are in.
Fluent Groovy API	Added new Fluent service types. • Form version Selector • Receipt Render • Transaction History Publisher • Virus Scan Service
Application packages	Added new configurations for Organization Properties, Delivery Channels, and Service Connections.
Security manger logging	Introduced a new Security Manager logger that can be used in all Security Manager Groovy scripts.
Update to latest Maestro form from within Manager	Enhanced Maestro form updates from Journey Manager, so it uses the latest build of the Maestro form without exporting a Maestro Form Version Archive and then importing it Manager.
Maestro localization support	Introduced localization to Journey Manager and Maestro, which allows multiple languages to be added to a single form. This feature allows a form to change languages based on the users of the form.
Collaboration Jobs	Form and service version pinning: • The form code for Review Tasks is no longer necessary. Task Assign actions that do not specify a Form Code will by default use the same form code and version as the previous submission. • The Form Version can be explicitly set in the Job Definition. • The Job Action Version can be explicitly set in the Job Definition.
Collaboration Jobs	History logging, so the state of the job is recorded and logged at the end of each branch of processing completes.

For more information, see Manager 5.1 Release Notes.

5.0

Release Key Features	Description
Fluent Groovy API	Introduced completely new and improved API, which can be used from Groovy scripts.
Maestro form version import	Added support for Maestro form version import defaults, which provides specialized import behavior if a form version archive was created in Maestro.

For more information see Manager 5.0 Release Notes.

4.3

Release Key Features	Description
Application packages	Added support for: • Form bundle and form chaining • Receipts with Groovy scripts • Dynamic task assignment • Cancellable tasks • Publish and subscribe data extract sharing • Unified navigation support
AWS KMS	Added support for AWS Key Management System (KMS) data encryption in the Cloud Service Enterprise VDC edition.
Groovy service tests	Added comprehensive Groovy Services test framework to support accelerated development and development best practices.
REST API	Added Transactions, Service Definitions and Groovy Service Invoke REST API.
Security hardening	Added security role and group grant leases, improved SEM logging support and Apache Mod Security configurations.
Service improvements	Added Form Security Filter, Groovy Receipt service types, service versioning support, Groovy Service Logging, improved documentation,

	service editing and form and service definition Organization Application Packages.
Transact Integration Gateway	Introduced TIG, which provides a light weight on-premise integration module for performing deliveries and proxy calls to internal systems. This module replaces the TIA and TDA modules.

For more information, see Manager 4.3 Release Notes.

Next, learn more about deprecated features.

Data Protection

Journey Manager safeguards sensitive form data in transit.

Journey Manager protects your data as it travels between your device and our servers using industry-standard TLS[1] encryption. This robust encryption ensures that your information remains confidential and secure from unauthorized access. Consequently, data in transit is protected from eavesdropping and tampering.

Additional Data Protection

For an extra layer of protection, Manager offers the optional Form Data Obfuscation. This feature further scrambles sensitive data, making it even more difficult for potential attackers to decipher. Although the key exchange mechanism for this feature is weaker compared to TLS, it still serves as a valuable component of a defense-in-depth strategy. Given the robust security provided by TLS, this additional layer is typically unnecessary in industrial applications. However, this additional measure can be a valuable part of a comprehensive security strategy.

Why TLS is Sufficient for Most Use Cases

In most cases, as used by millions of people and businesses worldwide, the high level of security provided by TLS is sufficient to protect sensitive data in transit. By following industry best practices, such as using strong encryption algorithms and keeping the system up-to-date, we can ensure that your data remains safe and secure.

Next, learn about all Journey Manager's features.

[1]Transport Layer Security (TLS) is a cryptographic protocol designed to provide communications security over a computer network.

Unified Application Data Document Model

Journey Manager comes with the Unified Application Data Document Model that allows you to store integration data separately from a form data. This is the recommended approach as compared to the transaction properties.

The Temenos Journey Manager platform is essentially based on a document oriented architecture where we provide a seamless customer experience to capture data for further processing. During the application life cycle we persist application data in XML format that can be used to deliver data to the back end for customer onboarding. During the application process we also integrate with various third party to perform operations such as lookup, prefill, validation, authentication, authorization, funding, account onboarding and list goes on. These operations are executed on server side and responses may return back to application front end to perform next steps. End user may not need to see every single detail of the operation performed by the server, however it is important to persist the state and responses on the server for making decisions based on the outcome, or deliver to back-end for detailed outcome of those operations. So persisting that information becomes as critical as persisting user entered information.

In earlier days, developers persist server side information as Manager properties and use it for making a decision. These properties are saved as database entities against the transaction and available until transaction is purged. This is basically a key-value pair that can be retrieved by querying transaction property name.

Since Manager 17.10, you can enable the Unified Application Data Document Model mechanism in a form version by selecting the Unified App Data checkbox.

By default all transactions use `AvokaSmartForm` as the root node for transaction XML. Upon enabling this feature, Manager adds an additional wrapper called `Root` to `AvokaSmartForm`. During the form render request, Manager only sends or merges nodes under "AvokaSmartForm" and the rest of the information stays on the server. This allows the developer to persist all information that is not relevant or sensitive to the client on the server itself.

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
  <AvokaSmartForm>
    <Applicant>
      <FirstName>John</FirstName>
      <LastName>Doe</LastName>
    </Applicant>
  </AvokaSmartForm>
</Root>
```

When to use Unified App Data

Both Transaction Properties and Unified App data model allow the developer to persist server side information. Since there is no right or wrong approach but Unified App model was introduced to make the developer's life easier and serves many benefits over the conventional approach to save information as transaction properties such as:

- Data Integrity - It is important that application data is not manipulated by accessing form data JSON on client side. If we completely rely on form data, then there is a possibility that if someone intentionally try's to override the application outcome. Persisting data on server side XML allows developer to validate the information or use that information as a base for further processing rather than completely relying on data passed by front end.
- Data Marshaling - Manager provides APIs to allow developers to generate and populate VOs from XML structure. This allows developers to work with POJOs rather than arbitrary values. This helps to write better unit tests against various application states.
- Data validation - Data can be validated against XML Schema validation rules.
- Data Sensitivity - Critical information that doesn't need to be exposed will still be available on server by accessing transaction XML. Developer doesn't need to adopt any additional approach to persist anything beyond transaction XML.
- Data Delivery - Delivery API can deliver transaction XML which will automatically deliver the server side information to a customer where transaction properties require additional effort.
- Encryption - A transaction form XML data is always obfuscated in the database, while transaction properties are not. This protects against any PII data being exposed.

When you prefill the application or hydrate it with details like created account numbers, you should consider carefully what data to include in each tag because it goes to the browser. In most cases, you should only send masked values, not the raw ones. You should keep the raw values on the server-side only.

Springboard by default has some design patterns around this with the Lock Customer function, which you can use as a best practice.

When to use Transaction Properties

Transaction Properties and Unified App Data are not mutually exclusive. The developer can still use transaction properties for scenarios such as:

- Reporting - You may require a custom report on transaction data which becomes pain by parsing form xml for every transaction. A data extract or transaction property (as long as it doesn't contain PII) can be used to save the information that you can report on.
- Workspace - As at the time of writing this article, workspace API are designed to read transaction properties to display integration outcome / high level information on the screen. This is currently not designed to read form xml. There are plans to include form xml support in future.
- Immense data - There are some use cases where developer needs to persist huge amount of information (may be a raw response). This is still debatable but I personally don't feel to record immense amount of data as server xml, specially responses returned by 3rd party vendors as we don't know what the data format would be. Saving unknown information that may include unsupported xml values can lead into issues. This kind of information can persist as Transaction Property (blob) in the database.

Next, learn about the data model.

Save Challenge

Journey Manager comes with a Save Challenge feature that allows users to resume their form applications, so they can complete or cancel them. This is an important step in the onboarding process when a user needs to save a form application to get some required information or a web browser is closed or crashed. Therefore, the Save Challenge provides a secure mechanism to return to the form.

To enable a Save Challenge:

1. Select Forms > Forms.
2. Locate a form and select the Flow Config tab.
3. Select Enabled from the Save Online dropdown list. For more information, see Configure the User Flow options for the form
4. Provide a valid path in the Save Challenge XPath field in Form Configuration Mapping, for example, `//DateOfBirth`.
5. Check the `supportsSaveChallenge` is set to `true` in the SDK `form-def.json` file.

   NOTE

   The `application-package-config.xml` is updated with this option after running the `app-package` command.

6. Check the Supports Save Challenge checkbox is selected in a form version.
7. Check the Save Challenge Data checkbox is selected in Transaction Status.

Next, learn abut other Manager's features.

Temenos Journey Manager Integration

The Temenos Journey Manager enables digital interaction between an organization and its customers. Because of the importance of this interaction, there is often a need to integrate between the Temenos Journey Manager platform and other systems. In some cases, the integration required can be minimal, but integration can be quite extensive.

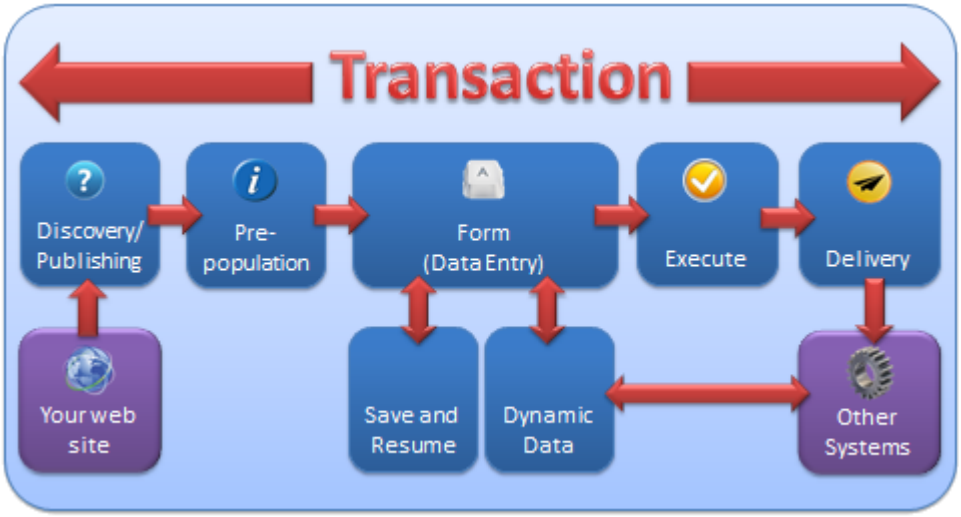
- Integration areas occur at different stages within the interaction between a user and the organization.
- Within each integration area, there are different integration implementations, to support different technologies used by different organizations.

Integration Areas

There are several key integration areas or stages in which integrations generally occur during transactions:

- Transactions Discovery
- Data Pre-population
- Data Entry Streamlining
- Execution
- Save and Resume
- Delivery

This is illustrated in the diagram below.



These integration stages are detailed described below.

Transactions Discovery

Your users need to be able to discover the transactions that you provide. There are a number of ways to do this. The simplest is to have a link on your web site which launches the transaction.

Things can get more sophisticated than this. For example:

- You may want to have customized links, which are different for different users. Generally this is so that you can provide custom pieces of data in the link URL, generally so that you can pre-populate some of the fields in the form from this data. This helps to make it easier for users to fill out forms (see Pre-population below).
- If your users are logged into your web site, you may want to pass their identity to the transaction. This again allows the form to pre-populate some of the fields.
- You may want to embed the form into your existing web site, rather than linking to it. This is generally not recommended in modern responsive and task-oriented designs, although it is possible to implement with Transact.
- Another option is to send the user a link to the transaction using some other mechanism, such as via an email. You can send them a link to a blank form, alternately you can also use Transaction Manager to create a custom link that contains a partially pre-filled form to make their lives easier.
- Users can also save a transaction that they have partially completed. This will generate a custom link for them, that will enable them to return to the transaction at a later date. There are also other ways for them to get back to the transaction if they lose the link. You can configure a challenge question to ensure that only the original author can access their saved transaction.

Data Pre-population

We want to make it as easy for your users to complete a transaction as possible. Transact is specifically designed to make the entire experience as intuitive and simple and engaging as possible.

You can make the data entry process even simpler by pre-populating (or pre-filling) information into the form that you already know about them, which means that they won't have to type the information themselves. Depending on what information is available, this can dramatically

We can generally provide the pre-fill data in one of the following ways:

- Front-end data passing. If your existing web site knows who the user is, you can pass specific pieces of data about that user directly to the transaction. This can be done via URL parameters, cookies, lookup tokens, and a variety of other techniques.

- Back-end look-ups. If the user has been authenticated by your web site, you can pass their identity to the Transaction Manager server. We can then use this identity to look up information in back-end business systems, or Transaction Manager's own profile database, to pre-fill the form.

Pre-population is also sometimes provided by using a previous submission's data to populate a new submission. This is useful when a transaction has to be repeated every year (or periodically). and the data doesn't change much from submission to submission.

Data Entry Streamlining

During the data entry process, we want to make it as easy as possible for users to complete the form, and get the product or service they are requesting.

There are a number of ways we can do this. For example:

- We can populate fields in the form from third party services, such as LinkedIn or Facebook. (This is similar to pre-population, but occurs once the form has already loaded, rather than prior to loading.)
- We can do calculations and lookups to provide them more information, or to validate their data entry. For example, we might verify the name of the account holder that they are trying to transfer funds to, or calculate the approximate repayments for their loan.
- We can provide them ways of speeding up their data entry. For example, instead of forcing them to type in an address laboriously, field by field, we can allow them to just start typing an address, and provide a matching set of possible candidate addresses, and then auto-complete the individual fields. (Like the address picker in Google Maps.)
- We can ask the user to upload a document, such as a photo of their license. We could use OCR technology to recognize the text on the document, and used to populate fields on the form.
- We can give the user real-time validation of their eligibility. For example, we could integrate with a credit scoring service to provide an indication about whether their loan is likely to be approved or not.

Data entry optimization is usually done using one of two techniques:

- Client-side integration. This occurs directly between the form (executing in the client's browser), and the information system. The LinkedIn pre-population widget is this type - no server-side installation or configuration is required.
- Dynamic Data. In this case, the form will invoke Transaction Manager, which will in turn invoke the actual service. Several Composer widgets are designed to work with Dynamic

Data Services. Transaction Manager can provide added value, such as data and protocol transformation and security.

Execution

Execution is the process of adding additional value to the transaction after the data entry step. It can consist of a number of added-value steps, some of which require may integration with back-office or third party systems.

Some examples of added value integrations include:

- Adding electronic signatures using third-party eSignature solutions.
- Processing payments using third-party payment gateways.
- Retrieving credit scores or identity verification results using third-party services.
- Using Enterprise Document Management systems to generate documents or welcome kits to provide to the customer.
- Providing a review-and-approval step prior to delivery.

Save and Resume

Saving a form and returning to it to resume editing is an important capability. This is shown on the diagram for completeness, but is in fact implemented purely inside of Transaction Manager. There are no user-modifiable extension points.

Delivery

Delivery is the process of delivering the results of the transaction, in both human-readable and machine-readable format, to the organization.

Of all the different points of integration, delivery is the most important. The reason is obvious - unless we deliver the data that was collected (in some way) to the organization, there is no value achieved for the organization.

The range of different delivery options is highly varied. It can be as simple as an email containing a PDF as an attachment which is processed manually, or as complicated as direct integration of different parts of the data into a number of different back-end business systems. Delivery can even be performed for transactions that haven't been completed by the end user, as this abandonment information can itself be useful.

Authenticated and Unauthenticated Transactions

Manager supports the use of both authenticated or unauthenticated transactions. Authentication is usually the process of identifying a user using a user-name and password, or sometimes more

sophisticated methods. An unauthenticated user is a user who hasn't logged into Transaction Manager, and doesn't have an identity in Transaction Manager. This does not necessarily mean that they are unknown, and usually some information will be known about them, often including their email address.

Similar capabilities are available for authenticated and unauthenticated users, but are implemented in slightly different ways:

Capability	Authenticated	Unauthenticated (Anonymous)
Pre-population	A user's identity can be used to pre-fill forms with information that is already known about that particular person. This assumes that the user's identity already exists in an existing system.	Pre-population can still be achieved, but is implemented without the user actually logging in. For example, pre-fill data can be passed to the transaction using URL parameters.
Save and Resume	An authenticated user can save a partially completed transaction. By logging in again later, the user can view their partially completed transactions (known as drafts), and resume editing. The user can also log in from a different device.	Save and resume is implemented via a simple reference code and challenge question (similar to an airline booking code).
Transaction History	When an authenticated user logs into Transaction Manager, they can see the history of their submissions. (Note that they will usually only be able to see a summary of the transaction, not the content, because generally data retention policies will result in the details being purged.)	Transaction history can be provided to in a number of different ways, such as a confirmation email. Historical information is often maintained in other systems that maintain a more complete history of the customer's interactions with the organization, such as CRM systems.
Tasks	A task can be assigned to a user who is authenticated in Transaction Manager, and appears in their task list.	A task can be assigned to an unauthenticated user via email or any other system that can display a URL. The user will simply click on the link in order to perform the task.

A question that is often asked is whether to use authenticated or unauthenticated access. Transaction Manager supports both types extremely well. However, it is important to note the following regarding authenticated transactions.

1. Many organizations already have the concept of identity and usernames/passwords. Usually the organization will not want to introduce a new and different username and password, but rather re-use the existing mechanism.
2. Users are often reluctant to sign up with a username and password just in order to be able to save and resume a transaction later. (We have data that supports this assumption.) This can increase transaction abandonment, because users simply don't want to register. It's usually wise to allow users to complete a transaction without requiring them to register.
3. Identity can also be complicated to implement, especially the identity of your customers outside the firewall. Transaction Manager does support both common signon using LDAP, and Single Sign On (using SAML), but these can add complexity to the implementation.

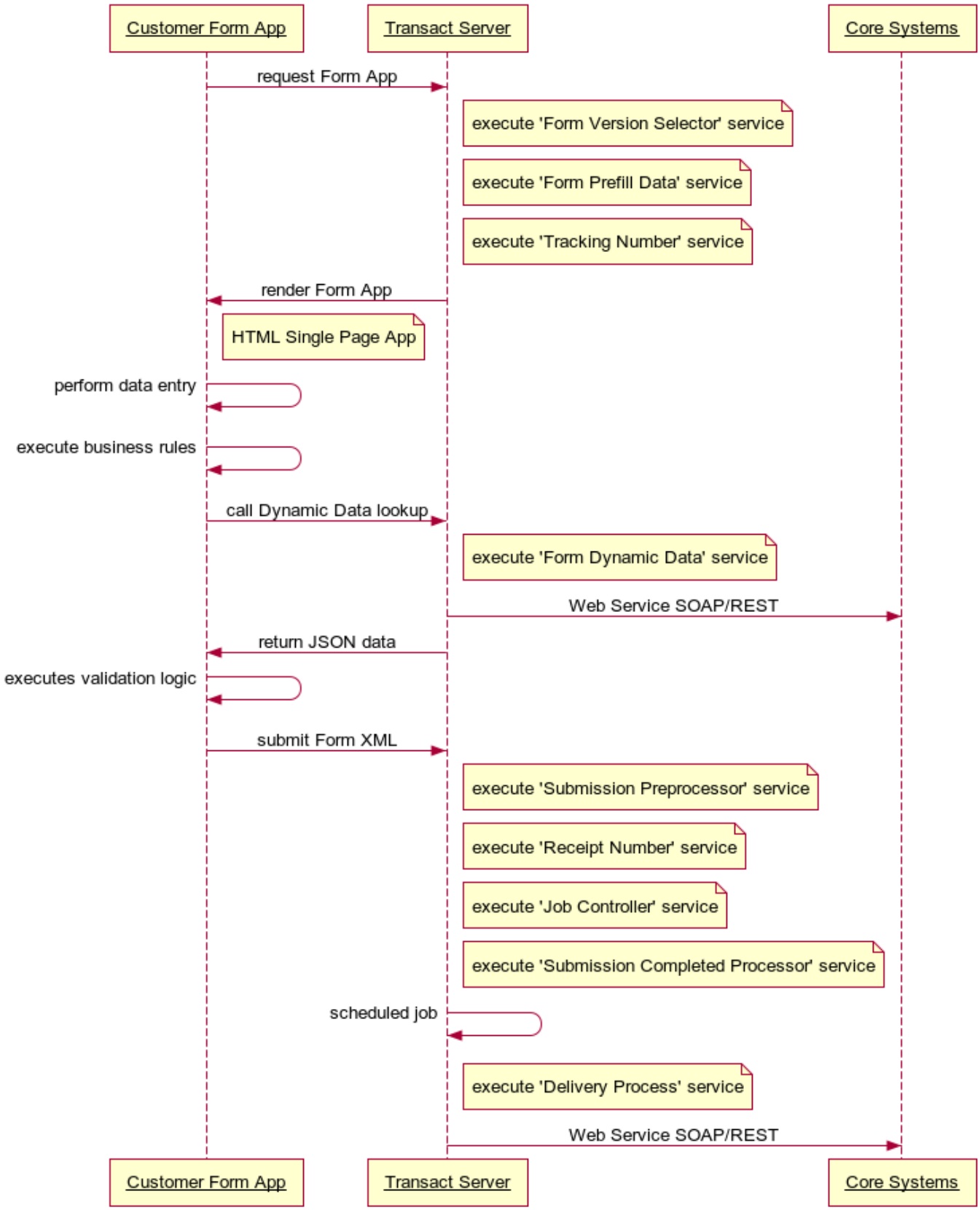
Because of the above, we generally recommend the following:

- External customers be treated as "anonymous" users, without trying to establish or confirm their identity online. This may have a minor impact on their ability to pre-fill their forms (although there are ways to achieve this), but in general, treating them as anonymous dramatically simplifies the integration requirements, the security implications, and the time to market. As outlined above, most features in Transact such as save and resume, task assignment and pre-fill have been designed to work with either anonymous or authenticated users.
- Internal users (such as help desk and call center staff) be treated as authenticated users. This is usually much simpler to implement, since all internal users are already authenticated on the network, and both LDAP and SSO are relatively simple to implement.

In some cases authenticated end users may be required for business reasons, and Transact does support this extremely well - but bear in mind that it may affect your time and cost of implementation.

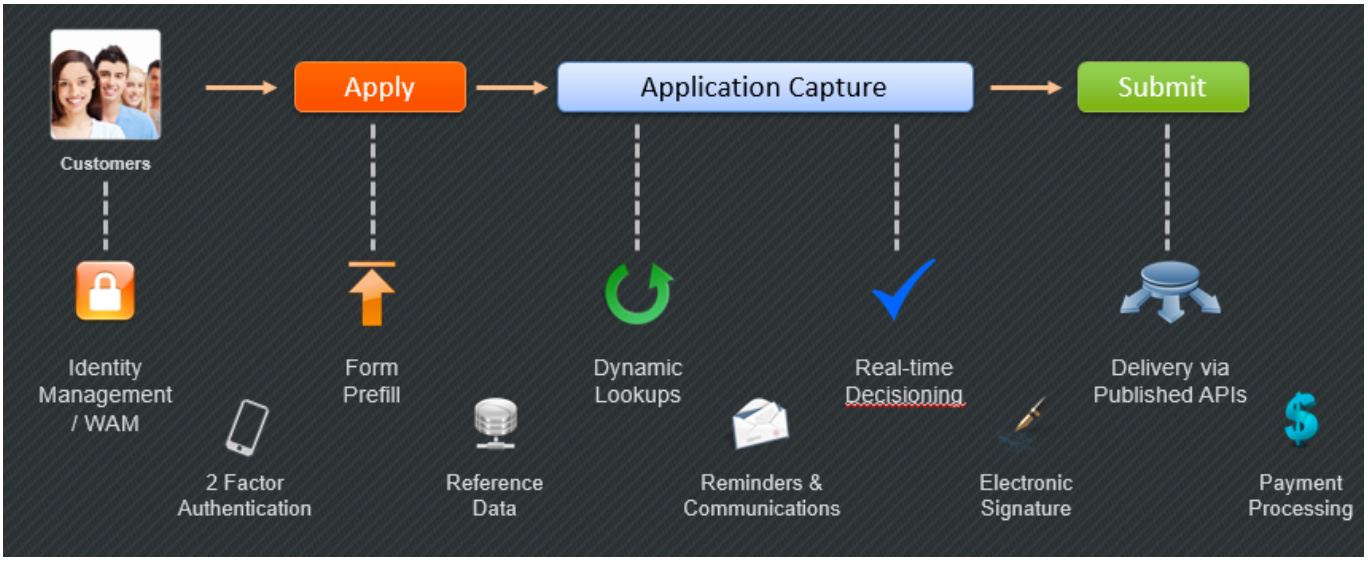
Extension Points

Integration is performed in various areas of the transaction life-cycle though the use of defined extension points. Developers can inject code or other invocations at these defined extension points. The main points are shown in the interaction or sequence diagram below, although some extension points have been omitted for simplicity.



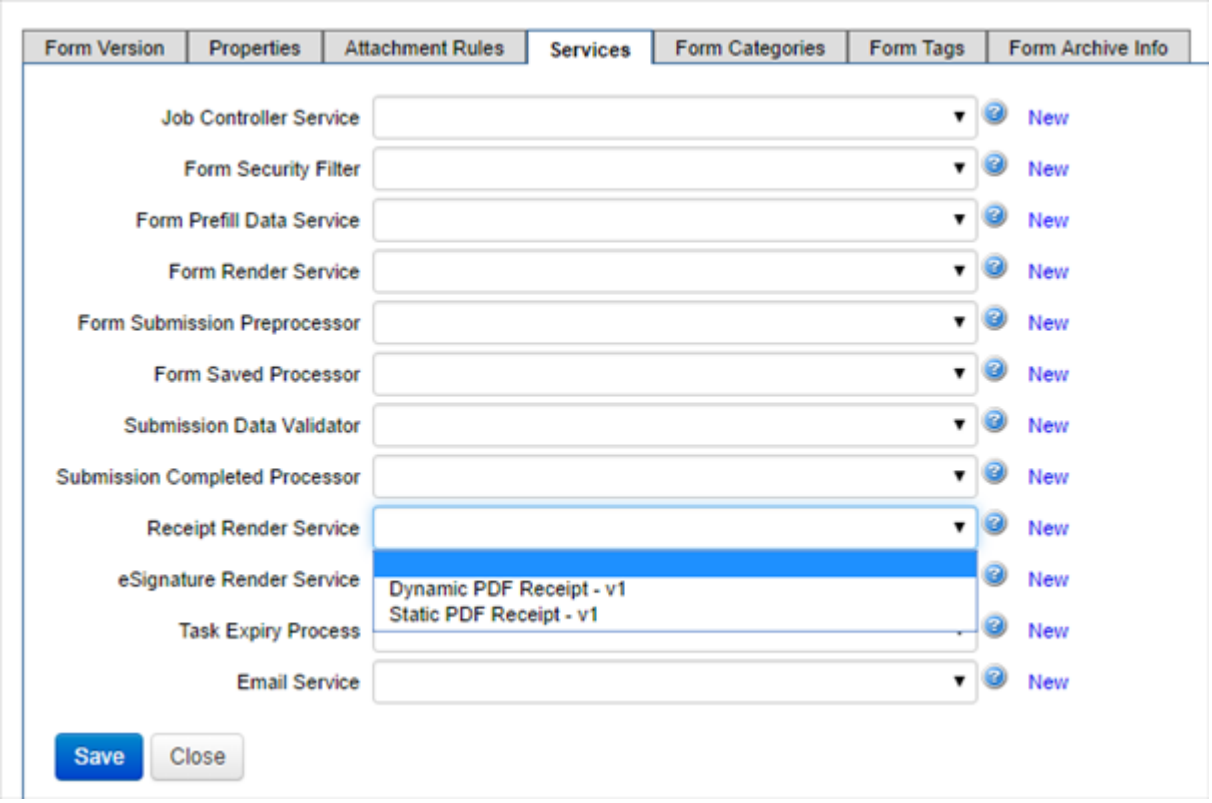
There are many different points in the life-cycle of a user interacting with a form, between the time that the user first clicks on a link to open the form, and the time that the transaction is

complete. We won't go into all the details, since there are over 20 different life-cycle events, but some of the key events are shown in the diagram below.



Some of these points are handled by built-in capabilities within the Temenos Journey Manager platform, but others require integration with third-party or organization systems.

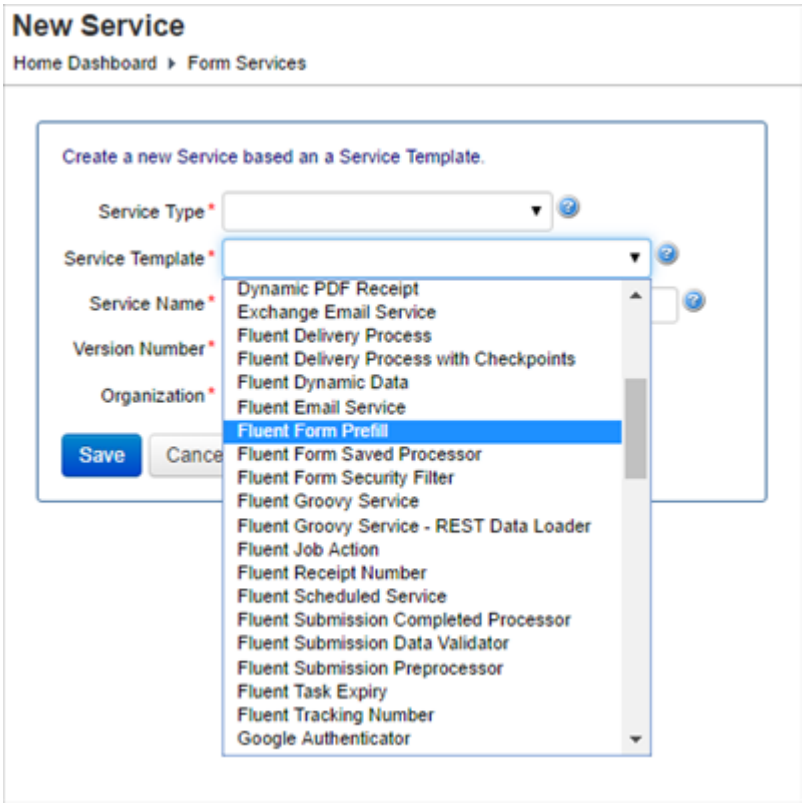
Manager provides extension points at many of these life-cycle events to allow integration between Manager and these other systems. The example of these extension points is shown below.



Each of the drop-down lists specifies a service that may be invoked at that particular point in the transaction life-cycle. You may select an existing extension point (as is shown in the Receipt Render Service in the screenshot), or you may create a new extension point by clicking the "New" button.

In order to build a new extension point, you author the extension point in a language called Groovy. Groovy is a simple yet powerful scripting environment that most developers find intuitive and productive.

When you create a new service, you are not simply presented with an empty screen and begin typing. Over the years, we have collected a series of templates from real projects that can be used as starting points for development of different types of services. A subset of the available templates is shown in the screenshot below:



The template will create a starter script which allows a programmer to simply "fill in the blanks" for their particular service.

Calling External Services and API

Once you start building a service, you will often need to invoke external services to actually perform the work. These services will usually be exposed as either SOAP or REST web services. These services may already exist, or they may be built as part of the Manager project. They can

be built in a number of different ways, including hand-built using coding languages, or the use of specialized integration system or an Enterprise Service Bus.

Libraries for calling these external services are built into the Temenos Journey Manager platform as shown in the example below.

Examples

The example below performs REST request and returns a JSON object

```
import com.avoka.tm.http.*
import com.avoka.tm.util.*

Map params = [:]
params.id = request.getParameter('customerId')
params.account = request.getParameter('accountId')

String username = svcDef.paramsMap.username
String password = svcDef.paramsMap.password

// execute GET request and return a HttpResponse object
HttpResponse response = new GetRequest('https://service.mycorp.com/secure/rest/accounts/')
      .setParams(params)
      .setBasicAuth(username, password)
      .execute()

if (response.isStatusOK()) {
    // get Path object from the response
    Path path = response.getPathContent()
    ...

} else if (response.isStatusNotFound()) {
    // object not found
    ...

} else {
    throw new RuntimeException(response.statusLine)
}
```

This illustrates how easy it is to invoke these external services. Of course, more sophisticated services can be developed.

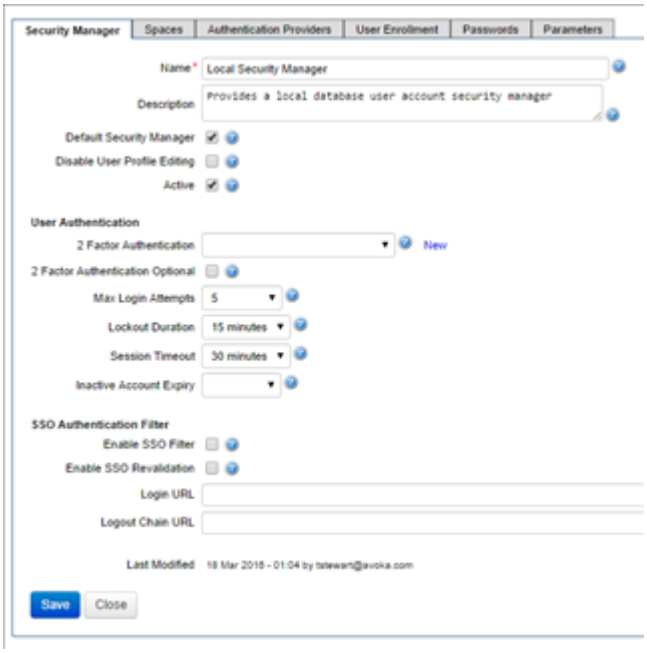
It is often also necessary to invoke Transact Manager itself in order to get more information from the system, or to update the transaction. This is also available through a set of comprehensive Transact API's.

Web Site Integration and Single Signon

It is almost always required to integrate a form into an existing web site. Usually this is achieve through a simple link or button. In the past, more complex integrations were performed, such as embedding forms within existing html pages, although in modern sites this is rare, largely due to the needs of smaller screens and mobile devices.

It is also sometimes necessary to provide single-sign-on integration with existing systems. This is provided by an extensible single-sign-on mechanism that is part of the security management

sub-system within Transact. SAML, oAuth and two-factor authentication are supported. A screen-shot is shown below:



Form Level Integration

Sometimes it is useful to perform integration directly between a form and another service. This can be performed in two ways:

- Directly between the form and the service. This can be implemented directly using JavaScript in the usual way. Generally the integration will be packaged up into re-usable component that can be simply dropped into any form that needs it.
- Using Transact Manager as a proxy service. This mechanism is known as Transact Dynamic Data services. The form uses a standard mechanism for passing data in and out, which both programmers and non-programmers can easily use and configure. This invokes a service on Transact Manager (written in Groovy). The Groovy service has all the power of the Groovy programming language and supplied libraries, and so arbitrary complexity can be simplified to ensure that the service exposed to the form is easy to use. There are several advantages to implementing the system this way, including ease of use, centralized configuration, integration with Transact licensing audit logs, and more. This is the technique usually used.

Next, learn about the five Temenos Journey Manager platform integration areas.

Integrate Manager with SharePoint

Journey Manager allows you to integrate with SharePoint[1], using a Java integration component, that supports the following SharePoint interfaces:

- `Copy.copyIntoItems`
- `Copy.getItems`
- `Lists.addAttachment`
- `Lists.getListItems`
- `Lists.getListCollection`
- `Lists.updateListItems`
- `Lists.checkInFile`
- `UserGroup.getUserInfo`
- `UserProfileService.getUserProfileByName`
- `Webs.getWeb`
- `Webs.getWebCollection`

Security of your applications is our priority, so, before using the SharePoint web services, you must ensure that a user, which you use to authenticate the connection, has the following permission: Use Remote Interfaces - Use SOAP, Web DAV, the Client Object Model or SharePoint Designer interfaces to access the Web site .

To call methods of SharePoint interface, you need to establish a SharePoint Connection. This can be done with a Groovy service. The `com.avoka.component.sharepoint.SPClient` class uses a Microsoft SharePoint Client class with web service factory methods.

The Groovy script example below illustrates using the `SPClient` object to create a `List SharePoint` web service object, which is then used to look up a collection, such as a `List`. In this example, configuration Service Parameters are accessed via the `params` map object .

[1]SharePoint is a web-based collaborative platform that integrates with Microsoft Office.

```
import com.avoka.component.sharepoint.SPClient
import com.avoka.component.sharepoint.service.List

// Create a SharePoint client using the service params map with the configured connection
details
def spClient = new SPClient(params.username, params.password, params.url, params.domain)

// Create a List web service
def listService = spClient.getLists()

// Get a List collection map
def collectionMap = listService.getListCollection(params.siteContext)
```

For more information, see API documentation.

Next, learn about the form dashboard.

Five Temenos Journey Manager Integration Areas

The Temenos Journey Manager comes with the following five integration areas:

1. Delivery
2. Form discovery
3. Pre-Population to accelerate data entry
4. Streamlining of data entry
5. Execution

Let's take a look at each integration in more detail.

Delivery

Delivery integration is extremely important to every organization. Unless you deliver the data collected to the organization in some way, you are ultimately not gaining any value. This area is listed first, because it is almost always implemented, whereas many of the other integration areas are optional.

There are generally three types of artifacts that are delivered:

- **Receipt.** A human-readable version of the form, usually in the form of a PDF document. This is known as a receipt, and can be generated in several different styles.
- **Data.** A computer-readable version of the data, in the form of an XML document. This is suitable for integrating into other systems.
- **Attachments.** Any supporting documents, known as attachments. These may be photographs, scanned documents, or electronic documents of various types.

The Temenos Journey Manager platform has a large number of different ways in which deliveries can be performed, ranging from simple to sophisticated.

Delivery Type	Description	Examples
Email	The PDF, XML and attachments are attached to an email, which is then emailed to an individual or group email box for processing. The person/people monitoring this email account process the PDF, XML and/or attachments manually. This approach is usually the simplest option for low	There is really not very much to do, other than making sure Manager has access to a mail server, and identi-

	volume and/or low value transactions, but it can also be used very successfully as a short term solution while more sophisticated integrations are being built.	fying the correct email recipients.
Fax Gateway	If your organization already has a fax gateway, then the inbound faxes will generally be delivered as attachments to an email message, or as files in a folder. Manager can easily be configured to do the same.	Temenos have previously integrated with the RightFax fax gateway.
Web Services (SOAP)	Managerhas a SOAP API that lets external systems retrieve submissions (including their PDF, XML and attachments). This API can be called by any client that is SOAP-compliant. This approach has the advantage that no firewall holes are required since this is a pull invocation.	Several of our customers have used the SOAP API in order to retrieve submissions.
TIA	The Temenos Journey Manager platform includes a pre-built (but extensible) application known as the Integration Agent. This Java-based application uses the SOAP API to retrieve submissions from Manager, and deliver these into external systems. A number of standard integration end-points are provided, such as folder drop, or customers can customize TIA to perform any type of integration that is possible from Java.	The Australian Dept of Industry uses TIA to deliver submissions into a number of different agencies in a multi-tenanted environment. This provides a level of separation between the Dept of Industry who manage and administer the system, and the agencies who only receive submissions destined for them.
ESB / Mulesoft	MuleSoft is an Enterprise Service Bus. Temenos has developed a Mule connector which uses the SOAP API to retrieve submissions from Manager. Once a submission has been retrieved, it can be	

	easily integrated with any of the systems that Mulesoft supports, which are listed here: https://www.mulesoft.org/connectors Mulesoft also has a sophisticated data mapping tool for mapping data from one form to another.	
REST API	Manager has a REST API that that lets external systems retrieve submissions (including their PDF, XML and attachments). This is similar to the SOAP API, but is a more modern protocol, easier to use, and supports high availability architectures.	
TMRDC	The Manager REST Deliver Client is a very simple and self-contained application built in the Go language. It uses the REST API to retrieve submissions from a Manager server. The default implementation is to store these submissions in a folder/directory. However, source code is provided, and can be customized to perform other types of integrations.	
Groovy	Manager allows delivery integrations (and other types of integrations) using the Groovy scripting language. Groovy is a simple yet powerful scripting language that has been built into Manager. Groovy gives you access to many of the internal Java API's provided by Manager, and also has capabilities for calling third party services and systems, including REST and Web Service API's, so it is a powerful tool for building sophisticated integrations. Groovy is often used to "glue" Manager to another system, rather than being used as a programming language. A full set of the available Manager API's can be provided upon request if desired.	Examples!!! They abound.
Java	For even more sophisticated integrations, Manager allows Java modules to be developed and uploaded into Manager, where the methods they	

	contain can be invoked at defined extension points. This provides access to the full power of Java, plus all of the Manager Java API's.

In addition to these general purpose delivery mechanisms, several specific integrations have been developed for particular end-points.

These have been integrated into Manager as pre-built integrations, and they include:

Delivery End-point	Description
Microsoft SharePoint	This integration allows submissions to be delivered directly into a Share-point instance.
SalesForce	This integration allows for: 1. OAuth 2 authentication against SalesForce identities. 2. Delivery of submission information directly into SalesForce for mar-shalling into destination objects. 3. API level form assignment to known and anonymous users directly from SalesForce, encapsulating APIs on Manager. 4. UI objects to present a categorized list of available forms directly inside SalesForce, for either opening immediately or pushing to SalesForce contacts. These may be automatically pre-filled with data from SalesForce objects. 5. UI objects to review the active and completed forms for a particular object within SalesForce.
Marketo	Creates or updates the visitor's Marketo LeadRecord using the submitted form data.
Adobe LiveCycle	Manager can deliver submissions directly into Adobe LiveCycle Process Management. Process Management is a BPM system that allows back-office business processes to be easily developed.

Finally, Temenos staff and our customers have used some of the above capabilities to integrate with a large number of off-the-shelf and custom systems. Some of these include:

- PEGA (BPM system)
- SAP (ERP system)
- Microsoft Dynamics (CRM system)

- FTP/SFTP (file transfer protocols)
- Shift (a dealer management system)
- Carflow (a dealer management system)
- Huddle (a document management system)
- K2 (a BPM system)
- IBM Lombardi (a BPM system)
- Telco systems (several bespoke systems at a large Telco)

The Delivery Service extension point with Manager provides a number of built-in added value services, including:

- **Retry capabilities.** If a delivery fails, it can be automatically or manually retried. This can be helpful for cases when a delivery end-point is temporarily unavailable.
- **Delivery checkpoints.** If a delivery consists of multiple steps (for example, send an email, then update a database, and then store a file in a directory), then each part of this delivery can be set up as an individual checkpoint. If one of the steps fails, but a previous step has completed successfully, then when the delivery is retried, it will start on the first incomplete checkpoint, rather than from the beginning. This would ensure that, for example, only one email ever gets sent.
- **Alerts.** The delivery subsystem provides administrative alerts to notify system administrators when a delivery has failed so that the underlying cause can be corrected.

Form Discovery

Every unique form in Manager has an associated URL link that you can place on your site. Any user who clicks on the links will launch the form. You can also use the form URL in any other way such as in an email. You can also link to the form from any type of "form finder" technology, or a wizard that assists a customer to find the appropriate forms. You can also embed links to forms in outbound communications such as email or social networks.

Another way for customers to complete a form is to send a customer a link to a form that has already been partially completed on their behalf. Manager includes a Task API, that allows you to create a partially completed form, generate a custom URL, and then pass that URL to the end user. This can be used to "push" forms from CRM or other systems.

An older approach is to embed the Temenos Journey Manager platform form directly inside the organization's web site. This approach is rarely used, mostly because of the advent of smartphones, which lack the screen real estate to display both the existing site and the form in the same screen. There are also other issues with this approach, including the importance of removing distractions and navigation links from the task-oriented process of completing the

transaction, as well as the difficulty of effectively designing responsive user interfaces in two separate web pages that effectively work together.

3 Pre-Population - Accelerating Data Entry

Manager provides many different techniques for pre-filling forms. One of the simplest ways to implement this is to pass pre-population data as URL parameters in the URL that links to the form. These parameters can be customized on a per user basis in your existing web site if required. Another way is to retrieve pre-population data from data stored in cookies.

Alternately, Manager has many extension points where calls can be made to external systems, and the results of these calls can be injected into the form's data before it is presented to the user. The sequence diagram above highlights the key extension points. The extension points can be implemented as Java or Groovy calls to external systems. These extension points also have access to many of the internal Manager API's.

Manager implements a form receipt-number generation service, to ensure that each form has a unique code. This service can be configured to auto-generate different styles of receipt number, or it can be extended to call external systems to generate the receipt number. This allows the form receipt number to match a reference number in an existing system.

Manager implements several different techniques for passing identity from an existing system to Manager, including SSO/SAML, LDAP integration, and others. If the user's identity has been established, then calls can be made to back-end system using this identity to retrieve user-specific data, which can then be injected into the form before being presented to the user.

Streamlining Data Entry

Once the user is actually within the form, we want to make their experience as streamlined as possible. One way we can do this is by designing and creating engaging, simple to use, guided experiences using Composer.

We can also streamline their experience by providing contextual data within the form that:

- Accelerates their data entry
- Validates their data entry
- Provides information to them that helps them to make decisions

Usually this type of integration is provided by custom user interface widgets within the form itself. There are generally two different mechanisms that these widgets use to communicate to external systems.

Mechanism	Explanation
Direct	The widget communicates directly to the third party service, usually (but not always) using JavaScript, REST and JSON. This is most useful for cloud-hosted third-party services that don't require usage keys, identity verification, or license/billing, and are not subject to security concerns.
Dynamic Data Service	A Composer widget communicates with a standardized dynamic data service within Manager. The dynamic data service protocol is predefined, and uses JSON data as the payload. The dynamic data service then communicates to the desired back-end service to verify or retrieve data, perform a calculation, or verify data. The results are then passed back to the widget. Dynamic data services are useful for several scenarios: • Licensing. Some external services require the user to specify a usage key of some sort. • Billing reconciliation. • Security (secure channel to backend services) • Data transformation or consolidation, protocol translation. Composer includes several pre-built widgets that support Dynamic Data calls, and custom widgets can also be created.
Hybrid	Occasionally it is useful to build widgets that use a combination of direct access and dynamic data services. For example, an initial call may use Dynamic Data to establish credentials with a third-party, but then use direct access to the third party service for greatest efficiency.

Some examples of pre-built services that communicate with third-party services include:

- Experian, which provides postal address lookup services.
- GreenID which provides identify verification services.
- Manager Reference Data, which allows data to be stored in Manager, and displayed in a variety of different ways.

Many our customers have used the generic Dynamic Data-aware widgets to expose their back-end web services, REST services, databases and bespoke systems in a variety of ways within the form.

Execution

Execution is largely about adding additional value to the overall transaction, after the basic data completion step has been completed. Some of these integrations are part of the process of completing the transaction, and are visible steps to the end users, whereas other types of integrations occur behind-the-scenes.

The following examples list some of the types of integrations that are possible at execution time with Transact.

Integration	Description
Electronic Signatures	Manager can integrate with Electronic Signature providers in order to provide an improved level of signature compliance. (Note that Manager provides several types of electronic signature capability directly in our system, without requiring third-party services.) Manager has already been integrated with DocuSign and Silanis, but the same principles can be extended to similar electronic signature providers. Electronic signatures are usually implemented in one of two ways: 1. **Immediate signature.** A receipt (or document of record) is created, and submitted to the third-party signature provider, along with signatory meta-data. The signature provider processes this request, prepares the document for signing, and returns the signatory id to Manager. It then redirects the user to the signature provider web site, where the document will be presented for signature. The signature providers may also request additional identification information. Once the user has signed the document, the signature provider will redirect back to the Manager web site, so that the transaction can be completed, delivered, etc. 2. **Signature by email.** A receipt is submitted to the signature provider, along with signatory meta-data.The signature provider will prepare the document for signing, and then send an email to the signer for processing. Manager will notify the applicant to check their email address. The user will click on the link in the email, and complete the signature process in the signature provider's web site. Manager will be notified of the signature completion, and will complete the transaction, including delivery etc.

	The immediate signature approach is a more seamless experience for the end user. However, the signature by email approach can have a higher level of identity for the signer, because it implicitly validates their email address. It is also generally the only way in which a document can be signed by multiple signers - this is usually supported by signature providers in either parallel or sequential flows.
Payment processing	Many transactions require an associated payment. The application form will calculate the fee, and also specify what payment methods are available. When the form has been submitted, Manager will redirect the user to a payment provider's website, along with meta-data about the payment, such as the amount. This site is known as a Payment Gateway. The Payment Gateway will process the payment, and then redirect back to Transact, passing payment meta-data. Manager will inform the user of the outcome, as well as store the payment-meta-data to allow for reconciliation reports to be run. never stores or processes credit card or banking information directly. This is a specialist service that requires specific levels of security and privacy compliance, and is always performed by a specialist payment providers. Temenos Journey Manager platformhas built-in integrations with several providers, and other providers can be implemented as required.
Additional Verification	Additional Verification of the user's information for purposes such as credit scores or identity verification can be implemented within Manager itself. These can also be implemented post-delivery in the organization's back-end processes, but performing them within Manager provides the ability for the end-user to be informed of the result while they are still within the transaction, and therefore have an opportunity to retry the application and correct the information.
Review and Approval	Manager includes a simple but powerful review and approval capability known as Collaboration Jobs. Transactions that have been completed by end users can be progressed through several review/approval steps, by different individuals or groups. Steps can also be designed to retrieve information, perform processing, or make decisions. Review and approval is often implemented in larger and more sophisticated BPM or Workflow systems, but the Review and Approval capability in Manager provides a number of benefits: • There is no need to purchase or maintain a BPM or Workflow system. • Review and Approval is very simple to configure, and very easy to use for the reviewers.

Document Generation	Manager produces a receipt automatically, in PDF format, providing a document of record for the customer and for the organization. Manager can produce a document of record using: • Composer-generated receipt templates (automatically) • Hand-created Adobe LIveCycle Designer receipt • Hand-created Adobe Acroform receipt (also known as pixel-perfect receipt). However, in some cases, the documents that need to be created are more sophisticated or different to what can be achieved in Manager natively. There are a number of systems known as Customer Communication Management systems which are designed to produce customer communication such as offers, receipts, contracts and information packs. These systems can be used by Manager to generate the document of record.

The Economics of Integration

It is very important to remember the consider why we do integration - we don't do integration just for the sake of it, we do integration order to provide specific value to the organization. Integration can often be the most time-consuming and costly part of any implementation. It's very easy to fall into the trap of doing integration because it's possible or seems obvious, but it is wise to always step back and consider what value integration is providing to the overall business objectives relative the cost of implementation and future maintenance.

Why do we do integration?

Usually we do integration in order to achieve operational efficiencies. Integration usually saves humans from having to perform manual tasks, which in turn reduces costs. These efficiencies may be realized by the end-user (eg save typing and frustration through pre-population) or by the organization itself (save money and reduce errors by eliminating

Assess the Value of Integration

A survey performed by the Forrester group in 2015 asked "What initiatives are likely to be your organization's top business priorities over the next 12 months?"

The top three responses were:

- Grow revenues (77%)
- Improve the experience of our customers (74%)
- Reduce costs (61%)

It is interesting that improving user experience is regarded as significantly more important than reducing costs. And that growing revenues is the most important criterion in the entire survey.

- The Temenos Journey Manager platform is all about improving customer experience, and growing revenues (although it also leads to improving efficiency).
- Integration is all about improving operational efficiencies, and reducing costs.

It is important to keep in mind that in many cases, the primary goal of implementing the Temenos Journey Manager platform is improving user experience, and growing revenue. You should therefore consider very carefully the value of doing integration compared to the cost. It may be that you can achieve a large proportion of the value of the Temenos Journey Manager platform without doing any integration.

We're not saying that you shouldn't do integration, just that you should be aware of the costs and the value.

Consider Deferring Integration

Another factor to consider is time to market. Manager can be implemented very quickly, and realize important benefits such as improved user experience and improved conversion rates. However, integration, because it touches back-office systems, needs to be very carefully developed and tested, and this can often take a significant amount of time.

One common approach is to implement Manager with little or simple integration, and realize the primary benefits as early as possible. More sophisticated integration can then be performed at a slower and more careful pace - and because Manager delivers the data in XML format, it's easy to add integration later. Integration can be prioritized for the most important transactions first, with lower value or lower volume transactions deferred till later.

Some of our most successful implementations had started with minimal integration at the beginning, but over time the integration has been phased in separately.

Consider Leveraging Existing Paper/image-based Processes

Ideally the goal is to completely replace paper/images with all-digital transactions. However, in reality we rarely achieve 100% success - there are almost always going to be some customers who insist on hand-writing on paper. This means that you almost always needs to maintain your existing back-office processes which currently handle paper or images of paper.

Since you're going to need to maintain these back-office processes anyway, one useful approach is to augment or plug into these back-office processes using digital artefacts, rather than trying to create a completely all-digital back-office process. In other words, deliver a digital

document (generally a PDF) into your existing paper- or image-based systems. If you do this, integration is usually very simple and quick.

Hidden Costs of Process Re-engineering

Process re-engineering is costly. This is not just about technology, but about people, skills, training, management, and related costs. Overall change management is often overlooked, and can be hugely costly. You don't want to introduce a new digital channel, and at the same time, completely re-engineer your back-office processes. Instead, one option is to simply plug digital artefacts into existing back-office systems. You get the benefits of a new digital customer-experience, but you don't have to retrain anyone, build new processes and procedures and manuals, hire new staff with different skills, or put new managers in place. You have the opportunity to re-engineer and optimize your existing back office processes at a later time.

Our strong recommendation is not to combine process re-engineering with a (user-experience improvement) project, otherwise there is a possibility that both projects will fail. Process re-engineering and change management are hard and expensive, and shouldn't be embarked upon lightly.

Next, learn about Temenos Journey Manager platform development.

Google Analytics Virtual Page Integration

The Temenos Journey Manager allows you to add page tracking using Google Analytics (GA), so each page will fire a page view event with a title and URL, which you can analyze to improve user onboarding. You can easily add GTM[1] to a Maestro form using Journey Manager and Journey Maestro tools. The simplified steps are outlined below. For detailed instructions, including Google Tag Manager configuration, see how to add Google Tag Manager to a Maestro form.

To add GA to your web app form:

1. Add GA code to the head of the page in Manager.
2. Create a Business Rule to call the GA method on page change in Maestro. You need to configure the Maestro form business rules with the following JavaScripts:

[1]Google Tag Manager (GTM) is a tag management system (TMS) that allows you to quickly and easily update tracking codes and related code fragments collectively known as tags on your website or mobile app.

- On Form Load:

```
// Init the global variable used to store the form story for GTM
dataLayer.push({event: "Form started"});

window.addPageToGTMTag = function() {
  var currentPage = Form.getCurrentPageId();
  dataLayer.push({ event: currentPage});
}

// Function to add Google Tag Manager leaving page event
window.addGTMLeavingPageTag = function() {
  // Push this event to GTM
  var currentPage = "leaving_" + Form.getCurrentPageId();
  dataLayer.push({ event: currentPage});
}


// Function to add the Google Tag Manager iFrame to the form
function addGTMTag() {
  $('body').prepend('<noscript></noscript>');
  $('body noscript').first().prepend('<iframe src-
c="https://www.googletagmanager.com/ns.html?id=GTM-PTNXXX" height="0" width-
h="0" style="display:none;visibility:hidden"></iframe>');
}
addGTMTag();
```

- On Page Load:

```
// Push this event to GTM
window.addPageToGTMTag();
```

- On Continue (Next Page Action):

```
// Push leave event to GTM
window.addGTMLeavingPageTag();
```

3. Use Google Analytics tool to view reports on user form usage.

Next, learn about Temenos Journey Manager platform development.

Cloud-hosted Temenos Journey Manager VDC & Enterprise Editions

When running the Temenos Journey Manager platform, several editions are available which are suitable for organizations with the highest requirements for privacy and security. These editions are the Private Cloud Enterprise Edition or Virtual Data Centre Edition (VDC). These editions are cloud based, on either AWS or Azure, to provide maximum agility and include multiple servers to ensure high availability.

The Temenos Journey Manager platform by design is a short term storage solution that does not retain **Personally Identifiable Information**[1] longer than a customer's data retention policies allow.

Use the matrix to compare the VDC and Enterprise Editions, depending on your requirements.

Features	VDC Edition	Enterprise Edition
Redundancy & Availability		
High Availability[2] Application Servers (Multi-AZ) (Traffic load balanced across application servers in separate data centers) TJM Application Server Nodes (Production) 2+2 = 2x /manager (admin) portal nodes + 2x /workspace portal nodes HA = High Availability (nodes deployed into separate AWS AZs) High Availability RDS Database Server Replicated across separate data centers (Multi-AZ)	4+ (HA) (2+2)	2 (HA)

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.
[2]High availability (HA) is a characteristic of a system which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.

Non-production environments (Test, UAT, Stag, etc) included	3	2
Production - Service Level Availability Target (monthly)	99.9%	99.5%
Number of countries supported (i.e. tenants) (TJM organizations cannot be used to segregate country solutions, or deploy additional instances)	1	1
Extra countries/tenants Available (POA)		N/A
RDBMS	Aurora	Aurora
Backup & Recovery		
Database Backup Frequency	Continuous	Continuous
Backup Frequency (RPO[1])	~ 5 minutes	~ 5 minutes
Database Recovery Time (RTO[2])	4 hours	8 hours
Security		
Customer Virtual Private Cloud (solution installed in customer-specific AWS VPC with private subnets)		
Data Sovereignty - All data resides in same geographic region. Regions are US (Oregon), EU (Ireland), Asia Pacific (Sydney), and other AWS Regions available as required		
Strong AES 256-bit data encryption for data in transit and at rest		
Separate security subnets for production and non-production environments		
IP White-listing		

[1]Recovery point objective (RPO) is the age of files that must be recovered from backup storage for normal operations to resume if a computer, system, or network goes down as a result of a hardware, program, or communications failure.

[2]Recovery Time Objective (RTO) is the targeted duration of time and a service level within which a business process must be restored after a disaster (or disruption) in order to avoid unacceptable consequences associated with a break in business continuity.

Automatic updates for OS & security vulnerability patches		
Anti-virus scanning		
Dedicated Database Server in same Virtual Private Cloud		**X**
Separate administration and end user portal servers *separate URL endpoint for Manager and Portal		**X**
Encryption Data Key Management	Amazon KMS	Amazon KMS
Internal vulnerability scanning – Continuous (Qualys Cloud Agent)		**X**
External vulnerability scanning - Monthly (Qualys WAS scan)		**X**
Third-party penetration testing - Annually	Available (POA)	Available (POA)
Akamai - for WAF and CDN	Available (POA)	Available (POA)
Network Connectivity Features		
AWS VPC Peering (Connect 2 AWS accounts; no service cost - AWS Doc)	Available	Available
AWS Direct Connect (Connect AWS to on premise; service cost not included - AWS Doc)	Available (POA)	Available (POA)
AWS VPN Site to Site	Available (POA)	Available (POA)
Advanced Security Features		
Web Application Firewall (WAF)	Apache ModSecurity (fixed customized rules)	Apache ModSecurity (fixed customized rules)
Network Protection (Deep packet inspection,	AWS Shield	AWS Shield

Flood protection, DoS, DDoS and port scan blocking, Country blocking)	Standard (no DPI and Country blocking)	Standard (no DPI and Country blocking)
Outbound Proxy (whitelist)		**X**
Security Extras		
AWS Dedicated Instances (dedicated hardware for application servers - AWS Doc)	Available (POA)	**X**
Additional IPS / UTM (Sophos Unified Threat Management) layer (AWS Doc)	Available (POA)	**X**
Platform Issue - Response Times and SLAs – Production		
Refer to your contract for all SLAs for Production Response Times and Production Resolution Times. These may vary between clients due to different support agreements negotiated between Temenos and the client.		

If you have any further questions or you need more information including pricing, contact your Customer Success Manager.

Next, learn about the Temenos Journey Manager platform system requirements.

Test Specification

Journey Manager is a complex software system, consisting of many interconnected components, with a wide range of features that we add and improve on a regular basis. To make sure our software functions as expected, we spend significant time of developing efforts to test all features.

We use the following testing frameworks and best practices to achieve the required test coverage as per Temenos' guidelines:

Unit Tests

Performance Tests

Automated Tests

Manager is certified against Keycloak so you can set it up for various test cases that require single sign-on with identity and access management.

Next, learn about Journey Manager features.

Performance Specification

Journey Manager is a Web server designed to perform a high number of user transactions per second, which are successful user form accesses, to satisfy a wide range of customer requirements. For each major release, we run multiple performance tests using several standard scenarios. Then we capture performance metrics and analyze them to ensure each software release runs as expected. Metrics can vary depending on Manager version and a profile.

> INFO
>
> Journey Manager has a pool of database connections, which is usually set to 100, so if it gets exhausted, then the max wait time for the next available JDBC connection is typically around 30 seconds.

TODO: Describe how we set up test environments and collect test data. Add a disclaimer that these result numbers are a guideline only.

For detailed performance result, contact our Sales team.

Scenario 1

This scenario uses the following configuration:

- A Credit Card application consisting of 44 forms
- CDN set-up
- Test time 10 min
- ELB, JM, MySQL 5.6.34 and RDS Aurora 5.6.10a - db r3.large - All AU/Sydney region (VDC infrastructure)
- Amazon S3 submission storage - KMS
- Render, submission - no user wait / think time

Performance results for a background receipt (with CDN) test case are summarized in a table below.

Profile	Manager / DB ver-	Successful sessions	Failed sessions	Successful pages	Failed pa-	Successful hits	Failed hits	Total KB sent	Total KB received	Avg Response time,

	sio-n				g-e-s					sec (wit-h pag-e ele-men-ts)
100 con-cur-rent user-s, 100 max jdbc poo-l-m4.l-arge	5.0.-3 / Aur-ora	140-84	0	281-78	0	183-212	0	13-49-13	53-89-54-1	2.07 (2.1-1)
200 con-cur-rent user-s, 100 max jdbc poo-l-m4.l-arge	5.0.-3 / Aur-ora	912-9	58-4	185-79	5-8-4	122-531	5-8-4	90-26-2	36-13-67-5	4.53 (4.5-7)
200	5.0.-3 /	891-6	22-1	182-47	2-2-	120-888	2-2-	88-46-	35-68-	5.61 (5.6-

con-cur-rent user-s, 200 max jdbc poo-l-m4.l-arge	Aur-ora				1		1	2	02-9	5)
100 con-cur-rent user-s, 100 max jdbc pool-m4.l-arge	5.1 / Aur-ora	284-70	0	569-91	0	114-033	0	16-43-01	22-50-59-6	1.03 (1.0-4)
100 con-cur-rent user-s, 100 max jdbc pool-	5.1 / Aur-ora	289-73	0	580-49	0	116-201	0	16-72-61	22-94-60-0	2.04 (2.0-5)

m4.large 200 concurrent users, 200 max jdbc pool - m4.large	5.1 / Aurora	237-73	0	476-65	0	954-49	0	13-72-86	18-85-40-3	2.49 (2.5-0)	

Successful sessions per second for a background receipt (with CDN) test are summarized in a table below.

Profile	Manager version	0:-0-0:-0-0:-0-1:-0-0	0:-0-1:-0-0:-0-2:-0-0	0:-0-2:-0-0:-0-3:-0-0	0:-0-3:-0-0:-0-4:-0-0	0:-0-4:-0-0:-0-5:-0-0	0:-0-5:-0-0:-0-6:-0-0	0:-0-6:-0-0:-0-7:-0-0	0:-0-7:-0-0:-0-8:-0-0	0:-0-8:-0-0:-0-9:-0-0	0:-0-9:-0-0:-1-0:-0-0	Total	Per Hour
100 concurrent us-	5.-0.-3 / Aurora	2-1.-4	2-1.-8	2-5	2-4.-1	2-3.-7	2-2.-8	2-4.-3	2-3.-9	2-4	2-3.-7	2-3-.-5	8-4-6-0-0

ers, 100 max jdbc pool m4.large 200 concurrent users, 100 max jdbc pool m4.large 200 co-	5.0.3 / Aurora	14.9	17.9	16.3	11	65.7	16	17	17.5	17.7	17.3	15.2	547.20
	5.0.3 /	10.6	14.3	14.6	15.9	15.4	15.6	14.9	14.6	16.1	16.5	14.-	536-

nc-urr-ent us-er-s, 20-0 ma-x jdb-c po-ol-m4-.lar-ge	A-ur-or-a											9	4-0
10-0 co-nc-urr-ent us-er-s, 10-0 ma-x jdb-c po-ol -m4-.lar-	5.-1 / A-ur-or-a	4-4.-7	4-6.-9	4-7.-5	4-7.-7	4-7.-8	4-7.-5	4-8	4-8	4-8	4-8.-4	4-7-.-5	1-7-1-0-0-0

ge 20-0 concurrent users, 10-0 max jdbc pool - m4-.large 20-0 concurrent users, 20-0 max jdbc	5.-1 / Aurora	4-6	4-8.-2	4-8.-6	4-9	4-7.-9	4-8.-6	4-8.-5	4-8.-8	4-8.-9	4-8.-5	4-8-.-3	1-7-3-8-8-0
	5.-1 / Aurora	2-8.-9	3-7.-6	3-9.-1	4-0.-2	4-1	4-0.-8	4-1.-8	4-2.-3	4-2.-2	4-2.-5	3-9-.-6	1-4-2-5-6-0

pool - m4-.large										

The performance analysis shows that:

Scenario 2

This scenario uses the following configuration:

- Ramp-up user - test time 10 min
- Ramp-up users, from 10 to 100 users with increment of 5 users every 30 secs
- 4 renders and 1 submission - user think time - last render: 1 sec and submission 1 sec

Performance results for a background receipt (with CDN) test case are summarized in a table below.

Profile	TM version / DB	Successful sessions	Failed sessions	Successful pages	Failed pages	Successful hits	Failed hits	Total KBytes sent	Total KBytes received	Avg Response time, sec (with page elements)
100 max jdbc pool - m4.-	5.-1 / Aurora	9221	0	4644-0	0	1208-03	0	97-07-9	290-611-3	0.30 (0.3-2)

large

Successful sessions per second for a background receipt (with CDN) test are summarized in a table below.

Profile	TMver-sion	0:-0-0:-0-0:-0-1:-0-0	0:-0-1:-0-0:-0-2:-0-0	0:-0-2:-0-0:-0-3:-0-0	0:-0-3:-0-0:-0-4:-0-0	0:-0-4:-0-0:-0-5:-0-0	0:-0-5:-0-0:-0-6:-0-0	0:-0-6:-0-0:-0-7:-0-0	0:-0-7:-0-0:-0-8:-0-0	0:-0-8:-0-0:-0-9:-0-0	0:-0-9:-0-0:-1-0:-0-0	Total	PerHour
1-00 maxjdbcpool-m4.large	5.1/Aurora	4.77	8.63	12.1	15.3	16.9	18.5	19.1	19.7	19.4	19.4	15.4	5544 0

The performance analysis shows that:

Next, learn about [Journey Manager features](#).

Resource Limitations

Journey Manager, as any software system, has some limitations that you should understand to design a robust solution.

JDBC Connections

Manager supports ? open JDBC connections.

Java Threads

Manager supports ? Java threads running at any given .

Transactions

Manager supports ? transactions to be loaded and viewed in a Manager Dashboard at any given time.

Services

Manager supports ? services running at the same time.

Forms

Manager supports ? form to be open at the same time.

User Accounts

Manager doesn't impose any limits on the number of user accounts you can have. For more information, see User Accounts Overview.

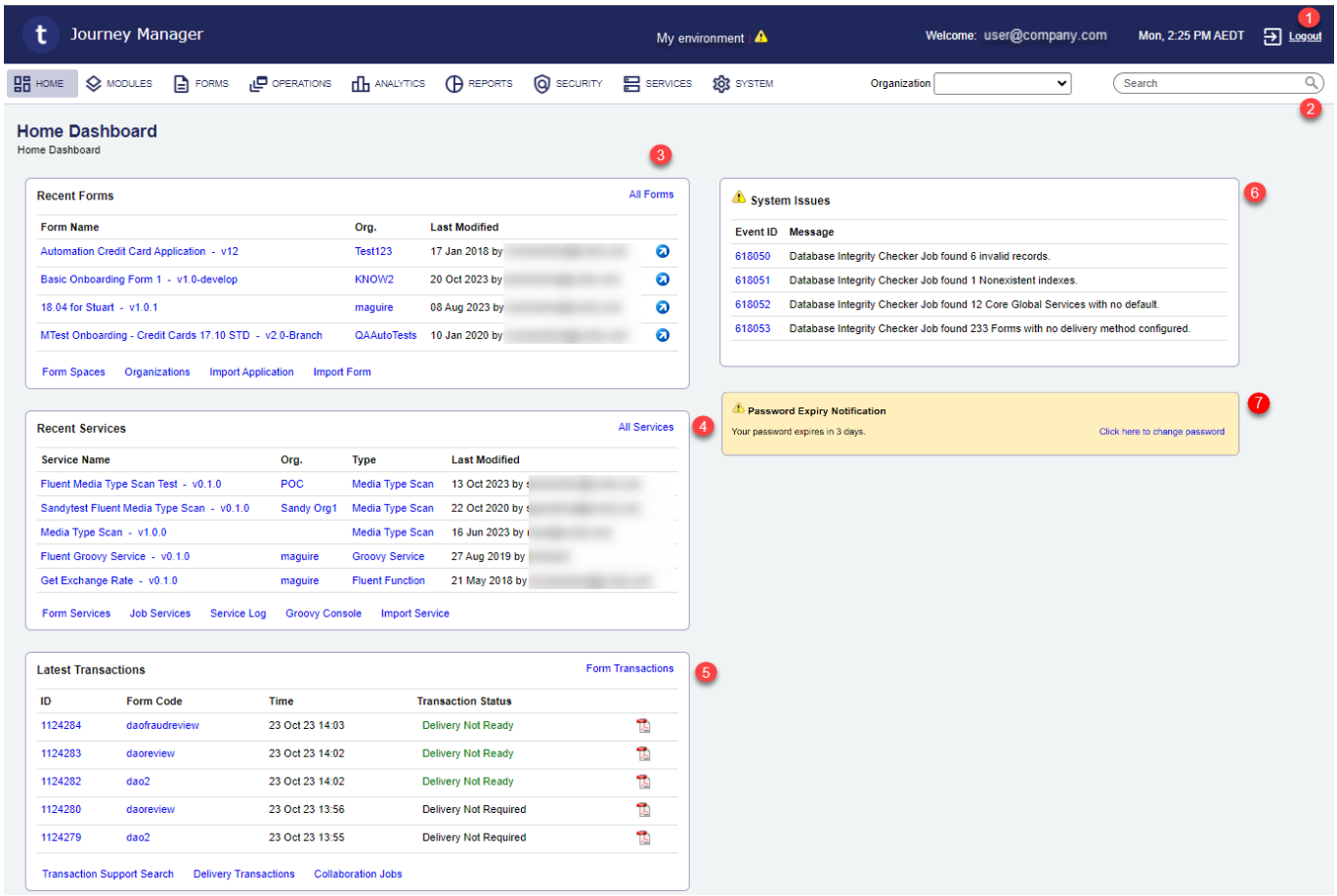
Next, learn about Journey Manager features.

Journey Manager Home Dashboard

Journey Manager provides you with the Home Dashboard where you can quickly access your recent forms, services, and transactions, as well as the current most important system issues. After you logged into Manager, the Home Dashboard is the first screen that is shown to you.

> NOTE
>
> The Manager Home Dashboard is often referred to as the Manager console.

To navigate to the Manager Home Dashboard, click Home.



The Home Dashboard consists of the following elements:

1. Screen Header
2. Menu Bar
3. Recent Forms Card
4. Recent Services Card
5. Latest Transactions Card

6. System Issues Card | 23.10 This feature was introduced in 23.10
7. Password Expiry Notification Card | 24.10 This feature was introduced in 24.10

NOTE

The Completed Transactions By Form pie chart and the Transaction Trend graph have been removed from the Home Dashboard | 22.10 This feature was removed in 22.10
You can find this information in Analyze Form Submissions and Analyze Transaction Trend correspondingly.

NOTE

You must have the certain permissions set by an administrator to see these cards. For more information, see Administrator's documentation.

Screen Header

The screen header displays some useful information, including:

- Product name: Journey Manager
- Environment name: My environment

  NOTE

  You can change the environment name in the Admin Environment Name property of the Transact Manager module.

- Warning sign if any system issues are present in Manager
- Name of a logged in user: user@company.com
- Current time of the server instance: Mn, 2:25 PM AEDT

The Screen Header provides you with the following actions:

- : log you out of Manager. We recommend logging out explicitly when you don't use the product. However, Manager will terminate all inactive user sessions after the specified time. You can change this time using the Session Timeout parameter of your security manager. For more information, see Configure a Security Manager.

Menu Bar

The Menu Bar provides you with a quick access to all functionality of the product. For more information on each menu option, see a corresponding topic in the Table of Content of our documentation.

Recent Forms Card

This card lists 5 recently viewed or modified forms. The forms are ordered by last modified date and time.

You can perform the following actions on this card:

- Click a form name to see its configuration in a form dashboard
- Click Org to view this organization configuration
- Click From Test Render to render this form in the default form space
- Click Form Spaces to view form spaces
- Click Organizations to view organizations
- Click Import Application to import an application package
- Click Import Form to import a form

Recent Services Card

This card lists 5 recently viewed and modified services. The services are ordered by last modified date and time.

You can perform the following actions on this card:

- Click a service name to see its service definition configuration
- Click Org to view this organization configuration
- Click Service Type to view services of this selected type
- Click Form Services to view form services
- Click Job Services to view job services
- Click Service Log to view Groovy service log
- Click Groovy Console to open the Groovy console
- Click Import Service to import a service

Latest Transactions Card

This card lists 5 latest transactions of users' submissions.

You can perform the following actions on this card:

- Click ID or Form Code to see the transaction details
- Click Transaction Status to view a transaction status
- Click View PDF Receipt to view and download a PDF receipt of a submitted form
- Click Transaction Support Search to view and search transactions
- Click Delivery Transactions to view transaction delivery
- Click Collaboration Jobs to view collaboration jobs
- Click Form Transactions to view form transactions

System Issues Card

This card lists 5 latest system events that are of importance to overall performance, stability and maintainability of Manager.

> NOTE
>
> You must be an administrator to see this card.

You can perform the following actions on this card:

- Click Event ID to see additional information about an event.

Password Expiry Notification Card

This card displays the Password Expiry Notification message when your password is about to expire, providing a 10 day notice, or when it has expired. The message looks like this: Your password expires in 2 days.

You can perform the following actions on this card:

- Click a link to change a password.

Next, learn about the form dashboard.

Journey Manager Modules

Journey Manager supports modular architecture, which allows you to deploy new software modules to add or enhance the existing functionality. A module is a WAR file that encapsulates this functionality. Some modules can be considered as products themselves, because of their rich functionality and the way they can be used, so you may refer to modules as products too, for example, the Maestro product.

Manager includes the following modules:

- Avoka Exchange: the catalog of pre-integrated FinTech[1] applications written for Maestro.
- Transact Insights: the analytics module that captures behavioral and completion analytics for supported applications.
- Transact Maestro: the tool for designing and building application forms that implement various customer onboarding journeys.
- Transact Manager: the server engine as well as the management dashboard.
- Transact Receipts: the background PDF receipts generator.
- Journey Operations: the Operations Portal. | 22.10 This feature was introduced in 22.10. | 24.10 This feature was removed in 24.10
- Manager API: the Journey Manager API (Fluent, REST, SOAP) endpoints that can be authenticated by using a security manager that is different to one configured in the Transact Manager module. | 24.10 This feature was introduced in 24.10

NOTE

Our product names have changed, but the modules will maintain their original names for backward compatibility. | 19.05 This feature was introduced in 19.05.

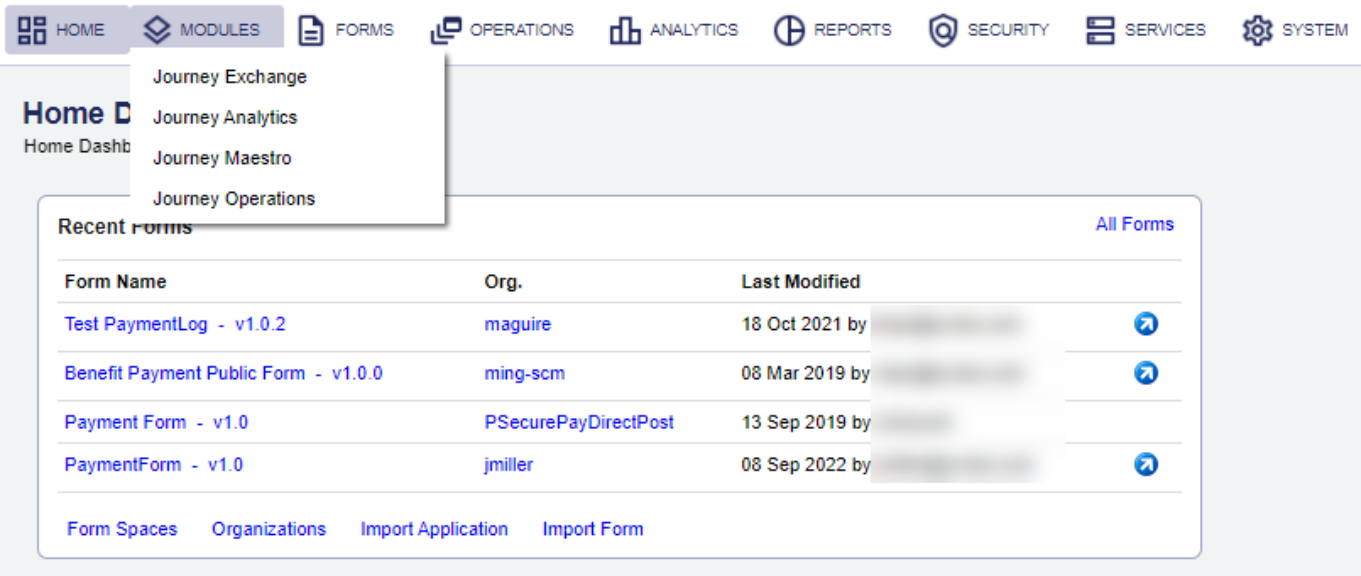
You choose which modules to install during the Journey Manager installation or upgrade. However, you should only install the modules you need, because each module incurs licensing fees and consumes both memory and processing power of a server.

Manager allows you to access the installed modules (products) by launching them in a separate browser's window.

---

[1]Financial Technology, also known as FinTech, is a new industry that uses new technology and innovation to improve activities in finance. It aims to compete with traditional financial methods in the delivery of financial services to customers by improving customer experience, reducing time-to-market, decreasing cost to mention a few.

To access a module, click Modules and select a product from the dropdown list.



If you can't see a module you need, check the following:

- The module has been installed to your Manager instance.
- A form space hosting this module has been assigned to your user account.
- A role with the correct permissions has been assigned to your user account.

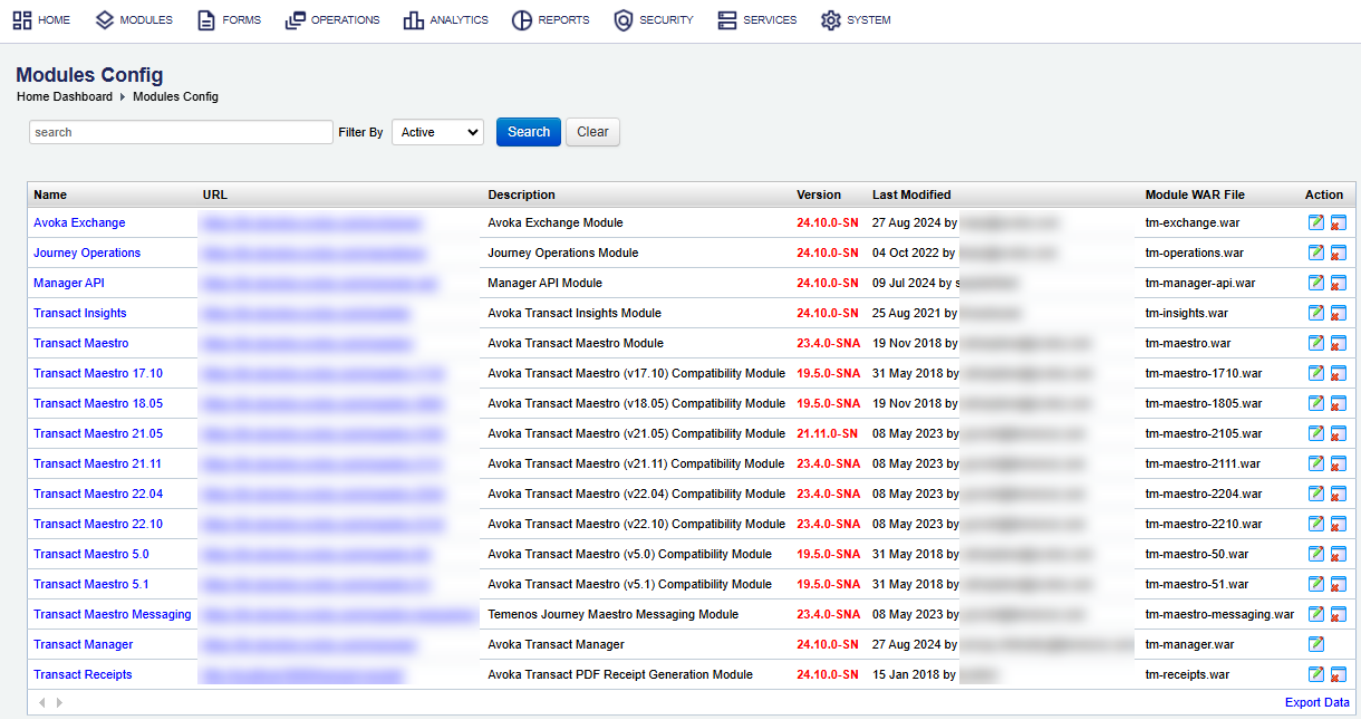
For example, if you don't see the Journey Operations module, you should try one of the following:

- Check with the Support Team the module has been installed.
- Ensure you have the Journey Operations form space assigned.
- Ensure you have the Journey Operations Staff role assigned.

For more information, see Provision a User Account.

Manager has a separate configuration UI allowing you to see all installed modules and update their configurations.

To view all installed modules and update their configurations, select System > Modules Config.



Next, learn how to configure modules.

Journey Manager Third-Party Libraries

Journey Manager is implemented in Java[1] and Groovy[2], which allows us to use a broad range of open-source[3] third-party libraries. We have carefully selected each library to make sure it's right for our purpose. We diligently update each library with new versions to ensure Manager runs as expected. If we get to know that a library contains a bug or a security vulnerability (CVE), we will update it as soon as possible, so each major release of Manager will include only updated libraries.

> NOTE
>
> Access to third-party Java[4] libraries by Fluent Groovy services is restricted via a security white list. For more information, see Fluent Security Configuration.

Licenses

Manager uses the third-party libraries according to their term and conditions outlined in the followings licenses:

- Apache License 2.0
- MIT License
- iText Commercial License
- NetBSD (2-line) License
- BSD 3-Clause License

---

[1]Java is a programming language that produces software for multiple platforms. When a programmer writes a Java application, the compiled code (known as bytecode) runs on most operating systems (OS), including Windows, Linux and Mac OS. Java derives much of its syntax from the C and C++ programming languages.

[2]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

[3]Open source software (OSS) is software with source code that anyone can inspect, modify, and enhance.

[4]Java is a programming language that produces software for multiple platforms. When a programmer writes a Java application, the compiled code (known as bytecode) runs on most operating systems (OS), including Windows, Linux and Mac OS. Java derives much of its syntax from the C and C++ programming languages.

- Microsoft JDBC Driver 4.0 For SQL Server License
- Oracle's dual-license JDBC Driver for MySQL
- PayPal, Inc. SDK License
- OTN License Agreement
- Gemalto Commercial License
- Mozilla Public License
- BSD 3-clause
- Open JDK

Java

Manager requires the following JDK[1] versions for each software release:

Java	21.11	22.04	22.10	23.04	23.10	24.04	24.10
Open JDK	11.0.1-1	11.0.1-1	17.0.4.-1	17.0.4.-1	17.0.4.-1	21.0.1+1-2	21.0.1+1-2

Being an application server, Manager relies on an Enterprise Edition (EE) Framework, which is used by various Java-based software, such as WildFly. You don't need to worry about this as long as you use compatible third-party libraries as listed below. However, if you run form applications on Journey Manager prior to 23.04.0 and wish to use Journey Manager 23.04.0+ or Journey SDK 23.04.0+, you must make them compatible with Jakarta EE 10. For more information, see Migrate to Jakarta EE.

Build Automation

Manager supports the following Ant and Maven versions for automating Journey SDK application builds.

Build tool	21.11	22.04	22.10	23.04	23.10	24.04	24.10
Mave-	N/A	3.6.1	3.6.1	3.6.1	3.6.1	3.6.1	3.6.1

[1]The Java Development Kit is a distribution of Java Technology by Oracle Corporation. It implements the Java Language Specification and the Java Virtual Machine Specification and provides the Standard Edition of the Java Application Programming Interface. For more information, see https://www.oracle.com/java/

n[1]							
Ant	1.10.1-1	1.10.1-1	1.10.1-1	1.10.1-1	1.10.1-1	1.10.1-1	1.10.1-1

1. Official support for Maven build automation was introduced in the 22.04 release. If your Journey SDK is from an earlier release, you must either use Ant or migrate to a newer SDK release that supports Maven.

NOTE

We recommend using Maven for build automation, if your Manager release supports it.

INFO

Journey SDK doesn't support Gradle for build automation.

For more information, see Setup Journey SDK.

Libraries

The main third-party library names and their versions are listed below for each Journey Manager major version.

NOTE

Journey SDK uses the same third-party libraries as Journey Manager.

Library Name / Java Package	21.11	22.0-4	22.1-0	23.0-4	23.1-0	24.0-4	24.1-0
Apache Cayenne org.apache.cayenne	4.1	4.1.1	4.1.1	4.1.1	4.1.1	4.1.1	4.1.1
Apache Click org.apache.click	2.4-RC8-TM	2.4-RC1-7-TM	2.4-RC1-8-TM	2.4-RC1-9-TM	2.4-Jakarta-RC1-9-TM	2.4-Jakarta-RC1-9-TM	2.4-Jakarta-RC1-9-TM
Apache Commons	1.3.1	1.3.1	1.3.1	1.3.1	1.3.1	1.3.1	1.3.1

Command Line Interface org.apache.commons.cli							
Apache Commons Bean Utils org.apache.-commons.beanutils	1.9.4	1.9.4	1.9.4	1.9.4	1.9.4	1.9.4	1.9.4
Apache Commons Codec org.apache.-commons.codec	1.14	1.14	1.14	1.15	1.15	1.15	1.15
Apache Commons Collections org.apache.-commons.collections	3.2.2	3.2.2	3.2.2	3.2.2	3.2.2	3.2.2	3.2.2
Apache Commons Collections4 org.apache.-commons.collection4	4.4	4.4	4.4	4.4	4.4	4.4	4.4
Apache Commons Compress org.apache.-commons.compress	1.21	1.21	1.21	1.21	1.21	1.26	1.26
Apache Commons	0.5	0.5	0.5	0.5	0.5	0.5	0.5

Discovery org.a-pache.-commons.discovery							
Apache Commons Email org.a-pache.-commons.email	1.5	1.5	1.5	1.5-jakart-a-tjm1	1.5-jakart-a-tjm1	1.5-jakart-a-tjm1	1.5-jakart-a-tjm1
Apache Commons FileUpload org.a-pache.-commons.fileupload	1.4	1.4	1.4	1.5-jakart-a-tjm1	1.5-jakart-a-tjm1	1.5-jakart-a-tjm1	1.5-jakart-a-tjm1
Apache Commons IO org.a-pache.commons.io	2.10	2.10.-0	2.10.-0	2.11.-0	2.11.-0	2.11.-0	2.11.-0
JSON for Java com.-fasterxml.jackson	2.11.3	2.12.-1	2.12.-1	2.14.-2	2.14.-2	2.15.-3	2.16.-1
Apache Commons Lang org.a-pache.commons.lang	2.6	2.6	2.6	2.6	2.6	2.6	2.6
Apache Commons Lang org.a-pache.-	3.10	3.11	3.11	3.12.-0	3.12.-0	3.14.-0	3.14.-0

commons.lang3							
Apache Commons Math org.apache.commons.math	1.2	1.2	1.2	1.2	1.2	1.2	1.2
Apache Commons Net org.apache.commons.net	3.7	3.8	3.8	3.9.0	3.9.0	3.9.0	3.9.0
Apache Commons Pool org.apache.commons.pool	1.6	1.6	1.6	1.6	1.6	1.6	1.6
Apache CXF org.apache.cxf	3.3.7	3.3.10	3.3.10	4.0.0	4.0.0	4.0.4	4.0.4
Apache FOP org.apache.fop	2.5	2.6	2.6	2.8	2.8	2.9	2.9
Apache HTTP Components org.apache.httpcomponents	4.5.13	4.5.13	4.5.13	4.5.14	4.5.14	4.5.14	4.5.14
Apache HTTP Components org.apache.httpcom-	4.4.14	4.4.15	4.4.15	4.4.15	4.4.16	4.4.16	4.4.16

ponents.httpcore							
Apache James Mime4J org.apache.james.mime4j	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Apache Neethi org.apache.neethi	3.1.1	3.1.1	3.1.1	3.1.1	3.1.1	3.1.1	3.1.1
Apache POI org.apache.poi	4.1.2	5.0	5.0	5.2.2	5.2.2	5.2.2	5.2.2
Apache Santuario org.apache.santuario.xmlsec	2.1.4	2.1.7	2.3.0	3.0.1	3.0.1	3.0.3	3.0.4
Apache Shiro org.apache.shiro	1.8.0	1.8.0	1.8.0	1.11.-0	1.11.-0	1.13.-0	1.13.-0
Apache Velocity org.apache.velocity	1.7	2.3	2.3	2.3	2.3	2.3	2.3
Apache WS Security org.apache.ws.security	2.2.5	2.2.5	2.2.5	3.0.02	3.0.02	3.0.02	3.0.02
Apache WS XML Schema org.apache.ws.xmlschema	2.2.5	2.2.5	2.2.5	2.3.02	2.3.02	2.3.02	2.3.02
Apache Xalan	2.7.1	2.7.1	2.7.1				

org.apache.xalan							
Apache Xerces org.apache.xerces	2.12.0-.SP03	2.12.-2	2.12.-2	2.12.-2	2.12.-2	2.12.-2	2.12.-2
Apache XML Resolver org.apache.xml.xml-resolver	1.2	1.2	1.2	1.2	1.2	1.2	1.2
Auth0 Java JWT com.auth0.java-jwt	2.2.2	2.2.2	3.19.-2	3.19.-2	3.19.-2	3.19.-2	3.19.-2
AWS SDK for Java com.amazonaws	1.11.8-65	1.11.-983	1.12.-286	1.12.-396	1.12.-396	1.12.-657	1.12.-657
Bouncy Castle org.bouncycastle	1.69	1.69	1.70	1.72	1.76	1.78	1.78.-1
ClamAVJ com.-philvarner.clamavj	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Google Gson com.google.gson	2.8.6	2.8.6	2.9.0	2.9.0	2.9.0	2.9.0	2.9.0
Google Guava com.google.guava	25.0-jre	25.0-jre	31.1-jre	31.1-jre	32.1.-2-jre	32.1.-2-jre	32.1.-2-jre
Groovy Lang org.codehaus.groovy	3.0.8	3.0.8	3.0.1-2	3.0.1-2	3.0.1-2	3.0.1-7	3.0.1-7
Groovy WS-Lite org.codehaus.groovy	0.8.0	0.8.0	0.8.0	0.8.0	0.8.0	0.8.0	0.8.0

Groovy HTTP Builder org.codehaus.groovy	0.7.1	0.7.1	0.7.1	0.7.1	0.7.1	0.7.1	0.7.1
Hessian Remoting com.caucho.hessian	4.0.63	4.0.6-5	4.0.6-5	4.0.6-5-jakart-a-tjm	4.0.6-5-jakart-a-tjm	4.0.6-5-jakart-a-tjm	4.0.6-5-jakart-a-tjm
Java Sysmon com.jezhumble.-javasysmon	0.3.3	0.3.3	0.3.3	0.3.3	0.3.5-patch-ed	0.3.5-patch-ed	0.3.5-patch-ed
Jaxen org.jaxen	1.1.6	1.2.0	1.2.0	1.2.0	1.2.0	1.2.0	1.2.0
Jayway JsonPath com.jayway.jsonpath	2.4.0	2.7.0	2.7.0	2.7.0	2.7.0	2.9.0	2.9.0
JBoss RESTEasy JAX-RS org.jboss.resteasy.res-teasy-jaxrs	3.15.1-.Final	4.7.4.-Final	4.7.7.-Final	4.7.7.-Final	6.2.5.-Final	6.2.7.-Final	6.2.7.-Final
Joda Time org.joda.time	2.9.7	2.9.7	2.11.-0	2.12.-1	2.12.-1	2.12.-5	2.12.-5
Jsch com.jcraft.jsch	0.1.55	0.1.5-5	0.1.5-5	0.1.5-5	0.1.5-5	0.1.5-5	0.1.5-5
Json-Lib net.sourceforge.json	2.4	2.5.0	2.5.0	2.7.0	2.7.0	2.7.0	2.7.0
JSoup	1.14.2	1.14.-2	1.15.-3	1.15.-3	1.15.-3	1.15.-3	1.15.-3

org.jsoup							
JUnit org.junit	4.13.1	4.13.-2	4.13.-2	4.13.-2	4.13.-2	4.13.-2	4.13.-2
Microsoft EWS Java API com.microsoft.ews-java-api	2.0	2.0	2.0	2.0	2.0	2.0	2.0
Kafka org.a-pache.kafka.clients			3.2.3	3.4.0	3.4.0	3.6.0	3.6.0
Microsoft JDBC Driver com.-microsoft.sqlserver	8.4.1.j-re8	9.2.1.-jre8	9.2.1.-jre8	9.2.1.-jre8	9.2.1.-jre8	9.2.1.-jre8	9.2.1.-jre8
Mockito org.mockito	3.5.13	3.8.0	3.10.-0	3.10.-0	3.10.-0	3.10.-0	3.10.-0
MVEL org.mvel	2.4.10-.Final	2.4.1-2.Fin-al	2.4.1-2.Fin-al	2.4.1-2.Fin-al	2.4.1-2.Fin-al	2.5.1.-Final	2.5.1.-Final
MySQL JDBC Driver com.mysql	8.0.21	8.0.2-3	8.0.2-9	8.0.2-9	8.0.2-9	8.2.0	8.2.0
PayPal Java SDK com.paypal.sdk							
QRGen net.glxn.qrgen	1.4	1.4	1.4	1.4	1.4	1.4	1.4

Oracle JDBC Driver com.oracle							
Open SAML org.opensaml	3.3.1	3.4.6	3.4.6	4.2.0	4.2.0	4.2.0	4.3.0
Quartz org.quartz	2.3.2	2.3.2	2.3.2	2.3.2	2.3.2	2.3.2	2.3.2
Recaptcha net.tane-sha.recaptcha	0.0.8	0.0.8	0.0.8	0.0.8	0.0.8	0.0.8	0.0.8
SafeNet Luna com.safenet.luna	5.3.0-11	5.3.0-11	5.3.0-11	5.3.0-11	5.3.0-11	5.3.0-11	5.3.0-11
SAXON net.-sourceforge.saxon	9.4.0.-6	9.4.0.-6	9.4.0.-6	9.4.0.-6	9.4.0.-6	9.4.0.-6	9.4.0.-6
SLF4J org.slf4j	1.7.30	1.7.3-2	1.7.3-6	2.0.7	2.0.9	2.0.9	2.0.1-3
Spring Framework org.s-pring-framework.spring	5.3.0	5.3.2-0	5.3.2-0	6.0.8	6.0.1-1	6.0.1-9	6.1.1-2
Spring Security org.s-pring-framework.security	5.5.1	5.6.5	5.6.5	6.0.3	6.1.4	6.1.8	6.1.8

Super CSV org.supercsv	2.4.0	2.4.0	2.4.0	2.4.0	2.4.0	2.4.0	2.4.0
Tika version.org.apache.tika	1.26	1.28.-2	1.28.-4	1.28.-4	1.28.-4	1.28.-4	1.28.-4
User-Agent-Utils eu.bitwalker.useragentutils	1.21	1.21	1.21	1.21	1.21	1.21	1.21
WildFly org.wildfly	24.0.1-.Final	26.1.-1.Final	26.1.-2.Final	28.0.-1.Final	30.0.-1.Final	31.0.-1.Final	33.0.-1.Final

The table above uses the following legend:

- - A library is still supported by Manager but it is not distributed with the installer. You may need to deploy it manually. For more information, see the installer guide of your Manager version.
- - A library is not available or introduced in the software release
- - A library is removed from the software release
- [1] - A custom library adjusted for the WildFly upgrade to ensure it is Jakarta compatible
- [2] - A part of the WildFly artifacts

INFO

For the complete list of third-party libraries, see a `pom.xml` file of your Manager version.

NOTE

If you have issues with the Java classloader resolving some of these libraries, you need to contact Temenos support for assistance.

TIP

If you want to run a library vulnerability check independently, you can use a library name and its version from the table above.

End-of life Releases

If you use an end-of-life (EOL) Manager version, you can check their third-party libraries here.

To view the list of third-party libraries in Manager prior to version 19.05, select System > About and click 3rd Party Libraries. | 19.05 This feature was removed in 19.05

Next, learn about our end-of-life policy.

Installation Overview

Before you can start using Journey Manager, you must correctly install and configure it. This is easy to do by following our installation guide, which is designed and written for product evaluators, administrators, DevOps, as well as developers. For Linux and Microsoft Windows installations, you should be familiar with your target operating system and its administration.

More often than not, Manager is installed on cloud so you don't need to do it yourself. However, we recommend getting familiar with this documentation as it helps you understand some product features and performance considerations as well as how to upgrade your product to newer versions.

Each major release of Manager comes with its own installation guide describing specific steps you must perform before, during and after installation or upgrade. The installation guides are PDF documents named `jm_installation_guide_v<MANAGER-MAJOR-VERSION>.pdf`, for example, `jm_installation_guide_v21.11.pdf`.

Download the JM Installation Guide

We've replaced the Manager Installation online documentation with a PDF version. Here's how you can get a copy of the PDF:

From Downloads

You can download the installation guide from the same location where the initial release of a major version of Manager downloads are hosted. For example, the installation guide for Manager 24.04 can be found in the `24.04.0` directory.
We do not update the installation guide for minor and maintenance releases of Manager.

> INFO
>
> Registered members of our resources website can request access to download Temenos Journey Manager products and documentation for supported Temenos Journey Manager releases.

From Teams

If you have access to the Temenos instance of Microsoft Teams, you can download the JM Installation Guide PDF from Teams. Follow these steps to download the version you need:

1. Login to Microsoft Teams.
2. Select the Temenos Journey Manager (TJM) team.
3. Select the Temenos Document Share channel.
4. Select Files and browse to the `JM Installation (INTERNAL USE ONLY)` folder.
5. Download the JM Installation Guide PDF (`jm_installation_guide_v24.04.pdf`).

By Request

Otherwise, contact us to request a copy of the JM Installation Guide PDF.

Plan the Installation

You need to plan the installation or upgrade of Manager carefully before you run installer in your environment or ask the cloud team to install it for you. There's no cover-it-all plan for every environment or configuration, but we have combined the best practices into a simple check list below. We strongly recommend reviewing it and commencing the installation only if all items of the list are satisfied.

The installation plan check list:

1. Ensure you have read and understood this installation guide. If something is unclear or your need more information, contact your Customer Service or Support.
2. Compile and build your current application packages and execute unit tests in a local IDE or on a build server using Journey SDK of the same Journey Manager version you are planning to install or upgrade to. Groovy compiler can reveal potential issues and thus provides you with more confidence to move on the next step of the upgrade.

   > IMPORTANT
   >
   > If any compilation or build errors arise, resolve them before commencing the installation. Resolutions may include the product configuration, Groovy scrips corrections, or cloud hosting configuration. For more information, contact your Customer Service or Support.

3. Review all release notes of your current and target Manager versions, understand them, and analyze the impact on your existing solution. In some cases, you may need to review your current solution, for example, due to critical library updates.
4. Ensure the back up and rollback procedures for your organizations, form spaces, forms, services, service connections work as expected, so you can restore them from back-ups.

5. Plan ahead to upgrade all your environments to the same target Manager version.

> IMPORTANT
>
> We don't recommend deploying any application packages in to a production environment until your test, staging, UAT, and production environments run the same Manager version.

Next, learn about Journey Manager features.

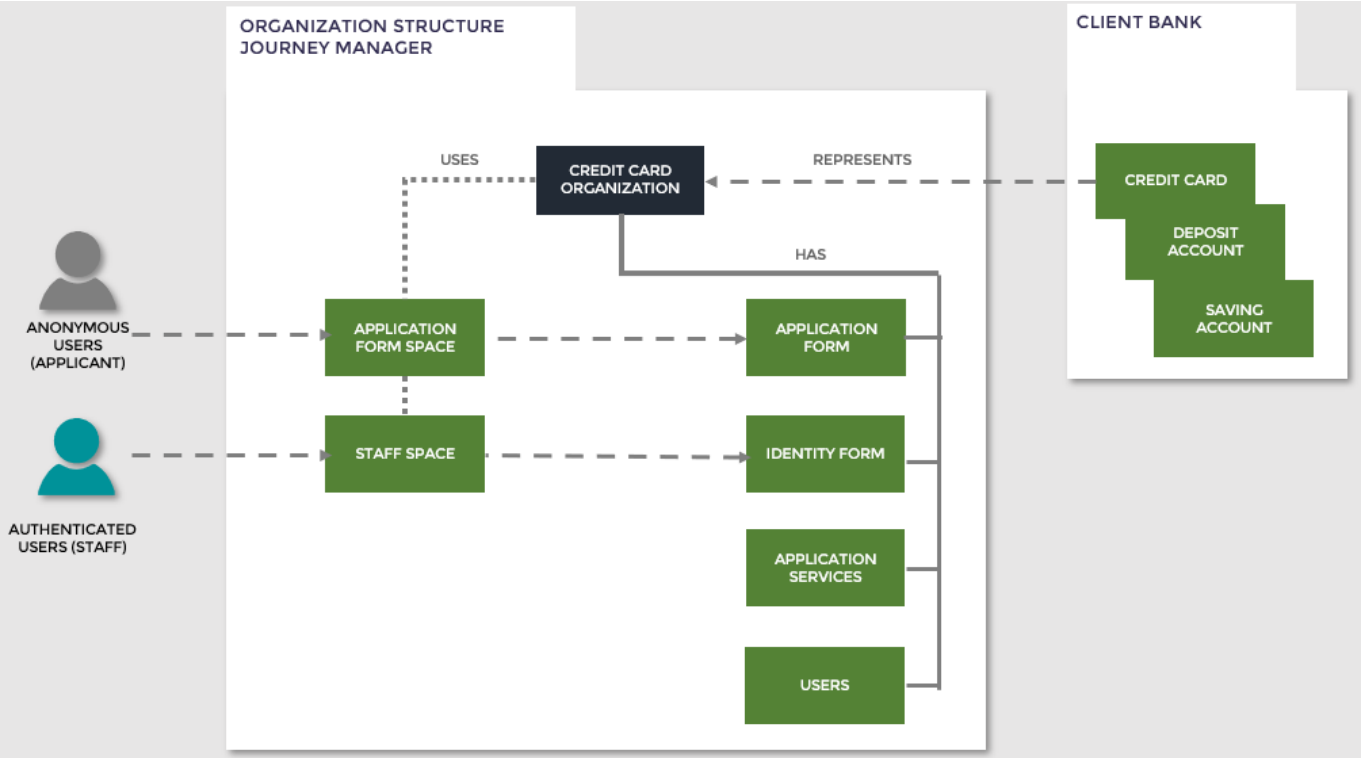
Organizations Overview

Manager uses the concept of organizations to help you represent your business that can be:

- A company
- A department or a branch within the company
- A government organization
- A product

An organization is a logical entity within Manager that owns forms and their assets, such as resources and libraries, so they can be grouped and managed as a whole. An organization can have many forms, but a form always belongs to one organization only. Each organization can use one or several form spaces to host its forms. Each organization has properties which will change the way forms appear and operate. Properties can be used to control things like colors, logos, addresses, signatures, form options, or almost any part of form behavior.

In the diagram below, Manager represents a client bank structure with the established credit card application process.



Manager is designed to accommodate anything from a single organization with a single form to multiple organizations with many forms having multiple form versions.

NOTE

We recommend creating a separate organization in Manager for its corresponding real life counterpart.

Organizations allow:

- Developers within each organization to access, deploy, and manage forms for their own organization.
- Administrators to manage one or more organizations.
- Operators to perform a broad range of day-to-day operations for one or more organizations.

Organizations allows you to create and configure forms with standard:

- Branding
- Delivery channels
- Properties
- Services
- Reference data
- Data retention

Manager also uses the following core concepts relevant to the organization:

- Users
- Forms
- Form Versions
- Form Data Configuration
- Properties
- XML Prefill
- Application Packages

We recommend getting familiar with these concepts before you start creating and configuring your organization.

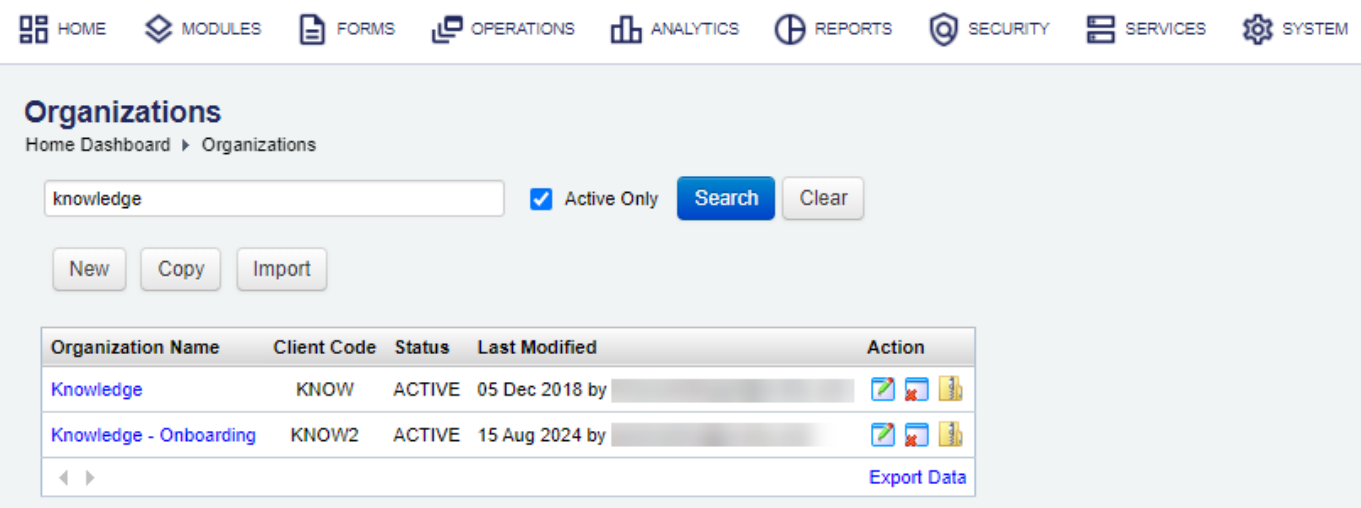
Next, learn how to view organizations.

View Organizations

Manager allows you to view the list of available organizations, so you can manage them efficiently.

To view organizations, select Forms > Organizations.

Organizations are shown page by page, with the maximum number of records per page configured via the user's preferences.



The list displays the following details:

- Organization Name: a globally unique organization name, which is often displayed on form's landing pages.
- Client Code: a globally unique organization ID, which must not include any spaces. A client code is used in a long form URL. For more information, see Form Friendly URL.
- Status: determines the availability of an organization and can be one of the following: | 21.11 This feature was introduced in 21.11.
  - Active: it is in use and readily available to all users. All functionality is allowed.
  - Offline: it can't be used at all. No functionality is available, including DDS, Delivery and so on.
  - Inactive: similar to Offline.
  - NoRender: it doesn't allow for new or saved and resumed forms, but ongoing open sessions can be completed. DDS, submits, saves, and delivery are allowed.
- Last modified: a date when the organization's configuration was updated last time and a user who did it.

To filter or search the roles by one or more criteria, specify the following settings and click Search:

- Organization's name or part of a name.
- Active Only. Only Active organizations are listed by default.
- Select an organization from the Organization dropdown list in the Toolbar.

Click Clear to reset the search criteria.

You can also perform the following actions:

- Create an organization
- Copy an organization
- Remove an organization
- Export an organization
- Import an organization

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to create an organization.

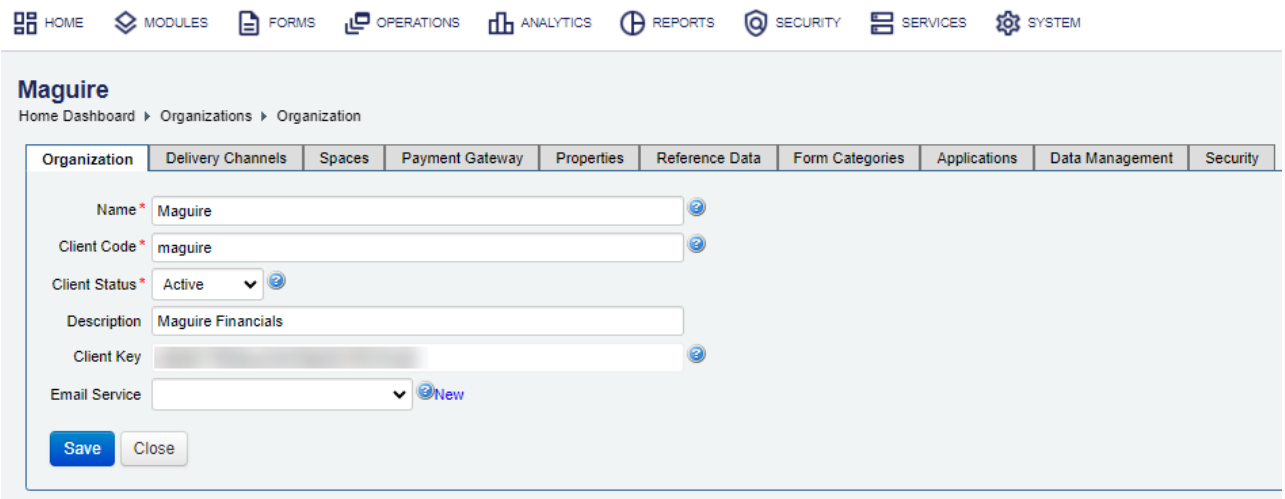
Create an Organization

Manager allows you to create an organization to encapsulate all relevant forms and resources.

> NOTE
>
> Only users with administrator privileges can create new organizations.

To create an organization:

1. Select Forms > Organizations.
2. Click New.



2. Enter a unique organization name in the Name field.
3. Enter a unique organization ID in the Client Code field.
4. Select the availability status of an organization from the Client Status dropdown list, which can be one of the following: | 21.11 This feature was introduced in 21.11.
   - Active: it is in use and readily available to all users. All functionality is allowed.
   - Offline: it can't be used at all. No functionality is available, including DDS, Delivery and so on.
   - Inactive: similar to Offline.
   - NoRender: it doesn't allow for new or saved and resumed forms, but ongoing open sessions can be completed. DDS, submits, saves, and delivery are allowed.
5. Provide a comprehensive description in the Description field. If a client has multiple organizations, a description may also be useful for those users who have global permissions to multiple organizations on a single Manager instance.
6. Provide an organization credential alias or client key in the Client Key field, which is the organization unique identifier used for subscription delivery Web Service integration.

7. Select the Active checkbox to allow this organization to use all facilities. Otherwise, it is inactive which means it is disabled and is not shown.

   NOTE

   It's not recommended to remove an organization. Instead, you should make the organization inactive.

8. Select an email service, used by this organization's forms to send submission related emails to users, from the Email Service dropdown list.
9. Click Save to update the changes.

Once the new organization is created, Manager automatically takes you to the Spaces tab so you can assign form space.

Next, learn how to configure an organization.

Copy an Organization

Manager allows you to copy an existing organization. You can use this functionality to:

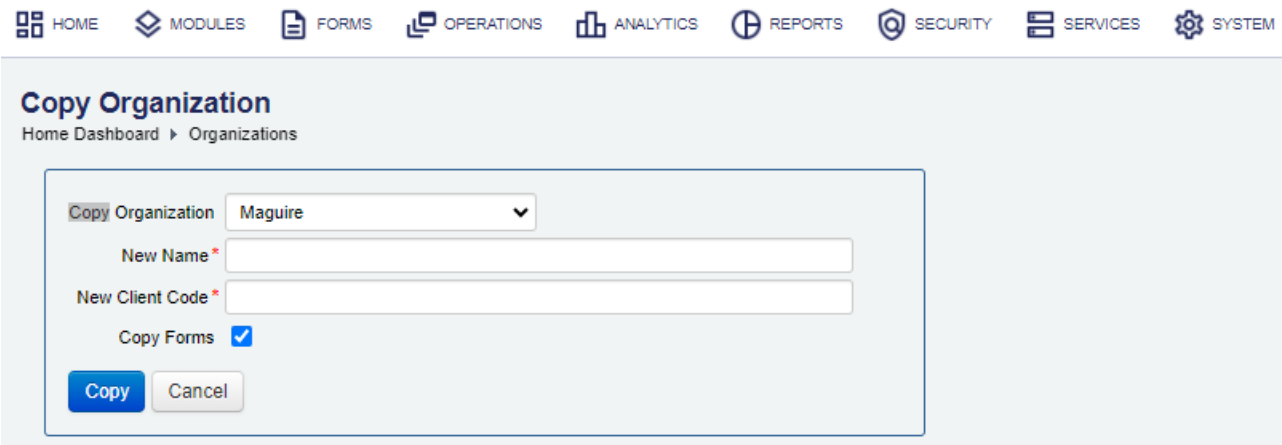
- Test organizations
- Create multiple copies of the same organization
- Configure a new organization from an existing organization

NOTE

The copied organization inherits the same configuration as the original one, which can include all of the organization's forms and resources.

To copy an existing organization:

1. Select Forms > Organizations.
2. Click Copy.



3. Select the organization you want to copy from the Copy Organization dropdown list.
4. Enter a unique organization name in the New Name field.
5. Enter a unique and abbreviated organization name in the New Client Code field.
6. Select the Copy Forms checkbox to copy all this organization's forms. If you do not want to copy all this organization's forms, clear this checkbox.
7. Click Copy.

NOTE

Organization's payment configuration, report schedules, delivery details and form manager references are not copied.

NOTE

The newly created organization will be in Active state.

Next, learn how to configure an organization.

Remove an Organization

Manager allows you to remove an organization and all its dependent resources, such as forms and submissions.

> NOTE
>
> It is recommended to make an organization inactive rather than remove it, especially if there have been submissions for any of its forms.

To remove an organization:

1. Select Forms > Organizations.
2. Click Remove and confirm your decision. This will permanently remove the organization.

Next, learn how to export an organization.

Export an Organization

Manager allows you to export organizations. You can use this functionality to:

- Backup or archive organizations
- Re-create one or several organizations in another Manager
- Promote or migrate one or several organizations to other environments

To export one or several organizations, select Forms > Organizations and click Export Archive.

Manager exports them into an organization archive ZIP file, which will be downloaded to your PC.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to import an organization.

Import an Organization

Manager allows you to import organizations. You can use this functionality to:

- Roll-back one or several organizations
- Re-create one or several organizations in another Manager
- Promote one or several organizations to other environments

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

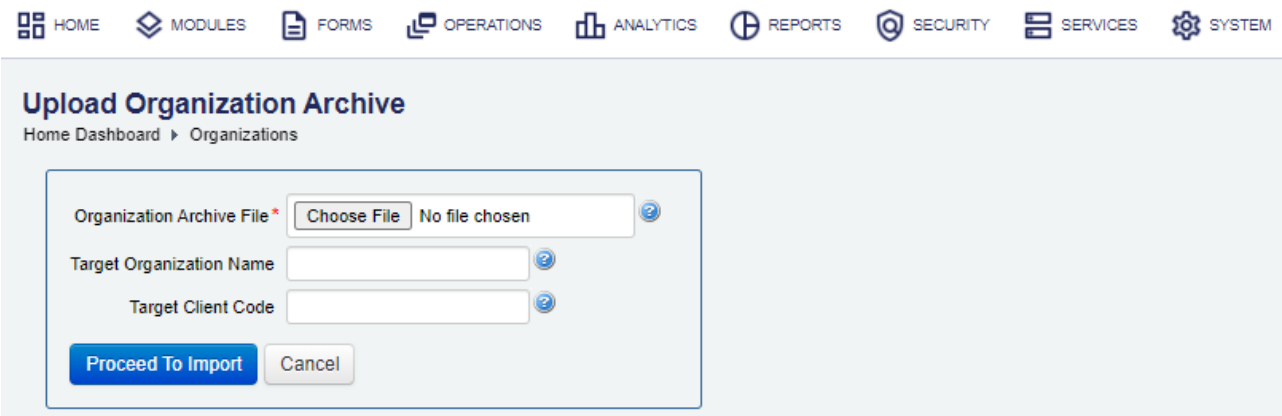
You can import an organization using one of following options:

- UI
- SDK
- API

Import Using UI

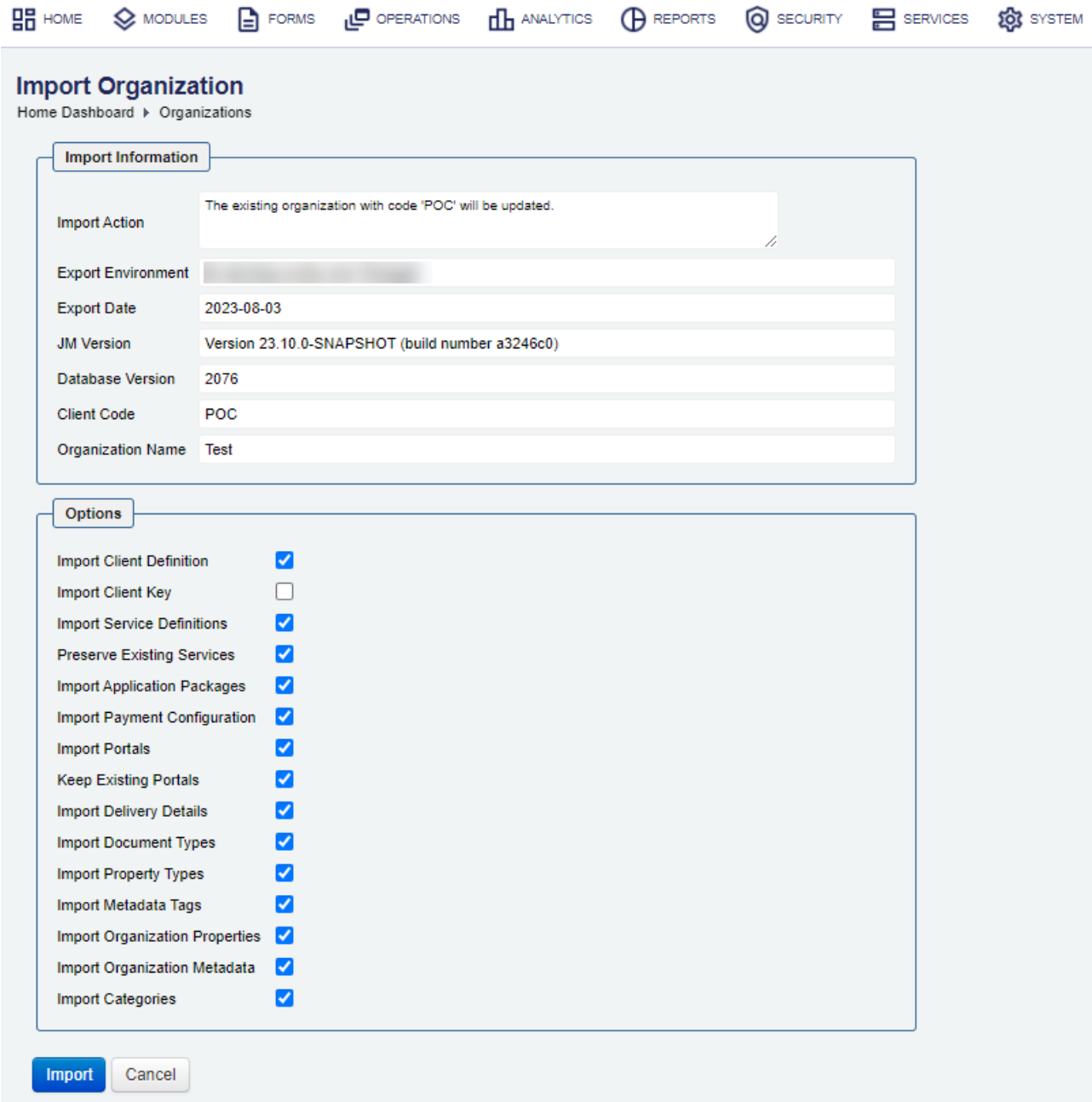
To import an organization:

1. Select Forms > Organizations and click Import.



2. Click Choose File and choose an organization archive ZIP file that was created during the organization export.
3. Provide a new Target Organization Name and Target Client Code to import the organization with a different name and code. You might want to do so if an organization with the same name and code already exist in the target environment. To import the organization as a new one in another Manager environment, leave Target Organization Name and Target Client Code fields blank.

4. Click Proceed To Import.
5. Review the Import Options.

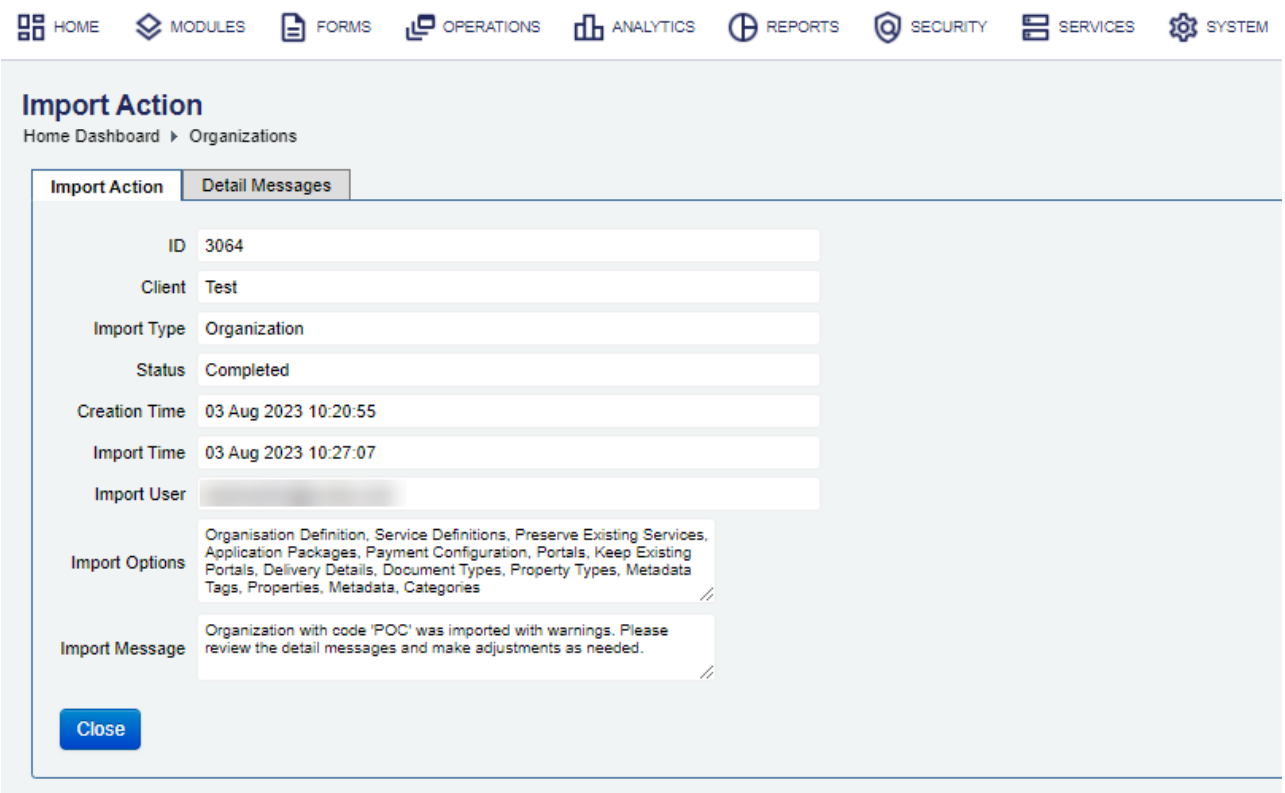


6. Select the Import Client Definition checkbox to import an organization definition.
7. Select the Import Client Key checkbox to import an organization key.
8. Select the Import Service Definitions checkbox to import service definitions.
9. Select the Preserve Existing Services checkbox to preserve existing services.
10. Select the Import Application Packages checkbox to import application packages.
11. Select the Import Payment Configuration checkbox to import payment configuration.
12. Select the Import Portals checkbox to import form spaces.

13. Select the Keep Existing Portals checkbox to preserve existing form spaces.
14. Select the Import Delivery Details checkbox to import delivery configuration.
15. Select the Import Property Types checkbox to import property types.
16. Select the Import Metadata Tags checkbox to import organization's metadata tags.

> NOTE
>
> The organization's metadata tags functionality has been removed since the Manager 18.05 release.

17. Select the Import Organization Properties checkbox to import organization properties.
18. Select the Import Organization Metadata checkbox to import organization's metadata.
19. Select the Import Categories checkbox to import form categories.
20. Click Import to complete the process. The expected result page looks like this.



21. Select the Detail Messages tab to see more information about the import. This tab pops up open if the import operation has any issues you may need to check and resolve.
22. Click Close and view organizations to verify the imported organizations have been added.

NOTE

You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

Import Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:archive-deploy goal to import an organization.

Import Using API

The Journey Manager REST API provides the endpoint URL to import an organization.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s):://<JM_SERVER>/manager/secure/rest/archive/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as Base64[1].

`--form ‘archiveFile=@“/Users/my-client.zip”’` - a path to an archive file to be imported.

`--form ‘importOptions=“ORGANISATION_DEFINITION, CLIENT_KEY,SERVICE_DEFINITIONS”’` - a comma-separated list of import options to perform. The option names are the same as described in the Import Using UI section but they use an underscore, for example, `ORGANISATION_DEFINITION` for Import Organization Definition.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

Example

```
curl --location --request PUT ‘https://<JM_SERVER>/manager/secure/rest/archive/v1/’
\
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
```

[1]Base64 is a binary to a text encoding scheme that represents binary data in an American Standard Code for Information Interchange (ASCII) string format.

```
--form ‘archiveFile=@“/Users/my-client.zip”’ \
--form ‘importOptions=“ORGANISATION_DEFINITION,CLIENT_KEY,SERVICE_
DEFINITIONS,PRESERVE_EXISTING_SERVICES,\
APPLICATION_PACKAGES,PAYMENT_CONFIGURATION,PORTALS,KEEP_EXISTING_PORTALS,DELIVERY_
DETAILS,DOCUMENT_TYPES,\
PROPERTY_TYPES,PROPERTIES,METADATA_TAGS,METADATA,CATEGORIES”’
```

Next, learn how to migrate an organization.

Migrate an Organization

Manager provides a mechanism to migrate organizations from one environment to another. It allows you to develop and test organizations in different environments before migrating them to the production environment.

When migrating an organization, all of its associated configurations are also migrated, except the organization administrator's entities and forms with all their data. Once the organization is migrated to a new environment, it will have configuration of the original environment, although entry IDs may change.

To migrate one or several organizations:

1. Export organizations.
2. Import organizations to another Manager environment.

> NOTE
>
> You can use the same process to perform an organization backup and rollback.

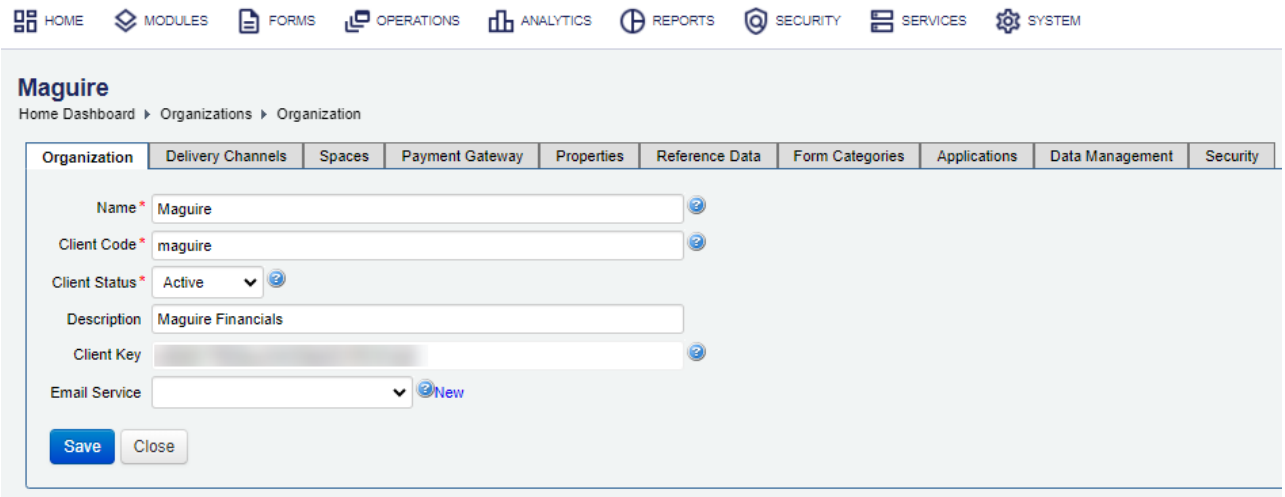
Next, learn how to configure an organization.

Configure an Organization

Manager allows you to create new and configure existing organizations.

To configure an organization:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.



3. Update the organization name in the Name field. For more information, see how to create an organization.
4. Update a unique and abbreviated organization name in the Client Code field.
5. Change the organization status.
6. Assign form spaces to host the organization's forms.
7. Create delivery channels that will be available to the organization and its forms and form versions. For more information on which delivery channel is used by which form and form version, see delivery channels overview.
8. Configure the payment gateway for the payment details required for any form within this organization to enable credit card payments.
9. Configure organization properties for customizing a broad range of features, such as emails.
10. Configure organization reference data for storing reference data that can be used by any form within this organization. | 17.10 This feature was introduced in 17.10..
11. Configure form categories for the ability to filter forms by categories within form spaces.
12. Configure applications for grouping organization specific services to facilitate their migration between Manager servers.
13. Configure this organization report schedules. | 19.05 This feature was removed in 19.05

14. Configure this organization data management.
15. Configure security for general settings relating to form rendering.

Next, learn how to assign form spaces to an organization.

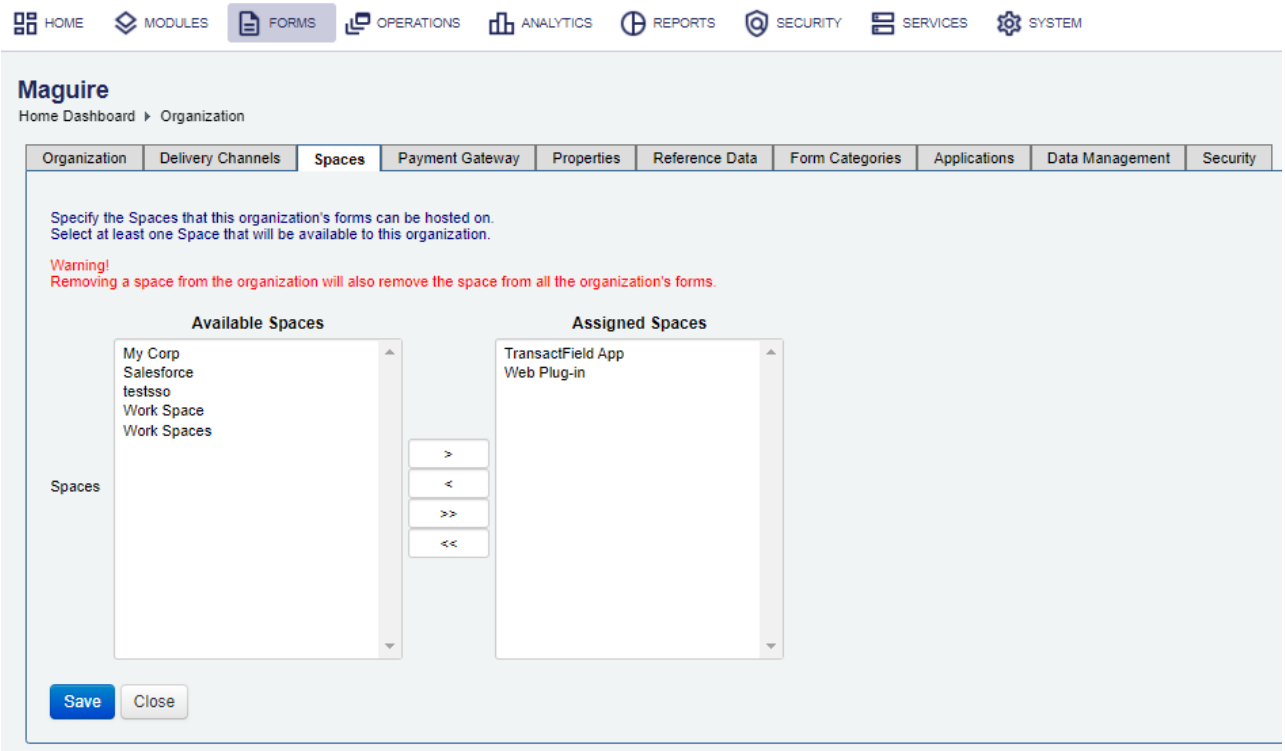
Assign Form Spaces to an Organization

Manager allows you to assign form spaces to an organization to host all organization's forms and their resources. You can assign both default form spaces, as well as any custom form spaces you have configured in your journey manager environment. This determines what form spaces will be available to forms assigned to the organization. You can also assign form spaces at the form level.

> NOTE
>
> You must assign the Web Plug-In form space to each organization so the Pull From Maestro button works.

To assign form spaces to an organization:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Spaces tab to see all assigned form spaces in the Assigned Spaces field.



4. Select one or more form spaces, which you want to assign to the organization, from Available Spaces.

NOTE

You can use the `Shift` and `Ctrl` keys to select multiple spaces on Windows PC.

5. Click > to move the selected form spaces from Available Spaces to Assigned Spaces.
6. Click < to revoke the selected form spaces from Assigned Spaces.

NOTE

Removing a form space from the organization will also remove the form space from all forms belonging to this organization.

7. Click >> to assign all form spaces.
8. Click << to revoke all form spaces.
9. Click Save to update the changes.

NOTE

You can assign or re-assign form spaces to an organization at any stage.

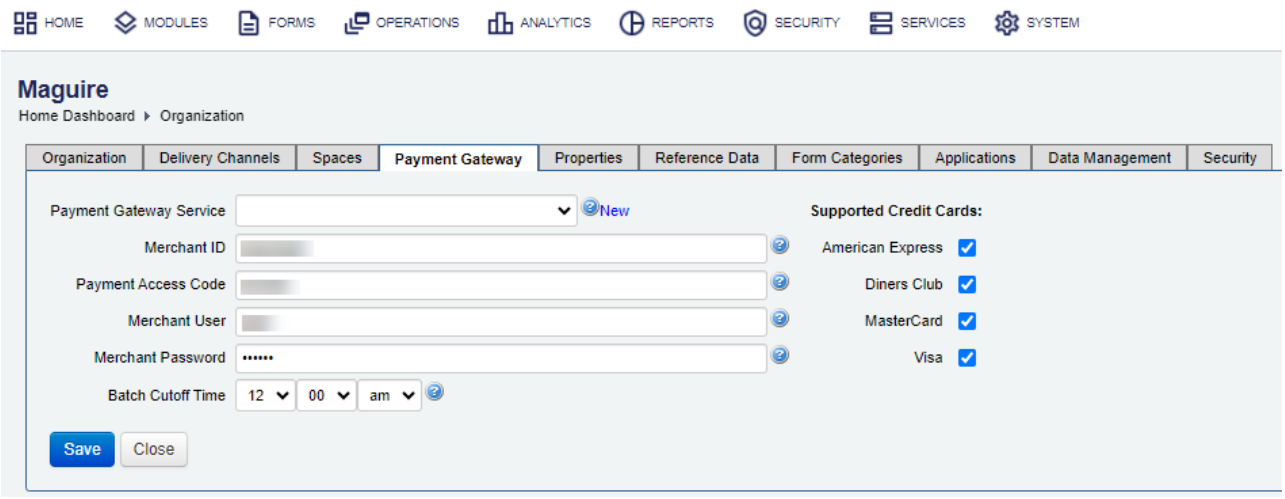
Next, learn how to configure an organization payment gateway.

Configure an Organization Payment Gateway

Manager allows you to enable and configure credit card payment gateways from any form within the selected organization. The payment gateway is necessary when a form requires additional payment details from a user as part of a successful form submission.

To configure a payment gateway:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Payment Gateway tab.



4. Select a payment gateway service from the Payment Gateway Service dropdown list that this organization uses when processing credit card payments.

   NOTE

   An administrator must create a service for the payment gateway and make it available to this organization.

5. Enter the merchant account number into the Merchant ID field that specifies the merchant's account to the payment gateway.
6. Enter the merchant payment access code into the Payment Access Code field that specifies the payment access code used to authenticate the merchant account on the payment gateway.
7. Enter the merchant user name into the Merchant User field specifies the merchant's user name if it is requested by the payment gateway.

8. Enter the merchant password into the Merchant Password field that specifies the merchant's password for when a password is requested by the payment gateway.
9. Select the cut-off time from the Batch Cutoff Time dropdown lists that specifies the time of day when payments stop being transferred to the organization's bank account on the same day.
10. Select the credit cards that this payment gateway accepts in the Supported Credit Cards section:
    - American Express
    - Diners Club
    - MasterCard
    - Visa
11. Click Save to update the changes.

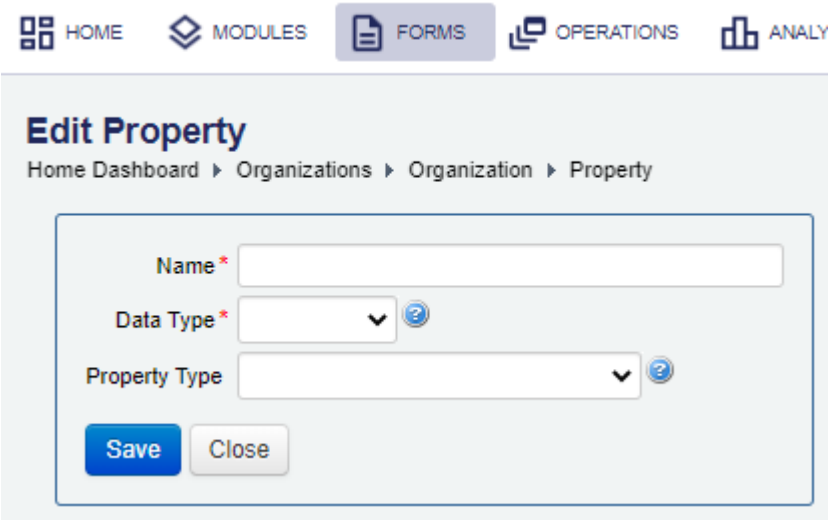
Next, learn how to configure organization properties.

Configure Organization Properties

Manager allows you to add new and configure existing organization's properties, which can be used to pre-populate forms or modify their behavior. You may want to use these property for form runtime branding when form's properties can be dynamically modified which will be instantaneously reflected at form render time.

To create a new property:

1. Select Forms > Organizations.
2. Locate an organization and click New.



3. Provide a unique name in the Name field.
4. Select a property data type from the Data Type dropdown list, which can be one of the following:
    - Boolean
    - CSV
    - HTML
    - Image
    - JSON
    - List
    - Long Text
    - Number
    - String

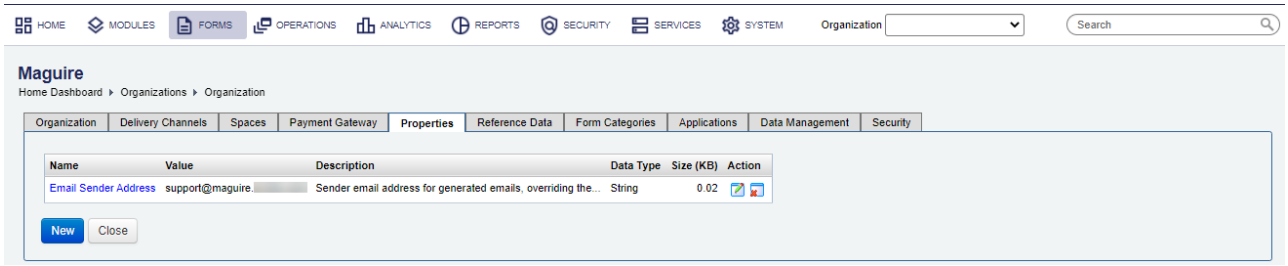
5. Select a property type from the Property Type dropdown list.

> NOTE
>
> If a property type is not in the list, you need to create a new property type of the same data type as the property you are creating.

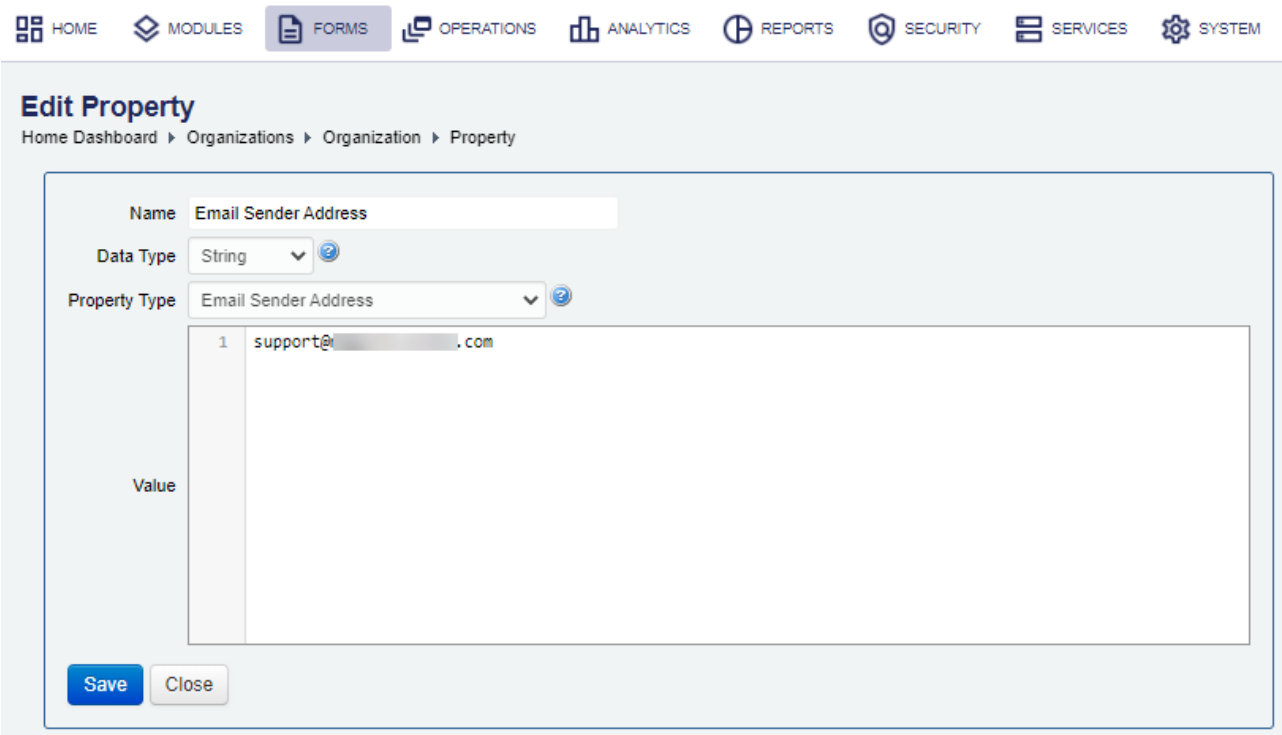
6. Click Save to update the changes.

To configure organization properties:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Properties tab to view, create and edit properties.



4. Select a property and click Edit.



5. Edit the property's value in the Value field.
6. Click Save to update the changes.

Next, learn how to configure reference data.

Configure Organization Metadata Tags

Manager allows you to configure metadata tag definitions for your organization. Metadata can be used to run queries on the set of forms defined in Manager using a Web Service API. If you want to define metadata for an organization, you can configure values for these metadata tags. All forms will inherit organization metadata when they are created.

To configure a metadata value:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Form Tags tab and click New.
4. Choose a metadata tag to assign a value to from the Tag Name dropdown.

   NOTE

   Only metadata tags with a scope of Client will be displayed

5. Set Tag Value to the metadata value for this organization and save. You will see that the metadata value has been added to the Form Tags tab.
6. Click Save to update the changes. All forms that are set up for this organization will inherit this metadata value.

Next, learn how to configure reference data.

Configure Organization Reference Data

Many applications require some type of reference data that doesn't change often, such as lists of countries, cities and postcodes, vehicle manufacturers and models, and so on. Manager allows you to view, create, configure, and test reference data for a selected organization. Since Maestro 17.10, reference data can be stored in Manager and accessed in Maestro forms by calling the `Transact.referenceData` function in the `Transact` name space. For more information, see Maestro API.

To start using reference data in your applications, you should understand more about its structure and types. You can think of reference data as an object that:

- is defined at the organization level.
- is uniquely identified in an organization by a name and locale.
- supports language translation through different locales.
- supports the CSV[1], JSON[2] and XML[3] data types.
- supports optional dynamic filtering and sorting of the CSV data type.
- supports server caching, with configurable refresh time-out specified for each reference data item.

Manager stores reference data using the following file formats:

CSV
: Supports filtering and sorting, so it has specific requirements described below. The first line of a CSV file must be a heading line containing comma-separated field names with corresponding data following on subsequent lines. Field names defined in the first line are used in filtering and sorting order options by Maestro clients. Field names in the heading line are case sensitive.

JSON and XML
: No specific format requirements and are returned as-is to the requesting Maestro client.

---

[1]CSV is a comma-separated values fil, which s a delimited text file that uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.
[2]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.
[3]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Every reference data item must be defined with a locale[1]. The locale, in conjunction with the reference data Name property, forms a unique combination. Both the name and locale must be specified when requesting reference data in a Maestro form.

An organization can have a set of properties that you can use in your applications, so it's important to understand the difference between the organization properties and reference data. This allows us to determine when one should be preferred over the other.

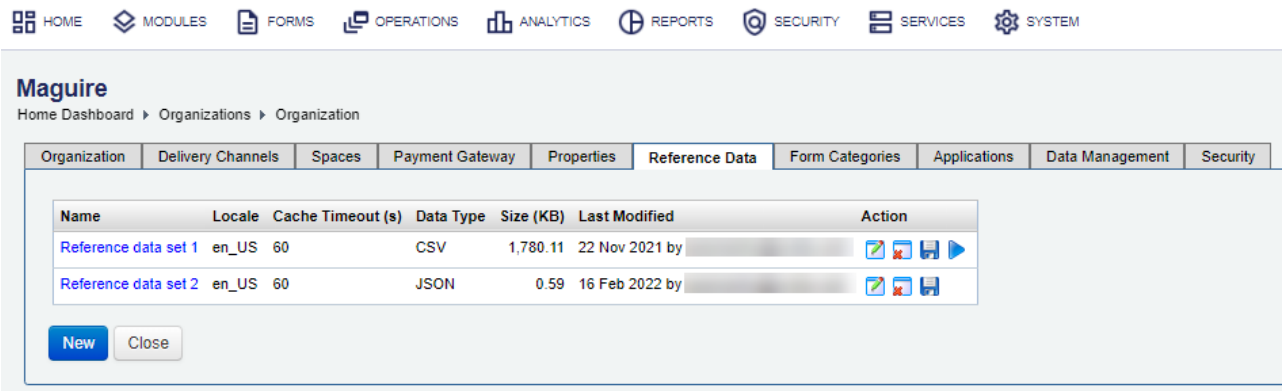
- Organization properties can be included in an application package, but reference data can't.
- Organization properties can be deployed through CI/CD pipelines, but reference data can't.
- Reference data elements include locale as part of a compound key, which means that you can have multiple reference data entries with the same name as long as the locale is different. Each organization property must be uniquely named, and there's no concept of locale.
- Organization properties support property prefill mapping.
- Reference data must be CSV, JSON, or XML. Organization properties can also be String, Boolean, Number, List or even Image.
- Reference data can be fetched from Maestro by using `Transact.referenceData(name, query, locale, hideProgressFlag, hideAlertError)`.
- Reference data is cached in Manager to improve performance, as opposed to running `PropertyQuery` every time data is needed.

To view organization reference data:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.

[1]Locale is a set of parameters that defines the user's language, region and any special variant preferences that the user wants to see in their user interface. Usually a locale identifier consists of at least a language code and a country/region code.

3. Select the Reference Data tab.



You can check the following details of all configured reference data:

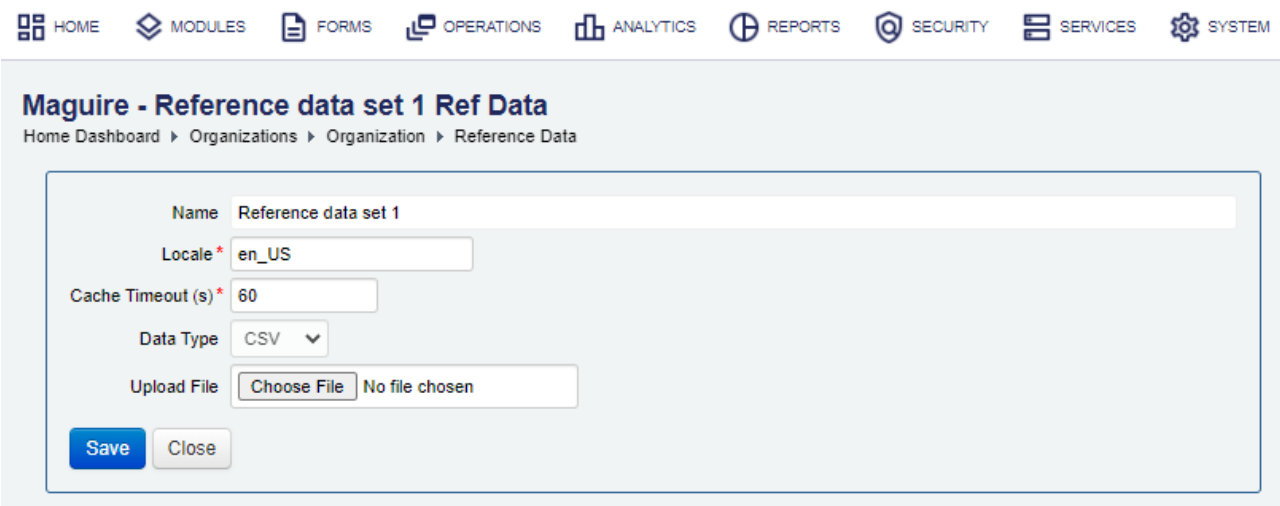
- Name of the reference data set
- Locale of the data set
- Cache timeout
- Data type
- Size of the reference data set
- Last modified

4. Click Edit to update the existing reference data.
5. Click Remove to delete a reference data.
6. Click Download as File to save a reference data set on your local PC.
7. Click Query Test to test your reference data set by running a test query against a reference data set.

> NOTE
>
> This option is only available for CSV data type.

To create or edit organization reference data object:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Reference Data tab.
4. Click New to create a new reference data set, or click Edit to update the existing reference data set.



5. Enter a unique name of reference data set in the Name field.
6. Enter a locale in the Locate field, which can be any non-empty string, for example `en_AU`.
7. Provide cache timeout in seconds in the Cache Timeout field.
8. Select a data type from the Data Type dropdown list, which can be one of the following:
   - CSV
   - JSON
   - XML
9. Click Choose File to upload a reference data file. A type of the reference data file and a file extension must match the selected data type, otherwise an error is thrown. | 24.04 This feature was introduced in 24.04
10. Click Save to update the changes.

After you've created a new reference data, you can test it verify everything works as expected.

> NOTE
>
> We recommend checking how to use reference data in a Maestro form.

Next, learn how to configure organization form categories.

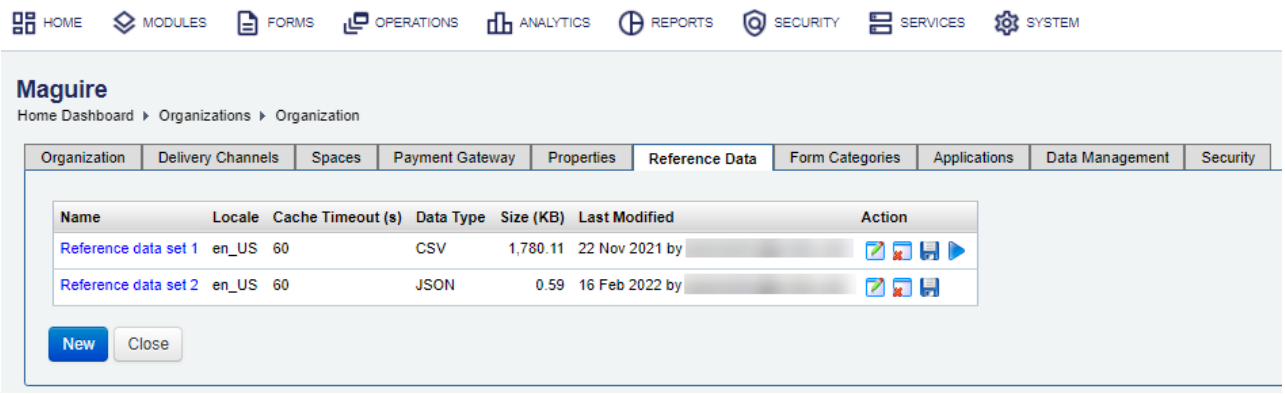
Test Organization Reference Data

Manager allows you to test reference data of a selected organization, so you can quickly validate if filtering and sorting of reference data work as expected. This is handy while building or troubleshooting Maestro forms.

> NOTE
>
> You can only test reference data defined as CSV[1] data type.

To test organization reference data:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Reference Data tab to see all configured reference data sets.



4. Choose reference data you want to test and click Query Test.

> NOTE
>
> This option is only available for CSV data type.

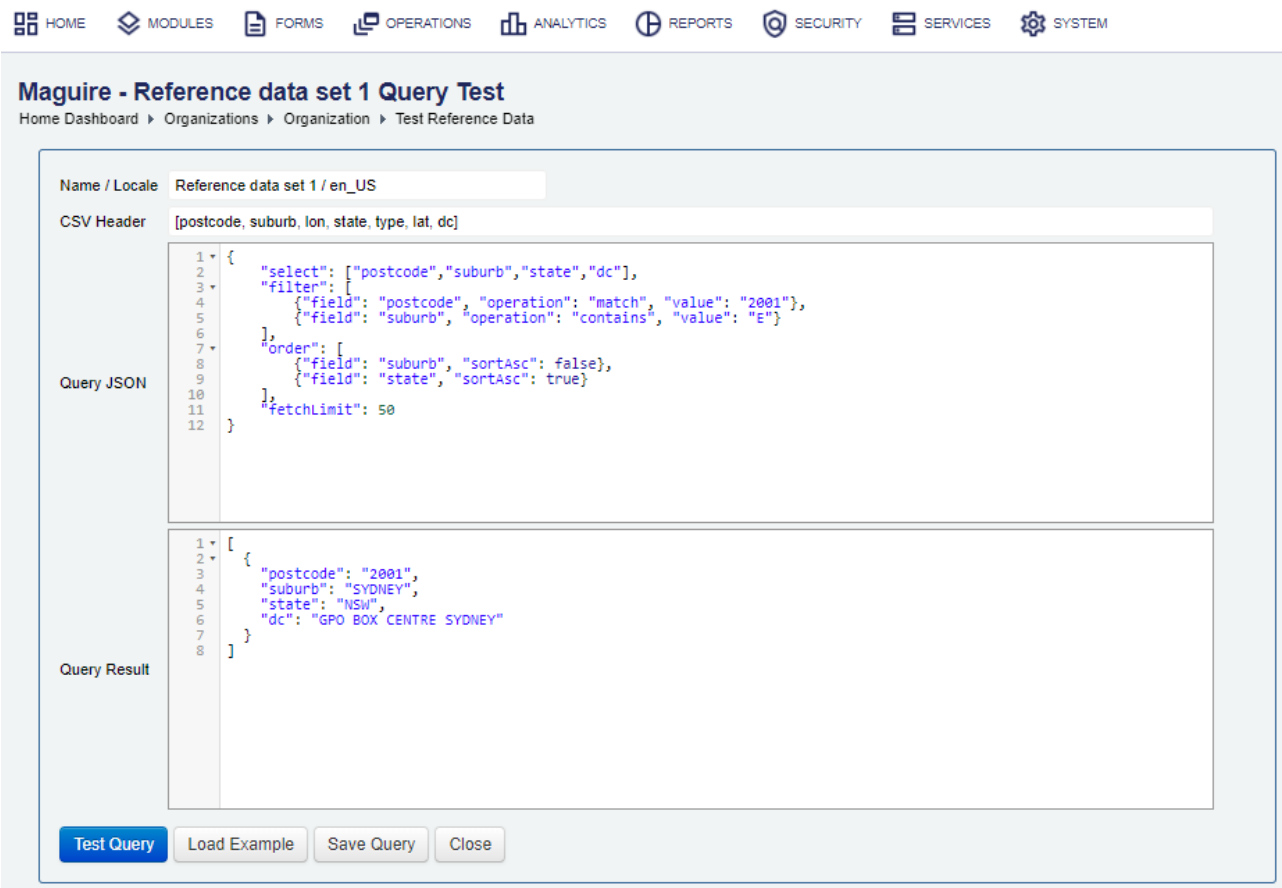
5. Update a query request in the Query JSON field and click Test Query to execute it. The example query is shown below.

---

[1]CSV is a comma-separated values fil, which s a delimited text file that uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

```
{
        "select": ["postcode","suburb","state","dc"],
        "filter": [
                {"field": "postcode", "operation": "match", "value": "2001"},
                {"field": "suburb", "operation": "contains", "value": "E"}
        ],
        "order": [
                {"field": "suburb", "sortAsc": false},
                {"field": "state", "sortAsc": true}
        ],
        "fetchLimit": 50
}
```

For more information, see query syntax.



6. Check a query result in the Query Result field.

```
[
  {
    "postcode": "2001",
    "suburb": "SYDNEY",
    "state": "NSW",
    "dc": "GPO BOX CENTRE SYDNEY"
  }
]
```

7. Click Load Example to upload a new query.
8. Click Save Query to save the current query for the future use.

Query Syntax

A query you write to sort and filter reference data is a JSON[1] object that is applied to the CSV data. You must adhere to valid JSON syntax, otherwise Manager throws an exception. For more information, see Introducing JSON.

The query object consists of four mandatory elements, which are listed below:

Element	Type	Description
`select`	array of string	The list of fields returned by a query.
`filter`	array of objects	A list of conditions to constrain the data returned by the query. Each filter condition is an object consisting of three elements: `field`, `operation` and `value`. The operation is used to perform a case-insensitive test of the field against the value, returning true for a successful test, and false otherwise.
`order`	array of objects	A list of sorting specifications that control the order of returned CSV rows. Each sorting specification is an object consisting of two elements: `field` and `sortAsc`.
`fetchLimit`	number	The maximum number of rows returned from the CSV file, applied after

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

the filter and sort operations have been applied.

The `select`, `filter` and `order` array elements are constructed using the following mandatory elements:

Element	Type	Description
`field`	string	A case sensitive string matching a column heading in the first line of the CSV file.
`operation`	string	A case sensitive string, which must be one of the following: `match`, `contains`, `startsWith`, `endsWith`, `greaterThan`, `lessThan`, `greaterThanEqual`, `lessThanEqual`.
`sortAsc`	string	Specifies how an order specification is applied to the output; either ascending (`true`) or descending (`false`).
`value`	string	A case-insensitive string used by a filter condition to test against the values in the column specified by the field property.

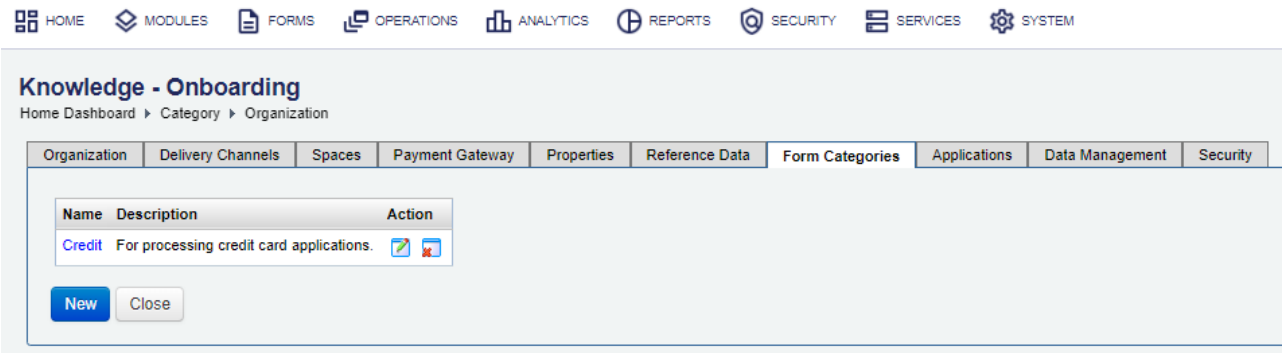
Next, learn how to configure organization form categories.

Configure Organization Form Categories

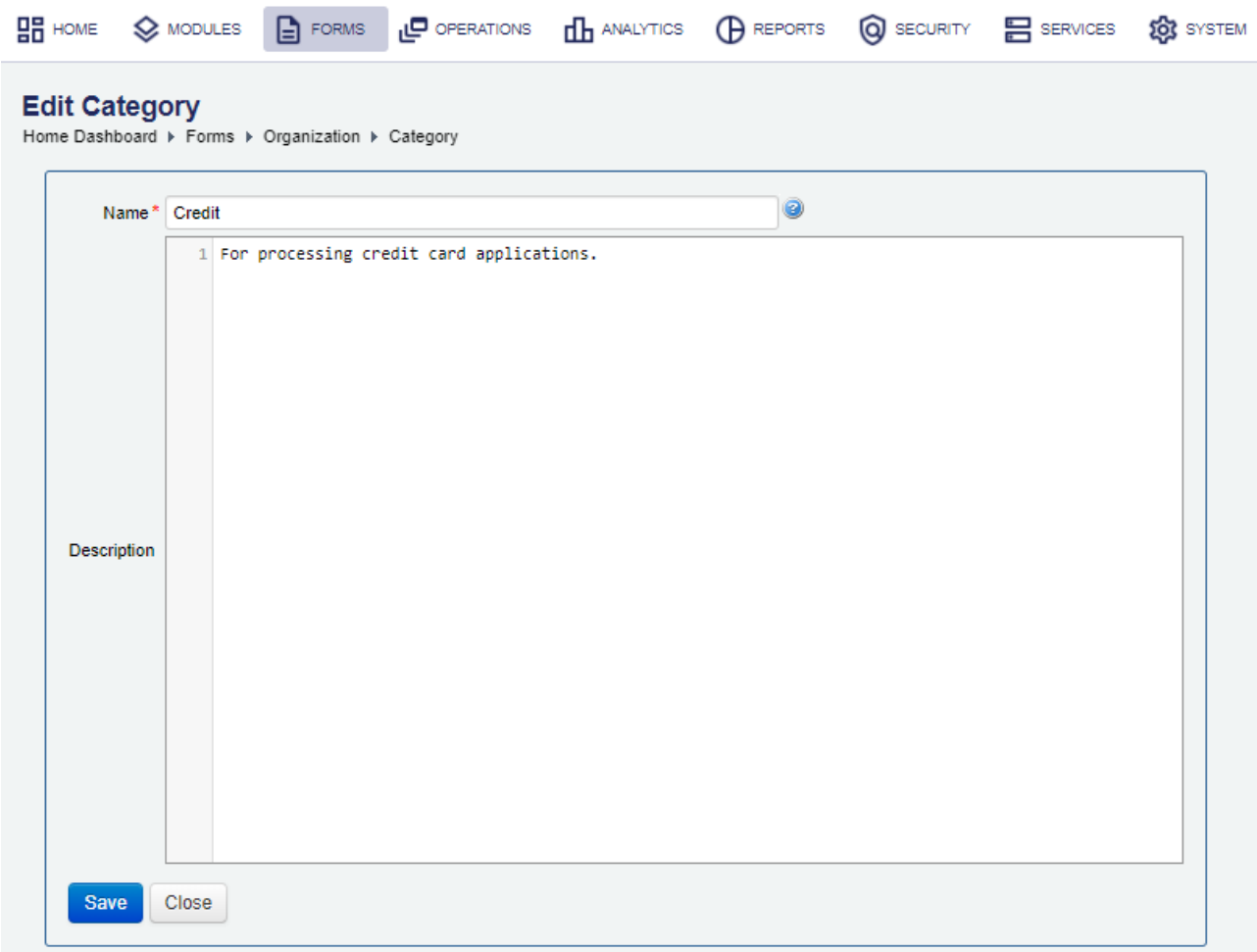
A form space, which you use to host forms belonging to your organization, may contain many forms, so with time it becomes difficult to find a form. Manager allows you to create various form categories to assist users with locating forms they need in the Forms tab. The standard form spaces support the form categories filter. However, If your form space has been customized, verify that this filter is still supported.

To create and configure a form category for an organization:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Form Categories tab to see all existing form categories.



4. Locate a form category and click Edit or click New to edit an existing or create a new form category.



5. Enter a unique form category name into the Name field.
6. Enter a suitable description into Description field.
7. Click Save to update the changes.
8. Assign each form version to the related form categories.
9. Check that a category selector appears on the form search page so you can filter forms by categories.

> NOTE
>
> For a category selector to work, a form space must support form categories.

Next, learn how to configure an application package for an organization.

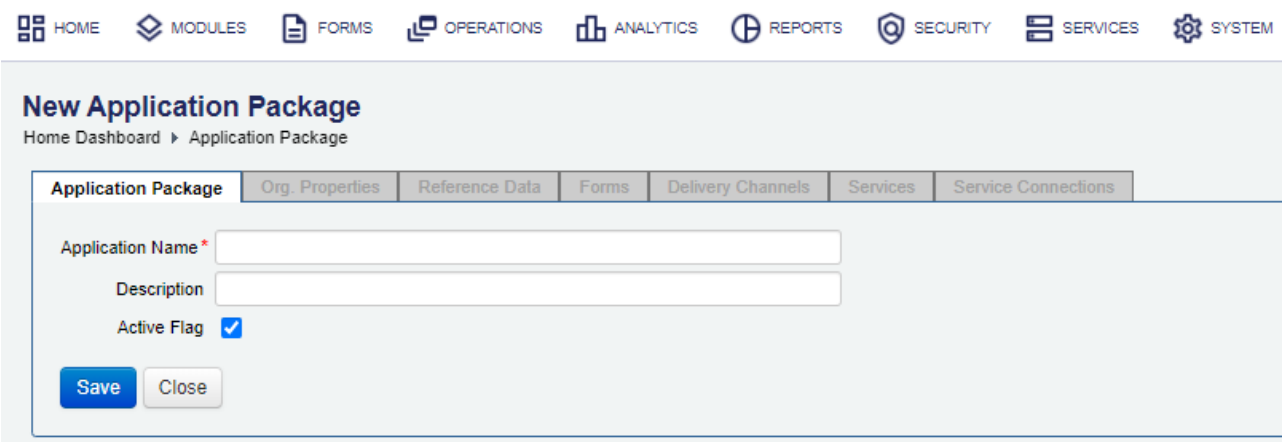
Create an Application Package

Manager allows you to package forms and related services and assets into an application package, so you can easily migrate it between organizations in different Manager instances.

To package a form, you need to create a new application package and assign the required services and forms to it. You can also assign the required organization properties, delivery channels, and service connections, where applicable.

To create an application package:

1. Select Forms > Organizations.
2. Locate an organization that you want to configure or manage existing application packages.
3. Select the Applications tab and click New.



4. Enter a unique name in the Application Name field.
5. Enter a meaningful description in the Description field.
6. Select the Active Flag checkbox, so the application package can be used.
7. Click Save to update the changes.

Now, the application package is created and you need to configure it, or at least its services.

Next, learn how to configure an application package.

View Application Packages

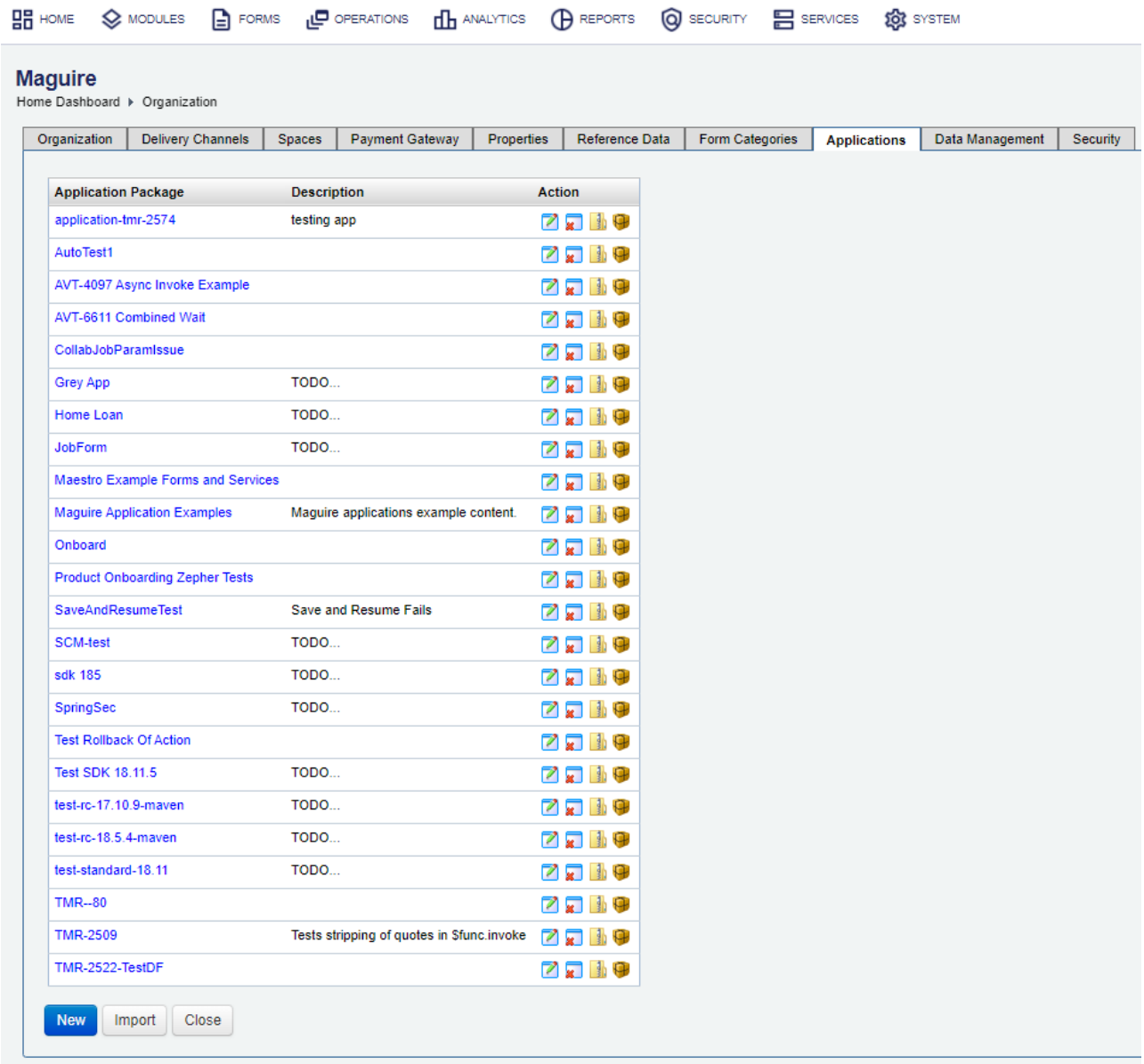
Manager comes with the concept of application packages that are archive files you can use to group organization specific forms, properties, reference data, services and service connections so that this configuration can be migrated between different Manager servers more easily.

To package a form, you need to create a new application package and assign the required services and forms to it. You can also assign the required organization properties, delivery channels, and service connections, where applicable. Manager allows you to view, create and configure application packages for the selected organization.

To view application packages:

1. Select Forms > Organizations.
2. Locate an organization.

3. Select the Applications tab to check details of all configured application packages.



The list displays the following details:

- Application Package: a name of an application package.
- Description: an application package description

You can also perform the following actions:

- Click Edit to update the existing application package.
- Click Remove Application Package to delete an application package.
- Click Export Archive (Current Form Versions) to export an application package including current form versions only.

- Click Export Archive (All Form Version) to export an application package including all form versions.
- Click New to create a new application package.
- Click Import to import an application package.

NOTE

You can also export and import application packages so you don't need to configure them every time in a new environment.

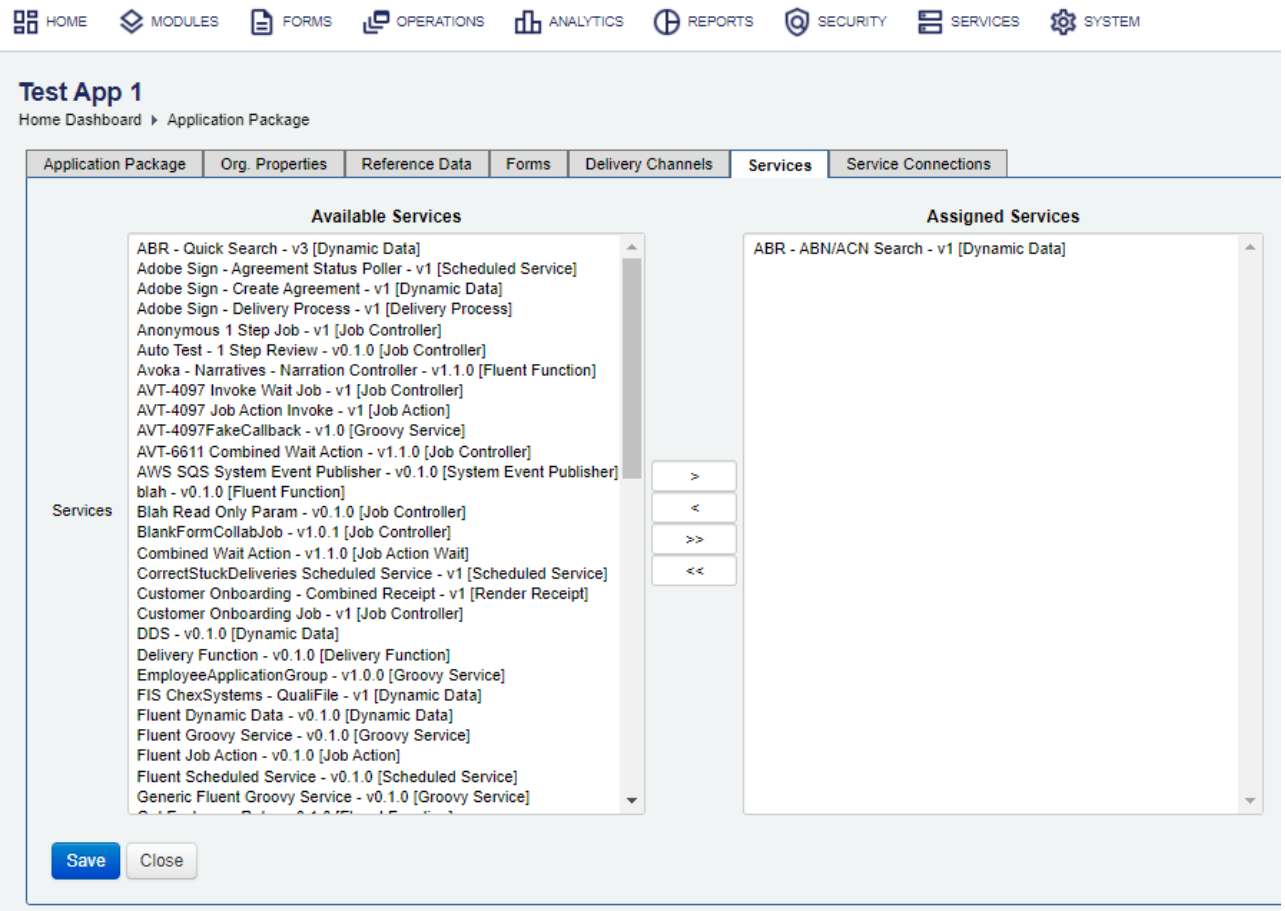
Next, learn how to create an application package.

Configure an Application Package

Manager allows you to configure application packages for the selected organization. You usually do this after creating a new application package.

To configure an application package:

1. Select Forms > Organizations.
2. Locate an organization and select the Applications tab.
3. Locate an application package and click Edit.
4. Assign the services from the Services tab to this application package.

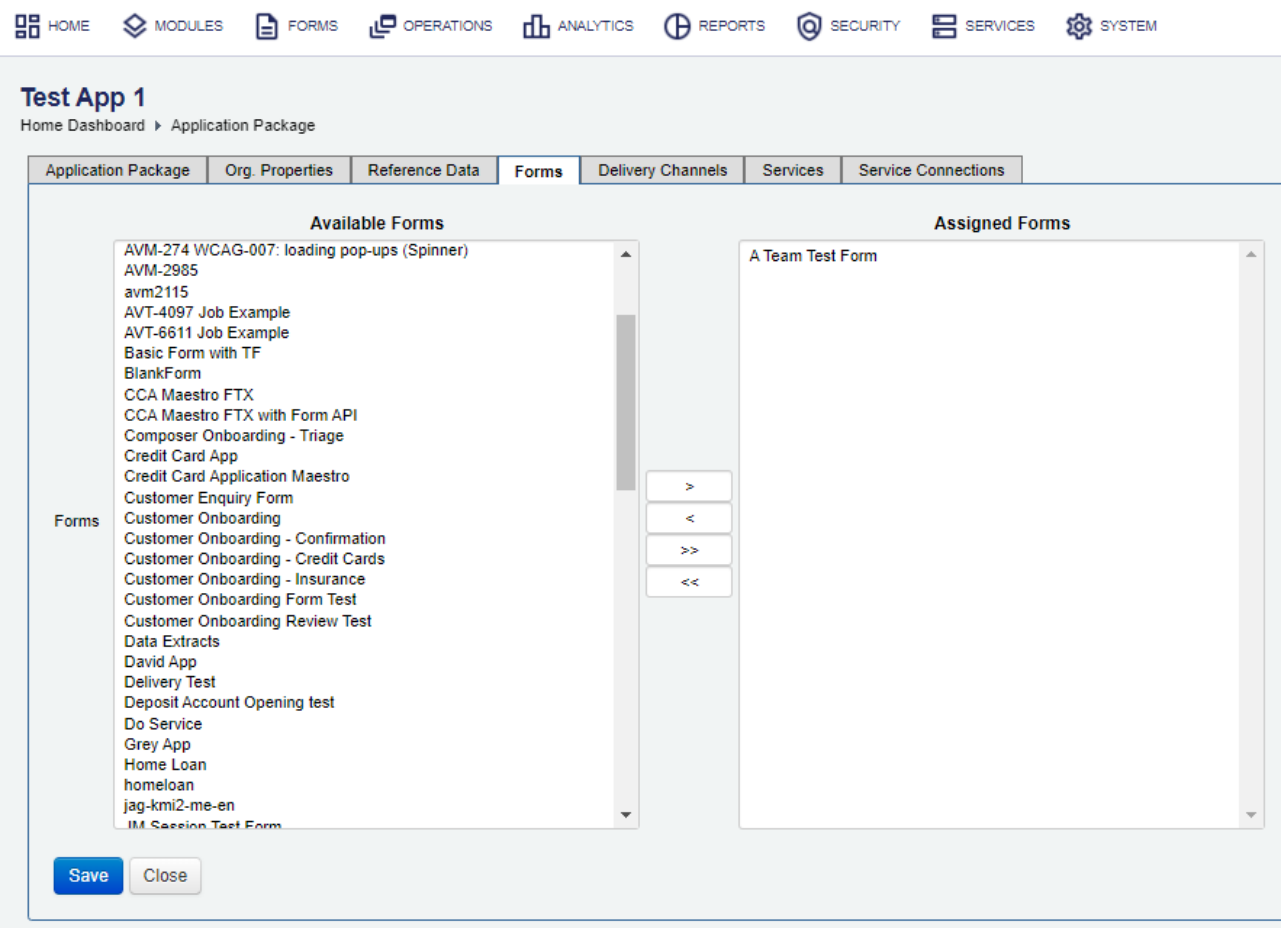


NOTE

You can't assign global services to an application package.

5. Select one or more services from Available Services that you want to assign to the application package. You can use the Shift and Ctrl keys to select multiple services on Windows PC.

6. Click > to move the selected services from Available Services to Assigned Services.
7. Select the services from Assigned Services and click <.
8. To assign all Services, click >>.
9. To revoke all Services, click <<.
10. Select the Forms tab to assign forms to the application package.



11. Select the Org. Properties tab to assign organization properties to the application package.
12. Select the Reference Data tab to assign reference data sets to the application package.
13. Select the Delivery Channel tab to assign delivery channels to the application package.
14. Select the Service Connections tab to assign service connections to the application package.
15. Click Save to update the changes.

NOTE

You can also export and import application packages so you don't need to configure them every time in a new environment.

Next, learn how to configure organization data retention.

Export an Application Package

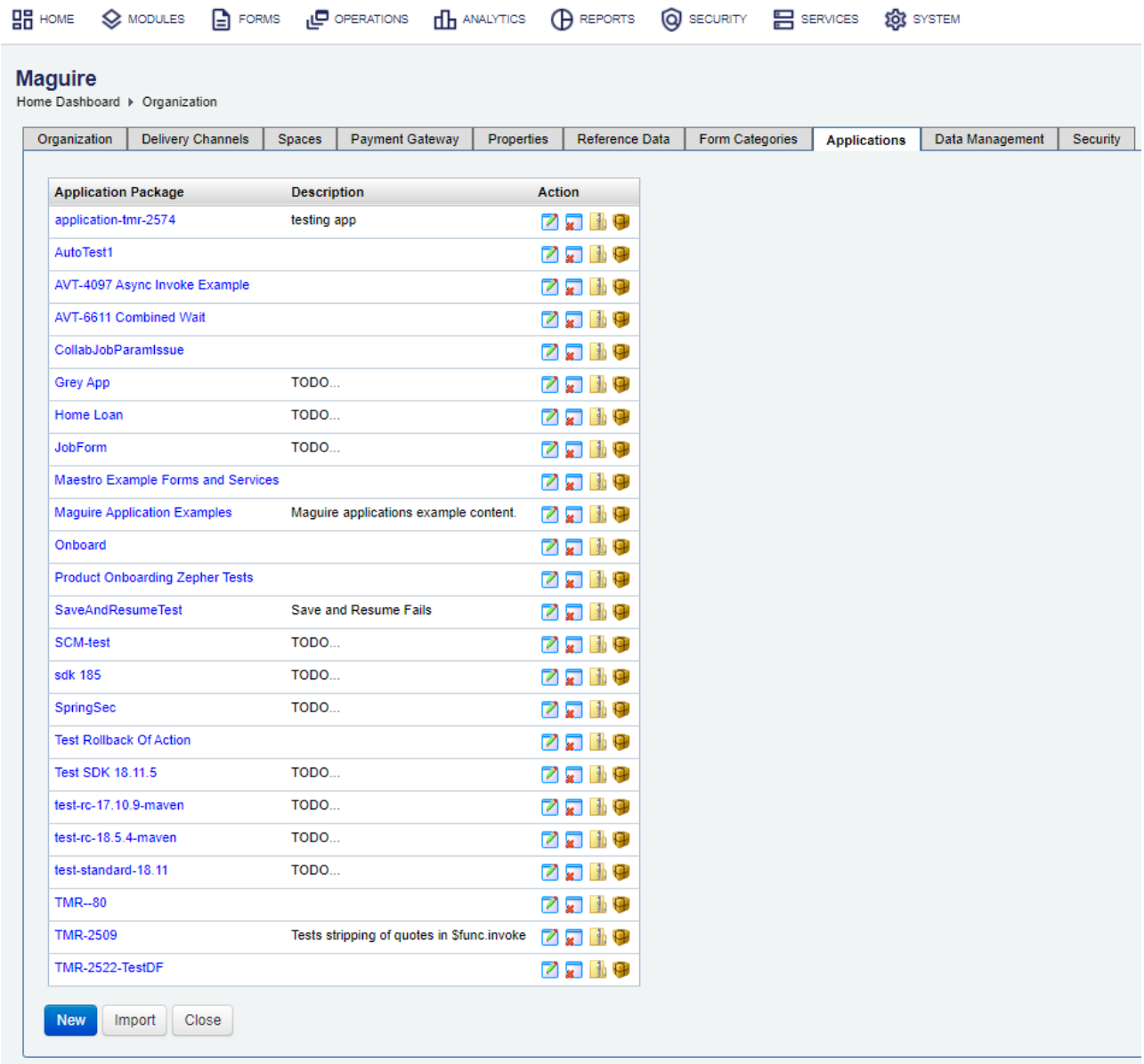
Manager allows you to export an application package from an organization. Application packages can be used to group organization specific services so that these services can be migrated between Manager servers more easily. You can use this functionality to:

- Backup or archive applications
- Re-create or migrate applications in another Manager
- Promote one or several applications to other environments

To export an application package:

1. Select Forms > Organizations.
2. Locate an organization that you want to configure or manage existing application packages.

3. Select the Applications tab.



4. Locate your application and click Export Archive (Current Form Versions) or Export Archive (All Form Versions). Manager exports them into an application archive ZIP file, which will be downloaded to your PC.

Next, learn how to [import an application package](#).

Import an Application Package

Manager allows you to import an application package into an organization. You can create and export an application package from one environment and then import it in another one. You can use this functionality to:

- Roll-back one or several applications.
- Re-create one or several applications in another Manager.
- Promote one or several applications to other environments.

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

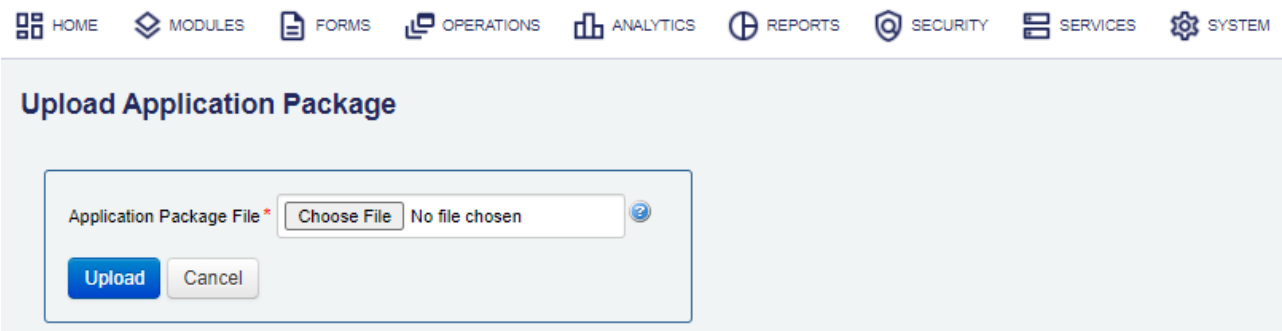
You can import an application package using one of following options:

- UI
- SDK
- API

Import Using UI

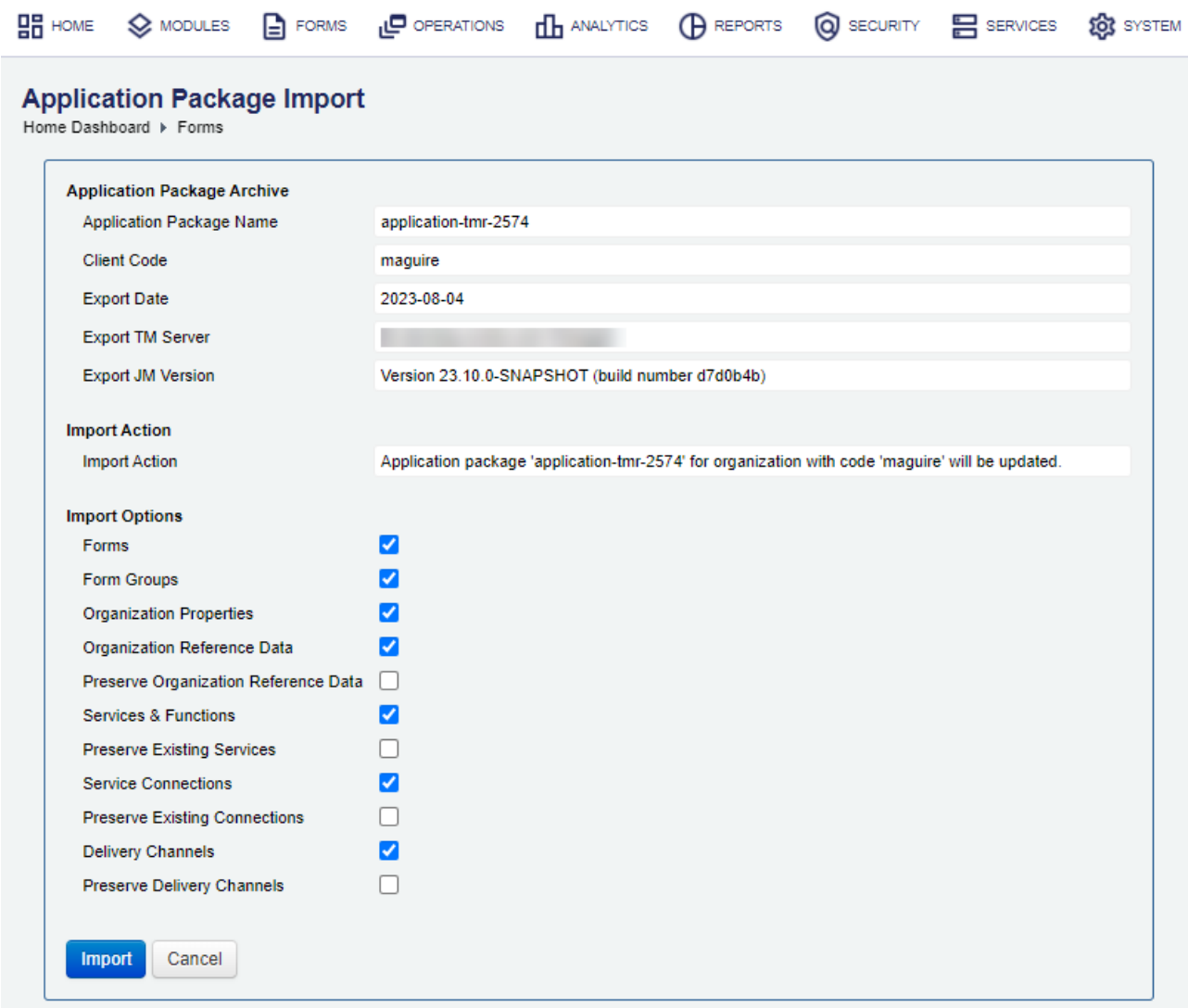
To import an application package:

1. Select Forms > Organizations.
2. Locate an organization that you want to configure or manage existing application packages.
3. Select the Applications tab.
4. Click Import.



5. Click Choose File and choose an application archive ZIP file.
6. Click Upload.

7. Review the Import Options as shown below.



8. Select Forms to import forms from the archive file into the target environment. The existing forms will be overwritten. This is the default option.
9. Select Form Groups to import form groups from the archive file into the target environment. The existing form groups will be overwritten. This is the default option.
10. Select Organization Properties to import organization properties from the archive file into the target environment. The existing organization properties will be overwritten. This is the default option.
11. Select the Organization Reference Data checkbox to import organization's reference data.
12. Select Services & Functions to import services and Transact Functions from the archive file into the target environment. The existing services and Transact Functions will be overwritten. This is the default option. For more information, see import services and import Transact Functions.

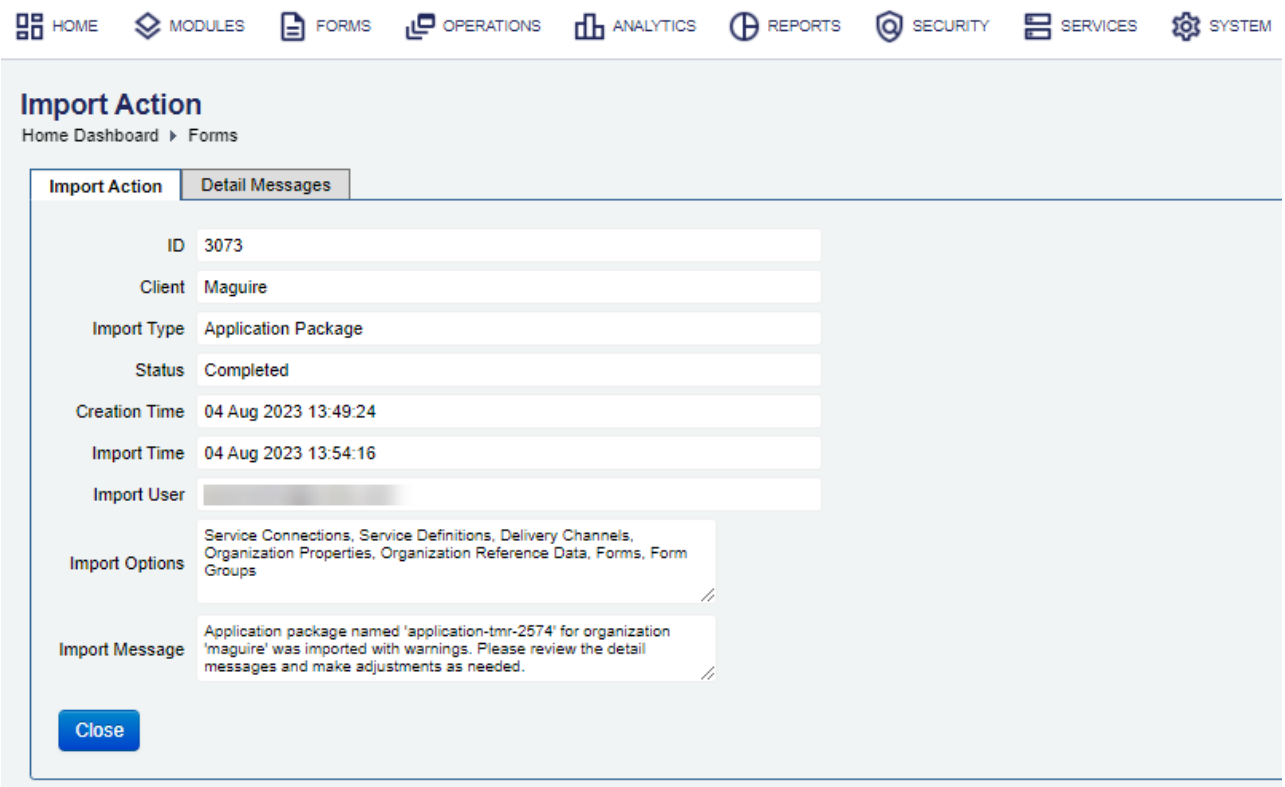
13. Select Preserve Existing Services, so any services that already exist in the target environment will not be modified in any way. Only new (to the target environment) services will be imported.

    NOTE

    If you import a new version of a service, clear the Preserve Existing Services, then manually activate this service version and update the configurations that refer to the old version of the service to point to the new version.

14. Select Service Connections to import service connections from the archive file into the target environment. The existing service connections will be overwritten. This is the default option. For more information, see import service connections.
15. Select the Preserve Existing Connections, so any service connections that already exist in the target environment will not be modified in any way. Only new (to the target environment) service connections will be created.
16. Select Delivery Channels to import delivery channels from the archive file into the target environment. The existing delivery channels will be overwritten. This is the default option.
17. Select Preserve Delivery Channels, so any delivery channels that already exist in the target environment will not be modified in any way. Only new (to the target environment) delivery channels will be created .

18. Click Import to complete the process. The expected result page is shown below.



19. Select the Detail Messages tab to see more information about the import. This tab pops up open if the import operation has any issues you may need to check and resolve.
20. Click Close.

NOTE

You can also check the Application Package REST API documentation for your Manager version.

Import Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:archive-deploy goal to import an application package.

Import Using API

The Journey Manager REST API provides the endpoint URL to import an application package.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s)://<JM_SERVER>/manager/secure/rest/archive/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as Base64[1].

`--form ‘archiveFile=@“/Users/my-application.zip”’` - a path to an archive file to be imported.

`--form ‘importOptions=“DELIVERY_CHANNELS,PRESERVE_EXISTING_DELIVERY_CHANNELS”’` - a comma-separated list of import options to perform. The option names are the same as described in the Import Using UI section but they use an underscore, for example, `DELIVERY_CHANNELS` for Delivery Channels.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

Example

```
curl --location --request PUT ‘https://<JM_SERVER>/manager/secure/rest/archive/v1/’
\
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
--form ‘archiveFile=@“/Users/my-application.zip”’ \
--form ‘importOptions=“DELIVERY_CHANNELS,PRESERVE_EXISTING_DELIVERY_CHANNELS,FORMS,
FORM_GROUPS, \
ORGANIZATION_PROPERTIES,ORGANIZATION_REFERENCE_DATA, PRESERVE_EXISTING_
ORGANIZATION_REFERENCE_DATA, \
SERVICE_CONNECTIONS,PRESERVE_EXISTING_SERVICE_CONNECTIONS,SERVICE_
DEFINITIONS,PRESERVE_EXISTING_SERVICES”’
```

Next, learn how to configure organization's security.

[1]Base64 is a binary to a text encoding scheme that represents binary data in an American Standard Code for Information Interchange (ASCII) string format.

Configure Organization Security

Manager allows you to configuring an organization security to allow or refuse form rendering on certain browsers or form requests with certain headers. Manager has a black list of older IE browsers that are not supported. Otherwise, it does a white list, so new browsers (meaning new mobile devices) may come into the market and they are not blocked automatically.

NOTE

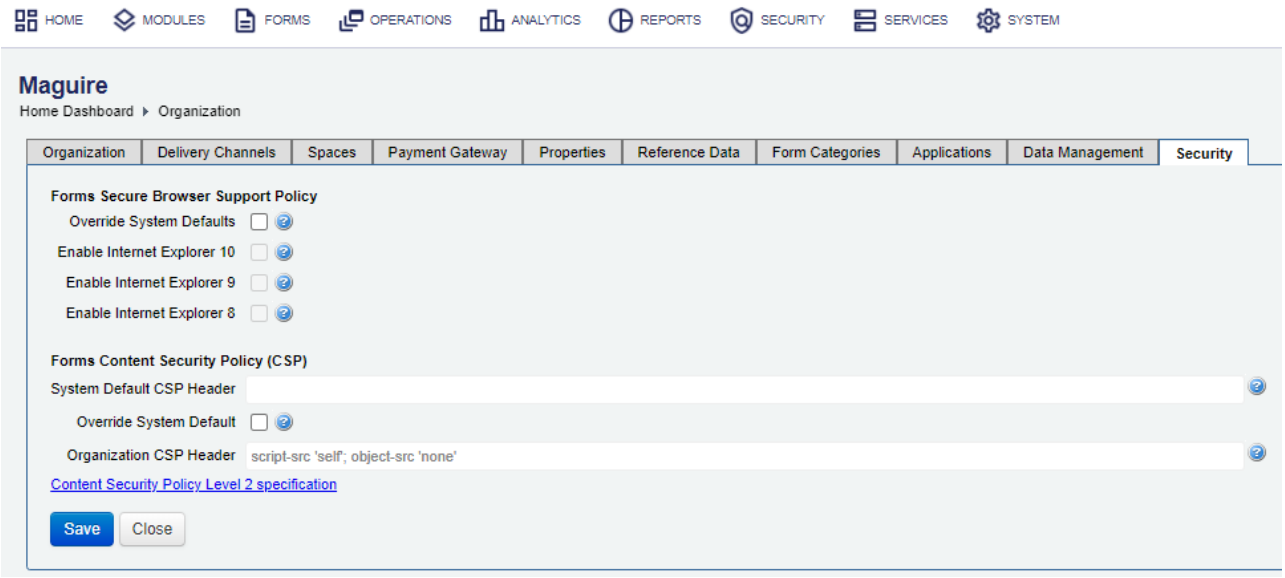
The system settings are defined in the default Form Submission Access Controller service.

NOTE

You can configure organization's security settings using the Configuration Service.
| 24.04 This feature was introduced in 24.04

To configure organization security:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Security tab to override browser's support policies, which will be used by all forms.



4. Select the Override System Defaults checkbox to override the system default supported secure browser configurations.

5. Select the Enable Internet Explorer 10 checkbox to enable the Microsoft Internet Explorer 10 (IE10) browser.
6. Select the Enable Internet Explorer 9 checkbox to enable the Microsoft Internet Explorer 9 (IE9) browser.
7. Select the Enable Internet Explorer 8 checkbox to enable the Microsoft Internet Explorer 8 (IE8) browser.

   NOTE

   Microsoft IE8, 9 and 10 are not supported by Microsoft and Manager for security reasons.

8. Provide the form CSP security HTTP header value used if not overridden by the organization CSP header value in the System Default CSP Header field.
9. Select the Override System Default checkbox to override the system default Content Security Policy (CSP) configuration.
10. Provide the form CSP security HTTP header value in the Organization CSP Header field. Any form space property CDN Server URL value will be automatically injected into `script-src-self` value.

    NOTE

    Check the Form CSP Header Value of the Form Submission Access Controller Service and the CORS Headers setting of your form space to ensure the correct CSP is used.

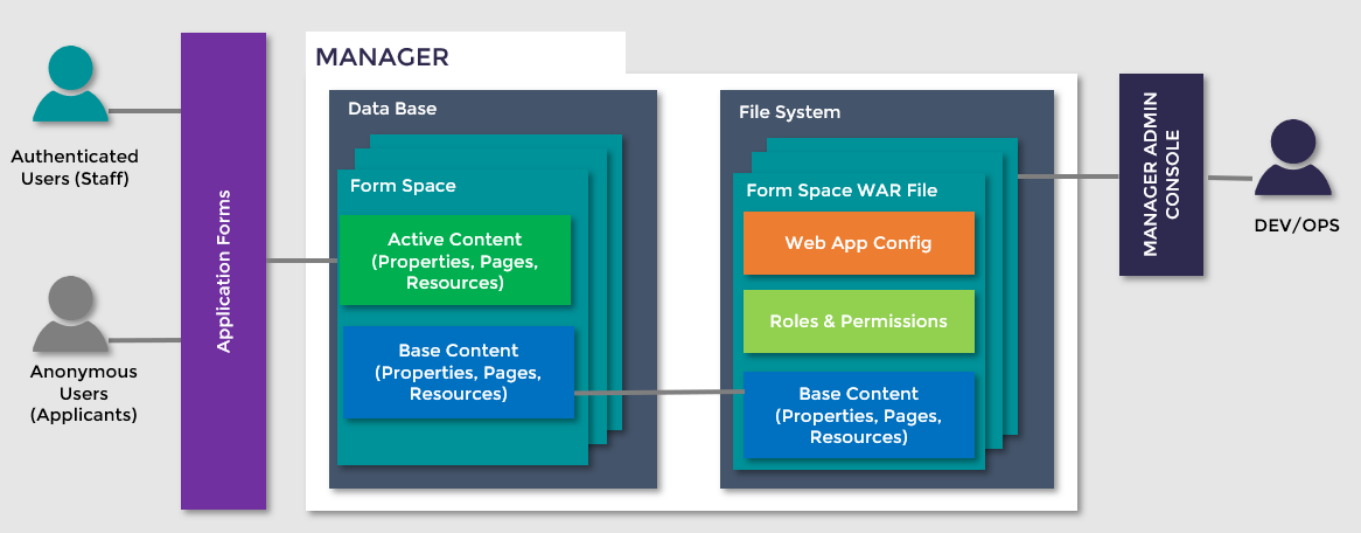
11. Click Save to update the changes.

Next, learn how to view organizations.

Form Spaces Overview

Manager provides form spaces that are used to host forms and expose them to users, handle form requests, such as form submissions, list tasks assigned to users, and perform user authentication. You can configure a form space to define whether hosted forms require authentication or not by selecting a security manager and authentication mode.

In Java Enterprise Edition (JEE) terms, a form space is a Java Web Application, which is deployed using a Web Application Archive (WAR) file. Unlike traditional JEE applications, Manager's form spaces WAR files are used to initialize the Manager database with appropriate configurations and HTML content. The Java application code is provided by the application server modules and the HTML content is rendered from the Manager database. We use this model to make application upgrades easier. J2EE Application Server restrictions mean that you are always required to deploy a WAR file to apply new changes. After that, the WAR files content becomes the base content in the database. Then, the active content in a form space is either edited in Manager or imported as an archive ZIP file. The active HTML content is served from the database.



NOTE

The form space name follows this pattern: `avoka-sf-portal-<portal context>.war`.

Every form in the Temenos Journey Manager platform is associated with one or more configured form spaces. If a form is associated with multiple form spaces, the form and all its saved and completed submissions will be visible and available to all of those spaces.

A form space can be licensed for internal and external transactions depending on the customer license agreement. You can generate transaction licensing reports to view internal and external transaction trends.

To create a new form space, you can import it as a WAR[1] file. You can then promote it from one environment to another.

Structure of Form Space WAR File

A form space WAR file includes configurations and HTML content only. It doesn't include any application source code. The typical form space WAR file contains:

Page Templates as HTML files, which are stored in:

- Unsecured root directory, which can be accessed anonymously.
- Secure directory, which requires an authentication user session. This directory contains the pages users sees on the form space after logging in, for example, the Tasks or To Do page.

Resources that are used to style a form space, including:

- css
- images
- includes: HTML snippets
- js: JavaScript files

WEB-INF that contains the list of XML files and the classes. The most important files are:

- web.xml: The web application deployment descriptor file describes how the web application should be deployed to the application server. It is required for every web application. This file has the context path space name substitutions applied by the installer.
- jboss-web.xml: This file has the context path space name substitutions applied by the installer and the context path specified in the installer has been written into the `context-root` element.

---

[1]WAR file (Web Application Resource or Web application ARchive) is a file used to distribute a collection of JAR-files, JavaServer Pages, Java Servlets, Java classes, XML files, tag libraries, static web pages (HTML and related files) and other resources that together constitute a web application.

WEB-INF/classes contains the compile Java packages and classes. The most important files are:

- db-config.xml: It is used to create all the new form space properties when the WAR file is deployed. The Roles for the form space are also defined here. This file has the context path space name substitutions applied by the installer.

Default Form Spaces

Manager comes with the following default form spaces, which you can opt in or out during Manager installation:

Salesforce
: Salesforce is the form space to enable integration with Salesforce.

TransactField App
: TransactField App is the business field work personnel form space that allows working offline. It provides a mobile app and PC application which enables business field staff to complete forms offline. Once online, the application allows them to sync their work with the Manager server, and receive new work tasks to complete. TransactField App supports authenticated users only. | Deprecated in 21.11 This feature was deprecated in 21.11.

Web Plug-in
: Web Plug-in is the default form space for forms to be discovered and linked to public web sites. It provides a form space for customer facing sales and service transactions. It is linked from your organization's website, but can also be embedded on your website if the form is fairly small. Web Plugin supports unauthenticated and authenticated users. Maestro requires this form space for building and publishing form versions.

Journey Space

> WARNING
>
> The Work Space form space has been renamed to Journey Space, so there is no confusion with the Workspaces portal. The Context Path URL has also been changed from `https://my-domain.com/workspace/` to `https://my-domain.com/journeyspace/`.
> For the changes to take effect, you need to have a fresh installation of Manager or the Journey Space form space. If you only upgrade Manager, you still have the old Work Space name with the old Context Path URL as `https://my-domain.com/workspace/`. | 23.10This feature was updated in 23.10.

Journey Space allows business staff to perform work functions as a group by creating tasks, collaboration jobs, completing ad-hoc forms, and providing help desk support. The work space is generally used by authenticated users but it also supports unauthenticated transactions as well.

> NOTE
>
> You should use Workspaces instead of Journey Space.

Pre-configured Form Spaces

Additionally, you can install the following pre-configured form space.

Workspaces
: Workspaces is the recommended form space for the business staff to perform work group functions. | 18.11 This feature was introduced in 18.11.

Maguire
: Maguire is the example of self-service form space. | 18.05 This feature was deprecated in 18.05.

> NOTE
>
> If a form space becomes deprecated, you may need to remove it manually from your Manager instance as it will not work.

Manager comes with a number of modules, which configuration might look similar to form spaces. However, modules are very different to form spaces - you can think of them as the separate products deployed in Manager.

Next, learn how to view form spaces.

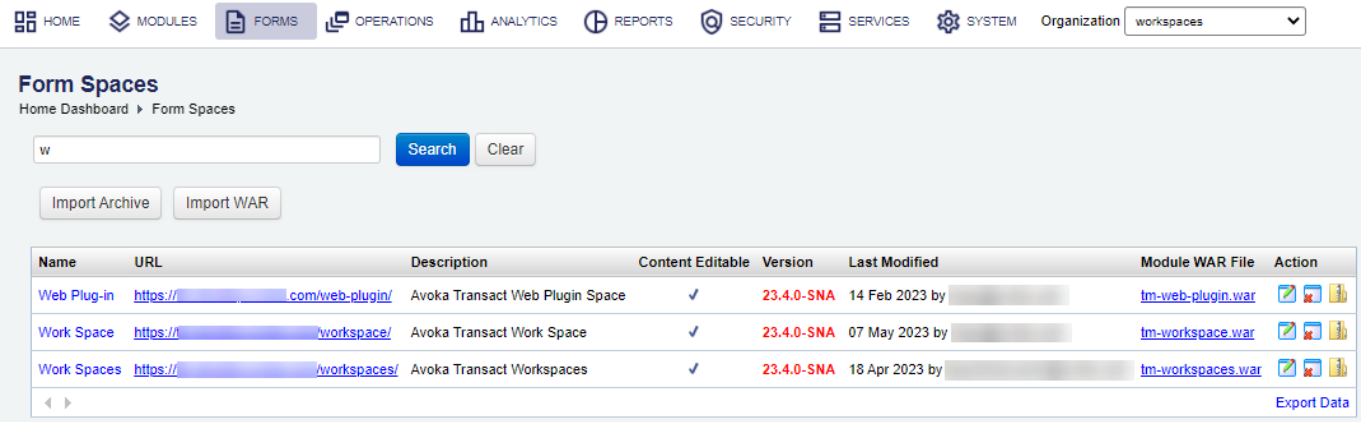
View Form Spaces

Manager allows you to view available form spaces, so you can manage them.

> NOTE
>
> You must be an administrator to perform some of these operations.

To view form spaces, select Forms > Form Spaces.

Form spaces are shown page by page, with the maximum number of records per page configured in the users preferences.



The list displays the following details:

- Name: a form space's name. Click a link to configure a selected form space.
- URL: a web application's fully qualified domain name[1] where forms are hosted and can be accessed by users. Click a link to open a web application hosted under this URL.
- Description: a meaningful description of the form space.
- Content Editable: whether an administrator can edit all fields in form space's content at run-time or not.

[1]Fully qualified domain name (FQDN), sometimes also referred to as an absolute domain name, is a domain name that specifies its exact location in the tree hierarchy of the Domain Name System (DNS). It specifies all domain levels, including at least a second-level domain and a top-level domain.

- Version: a version of a form space.

  NOTE

  The version is highlighted in red if:
  - A form space version is different from that of a Manager server's where the form space is deployed.
  - A form space WAR file needs to be rebuilt using the Manager installer.

- Last Modified: a date when a form space was last modified and a user account who modified it.
- Module WAR File: a name of a WAR file deployed as the form space. Click a link to download the file.

To search for specific form spaces, specify a name or a part of the name and click Search.

Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

NOTE

A red version number indicates that the form space version doesn't match the Manager version, so you might need to rebuild the form space.

You can also perform the following actions:

- Configure a form space.
- Remove a form space.

  WARNING

  If you remove a form space, any submission associated with it won't be accessible and won't render.

- Export a form space.
- Import a form space.

Next, learn how to configure a form space.

Create a Form Space

Manager allows administrators to create form spaces using the Journey Manager installer, which also performs string substitutions on the context path and form space name into the portal files. Each WAR file must have a unique context path per Manager environment, such as DEV, UAT, or PROD.

> NOTE
>
> It's not recommended to create a copy of an existing form space WAR file as it can result in deployment issues, if all the required substitutions are not done. If the form space copy has resulted in two WAR files with the same context, the J2EE application server will only deploy one WAR file.

To create a form space:

1. Run the Journey Manager installer - `Avoka-TM-Setup.jar` file.
2. Select Create a Transact Work Space and click Next.
3. Provide a unique form space name in the Space Name field, which Manager uses as the primary key when the form space is imported.

   > NOTE
   >
   > Do not use the ampersand (&) character in the form space name.

4. Type a unique context path in the Context Path field, which is a part of the URL as per the following pattern: `https://{Server Domain Name}/{context path}`. It's the base path to access the Web Application Archive WAR file. If the URL is `https://creativewave.transactcentral.com/cwf`, the context path will be `cwf`. In this case, the Manager URL will be `https://creativewave.transactcentral.com/manager`.

   > NOTE
   >
   > The context path must be unique per environment. Do not use the ampersand (&) character in the context path.

5. Provide a meaningful description in the Description field and click Next.
6. Select the Production servers MUST use secure cookies checkbox. It is selected by default. You can clear this setting for a test form space to run on your own PC. Do not configure test form spaces for clients.

7. Select the Groovy Services checkbox. It is selected by default. You can clear this setting for a test form space to run on your own PC.
8. Click Next.
9. Specify a directory where the installer creates a form space in the Installation Directory field. Default is the `C:\avoka-tm-custom-work-space` directory on a Windows PC.

   > NOTE
   >
   > If you are creating several form spaces, it's recommended to create a unique installation directory, for example, C:\avoka-tm-custom-work-space-cwf.

10. Click Next.
11. Click Install to create the form space WAR file in the specified directory.
12. Locate the form space WAR file in the `deploy` directory created in your installation directory, for example, `C:\avoka-tm-custom-work-space-cwf\deploy\avoke-sf-portal-cwf.war`.
13. Open the `readmy.html` file, located in the installation directory, for example, C:\avoka-tm-custom-work-space-cwf\readme.html, to check how to use the form space.
14. Import the form space into Manager.

Next, learn how to configure form space properties.

Configure a Form Space

Manager allows you to configure form spaces according to your specific requirements, including 2 Factor Authentication[1] that significantly improves a form space security.

NOTE

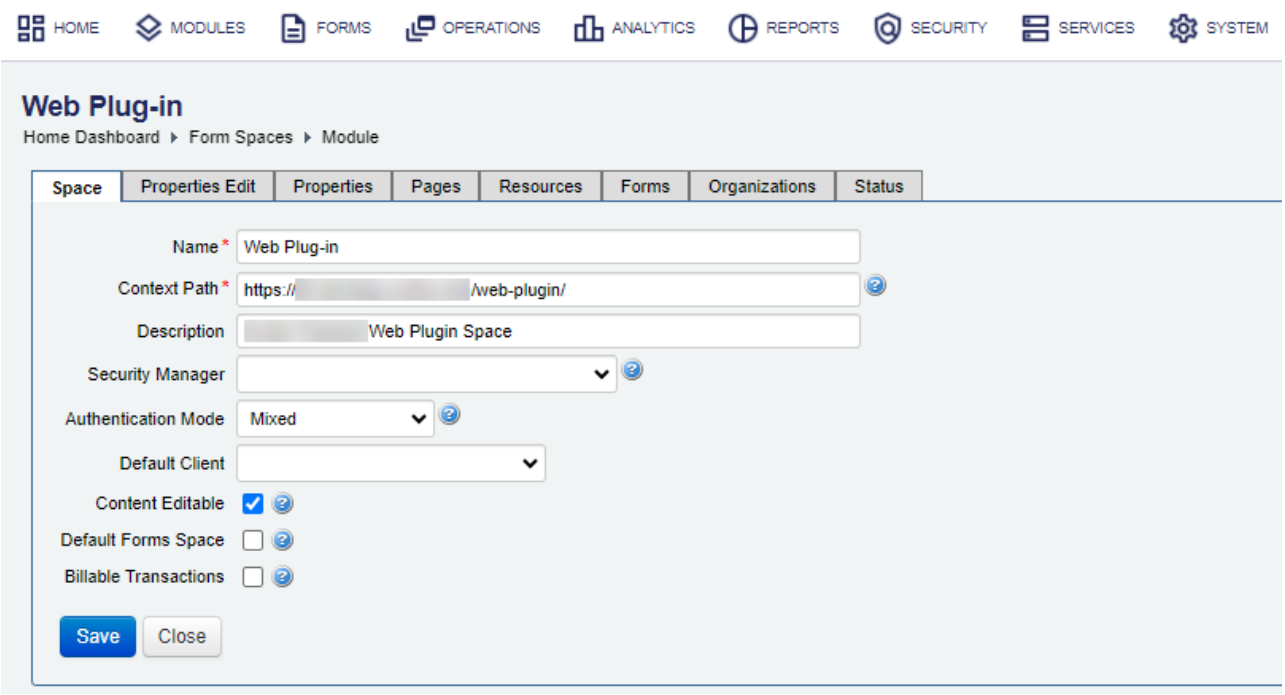
You must have an administrator role assigned to your account to perform this configuration.

To configure a form space:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.

[1]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.

3. Select the Space tab.



4. Edit the name of the form space in the Name field.
5. Provide the web application's fully qualified domain name[1] in the Context Path field, using the following format: `http://hostname.com/contextPath/servletPath/pathInfo`, for example, `https://mydomain.com/feedback/`. This is where forms are hosted and can be accessed by users.

> NOTE
>
> When a Manager server starts up, a context path is created for each deployed form space unless an entry with the same form space name or context path already exists.

[1]Fully qualified domain name (FQDN), sometimes also referred to as an absolute domain name, is a domain name that specifies its exact location in the tree hierarchy of the Domain Name System (DNS). It specifies all domain levels, including at least a second-level domain and a top-level domain.

6. Change the context path to allow Apache[1] mappings. This should be the URL where the form space is accessible to users. This context path is used during a form submission. If the URL does not match the form space context path (including the port number), the submission will not be accepted.

```
ProxyPassMatch ^/cwf(.*)$ ajp://mydomain.com:9009/feedback$1
ProxyPassReverse /cwf ajp://mydomain.com:9009/feedback
```

7. Provide a meaningful description of the form space in the Description field.
8. Select a security manager[2] to authenticate user login from the Security Manager dropdown list. Different security managers allow you to implement various user authentication flows, for example, 2 Factor Authentication[3].

> NOTE
>
> You can leave this field blank but assign this form space to a required security manager. For more information, see the Configure a Security Manager and Assign Form Spaces to a Security Manager chapters of security managers documentation.

---

[1]Apache HTTP Server is free and open-source cross-platform web server software, released under the terms of Apache License 2.0. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.

[2]Security managers are the Journey Manager's components that are responsible for User Authentication, User Authorization, User Self-Registration, and Session Management. Security mangers are designed to handle various sets of users requiring access to different form spaces and can be associated with one or more form spaces. Users can be external or internal, and they can require access to form spaces or a combination of form spaces and modules in different environments. You can find security managers under Security > Security Managers.

[3]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.

9. Select the level of authentication that is required to access the form space from the Authentication Mode dropdown list, which can be one of the following:
   - Mixed - both authenticated and anonymous users can access forms in the form space.
   - Authenticated Only - only authenticated users can access forms in the form space. An authenticated user is one that has entered a username and password that has been accepted by the system.
   - Anonymous Only - only anonymous users can access forms in the space. An anonymous user is one that has not provided a username and password to access the form.

   NOTE

   Forms configured with a different authentication mode to the authentication configured in the form space will be highlighted with a warning message.

10. Select an organization from the Default Client dropdown list, which is used to populate transactions of global forms served through this form space. | 23.10 This feature was introduced in 23.10
11. Select the Content Editable checkbox to allow an administrator to edit all fields in form space's content at run-time. Otherwise, fields are read-only and form space content can't be changed and the Pages and Resources tabs are not shown.

    NOTE

    Changes made to the Content Editable checkbox can only be applied once you clicked the Save button.

12. Select the Default Form Space checkbox to make this form space the default space to render forms.

    NOTE

    You can use this setting for testing forms in the Manager console.

13. Select the Billable Transactions checkbox to specify whether this form space is licensed for Billable transactions as well as Non-Billable transactions. If not selected, this form space is licensed for non-billable transactions only. For more information, see transaction licensing report.

NOTE

Prior to version 22.10, this option was called Internal Transactions where you could specify whether this form space is licensed for Internal (business users) transactions as well as External (customer) transactions. If not selected, this form space is licensed for external transactions only..

NOTE

You can change the type of licensing transactions of each form. | 18.11 This feature was introduced in 18.11.

14. Click Save to update the changes.

NOTE

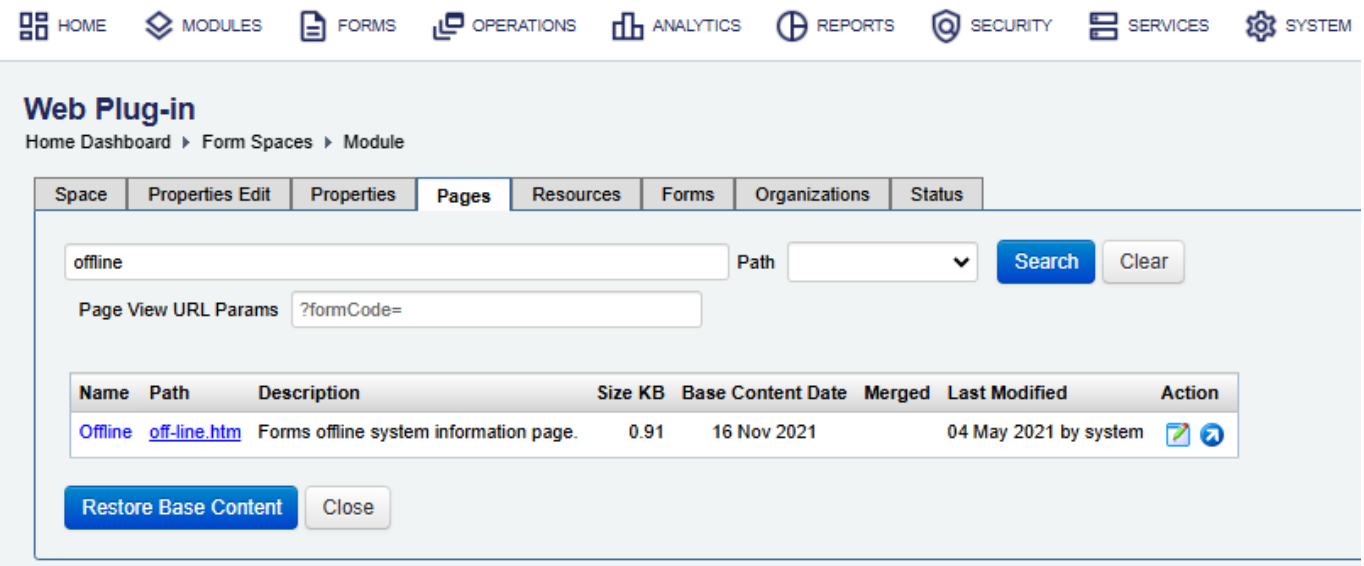
You can configure form space's properties in the Properties Edit tab.

Next, learn how to configure form space properties.

Configure a Form Space Status

Manager allows you to restrict access to a form space by changing its status. You can use this functionality to disable a form space when it's not required any more.

All requests made to any content of an unavailable form space are redirected to the Offline space page as shown below.



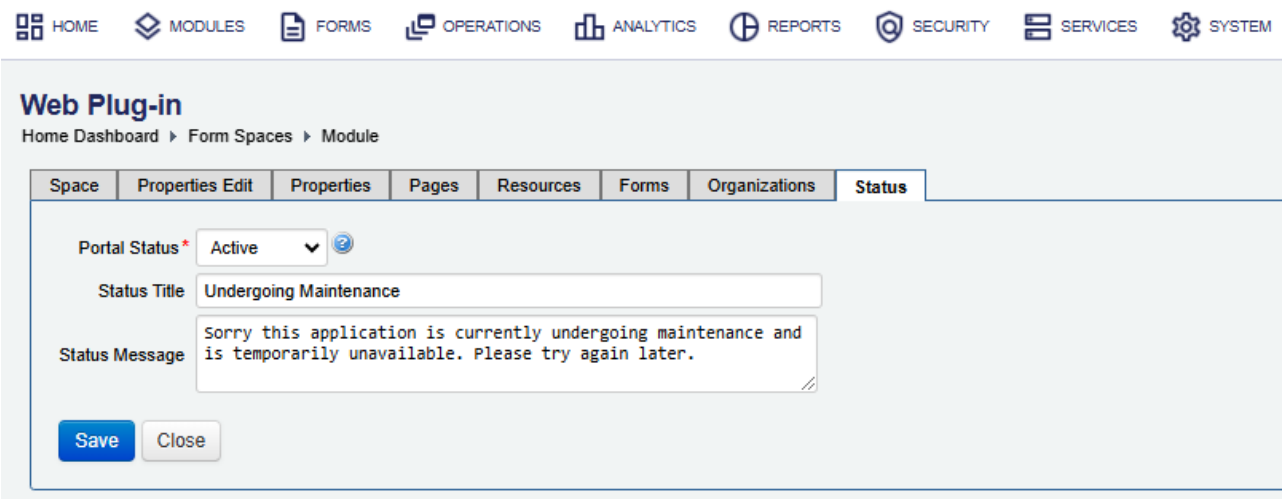
NOTE

Since Manager 22.04, you don't need to do this form space status change manually on a day you want the status to change. Instead, you can schedule the status change using Notifications and Status Change. This is handy for automated deployments and maintenance tasks. Additionally, you can change the status of forms and form versions, which gives you extra flexibility and also reduce the scope of your application unavailability, as compared to applying the status change to the whole form space.

To configure the form space status:

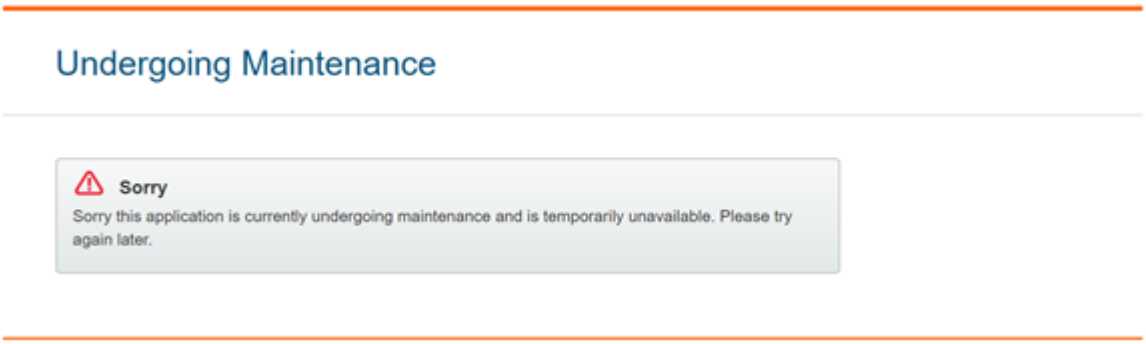
1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.

3. Select the Status tab.



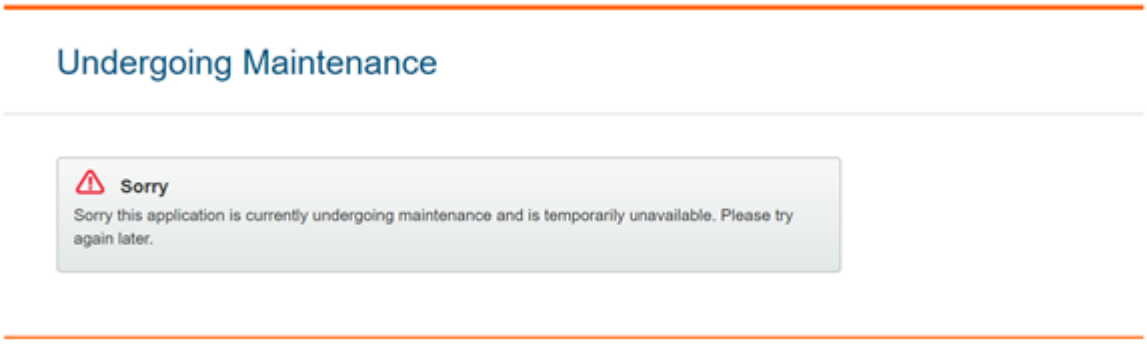
4. Select a status of the form space from the Portal Status dropdown list, which can be one of the following: | 21.11 This feature was introduced in 21.11.
   - Active: it is in use and readily available to all users. All functionality is allowed.
   - Offline: it can't be used at all. No functionality is available, including DDS, Delivery and so on.
   - Inactive: similar to Offline.
   - NoRender: it doesn't allow for new or saved and resumed forms, but ongoing open sessions can be completed. DDS, submits, saves, and delivery are allowed.

   The example of an offline form space is shown below.



5. Edit a page heading that is shown on the page in the Status Title field.
6. Edit a message that is displayed on the page in the Status Message field. This is only relevant when the space is in offline status. Each status can have a message configured to explain why the form space is restricted. Once the form space is back to an Active status, all content is accessible.

The example of an offline form space message is shown below.



NOTE

Form requests can be better handled by setting up a status message specific to the form. To do that, add an Offline Status Message property to the form. This message overrides the status message when form space users attempt to access a page specific to the form.

7. Click Save to update the changes.

Next, learn how to export a form space.

Assign Organizations to a Form Space

Manager allows you to assign organizations to a form space. All forms that will be accessible to users must be associated with at least one form space. The form spaces available to a form are determined by the form spaces associated with the organization.

The Web Plug-In space is the default space, used for building form versions from Maestro. It is required for all users and organizations working with Maestro forms. If the Web Plug-In form space is not assigned, you will receive an error message when building the form version.

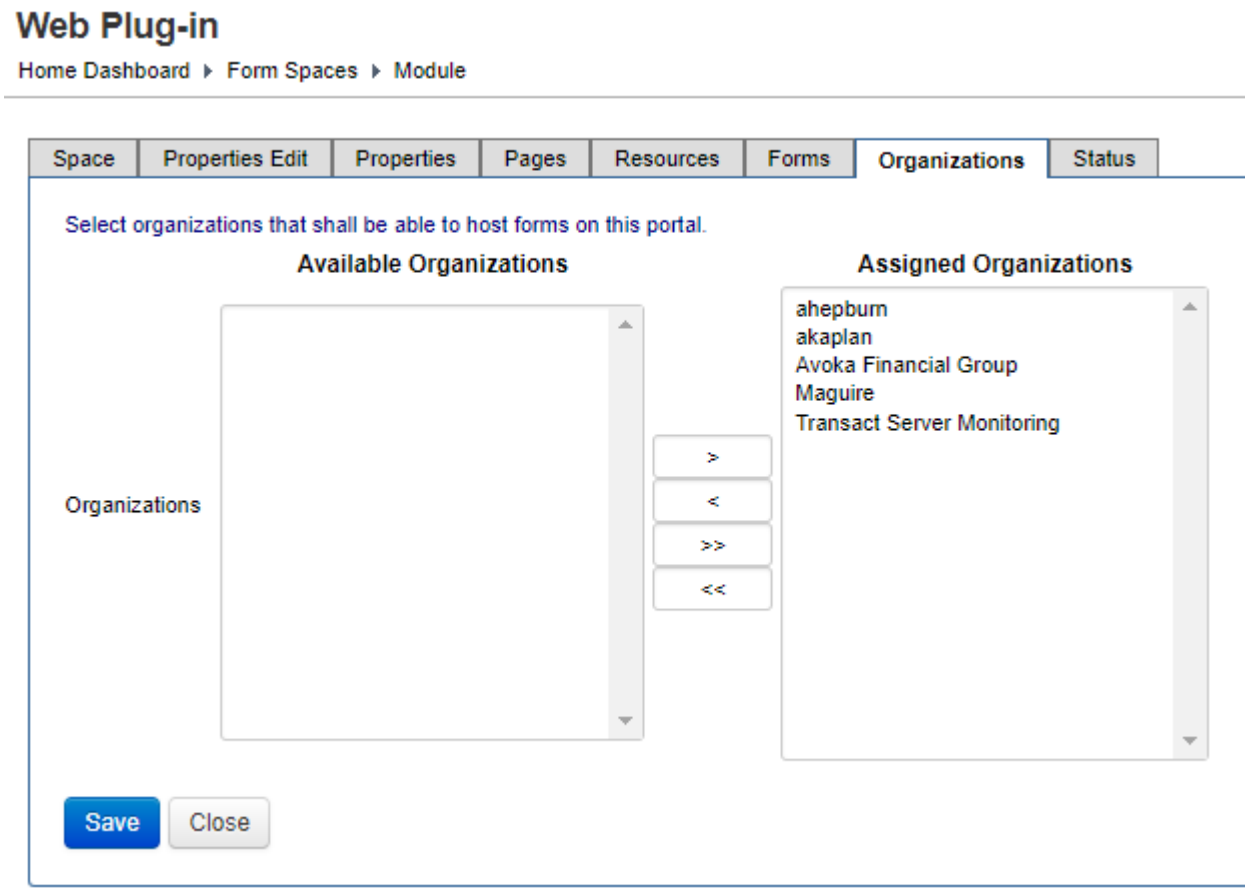
To assign organizations to a selected form space:

1. Select Forms > Form Spaces.
2. Locate a form space and click the Edit.

   NOTE

   The form space must be editable.

3. Select the Organizations tab to see all assigned organizations.



4. Select one or more organizations from the Available Organizations that you want to assign to a form space
5. Click the > to move the selected organization from the Available Organizations to the Assigned Organizations.
6. Click the < to move the selected organization from the Assigned Organizations to the Available Organizations.
7. Click the >> to assign all organizations.
8. Click the << to revoke all organizations.
9. Click Save to update the changes.

NOTE

You can also assign form spaces to an organization.

NOTE

Associating form spaces with organizations may require changes to your currently configured system when using Maestro 5.1 and higher.

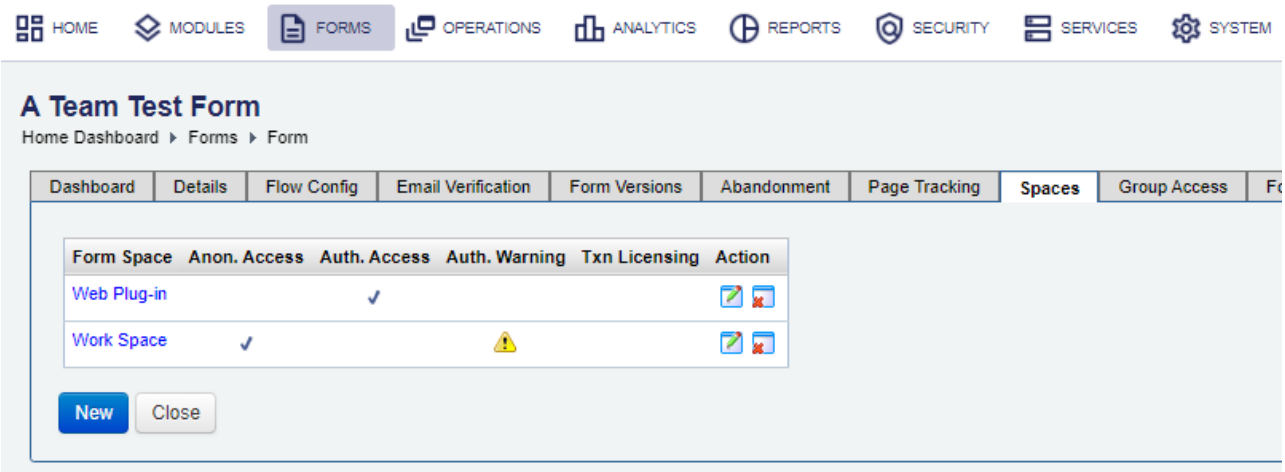
Next, learn how to assign form spaces.

Assign Form Spaces to a Form

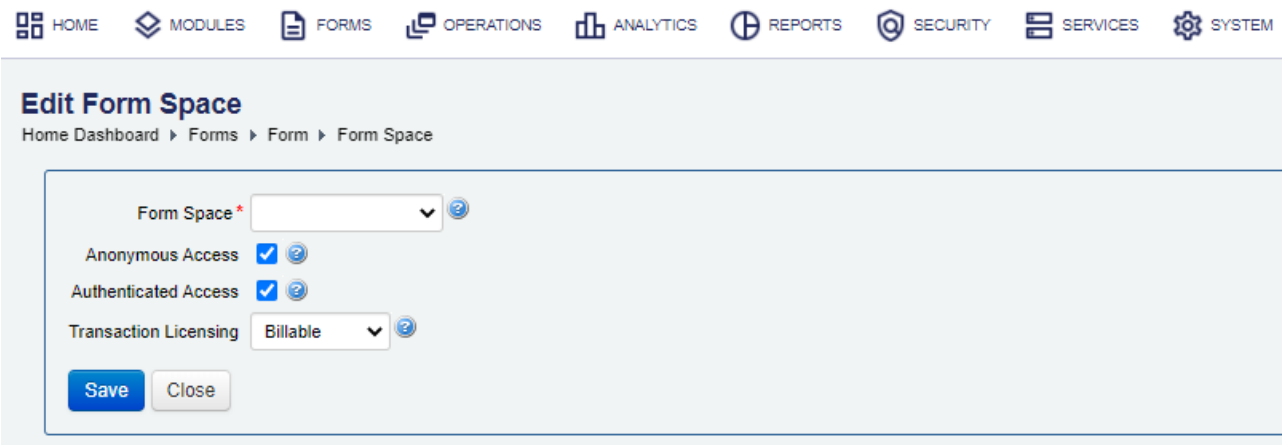
Manager allows you to assign form spaces to a form and configure user access to the form.

To assign a form space to a form:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select the Spaces tab to see the current form spaces assigned to this form and their configuration.



4. Click New to assign a form space.



5. Select a form space from the Form Space dropdown list to assign the form to.
6. Select the Anonymous Access checkbox to allow anonymous users to access the form.
7. Select the Authenticated Access checkbox to allow only authenticated users to access the form.

NOTE

You can select both the Anonymous Access and Authenticated Access. However, if the access authorized at the form space level is different to that one set at a form level, a warning message will be shown in the Auth Warning column for the selected space.

8. Select a transaction licensing type from the Transaction Licensing dropdown list, which can be one of the following: | 22.10 This feature was introduced in 22.10.
   - Billable - billable transactions
   - Non-Billable - non-billable transactions
   - Blank - to use the form space's Non-Billable licensing configuration. | 18.11 This feature was introduced in 18.11.
9. Click Save to update the changes.

Next, learn how to assign organizations to a form space.

Remove a Form Space

Manager allows you to remove a form space when you don't need it any more, for example, when a form space is deprecated. However, we recommend disabling unused form spaces rather than removing them.

> WARNING
>
> If you remove a form space, any submission associated with it won't be accessible and won't render.

To remove a form space:

1. Select Forms > Form Spaces.
2. Locate a form space.
3. Find all forms associated with this form space and assign them to a different form space to avoid any form rendering issues.
4. Click Remove.
5. Click OK to confirm it.

To learn how to create a new form space, see Create a Form Space.

Next, learn how view form space.

Export Form Spaces

Manager allows you to export form spaces as ZIP or WAR archive files. The main difference between these exports is that a ZIP archive contains only configurations related to a specific form, whilst a WAR archive contains all form space configurations. You can use the export form space functionality to:

- Backup or archive form spaces.
- Re-create or migrate form spaces in another Manager.

To export a form space as a ZIP archive:

1. Select Forms > Form Spaces.
2. Click Export Archive to export the form space as a form space archive ZIP file, which will be downloaded to your PC.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to import form spaces.

Import Form Spaces

Manager allows you to import form spaces as ZIP or WAR archive files. The main difference between the imports is a ZIP archive contains only form's pages related configuration, whilst a WAR archive contains the whole form space configuration. You can use the import form spaces functionality to:

- Migrate one or several form spaces.
- Re-create one or several form spaces in another Manager.
- Promote one or several form spaces to other environments.

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

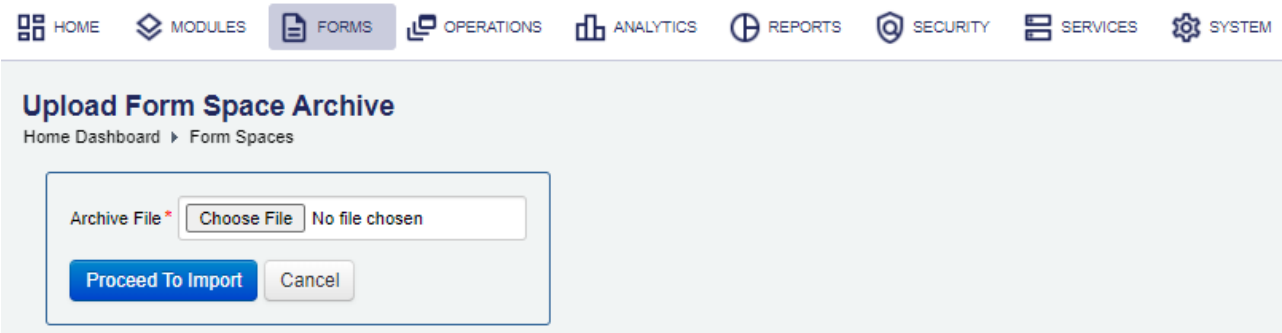
You can import a form space using one of following options:

- UI
- SDK
- API

Import Using UI

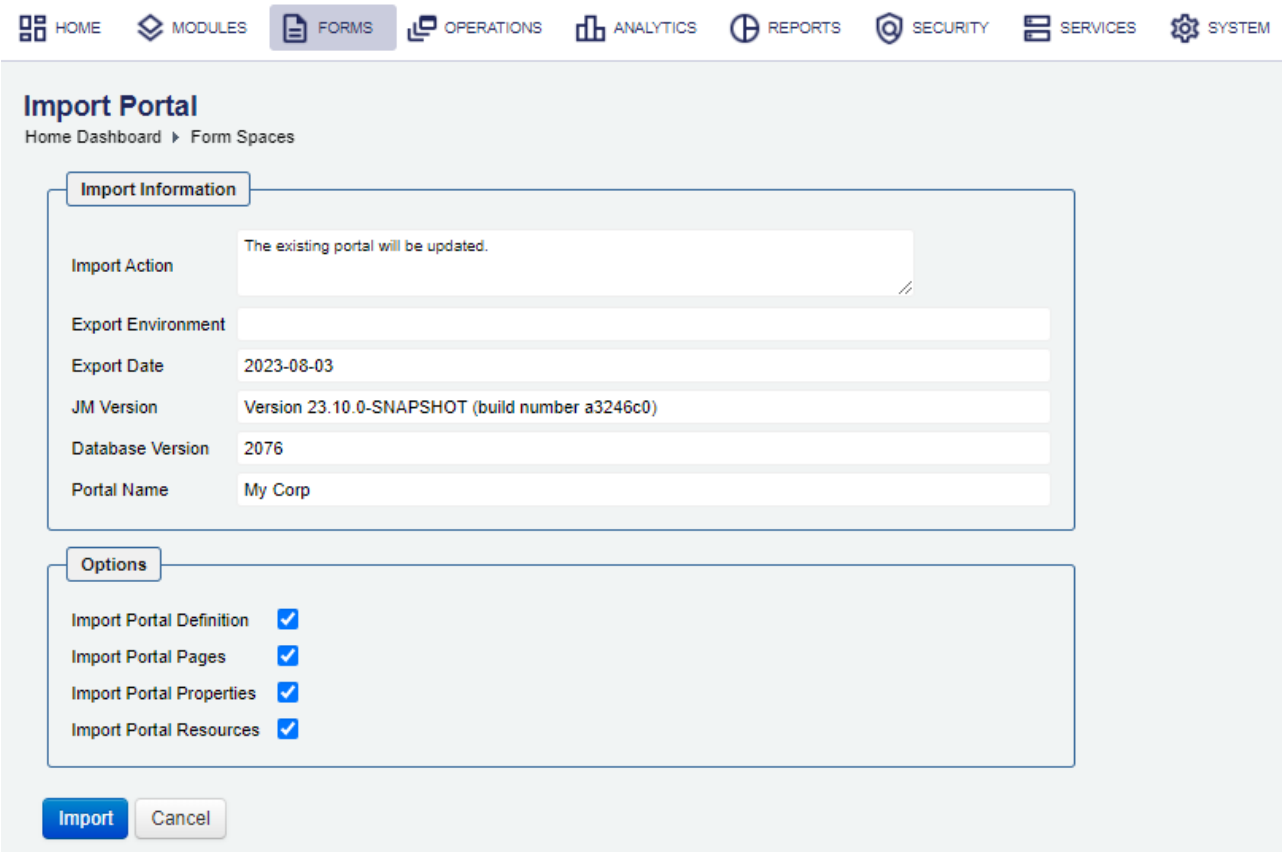
To import a form space as a ZIP archive:

1. Select Forms > Form Spaces.
2. Click Import Archive.



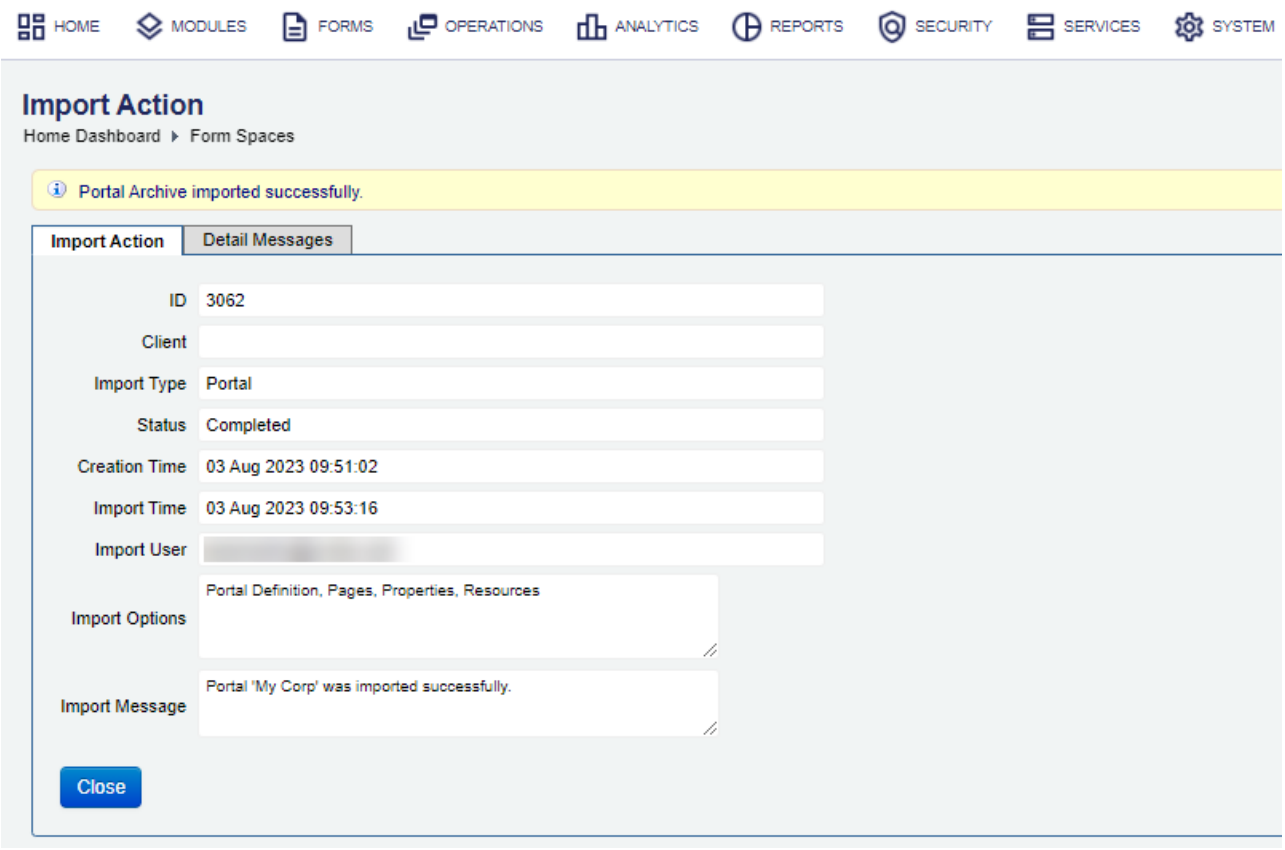
3. Click Choose File and choose a form space archive ZIP file, which has been previously exported.
4. Click Proceed To Import.

5. Review the Import Options.



6. Select the Import Portal Pages checkbox to import pages associated with this form space.
7. Select the Import Portal Resources checkbox to import resources associated with this form space.
8. Select the Import Portal Properties checkbox to import properties associated with this form space.
9. Select the Import Offline Submission Form checkbox to import offline submission forms associated with this form space.

10. Click Import to complete the process.



11. Click Close and check the list of form spaces to verify the imported form spaces have been added.

NOTE

You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

Import Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:archive-deploy goal to import a form space.

Import Using API

The Journey Manager REST API provides the endpoint URL to import a form space.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s)://<JM_SERVER>/manager/secure/rest/archive/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as Base64[1].

`--form ‘archiveFile=@“/Users/my-portal.zip”’` - a path to an archive file to be imported.

`--form ‘importOptions=“PORTAL_DEFINITION,PAGES,PROPERTIES,RESOURCES”’` - a comma-separated list of import options to perform. The option names are the same as described in the Import Using UI section but they use an underscore, for example, `PORTAL_DEFINITION` for Import Portal Definition.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

Example

```
curl --location --request PUT ‘https://<JM_SERVER>/manager/secure/rest/archive/v1/’
\
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
--form ‘archiveFile=@“/Users/my-portal.zip”’ \
--form ‘importOptions=“PORTAL_DEFINITION,PAGES,PROPERTIES,RESOURCES”’
```

Next, learn how to export form spaces.

[1]Base64 is a binary to a text encoding scheme that represents binary data in an American Standard Code for Information Interchange (ASCII) string format.

Migrate Form Spaces

Manager provides a mechanism to migrate form spaces from one environment to another. This allows you to develop and test form spaces in different environments before deploying them into production. You can also use this process to back up your form spaces before major changes, such as a Manager software upgrade, so you will be able to restore them in case of any unforeseen errors.

Migrating a form space doesn't migrate forms associated with it. It does, however, migrate the following resources:

- Pages
- Resources
- Properties

To migrate one or several form spaces, follow the process below:

1. Export form spaces.
2. Import form spaces to another Manager environment.

Manager allows you to migrate form spaces as ZIP or WAR archive files. The main difference between ZIP and WAR files is that a ZIP archive contains only the form's pages related configuration, whilst a WAR archive contains the whole form space configuration. This is illustrated below.

A ZIP archive has:

```
osf/
pages/
portal-config.xml
resources/
```

A WAR archive has:

```
account-pending.htm            login.htm
border-dummy-content.htm       logout.htm
border-template.htm            META-INF/
card-cvv-info.htm              not-authorized.htm
card-payment.htm               not-supported.htm
click/                         off-line.htm
confirmation.htm               receipt-challenge.htm
create-account.htm             redirect.jsp
error.htm                      resources/
error-500.jsp                  save-challenge.htm
forgotten-password.htm         secure/
form.htm                       server-busy.htm
form-attachments.htm           session-expired.htm
form-expired.htm               submission-cancelled.htm
form-not-found.htm             submission-email-verification.htm
hosted-card-payment.htm        submission-errors.htm
hosted-payment.htm             submission-failed.htm
incompatible-adobe-reader.htm  submission-saved.htm
index.htm                      terms-and-conditions.htm
job-challenge.htm              WEB-INF/
job-tasks.htm                  wet-signature.htm
landing.htm
```

NOTE

You can use the same process to perform a form space rollback.

Next, learn about form space hot deployment.

Export a Form Space as a WAR Archive

Manager allows you to export form spaces as WAR or ZIP archive files.

To export a form space as a WAR archive, follow the process below:

1. Select Forms > Form Spaces.
2. Locate a form space.
3. Click Module WAR File.
4. Manager exports the form space into a form space archive WAR file, which will be downloaded to your PC.

Next, learn how to import a form space as a WAR archive.

Import a Form Space as a WAR Archive

Manager allows you to import form spaces as WAR or ZIP archive files, for example, Maestro, as well as perform a form space hot deployment.

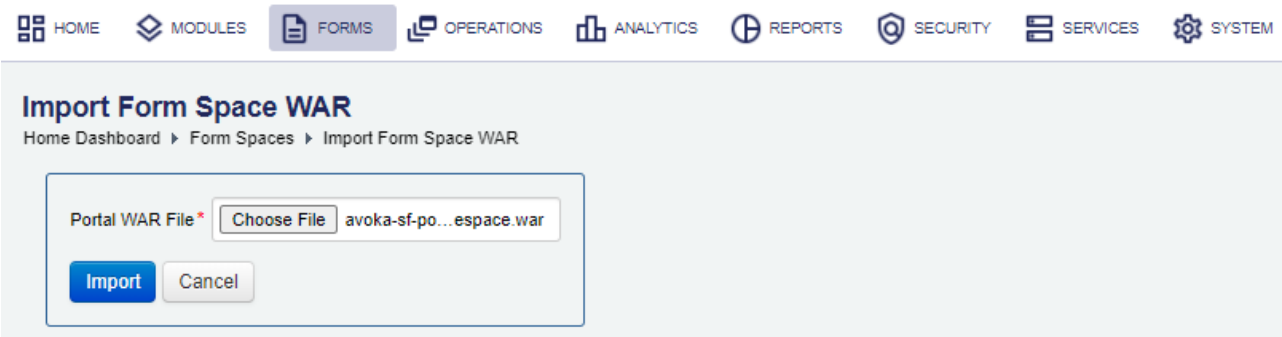
> NOTE
>
> Manager has permissions for user's export and import of a form space configuration as a ZIP archive, but not the WAR files. Therefore, for a user to import a form space archive, they need the Import Action Edit permission.

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

To import a form space as a WAR archive:

1. Select Forms > Form Spaces.
2. Click Import WAR.

   > NOTE
   >
   > The Import WAR button is only available on a single node system. If you have multiple server nodes, this button is grayed out and you will have to deploy a WAR file using a SSH session on your server.



3. Click Choose File and locate a form space archive WAR file, which has been previously exported.
4. Click Import. It might take 5-10 seconds for the import to work.
5. Check that the success message "WAR file deployed, please wait a little while it is initializing." is displayed on the top of the page.

6. Press Close and check the list of form spaces to verify the imported form spaces have been added.

   NOTE

   If you don't see the imported form spaces in the list, it could mean there is a problem with your user account configuration. Manager prior to version 4.3.0 allows an organization to be assigned for users with the Global Access. This is an issue because you can only see the form spaces that belong to the assigned organization. When a new form space is created, it doesn't have an organization assigned and, hence, it may be hidden from you.
   You might also need to restart a Manager server for the change to take effect.

NOTE

You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

You can also import a WAR file into Manager by copying it directly to a Manager server, if you have read and write access to the `deployments` directory.

To copy one or several form spaces as WAR archives:

1. Log onto the Manager server.
2. Copy the form space WAR file to the `/avoka/transact/manager/server/standalone/deployments` directory.
3. Log onto Manager.
4. Check the new form spaces have been added.

Next, learn how to export a form space as a WAR archive.

Standard Form Space Properties

Each form space has a number of predefined properties that allow you to customize a form space without any extra development efforts. For example, you can easily improve form space access security by using the CORS headers. Manager allows you to edit form space's properties, so you can implement additional customization as required.

Some properties are straight-forward but some require more understanding, so we describe them in details below.

- Cache Headers
- CORS Headers
- Security Headers
- X-Frame-Options

Cache Headers

The cache HTTP header allows you to set the following values:

- `Pragma`
- `Cache-Control`
- `Expires`

For more information, see Cache-Control.

The example of Cache Headers configuration is shown below:

```
Pragma: no-cache
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0, private
Expires: 1
```

CORS Headers

Manager evaluates the following conditions of a request and if they all match, then the content of `CORS_HEADERS` is applied to a response.

- `ApplyCORSHeaders` - if set to `true`, Manager applies `CORS_HEADERS` to the request as described below. Otherwise, if set to `false`, this functionality is disabled.
- `CORSApprovedOrigins` - a list of approved origins, which are used to validate the origin on a request. If an approved origin matches the `${request-origin}`, it is replaced with the provided origin. | 19.05 This feature was introduced in 19.05.

- `ExcludedCORSPaths` - a comma-separated list of paths that Manager uses to match any part of an endpoint URL of the request. If the match exists, Manager excludes it from having the headers applied. For example, if you have `"ExcludedCORSPaths":"/servlet"` configured, then all requests from the `/my-site/servlet/FormDynamicDataServlet` endpoint will be excluded and the headers won't be applied. This is because `/my-site/servlet/FormDynamicDataServlet` contains `/servlet`.

  > WARNING
  >
  > If you have an empty `ExcludedCORSPaths` condition, for example, `"ExcludedCORSPaths":""`, Manager won't apply `CORS Headers` to the response, regardless of any other conditions. This behavior will be fixed in | 23.04This feature will be introduced in 23.04

The Cross-Origin-Resource-Sharing[1] (CORS) security HTTP header allows you to set various parameters to a response. Here we only list the most common ones that are pre-configured in a standard form space:

- `Access-Control-Allow-Origin`
- `Access-Control-Allow-Methods`
- `Access-Control-Allow-Credentials`
- `Access-Control-Max-Age`
- `Content-Security-Policy`

For more information, see Cross-Origin Resource Sharing (CORS).

The example of CORS configuration is shown below:

[1]Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to tell a browser to let a web application running at one origin (domain) have permission to access selected resources from a server at a different origin. A web application executes a cross-origin HTTP request when it requests a resource that has a different origin (domain, protocol, and port) than its own origin.

```
{
        "ApplyCORSHeaders":"true",
        "CORSApprovedOrigins":"test-cors.org,avoka.com,temenos.com",
        "ExcludedCORSPaths":"/secure/servlet,/secure/app,/cache",
        "CORS_HEADERS":{
                "Access-Control-Allow-Origin":"${request-origin}",
                "Access-Control-Allow-Methods":"POST,GET",
                "Access-Control-Allow-Credentials":"true",
                "Access-Control-Max-Age":"86400",
                "Content-Security-Policy":"script-src 'self' ${cdn} 'unsafe-inline'; connect-src
'self'; object-src 'none'
        }
}
```

Security Headers

The Content Security Policy[1] (CSP) security HTTP header improves overall forms' security. It is applied to all form's resources that are passed via each form space. If it is not set, the Form CSP Header Value parameter of the Form Submission Access Controller server is used as the default. | 18.11 This feature was introduced in 18.11.

The CSP response header helps you reduce XSS risks on modern browsers by declaring what dynamic resources are allowed to load via a HTTP Header, thereby protecting your solution from a range of vulnerability attacks.

For more information, see Content Security Policy (CSP).

The example of the CSP response header is shown below:

[1]Content Security Policy (CSP) is a computer security standard introduced to prevent cross-site scripting (XSS), clickjacking and other code injection attacks resulting from execution of malicious content in the trusted web page context. CSP provides a standard method for website owners to declare approved origins of content that browsers should be allowed to load on that website—covered types are JavaScript, CSS, HTML frames, web workers, fonts, images, embeddable objects such as Java applets, ActiveX, audio and video files, and other HTML5 features. For more information, see https://www.w3.org/TR/CSP/

```
Content-Security-Policy: script-src 'self' 'unsafe-eval'; object-src 'none'; connect-src
'self'
X-Frame-Opticns: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Prctection: l; mode=block
```

By default, Manager has strict CSP settings configured. Occasionally, you need to include a capability or a 3rd party service in your forms that require changes to the default CSP settings. You can configure the following settings in Manager in order to allow your solution to operate as expected yet still maintain a strong CSP configuration.

To configure CSP:

System Wide Settings

1. Log in to Manager as a Global Administrator.
2. Select Services > All Services.
3. Select Form Submission Access Controller from the Type dropdown list. There should be only one item of this type.
4. Click the service and select the Parameters Edit tab.
5. Modify a `CSP` in the Form CSP Header Value field as required and click Save. For more information, see Form Submission Access Controller Service.

Organization Level Settings

1. Log in to Manager as an Organization Administrator.
2. Select Forms > Organizations.
3. Find your organization and select the Security tab.
4. Under Forms Content Security Policy (CSP), select the Override System Defaults checkbox and click Save.
5. Modify the value of Organization CSP Header as required and click Save . For more information, see Configure Organization Security.

NOTE

The Organization CSP header limit is 2 KB. | 21.11 This feature was updated in 21.11.

LinkedIn Example

Let's illustrate how to use these headers with a simple example.

The LinkedIn SDK requires a JavaScript library to be loaded from their remote server. The default CSP settings are usually something like

```
script-src 'self'; object-src 'none'
```

These settings permit scripts to be loaded from the same server, but no other. To permit the remote LinkedIn JavaScript file to load we must modify these setting to allow JavaScript sources from the relevant LinkedIn domains as follows:

```
script-src 'self' https://platform.linkedin.com https://www.linkedin.com 'unsafe-inline';
object-src 'none';
```

Because the LinkedIn integration uses inline JavaScript, we need to add the `'unsafe-inline'` directive. When you want torelax these settings, you have to add only known and trusted domains to the configuration, as opposed to using wildcards which could compromise this security layer.

X-Frame-Options

The X-Frame-Options[1] HTTP header improves overall forms' security. It is applied to all form's resources that are passed via each form space. The X-Frame-Options include:

- `NONE`
- `SAMEORIGIN` - the page can only be displayed in a frame on the same origin as the page itself
- `ALLOW-FROM URI` - the page can only be displayed in a frame on the specified origin, for example `www.example.com`
- `blank` - no X-Frame-Options header will be set

For more information, see X-Frame-Options.

Next, learn how to configure form space properties.

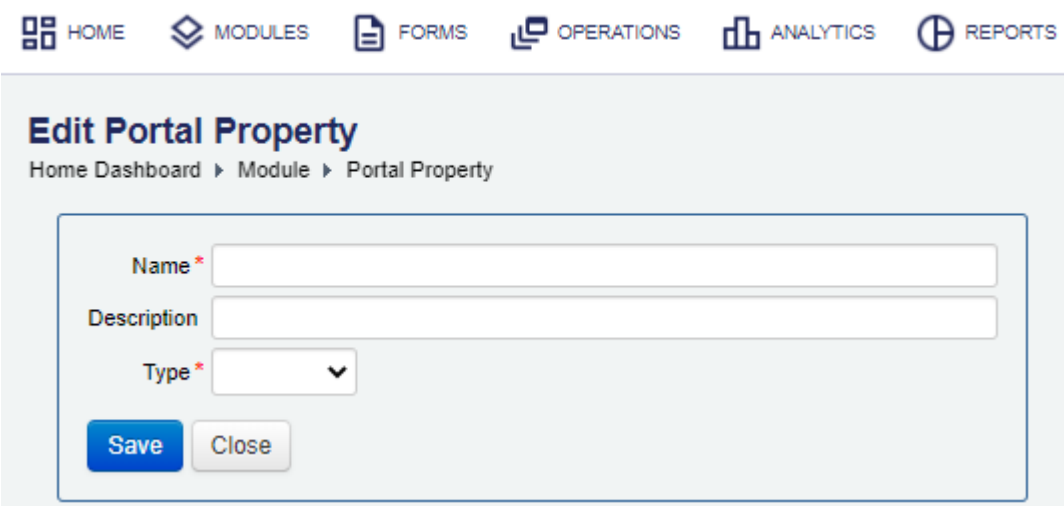
[1]X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a <frame>, <iframe>, <embed> or <object> . Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.

Create Form Space Properties

Each form space has a set of properties, which are used to customize its business logic. Manager comes with several predefined form space's properties that have their own unique functionality. You can create, configure, and remove properties to implement your specific requirements.

To create a new property:

1. Select Forms > Form Spaces.
2. Locate a form space and then click Edit.
3. Select the Properties tab.
4. Click New.



5. Enter a property unique name.
6. Provide a property description.
7. Select a property data type from the Type dropdown list, which can be one of the following:
    - Boolean
    - CSV
    - HTML
    - Image
    - JSON
    - List
    - Long Text
    - Number
    - String

8. Click Save to update the changes.
9. Check the newly created property in the Properties Edit tab.

To remove a property:

1. Select Forms > Form Spaces.
2. Locate a form space and then click Edit.
3. Select the Properties tab and locate a property you don't need any more.
4. Click Remove.

Next, learn how to configure form space properties.

Configure Form Space Properties

Manager comes with a set of standard form space properties that can be applied to all spaces that are created in this Manager instance. These properties have default values that can be configured via the Properties and Properties Edit tabs. You can use form spaces properties to override organization's properties. Subsequently, you can create form's properties to override the form space's properties.

Manager automatically sends emails to form users to notify the form has been changed. You can adjust the email format using the settings below.

> NOTE
>
> Some of form space properties are used to configure Journey Space's tabs.

> NOTE
>
> A form space can be set as not editable, so you can’t change its configuration. To enable this, select the Content Editable checkbox in the Space tab.

To configure the properties in the Properties tab:

1. Select Forms > Form Spaces.
2. Locate a form space and then click Edit.
3. Select the Properties tab.
4. Click Edit to configure the selected property. Check the standard properties to learn more about their functionality.
5. Edit the value of the property.
6. Click Save to update the changes.

To configure form space properties in the Properties Edit tab:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Select the Properties Edit tab to view and edit properties. Check the standard properties to

learn more about their functionality.

4. Edit Account Locked Message, which is a message displayed at login to users whose accounts are locked.
5. Select the Account Quick Lists Enabled checkbox to show Account Home Page quick selection lists and Tasks menu count. Note enabling this option can impact performance.
6. Select the Allow Form Cancel checkbox to allow users to cancel a submission at the payment stage.
7. Select the Allow Form Retry checkbox to allow users to go back to the form if payment has not been completed yet.
8. Select the Allow XML Redirect Target checkbox to allow XML form data to specify redirect target.
9. Select the Allow XML Request Param Form Prefill checkbox to allow XML request parameters, such as `xmlData` and `schemaSeed`, to be used in form prefill.
10. Edit a timeout for an anonymous user to access submitted form receipts in the Anonymous Submission Access Timeout field. After an anonymous user submits a form, Manager sends them a URL link to download a form receipt. This link will timeout after the Anonymous Submission Access Timeout period. However, if the user accesses the URL link using the same user agent[1] (browser) with the same IP address[2], this timeout doesn't apply, because Manager re-establishes a session. In this case, the user can access the form receipt multiple times without any time restrictions. The default timeout is 15 min.

    NOTE

    Check the Session Timeout option of a Security Manager used by the form space to adjust time after which a user session times out.
    Alternatively, you can implement session expiry within the form logic itself independently of Manager.

[1]User agent is software (a software agent) that is acting on behalf of a user. One common use of the term refers to a web browser that retrieves, renders and facilitates end user interaction with Web content.

[2]IP address is a unique string of numbers separated by full stops that identifies each computer using the Internet Protocol to communicate over a network.

11. Specify an optional URL of a CDN Server[1], which is used for caching static content, in the CDN Server URL field.
12. Click the CORS Headers link to edit a list of Cross-Origin-Resource-Sharing (CORS)[2] security HTTP header values. For more information, see CORS Headers.
13. Select the CSRF Protection Host Check checkbox to enable Cross Site Request Forgery (CSRF[3]) protection by validating the host name in a request.
14. Select the CSRF Protection checkbox to enable CSRF protection by using an extra token in a request. A token is a unique secure random token that is used to prevent CSRF attacks. This is selected by default. | 21.11 This feature was introduced in 21.11.
15. Specify a comma-separated list of URLs, which will be excluded from applying CSRF Protection tokens, in the CSRF Protection Ignore Patterns field. For example, if you provide the `/secure/index.htm, /secure/app/**` pattern, requests to these URLs won't include random CSRF tokens.

    NOTE

    This option is used only if the CSRF Protection checkbox is selected.

    WARNING

    You must restart a Manager server or re-deploy this form space for the CSRF Protection and CSRF Protection Ignore Patterns changes to take effect.

16. Edit date and time used in the Date Pickers in the Datetime Format field.

---

[1]A content delivery network or Content Distribution Network (CDN) is a system of distributed servers (network) that deliver pages and other Web content to a user, based on the geographic locations of the user, the origin of the webpage and the content delivery server.
[2]Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to tell a browser to let a web application running at one origin (domain) have permission to access selected resources from a server at a different origin. A web application executes a cross-origin HTTP request when it requests a resource that has a different origin (domain, protocol, and port) than its own origin.
[3]Cross-site request forgery (CSRF), also known as one-click attack or session riding, is a type of malicious exploit of a website where unauthorized commands are transmitted from a user that the web application trusts.

17. Edit an email message, which is sent to a user, for 2FA[1] user account enrolment in the Email 2FA Message field.
18. Edit an email subject, which is sent to a user, for 2FA user account enrolment in the Email 2FA Subject field.
19. Edit an email message for form confirmation or receipt email in the Email Form Receipt Message field.
20. Edit an email subject for form confirmation or receipt email in the Email Form Receipt Subject field.
21. Edit a lost password reset email message in the Email Lost Password Message field.
22. Edit a lost password reset email subject in the Email Lost Password Subject field.
23. Edit an email message for user saved form email in the Email Saved Form Message field.
24. Edit an email subject for user saved form email in the Email Saved Form Subject field.
25. Edit a sender email address for generated emails and save challenge validations in the Email Sender Address field. This allows you to override the global email sender address configured on an organization's level. For more information, see Configure Organization Properties. You can also overrides this email address on a form's level by adding the Email Sender Address property to a form version.

    WARNING

    If the address is incorrect or invalid, the SMTP server will throw an error.

26. Edit an email message for user shared form email in the Email Share Form Message field.
27. Edit an email share form subject in the Email Share Form Subject field.
28. Edit an email message for submission processing status update in the Email Submission Status Message field.
29. Edit an email subject for submission processing status update in the Email Submission Status Subject field. For example, `${submission.formName} Update`.

---

[1]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.

30. Edit an email message for task notification in the Email Task Notification Message field.
31. Edit an email subject for task notification in the Email Task Notification Subject field.
32. Edit an email border template HTML in the Email Template HTML field.
33. Edit a location of the email HTML border template to use in the Email Template Mode field.
34. Edit a form's Terms and Conditions statement the user must accept before accessing the form in the Form Terms And Conditions field.
35. Edit a Google API Services Access Key in the Google API Key field.
36. Edit a maximum number of save challenge requests that can be performed when returning to a saved form before a user is locked out in the Max Save Challenge Requests field. This maximum is inclusive of any attempts from other sessions for the same submission. The default is 3.
37. Select the Require Client Key checkbox to append a client key of an organization, where the form is deployed, to a form render URL. This prevents unauthorized access to a rendered form when a hacker guesses a form code and uses it to generate a valid URL. | 23.04 This feature was introduced in 23.04

    WARNING

    If there is a missing or mismatching client key in the URL, Manager displays the Form Not Found error page with The form you are looking for could not be found. Please check the form name and try again. error message.

38. Edit a duration (in minutes) after which a submission is locked out when the maximum number of save challenge requests has been exceeded, as per the Max Save Challenge Requests property in the Save Challenge Lockout Minutes field. If the Save Challenge Lockout Minutes is set to 0, then the submission is permanently locked out, which requires an administrator to manually unlock this submission. The default is 15.

    NOTE

    You can also configure the Save Challenge Lockout Minutes property in a form version to override this form space level's property. | 21.05 This feature was introduced in 21.05.

39. Select the Search Filter Group Select checkbox to enable the Groups dropdown list in the Journey Space Tasks and History tabs. This is the default option.
40. Select the Search Consistent Read checkbox to search using database Consistent Read in the Task List, History, Reviews and Help Desk pages.

41. Select the Search Data Extracts checkbox to search for the Submission Data extracts in Task List, History, Reviews and Help Desk pages.
42. Select a data load type from the Search Data Load dropdown list, such as Eager Data Load or Lazy Data Load, to apply to certain portal pages (Help Desk). If "Lazy Data Load" is used, the page will not attempt to load result data on initial access, instead waiting for the user to initiate a search. This can help with performance on systems with large transaction volumes. "Eager Data Load" means that the page will automatically load data when initially accessed.
43. Select the search results fetch limit for the Task List, History, Reviews and Help Desk pages from the Search Fetch Limit dropdown list. Default is 200.
44. Select the Search Filter Group Select checkbox to show the Group Select in the Search Filter on Task, History pages.
45. Select the Search Filter Jobs by User Org checkbox to enable the Job Groups dropdown list in the Journey Space Reviews tab and allows only job coordinators to see collaboration jobs that they are responsible for.
46. Select the default Search Period for the Help Desk pages from the Search Period Help Desk dropdown list.
47. Select the default Search Period for the History pages from the Search Period History dropdown list.
48. Select the default Search Period for the Reviews pages from the Search Period Reviews dropdown list.
49. Select the default Search Period for the Task List pages from the Search Period Tasks dropdown list.
50. Select the Search Task Attributes checkbox to search for the Task attributes Subject, Message and Address in the Task List and History pages.
51. Edit an external resource URL to redirect users to if the server is too busy to handle the form request in the Server Busy Redirect URL field. The default is the Server Busy form space page.
52. Select the Use Email for Login checkbox to use the email address as the user login name for Local Manager user accounts. It's not selected by default.
53. Specify an alternative user create account URL in the User Account Creation URL field.
54. Edit a default welcome page URL in the Welcome Page URL field. For example, `index.htm`.
55. Edit an X-Frame-Options HTTP header value in the X-Frame-Options field. The options can be such as: `[ NONE | SAMEORIGIN | ALLOW-FROM www.example.com ]`. If blank then no 'X-Frame-Options' header will be set. For more information, see X-Frame-Options.
56. Click Save to update the changes.

NOTE

If you are using a customized form space, some of these properties may not be implemented.

To create a new form space property in the Properties Edit tab:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Click New to create a new property.
4. Provide a property unique name in the Name field.
5. Select a property type from the Type dropdown list.
6. Click Save to update the changes.

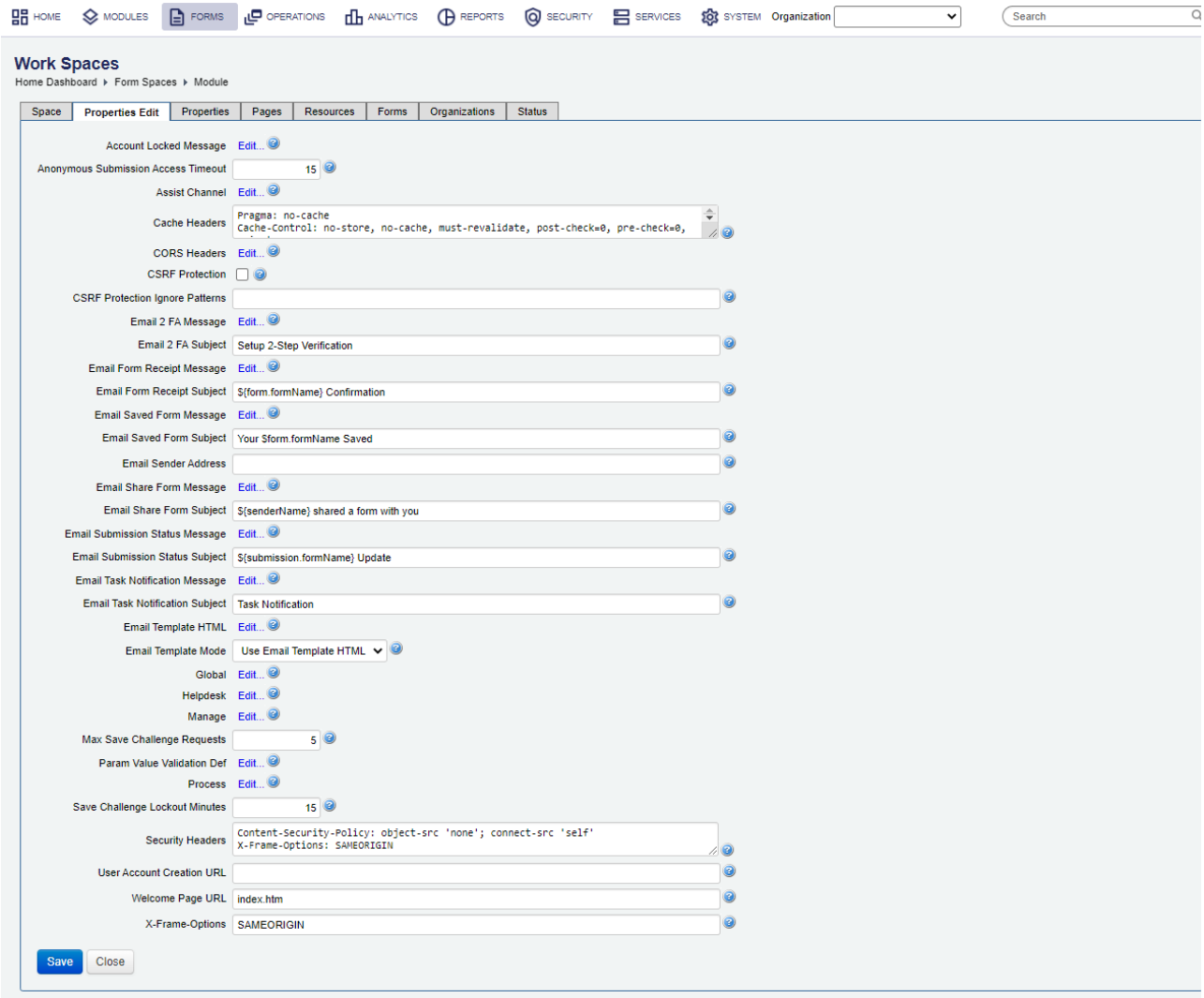
Next, learn how to a configure form space email template.

Configure Workspaces Properties

Manager allows you to configure specific properties of Workspaces pages, which are not part of the standard form space properties. You can adjust some of the properties to improve Workspaces data fetch performance that will result in a better user experience.

To configure the Workspaces's properties:

1. Select Forms > Form Spaces.
2. Locate the Workspaces and click Edit.
3. Select the Properties Edit tab to view and edit properties. For more information, see Configure Form Space Properties.



4. Select the Enhanced PII Search checkbox to enable enhanced searching for submission extracts or properties, which can significantly increase Workspaces data fetch performance. This option is selected by default. | 24.10 This feature was introduced in 24.10

> NOTE
>
> In versions prior to | 24.10 This feature was introduced in 24.10, this option is not created by default so you need to create it manually.

5. There's an optional search limit, which can be configured in the Enhanced PII Search Limit field, to restricts SQL predicate results used within the Enhanced PII Search logic. The default is 999. | 19.11.3 This feature was introduced in 19.11.3.

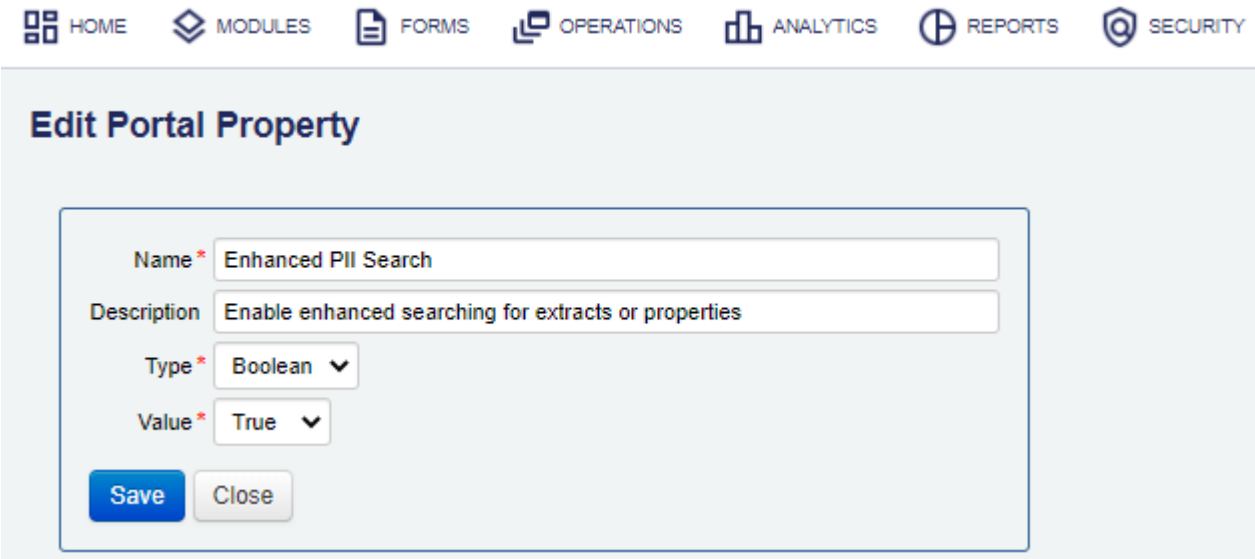
> NOTE
>
> It is not recommended to change this setting unless you are experiencing slow search queries by any transaction fields or it is advised by Temenos. This option is not created by default, so you must add it as a new property or contact Temenosto do it for you.

6. Click Save to update the changes.

Starting from | 19.11.3 This feature was introduced in 19.11.3., you can create this property manually to improve PII search functionality.

To create the Enhanced PII Search property:

1. Select Forms > Form Spaces.
2. Locate the Workspaces and click Edit.
3. Create the new Boolean property as shown below:



4. Click Save to update the changes.

Next, learn how to a configure form space email template.

Configure a Form Space Email Template

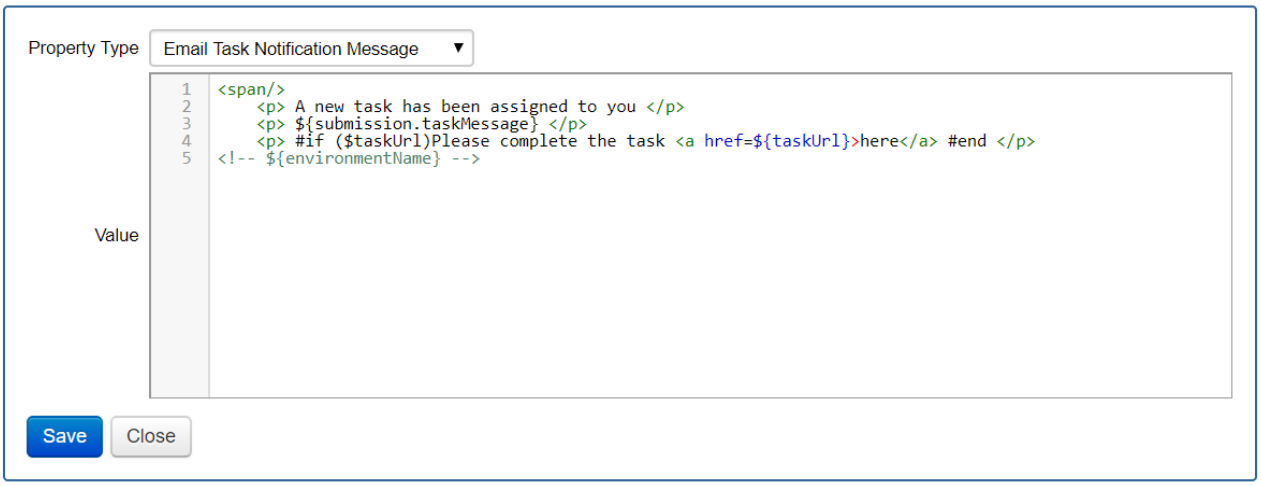
Manager extensively uses emails to send various notifications to form users and administrators. Manager applies email templates to insert form's dynamic content into generated emails. For example, you can have a form that lets a user to select multiple products and, when the form is completed, to send an email with a subject displaying all the products entered in the form. You can achieve this by configuring an email template which has access to a data model from which it can extract product names or iterate over them.

Email templates are created using the Velocity Template Language (VTL) markup, which is one of the simplest and easiest methods of incorporating dynamic data into an HTML document and is made up of References and Directives. References begin with a `$` and are used to retrieve a value, for example, $a = FirstName. Directives begin with a `#` and are used to perform an action, for example, #get, #set, #end.

> NOTE
>
> The email templates can be customized at organization, form space and form levels.

To configure a form space email template:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Select the Properties tab to see several email templates available.
4. Locate a template and click Edit. The common email template is an Email Task Notification Message one shown below.



5. Update the template email body as an HTML text in the Value field. In this example, the template describes a task for a recipient to complete. It consists of both the static text, “A new task has been assigned to you”, and dynamic text, $taskUrl, which is replaced with the URL when the email is rendered.
6. Click Save to update the changes.

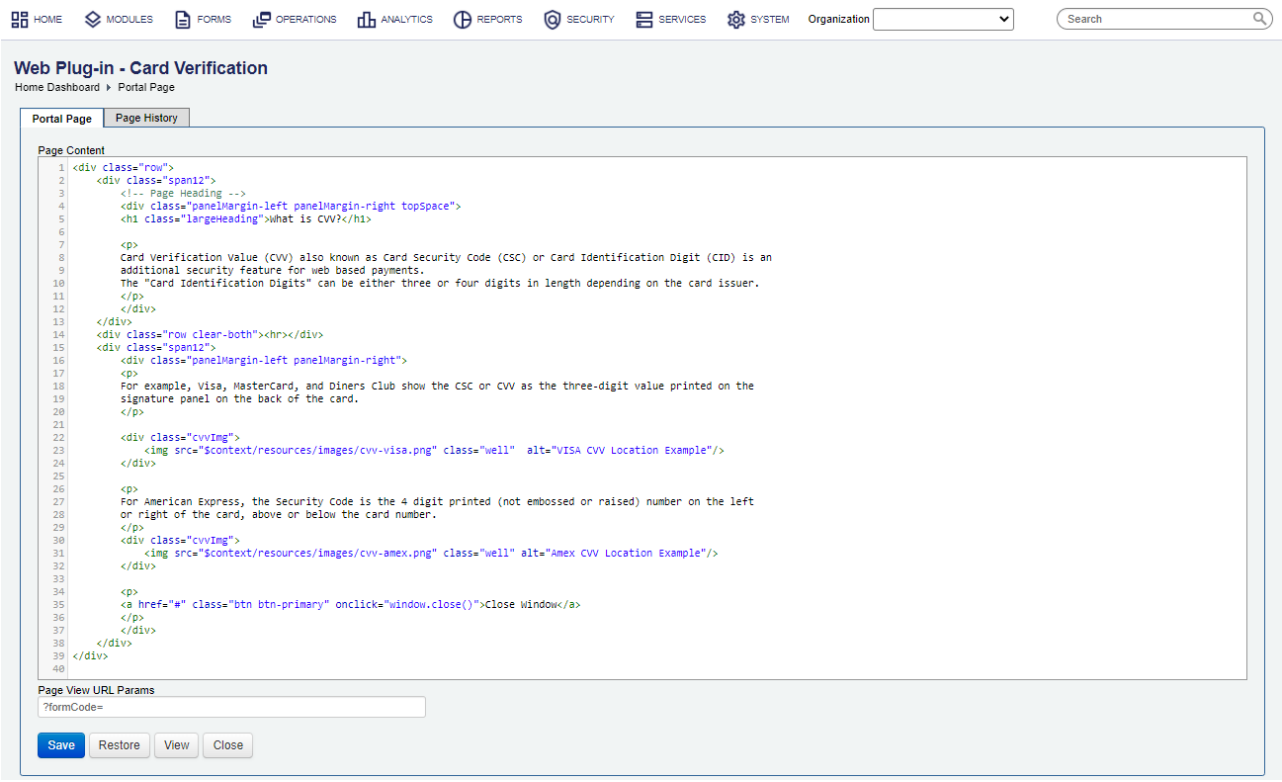
Next, learn how to view form space pages.

Configure Form Space Pages

Manager comes with pre-configured HTML pages, which are displayed to form users to enhance their form journey, for each form space.You can edit some HTML elements of a page to better suit your forms. However, you can't change the page behavior nor response.

To configure a page:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Select the Pages tab.
4. Find the page that you want to edit and then click Edit.



5. Edit the content of an HTML page in the Page Content field.

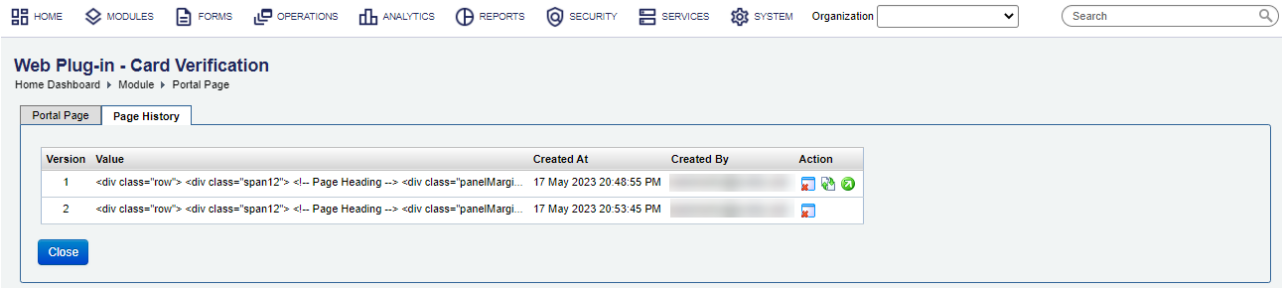
> NOTE
>
> To configure a pages, ensure that the Content Editable checkbox is selected.

6. Update the URL to this page in the Page View URL Params field.
7. Click Restore to cancel the latest changes and restore the page content to its base content.

8. Click View to check how the updated page looks. A new version of the page is created each time the page is edited and saved.
9. Click Save to update the changes.

To view the history of the page:

1. Click the Page History tab to see the changes made to the page.



2. Click Remove Portal Page History to delete the selected record from the history.
3. Click Restore this version's content as the current content to make this content the current one.
4. Click View Content Diff to Base Page Content to compare the current modified version of the page with the base version of the page.

Next, learn how view form space resources.

View Form Space Pages

Manager provides a range of generic HTML pages that support all forms associated with a form space. These pages are not part of the form, but rather are a part of the entire application that represents the form. Common examples are a Forgotten Password page, a Form Not Found page, and a Terms And Conditions page. Manager allows you to view and configure the pages.

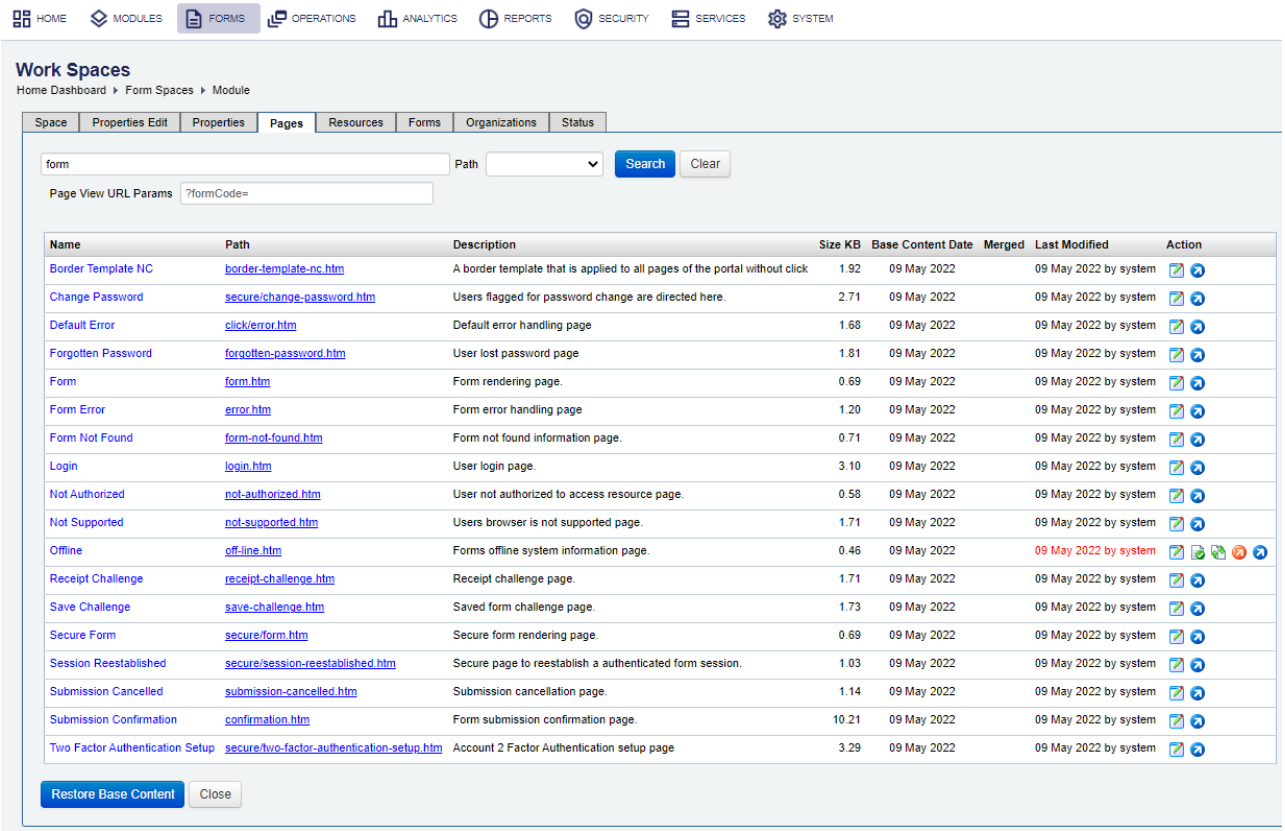
> NOTE
>
> As part of the ongoing modernization of the platform, since version 22.04, we have deprecated a number of Work Space pages: the Wet Signature, Landing, and Terms And Conditions pages. These pages are part of the old Non-FTX submission life cycle that is now handled within form design itself.

To view form space's pages:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Select the Pages tab.

   > NOTE
   >
   > If the Pages tab is not shown, select the Content Editable checkbox in Configure a Form Space.

   > NOTE
   >
   > You must have the Portal Content Edit permission to be able to create new resources.



Name	Path	Description	Size KB	Base Content Date	Merged	Last Modified	Action
Border Template NC	border-template-nc.htm	A border template that is applied to all pages of the portal without click	1.92	09 May 2022		09 May 2022 by system	
Change Password	secure/change-password.htm	Users flagged for password change are directed here.	2.71	09 May 2022		09 May 2022 by system	
Default Error	click/error.htm	Default error handling page	1.68	09 May 2022		09 May 2022 by system	
Forgotten Password	forgotten-password.htm	User lost password page	1.81	09 May 2022		09 May 2022 by system	
Form	form.htm	Form rendering page.	0.69	09 May 2022		09 May 2022 by system	
Form Error	error.htm	Form error handling page	1.20	09 May 2022		09 May 2022 by system	
Form Not Found	form-not-found.htm	Form not found information page.	0.71	09 May 2022		09 May 2022 by system	
Login	login.htm	User login page.	3.10	09 May 2022		09 May 2022 by system	
Not Authorized	not-authorized.htm	User not authorized to access resource page.	0.58	09 May 2022		09 May 2022 by system	
Not Supported	not-supported.htm	Users browser is not supported page.	1.71	09 May 2022		09 May 2022 by system	
Offline	off-line.htm	Forms offline system information page.	0.46	09 May 2022		09 May 2022 by system	
Receipt Challenge	receipt-challenge.htm	Receipt challenge page.	1.71	09 May 2022		09 May 2022 by system	
Save Challenge	save-challenge.htm	Saved form challenge page.	1.73	09 May 2022		09 May 2022 by system	
Secure Form	secure/form.htm	Secure form rendering page.	0.69	09 May 2022		09 May 2022 by system	
Session Reestablished	secure/session-reestablished.htm	Secure page to reestablish a authenticated form session.	1.03	09 May 2022		09 May 2022 by system	
Submission Cancelled	submission-cancelled.htm	Submission cancellation page.	1.14	09 May 2022		09 May 2022 by system	
Submission Confirmation	confirmation.htm	Form submission confirmation page.	10.21	09 May 2022		09 May 2022 by system	
Two Factor Authentication Setup	secure/two-factor-authentication-setup.htm	Account 2 Factor Authentication setup page	3.29	09 May 2022		09 May 2022 by system	

The page list displays the following details:

- Name - a name of a page
- Path - a relative path to the page
- Description - a meaningful description of a page
- Size KB - the page size in KB
- Base Content Date - a date of the page's content as defined in the form space's WAR file
- Merged - whether the content was merged or not
- Last Modified - a date when the page was modified and a name of a user who modified it

> NOTE
>
> If the content is different from the base content, the page is highlighted in red, and the following actions are made available: Mark Content Merged, Restore Base Content, and View Content Diff to Base Page Content

4. To filter or search the form spaces by one or more criteria, specify the following settings and click Search:

- Page name or a part of its name
- Path to the page

Click Clear to reset the search criteria.

You can perform one of the following actions on a selected page:

> NOTE
>
> You must have the certain permissions set by an Administrator to be able to perform the following actions with the exception of editing and viewing pages that any user can do.

- Click Edit to configure a page.
- Click Restore Base Content to restore the page content back to the original base content. This removes all latest modifications and returns the content back to its original form.

  > NOTE
  >
  > A page base content is the page's content as defined in the form space's WAR file.

- Click View Content Diff to Base Page Content to compare the current modified version of the page with the base version of the page.
- Click Mark Content Merged to promote the modified page to the base page. This marks the modified page as a new generic base version of the page.
- Click View to view the page in a browser. If the page needs certain URL parameters to display correctly, for example, a form code, enter them into the Page View URL Params field.

Next, learn about form space resources.

Create a Form Space Resource

Manager allows you to create different types of resources to be hosted in a form space.

You can create a resource using one of following options:

- UI
- Fluent API
- REST API

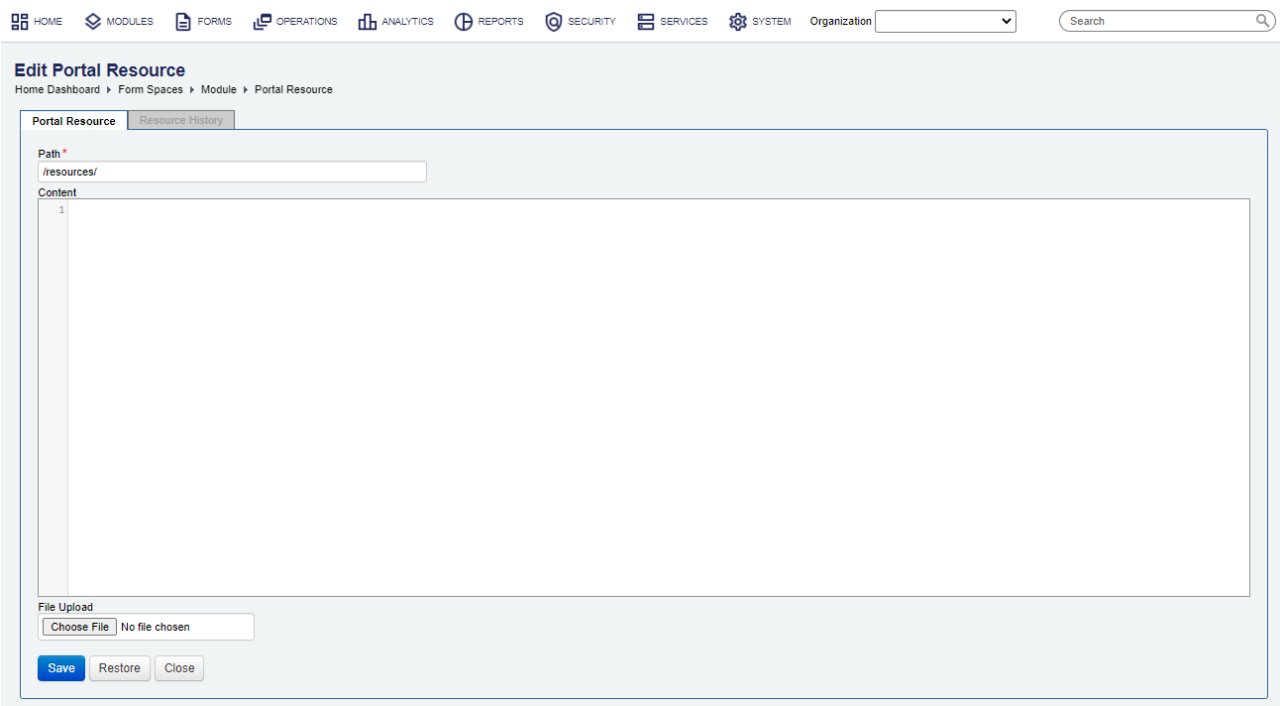
Create Using UI

To create a resource:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Click the Resources tab and then click New.

> NOTE
>
> If the Resources tab is not shown, select the Content Editable checkbox in Configure a Form Space.

> NOTE
>
> You must have the Portal Content Edit permission to be able to create new resources.



4. Enter a path of a new resource where it will be hosted. The path is relative to the form space context path, for example, `resources/CSS/learning.css`.
5. Enter a content of the new resource. You can add content directly into the content text area for a CSS resource or any other textual resources, for example, a Word document.
6. Click File Upload to attach a file as a resource, such as an image, translation CSV file and PDF file. You can upload files only if their file name extensions are white-listed in the Allowed File Name Extensions In JM deployment property. | 24.04 This feature was introduced in 24.04
7. Click Save to update the changes. This creates the resource.

Create Using Fluent API

The Journey Manager provides the Fluent API to create and update form space resources. The methods are `create`, `update` and `createOrUpdate` and they require either a form space ID or name to be set to create or modify a resource. This is illustrated below:

```
new SpaceResourceBuilder(txn)
    .setSpaceId(733)
    .setSpaceName("workspaces")
    .setResourcePath("resources/html/test.html")
    .setFileData(new byte[] {1,2,5})
    .createOrUpdate
```

Create Using REST API

The Journey Manager REST API provides the endpoint URL to create and update form space resources.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s):://{JM_SERVER}/manager/secure/rest/fluentapi/spaceresourcebuilder/create
```

Parameters

`Content-Type:application/json` - a content type.

`Accept:application/json` - a data load in JSON format including `setSpaceId`, `setResourcePath` and `setFileData`.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code:

`fileSize` - a size of a resource.

`lastModified` - when the resource was updated.

`modifiedBy` - a user who updated the resource.

`space` - a form space where the resource belongs to.

`resourcePath` - a relative path to the resource.

Example

A request:

```
curl --location --request POST ‘POST https://{JM_SERVER}/-
manager/secure/rest/fluentapi/spaceresourcebuilder/create \
Content-Type:application/json \
Accept:application/json \
{ \
   "setSpaceId" : "733", \
   "setResourcePath" : "resources/html/test.html", \
   "setFileData" : "the base64 converted string value of byte data" \
}’
```

A response:

```
{
```

```
    "fileSize": "1.00 KB",
    "lastModified": "Wed Sep 04 14:49:41 AEST 2024",
    "modifiedBy": "administrator",
    "space": "Work Spaces",
    "resourcePath": "/resources/html/test.html"
}
```

Next, learn how to configure a form space resource.

Configure a Form Space Resource

Manager allows you to edit resources belonging to a form space.

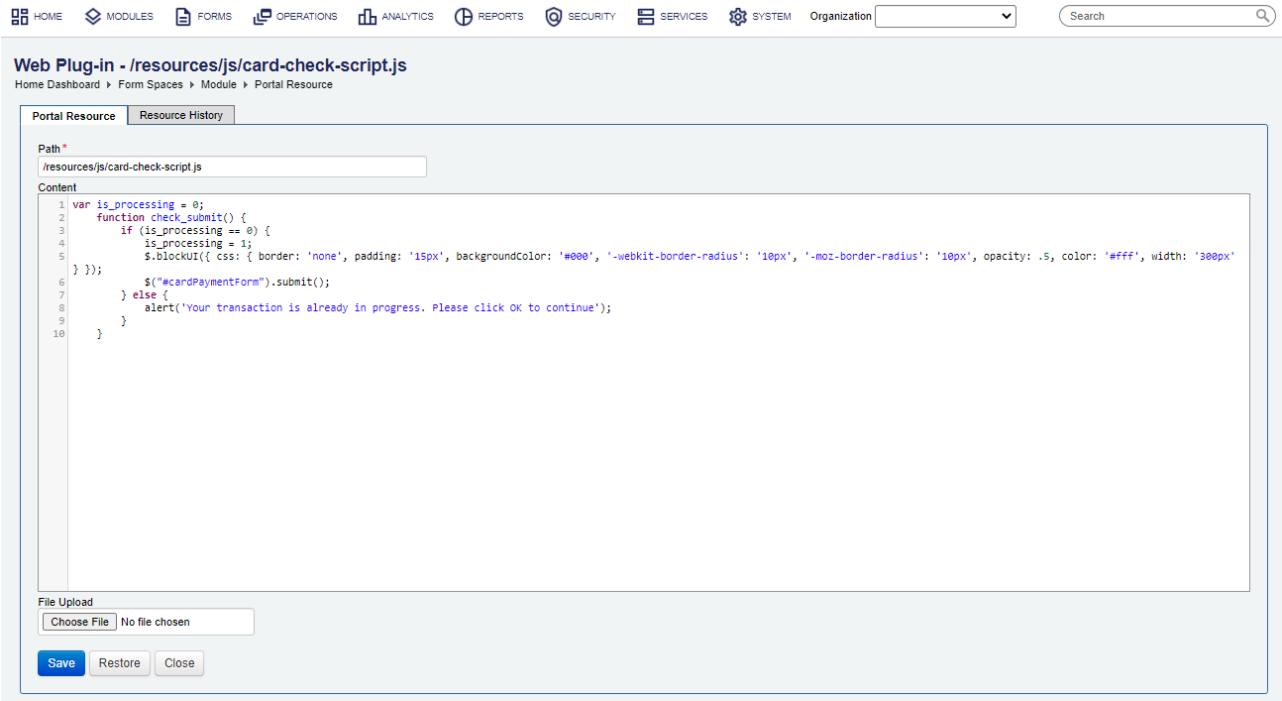
You can update a resource using one of following options:

- UI
- SDK
- API

Update Using UI

To configure a form space resource:

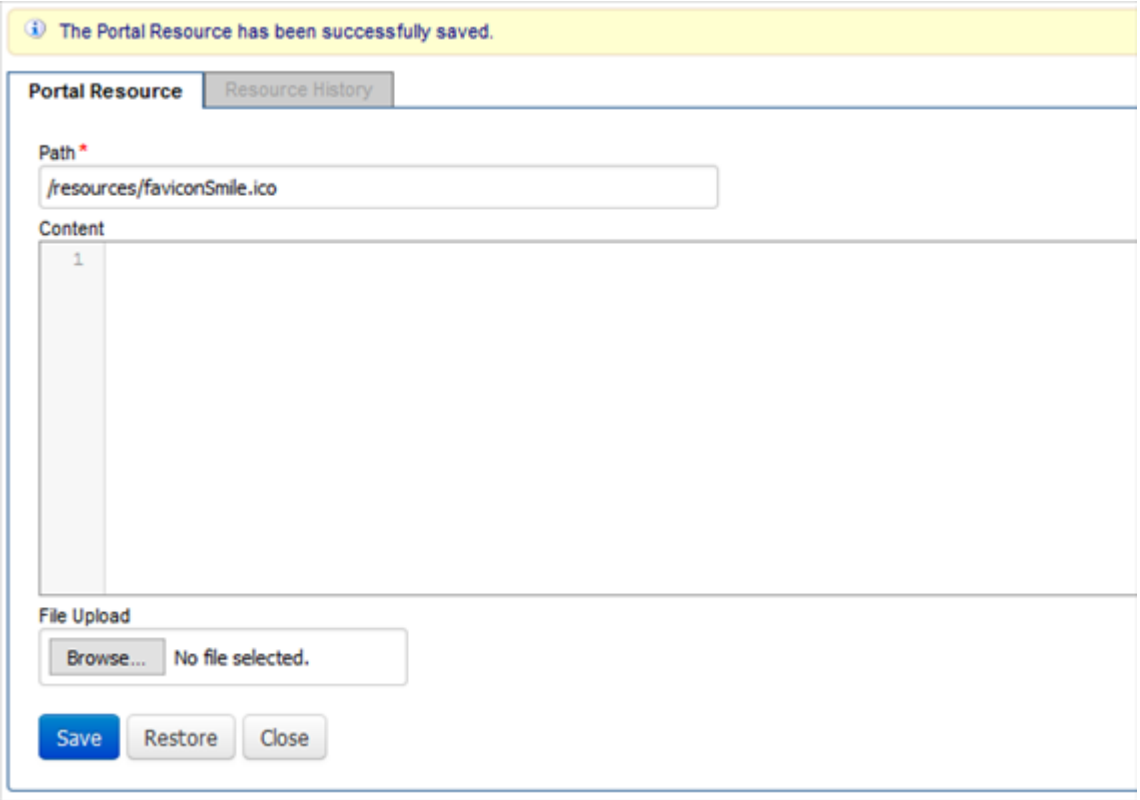
1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Select the Resources tab.
4. Locate a resource and click Edit.



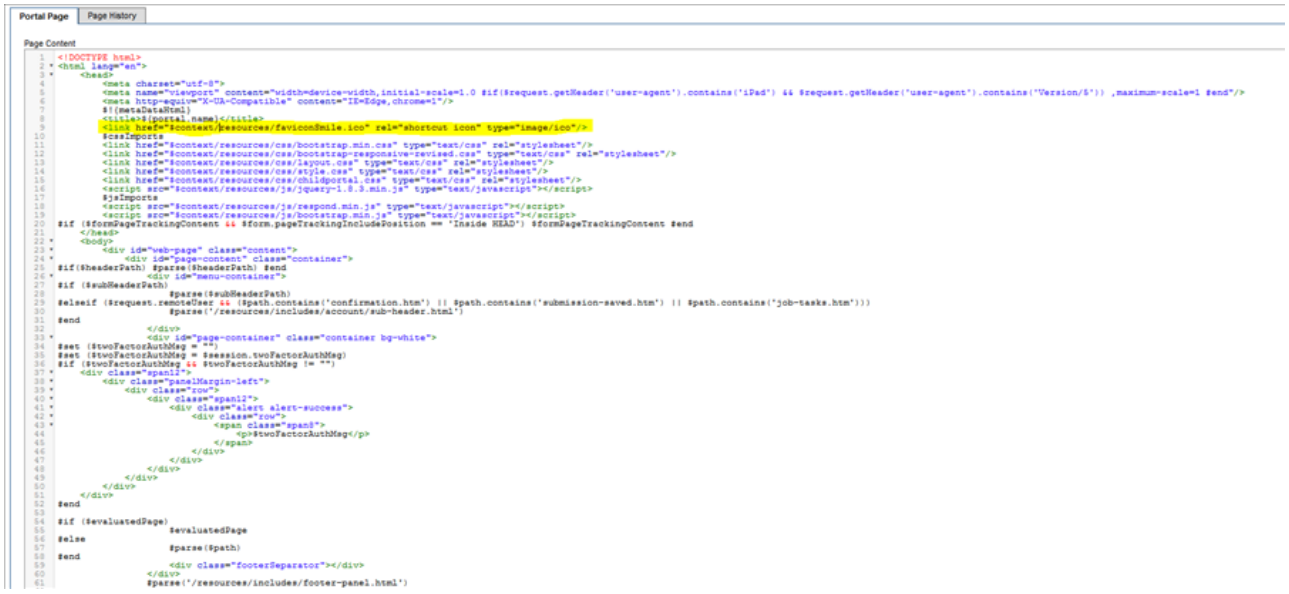
5. Edit a path to the resource in the Path field.
6. Edit the content of a text resource in the Content field, for example a CSS or HTML file.
7. Click Choose File to upload a new resource, for example a GIF image file.
8. Click Save to update the changes.
9. Click Restore to undo any new changes.

To add a new fav icon file:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Select the Resources tab.
4. Click New and upload a file. Give it a different file name from the standard one as browsers cache these things very aggressively, and they are hard to clear out. Make sure it is in the correct format.



5. Edit the Space Page “Border Template” and set the path to the new fav icon image.



To view a form space resource history:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Select the Resources tab.
4. Locate a resource and click Edit
5. Click the Resource History tab to view the history of a form space resource showing every edit that has ever been made to this resource.
6. Click Restore to roll back the resource to a previous version.
7. Click Compare to compare the content of a previous version of the resource with the current one.
8. Click Remove to delete the selected version of the resource.
9. Click Close.

Next, learn how to view all form resources.

View Form Space Resources

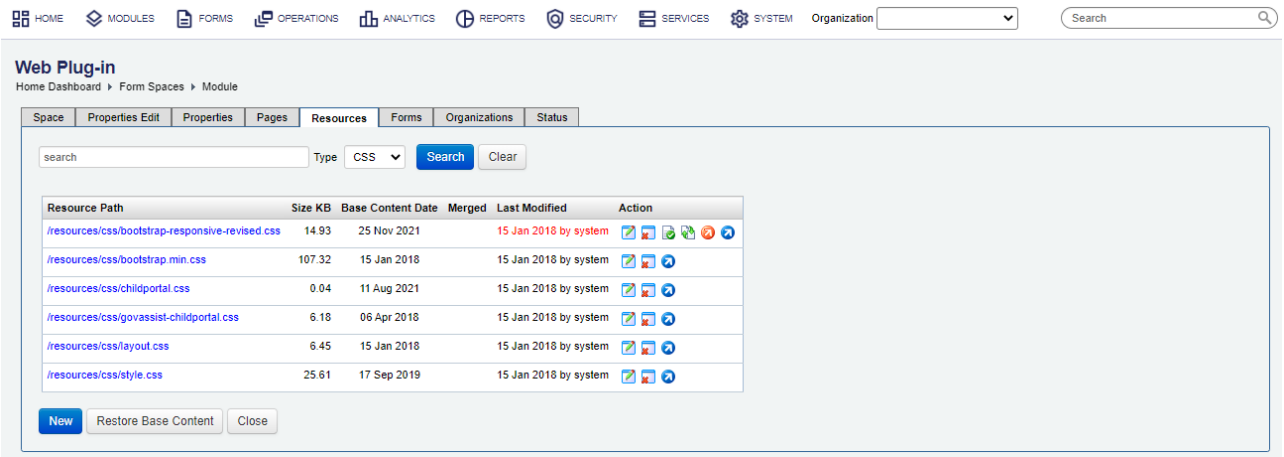
Manager allows you to view resources of a selected form space, create new resources, and update the existing ones. Resources are various files that a form requires to work and render correctly. A form resource type can be one of the following:

- CSS
- Image: GIF, JPG, PNG
- PDF
- HTML
- JavaScript
- Plain text

An example of a resource is an image file that is used in the Forgotten Password page.

To view form space resources:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Select the Resources tab.



The list displays the following details:

- Resource Path: a relative path to the resource
- Size KB: the resource size in KB
- Base Content Date: a date of the resource's content as defined in the form space's WAR file
- Merged: whether the content was merged or not
- Last Modified: a date when the resource was modified and a name of a user who modified it

> NOTE
>
> If the content is different from the base content, the resource is highlighted in red, and the following actions are made available: Mark Content Merged, Restore All Portal Resource with their Base Content, and View Content Diff to Base Resource

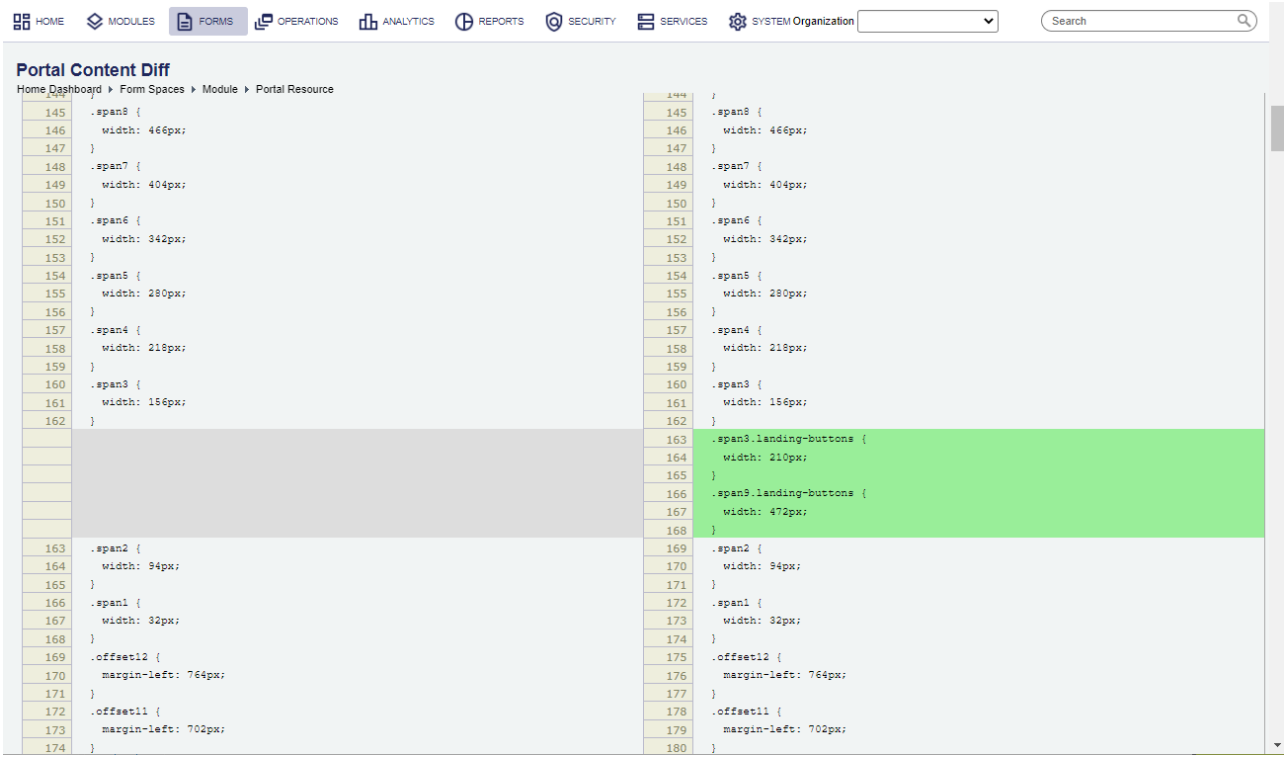
4. To filter or search the resources list by one or more criteria, specify the following settings and click Search:
   - Any text message that resource name and content may include
   - Resource type:
     - CSS
     - Image: GIF, JPG, PNG
     - PDF
     - HTML
     - JavaScript
     - Plain text
5. Click Clear to reset the search criteria.

You can perform the following actions on a selected resource:

- Click Edit to configure the content of a resource
- Click Remove Portal Resource to delete a resource
- Click Restore Base Content to restore the resource's content back to the original base content.

  > NOTE
  >
  > This removes all modifications and returns the resource's content back to its base, as well as replaces all styling cached in the database.

- Click Mark Content Merged to promote the modified resource to the base resource, which makes the modified resource the new generic (base) version of the resource.
- Click Restore All Portal Resource with their Base Content to restore the base content of the resource
- Click View Content Diff to Base Resource to view the differences between the current and base content of the resource. Select how you want to view the difference from the View Type dropdown list, which has the following options: Side by Side and Inline.



> NOTE
>
> You can't edit the content of the resource in this window.

- Click View Content to render and display a resource in a new tab.

> NOTE
>
> This option is not shown if the Disable Render Link for Portal Resources deployment property is set to `true`. | 24.04 This feature was introduced in 24.04

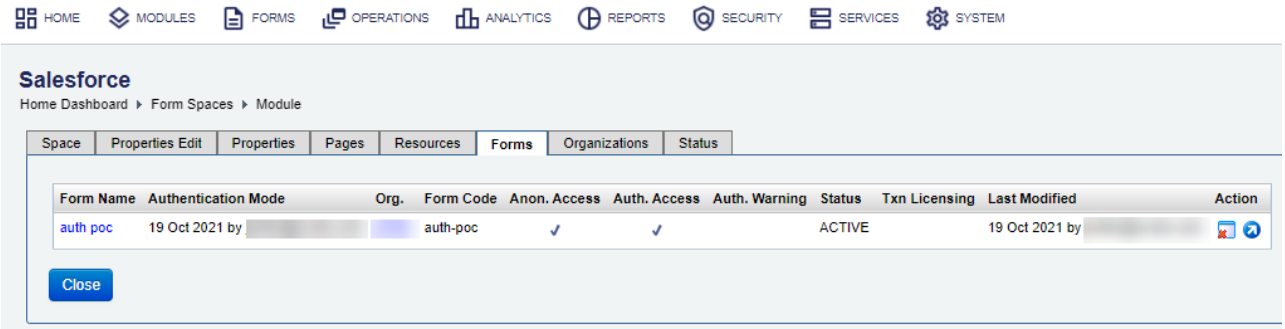
Next, learn how to view all forms hosted on a form space.

View Forms Hosted on a Form Space

Manager allows you to view all forms hosted on each form space, so you can administer and manage them.

To view all forms hosted on a form spaces:

1. Select Forms > Form Spaces.
2. Locate a form space and click Edit.
3. Select the Forms tab to see all forms listed page by page, with the maximum number of records per page configured in the users preferences.



The list displays the following details:

- Form Name: a name of a form.
- Authentication Mode:
- Org: an organization the form belongs to.
- Form Code: an ID of a form.
- Anon. Access: whether the anonymous access is allowed or not.
- Auth. Access: whether the authenticated access is allowed or not.
- Auth. Warning: shows any authentication warning, for example, The form space has been configured as Authenticated Only, but the form has the Anonymous flag set.
- Status: a status of a form, which can be one of the following:
  - Active: it is in use and readily available to all users. All functionality is allowed.
  - Offline: it can't be used at all. No functionality is available, including DDS, Delivery and so on.
  - Inactive: similar to Offline.
  - NoRender: it doesn't allow for new or saved and resumed forms, but ongoing open sessions can be completed. DDS, submits, saves, and delivery are allowed.

    - Txn Licensing: whether the form is billable or not.
    - Last Modified: when the form was modified and by what user.
4. Click Close.

You can also perform the following actions:

- View a form
- Remove a form from this form space
- Form test render

Next, learn how to assign organizations to a form space.

Form Space Hot Deployment

Hot deployment of a form space is the process of updating a form space WAR file without restarting Manager. This enables developers to update a form space with few changes, re-build it as a WAR file, and then deploy it to Manager on-the-fly, so the changes can be available to the form users right away.

To perform a hot deployment, prepare a WAR file and then import it.

Use this functionality in a development environment only and restrict user's access to form space import to prevent or minimize accidental or malicious deployment of WAR files that can potentially be very harmful to the Manager environment.

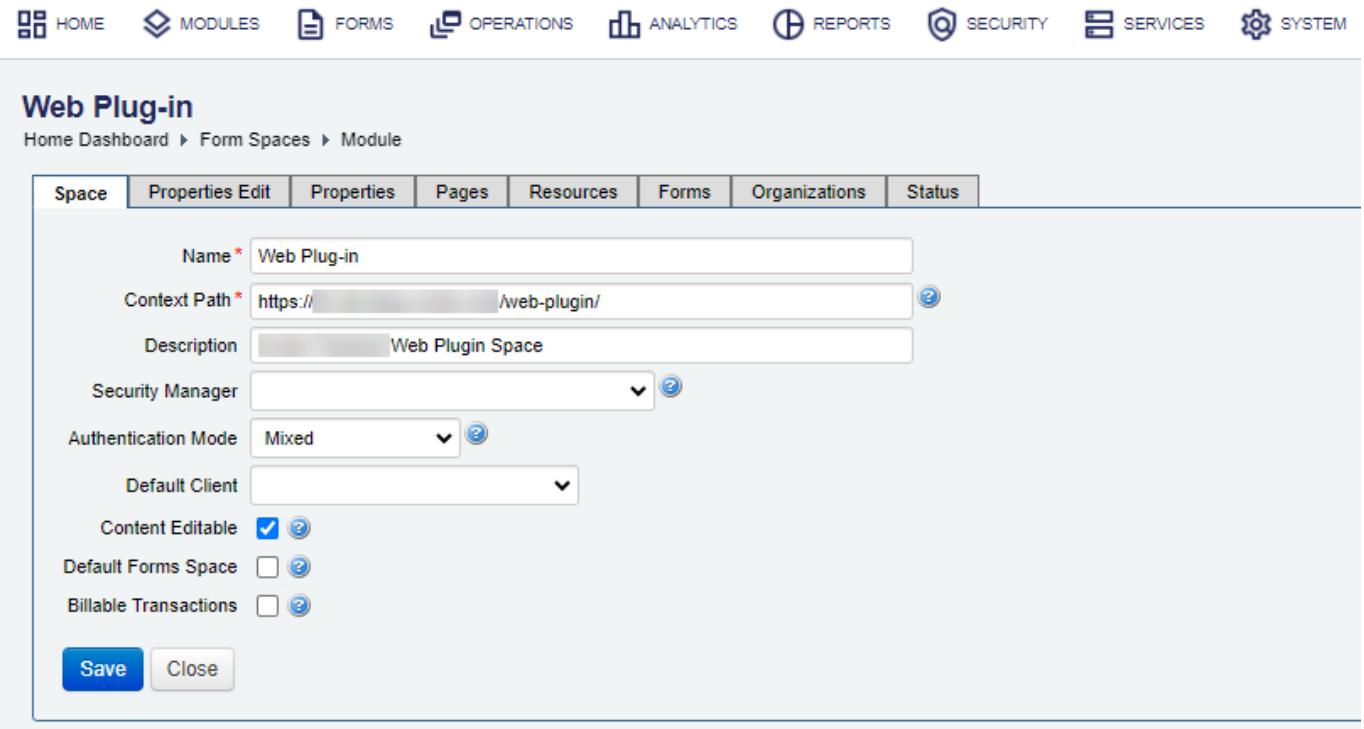
> NOTE
>
> Do not use this feature to upgrade to a new version of Manager. Follow the Manager upgrade instructions instead. If you have already used this feature to upgrade, verify that your form space is operating as expected.

Next, learn how to migrate a form space.

Web Plug-In Form Space

Manager comes with the default Web Plug-In form space, which exists on every server, to ensure no errors occur on client systems during importing and publishing forms. It's also used to monitor a Manager server, so you should not remove it. However, you can change the form space's configuration and update its properties.

You can log in to this form space without user authentication, as there is no security manager configured and the Authentication Mode is set to Anonymous Only, as shown below.



This form space comes with the default pages, which HTML source code you can edit to meet your particular requirements. Select the Pages tab to view them. It's important to know what pages are there and what's been manually modified, as it helps in smoother environment migrations and upgrades. Some releases of Manager has pages upgraded or even removed.

Page Name	Description
Logout	User logout page.
Not Authorized	User not authorized to access resource page.
Not Supported	Users browser is not supported page.
Offline	Forms offline system information page.
Receipt Challenge	Receipt challenge page.

Save Challenge	Saved form challenge page.
Secure Form	Secure form rendering page.
Secure Job Tasks	Secure job tasks page.
Server Busy	Server overloaded busy page.
Session Expired	Form session expiry page.
Session Reestablished	Secure page to reestablish a authenticated form session.
Submission Attachment	Form submission attachments page.
Submission Attachment Secure	Secure form submission attachments page.
Submission Cancelled	Submission cancellation page.
Submission Card Payment Hosted	Form submission card hosted payment page.
Submission Card Payment Hosted Secure	Secure form submission card hosted payment page.
Submission Confirmation	Form submission confirmation page.
Submission Confirmation Secure	Secure form submission confirmation page.
Submission Email Verification	Submission email verification page.
Submission Errors	Form submission errors page.
Submission Expired	Form has expired submission information page.
Submission Failed	Form submission failed information page.
Submission Payment Hosted	Form submission hosted payment page.
Submission Payment Hosted Secure	Secure form submission hosted payment page.
Submission Saved	Form saved submission confirmation page.
Submission Saved Secure	Secure form saved submission confirmation page.
Two Factor Authentication Setup	Account 2 Factor Authentication setup page

If you try to build a Manager form version in Maestro from an organization that does not have access to the Web Plug-In form space, you will get the error: `This design does not have access to the Web Plug-in Space`. In this case, you have to assign your organization to the Web Plug-In form space.

Once the form has been imported into Manager, you can assign a form space to the form.

NOTE

The Maguire form space was the default form space in previous versions of Maestro. The Maguire form space is an example only and should only be used in a development environment. The Maguire form space is an installation verification sample, which is used to ensure a server is configured properly.

Next, learn how to view all form spaces.

Host an Embedded Form

Manager allows you to host an embedded form in a form space's web page, so your form appear as if it were a seamless part of an existing website. There are various ways of doing this, which are discussed in integrate forms to existing websites.

This article focuses on the process of embedding a form inside an iframe element. In particular, it shows how to use a JavaScript helper method, showEmbeddedForm, to embed your form.

Reasons for employing the iframe method can include:

- Customers may not wish to have branding included in the form, preferring instead to manage their own branding within the pages where they want to show their minimally or non-branded forms
- Customers may wish to take advantage of controlled advertising services to surround the form with dynamic ads
- URLs will always reflect the customer's domain, rather than an external URL if the form is presented via a redirection
- Embedding the form may provide a more consistent user experience across the whole web site

How does it work?

Transact Manager includes the ability to embed a form inside a Form Space page. This is accomplished by JavaScript code which injects an iframe element into the page, and renders a form into it.

This guide will show you how to configure Transact Manager through the management console to accomplish this.

In following this guide, you will be:

- Setting up appropriate security access permissions to your form:
    - Assigning the Organization which owns the form to the Form Space
    - Assigning the Form to the Form Space
- Editing a Page in the Form Space to force it to show the Form. In this article, we will be editing the 'index.htm' page to host the form
- Inserting JavaScript in the page to cause it to create an `iframe` and render the form into it

The prerequisite for this guide is that you already have created your form in your Organization.

Step-by-step guide

1. If you need to create a new Form Space, follow the instructions atManaging Form Spaces
2. Ensure your Form Space is functioning correctly and can be accessed from a browser
3. Ensure your Form is built and can be rendered in one of the standard Form Spaces, such as Maguire or Web Plug-In
4. Open the Form Space in Transact Manager, select the Organizations tab, and assign the Organization which owns the form to the Form Space
5. Open the Form in Transact Manager, select the Spaces tab and assign the Form Space to the Form. Take note of the Form Code in the Form Details pane of the Form's Dashboard tab
6. Open the Form Space again, and select the Pages tab
7. Click the 'Index' link in the Name column of the Pages list (do not click the index.htm link in the Path column)
8. Paste this code into the Page Content box, change 'your-form'code' to your own form code, then click Save. You might also want to change the style to suit your form.

```
<style>
    #formDiv {height: 1000px;}
    .topSpace {display: none;}
    iframe {height: 100%;}
</style>
<div id="formDiv">
</div>
<script type="text/javascript" src="$context/smartform-webplugin.js"></script>
<script type="text/javascript">
    var replacableParamsMap = { id: '', formCode: '' };
    var queryParamsMap = { id: 'formDiv', formCode: 'your-form-code' }, tuple;
    var queryParams = document.URL.split('?')[1];
    if (typeof queryParams != 'undefined') {
        queryParams.split('&').forEach(function (queryParam) {
            tuple = queryParam.split('=');
            var paramName = tuple[0];
            var paramValue = tuple[1];
            if (typeof replacableParamsMap[paramName] === 'undefined' && typeof
queryParamsMap[paramName] !==  'undefined') {
                queryParamsMap[paramName] = [queryParamsMap[paramName]];
                queryParamsMap[paramName].push(paramValue);
            }
            else {
                queryParamsMap[paramName] = paramValue
            }
        });
    }
    Avoka.SmartForm.WebPlugin.showEmbeddedForm(queryParamsMap);

</script>
```

9. Open the Form Space again and select the Pages tab
10. Now click on the 'index.htm' link in the Path column - a new web browser tab or window should open, and the form should render into it

Explanation of Form Page Content

The JavaScript code above simply gathers query parameters from the URL, adds a couple of default values, and calls the showEmbeddedForm function to do the heavy lifting, passing the collected parameters.

The code contains the following:

- Some style to give the embedded form some height and to hide headers
- A JavaScript library which contains the showEmbeddedForm function
- A JavaScript script containing
    - A queryParamsMap map object which is populated with:
    - default 'id' value which identifies a div on the web page to contain the injected iframe
    - default 'formCode' value to identify the form to be rendered into the iframe
    - Some code to take all of the query-string parameters from the URL and add them to the queryParamsMap
    - If the parameter name is reserved 'id' or 'formCode', it will be replaced in the map with the value from the query-string parameter
    - If the parameter name is not reserved, it will be added to the map if it is not already mapped
    - If an unreserved parameter name has already been mapped, the map value will be converted to an array and the new value appended to it
    - A call to the showEmbeddedForm function, which is passed the accumulated map as a parameter
    - The function will use the 'id' and 'formCode' values to inject an iframe into the 'id' HTML element and render the 'formCode' form into it
    - All other parameters will be passed to the showEmbeddedForm function

This is illustrated below:

Page Content

```
1  <style>
2      #formDiv {height: 1000px;}
3      .topSpace {display: none;}
4      iframe {height: 100%;}
5  </style>
6  <div id="formDiv">
7  </div>
8
9  <script type="text/javascript" src="$context/smartform-webplugin.js"></script>
10 <script type="text/javascript">
11     var replacableParamsMap = { id: '', formCode: '' };
12     var queryParamsMap = { id: 'formDiv', formCode: 'creditcard' }, tuple;
13     var queryParams = document.URL.split('?')[1];
14
15     if (typeof queryParams != 'undefined') {
16         queryParams.split('&').forEach(function (queryParam) {
17             tuple = queryParam.split('=');
18             var paramName = tuple[0];
19             var paramValue = tuple[1];
20
21             if (typeof replacableParamsMap[paramName] === 'undefined' && typeof queryParamsMap[paramName] !==  'undefined') {
22                 queryParamsMap[paramName] = [queryParamsMap[paramName]];
23                 queryParamsMap[paramName].push(paramValue);
24             }
25             else {
26                 queryParamsMap[paramName] = paramValue
27             }
28         });
29     }
30
31     Avoka.SmartForm.WebPlugin.showEmbeddedForm(queryParamsMap);
32
33 </script>
34
```

You may need some style to show the form the way you like it

This script contains the showEmbeddedForm function

This identifies the formCode and the HTML element into which an iframe will be created and the form rendered

This code will render the form into an iframe

Next, learn how to [configure form space properties](#).

Map a Template to a Model

Manager allows you to map a template to a model

The following table lists the model variable names and types for each page.

- Java type names are not fully-qualified unless clarity is needed. For example, class String is the normal java.lang.String unless otherwise noted.
- Java types marked as an Entity are in the package com.avoka.fc.core.entity unless otherwise noted.
- Model classes that "inherit from" other model classes inherit the variables of those classes. For example, the model for ErrorPage not only includes the variables listed for it, but also all of the variables for BorderPage.
- The fact that the model can include a given variable does not mean that the variable is non-null at runtime.

Page	Model Class	Variable Name	Java Type	Inherits From
(global)	PortalUtils-Form This is not a Model class, but a utility method used by Model classes to set certain properties in the Model from the Form PortalUtils.addFormAndVersionProperties (Page, Form)	form	(Entity) Form	
formProperties	Map<String, String>			
	BorderPage	headerPath	String	
serverBuild	String			
portal	(Entity) Portal			
detectJavaScriptTimeout	int			
disableUserProfileEdit	boolean			

allowUser-Creation	boolean			
permissionSet	Set<String>			
error.htm	ErrorPage	entityId	String	Border-Page
errorLog	(Entity) ErrorLog			
errorContext	String			
showDetails	Boolean			
messageKey	String			
errorMessage	String			
errorTitle	String			
submission-can-celled.htm	Sub-missionCancelledPage	formRend-erUrl	String	Border-Page PortalU-tils-Form
formCode	String			
submission-saved.htm	SubmissionSavedPage	sub-mission	(Entity) Sub-mission	Border-Page PortalU-tils-Form
		isAnonym-ous	boolean	
		formUrl	String	
		user-Account	(Entity) User-Account	
		user-Name	String	
		logoutMs-g	String	
		todoCoun-t	int	
		eval-uatedPag-	String	

e

Next, learn how to configure a form space email template.

How to Unlock a Transaction

Manager allows you to configure a form space as well as a form version with the Save Challenge Lockout Minutes property, which is the duration in minutes a submission is locked out when the maximum number of save challenge requests has been exceeded. This property can be set to 0, which makes a form transaction permanently locked out. In this case, you need an administrator to manually unlock it.

Another option is to unlock a form transaction using an API. Unfortunately, Manager doesn't have any setting or a Fluent API to unlock such a transaction, so the only way to do this is by using Core API Groovy code.

To unlock the transaction with a script:

1. Copy the following Groovy script to the Groovy console.

```
import com.avoka.fc.core.dao.DaoFactory
import com.avoka.fc.core.entity.Submission
import org.apache.cayenne.access.DataContext

Submission submission = DaoFactory.submissionDao.getSubmissionByTrackingNumber
("9367VSY")
submission.setSaveChallengeFailures(0)
submission.setSaveChallengeLastFailure(null)
DataContext.getThreadObjectContext().commitChanges()
```

2. Update the tracking code with the transaction reference you want to unlock.
3. Click Execute Script.

After you unlocked the transaction, update the Save Challenge Lockout Minutes property either on a form space or a form version level to make sure it will not happen again.

Next, learn how to a configure form space email template.

Upgrade a Form Space

Every time you upgrade a Manager server, you should also upgrade form spaces hosted on that server instance, so each form space you have configured in your environment will have a version number which matches the version of Manager you have upgraded to.

While the form spaces may continue to work without the upgrade, important security updates and new features may not be available to them. This can have detrimental effect on your application performance as well security scans, which might find raise issues because the form space mark-up has not been merged with the new mark-up of the latest product.

The most important thing when upgrading a form space is that you want to replace the old form space with a new one and maintain the existing forms and user relationships within Manager, so the forms continue to render and work as expected. To achieve this, you should follow several simple recommendations listed below:

- The form space WAR file name has not been changed
- The form space context has not been changed
- The form space name is the same. This is set in a Manager installer when creating default form spaces or directly in the `db-config1.xml` file of a portal element, for example, a portal name is `testPortal`, if you are manually building a form space.
- Replace the form space in the deployment directory. On a single node system, you can deploy the war file through the Manager dashboard.
- Merge styles and mark-up changes

Depending on the content hosted by form spaces, the upgrade process can be one of the following:

Content Only Spaces

The Manager installer automatically upgrades the default form spaces, such as Workspaces (`tm-workspaces.war`), Web Plug-in and so on, but you have to rebuild any other form spaces manually.

For an environment that has content only form spaces, which are not custom Java spaces, this is a relatively simple process. A content only form space contains the base page mark-up, which is used in Manager by the page and resource merging facilities. If the WAR file is not rebuilt to represent the Manager version you are upgrading to, then you will not see any changes to the mark-up that have been made as part of the product upgrade. These may include new features and bug fixes or security patches.

The custom form spaces with content only spaces are upgraded simply by running the Manager installer used in the server upgrade. You can use a menu selection to provide the name of the form space and its context path as input.

Custom Java Spaces

Custom Java Spaces are more complex and the form spaces need to be manually prepared and the code level customization need to be merged into a clean Java Space for the version of Manager you are upgrading to. This can be a complex and time consuming activity, so we recommend you involve a form space developer tor help you re-build the form space.

Styles and Mark-up

Merging form and pages styles and mark-up is an important step which you may need to do to upgrade your form space. However, tt can be time consuming, especially if the styles or mark-up have been significantly changed from the standard product's set. Depending on where the styles and mark-ups are, you have the following options:

- If the form space WAR file includes all the styled pages, then you can use the Restore Base Content button located on both the View Form Space Resources and View Form Space Pages tabs of Manager.

  NOTE

  This will replace all styles cached in the Manager database.

- If the styles is in Manager then you have to merge in the pages and resources manually within the Manager console.

You usually need a developer if the original form space was customized. When the form space assets ZIP file is imported, it can bring with it some legacy code that ignores or misuses feature and security upgrades. Resolving this requires a manual review, so you need to do the following:

1. View form space resources.
2. Take note of the Base Content Date, Last Modified, and Merged columns. The Last Modified column will be populated in a red font if that resource has been changed. Comparing the Base Content Date and date in Last Modified can give a clue to the age of the modified content.

3. Click View Content Diff to Base Resource to open a new tab displaying the base resource next to the modified resource. Differences between the files are highlighted in various colors to make changes, additions, and deletions apparent.
4. Review all changes, and evaluate for functional impact. If a change only modifies a color, or font, or ordering or placement of screen elements, it may not impact security or new features and can be left in. Base content changes that introduce a new feature, or patch a bug, need to be examined and incorporated into the resource.
5. Use an editor suitable for the type of resource to make changes, and load the updated resource file into Manager. Depending on the file type, you can use the built-in text editor, or re-upload the file.
6. Test the updated resources, then click Mark Content Merged when the resource edits and tests are complete.
7. View form space pages and perform the same actions as for form space resources. The pages list only the page definitions of the form space. Pages differ from HTML stored as a resource. Pages can be navigated to as they are referenced as links, buttons, or tabs, in the form space and generally have a Java object backing them. Pages can be altered from base content and therefore need to be reviewed during an upgrade just like resources.

Next, learn how to a configure form space email template.

Journey Space

WARNING

The Work Space form space has been renamed to Journey Space, so there is no confusion with the Workspaces portal. The Context Path URL has also been changed from `https://my-domain.com/workspace/` to `https://my-domain.com/journeyspace/`.
For the changes to take effect, you need to have a fresh installation of Manager or the Journey Space form space. If you only upgrade Manager, you still have the old Work Space name with the old Context Path URL as `https://my-domain.com/workspace/`.

| 23.10This feature was updated in 23.10.

Manager comes with a Journey Space web application deployed as a specialized form space, which allows users and business staff to perform the following actions:

- Work with tasks and collaboration jobs.
- Complete and review forms.
- Access draft and saved forms.
- Provide help desk support.

Usually, a form is hosted as an unsecured page on the form space. An anonymous user must provide an email address when filling the form. Manager relies on this email to send status updates and saved transactions, and assign tasks back to the user. Additionally, one of the form fields can be used to provide a security challenge question, which the user must enter to open a saved form or anonymous task created by a collaboration job. However, Journey Space can be configured to apply various user authentication flows, including 2FA[1], to strengthen its security. In this case, you need to log in to use the web application.

Journey Space provides users with the following views:

---

[1]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.

- Home is the Journey Space dashboard.
- Forms allows you to find and select from the list of available forms.
- Tasks allows you to complete your outstanding forms and tasks.
- History provides you with the history view of forms you have submitted and their processing status.
- Reviews allows you to search for review and approval collaboration jobs.
- Help Desk allows you to search for forms saved or completed by your customers.
- Account allows you to edit your account settings and change your password.

> NOTE
>
> Some views are only shown to users with certain roles and privileges, see each view's documentation for details.

Journey Space is generally used for dealing with authenticated user transactions, but it also supports unauthenticated anonymous transactions. These permissions must be configured in Manager.

You should configure the group's permission to edit tasks, claim, assign and return them to a group in order to start using Journey Space. Manager provides several authenticated user roles for the Journey Space portal.

> NOTE
>
> You should use the Workspaces instead.

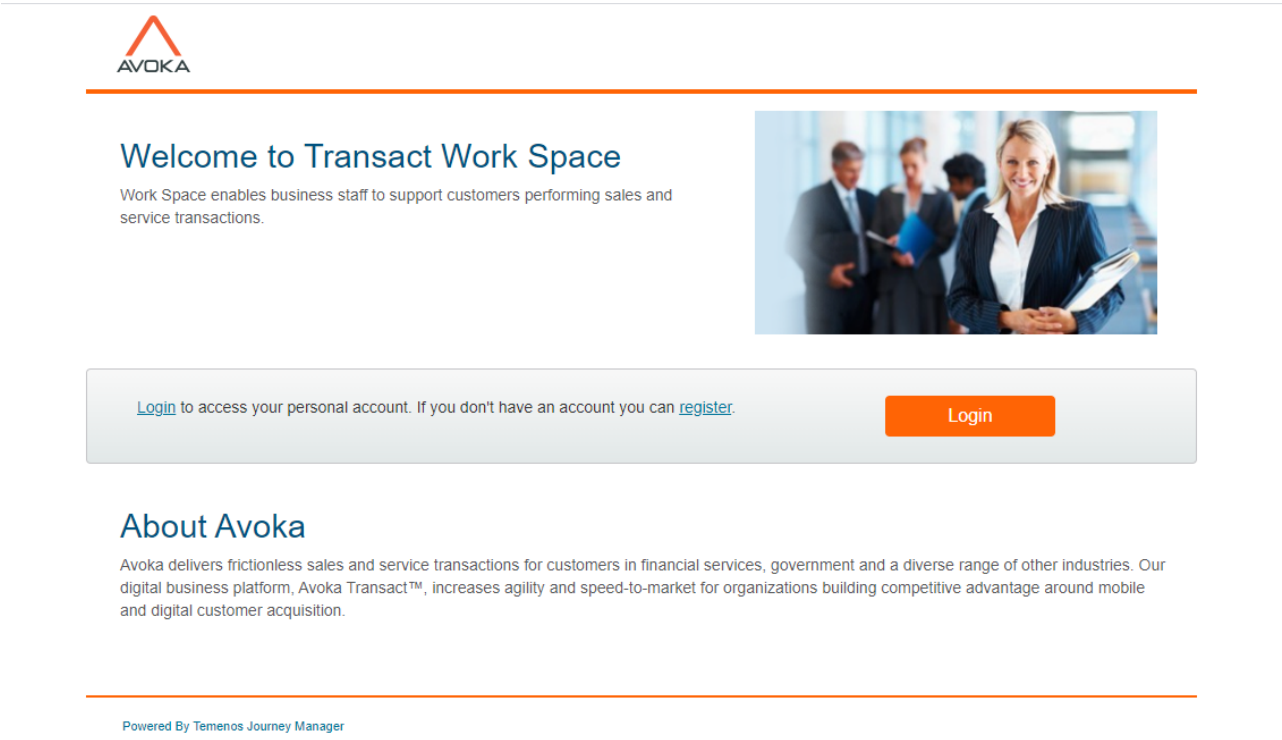
Next, learn how to configure Journey Space properties.

Journey Space Login

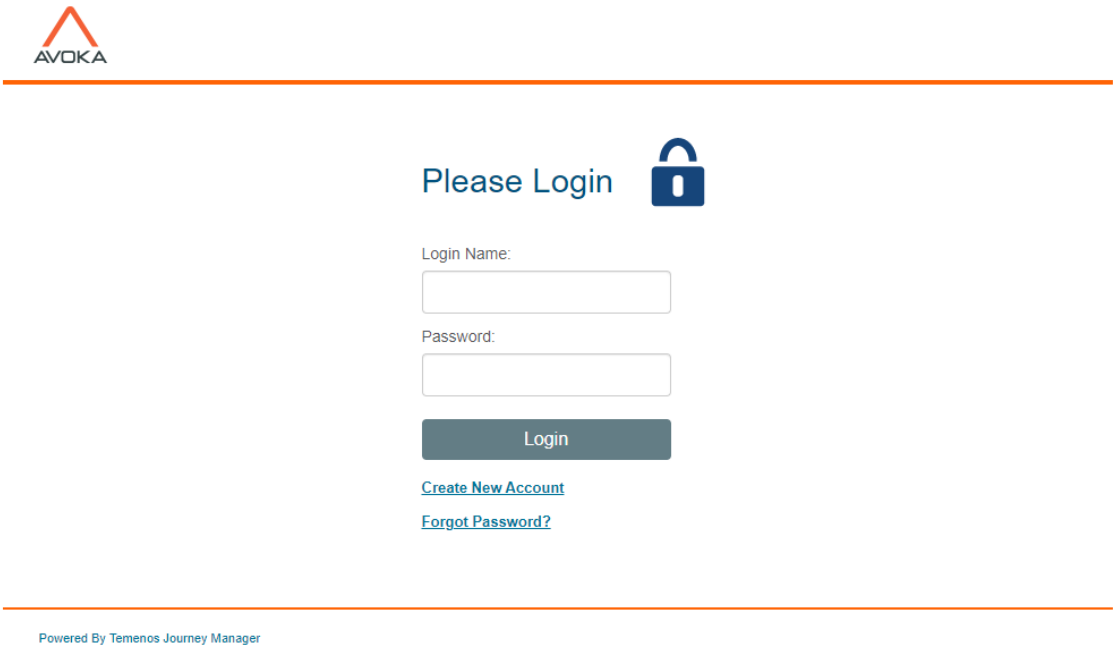
Journey Space can be configured to authenticate users before allowing them to use it. In this case, you need to log in using your account credentials. If you don't have your account, contact your system administrator or support team to create one for you.

To log in to Journey Space:

1. Go to the Journey Space URL in your browser, for example, `https://my-domain.-com/workspace/login.htm`

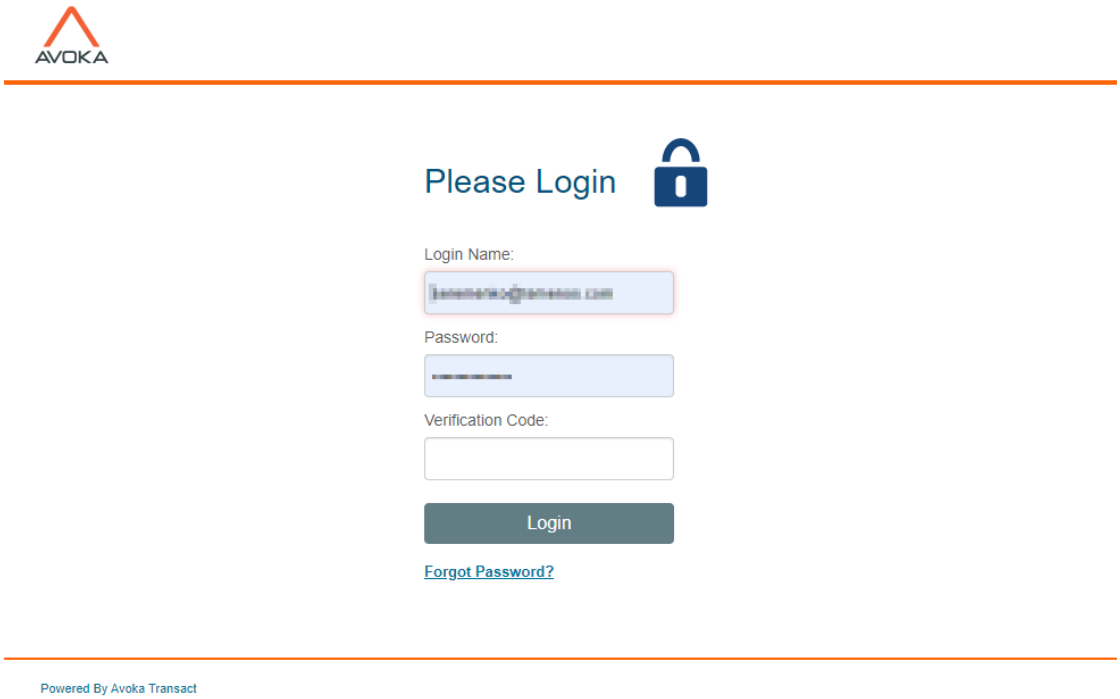


2. Click Login. Manager displays a login screen for you to enter your credentials.



Journey Space can be configured to use 2FA[1] to verify your identity, so you must provide an additional verification code to login.

[1]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.



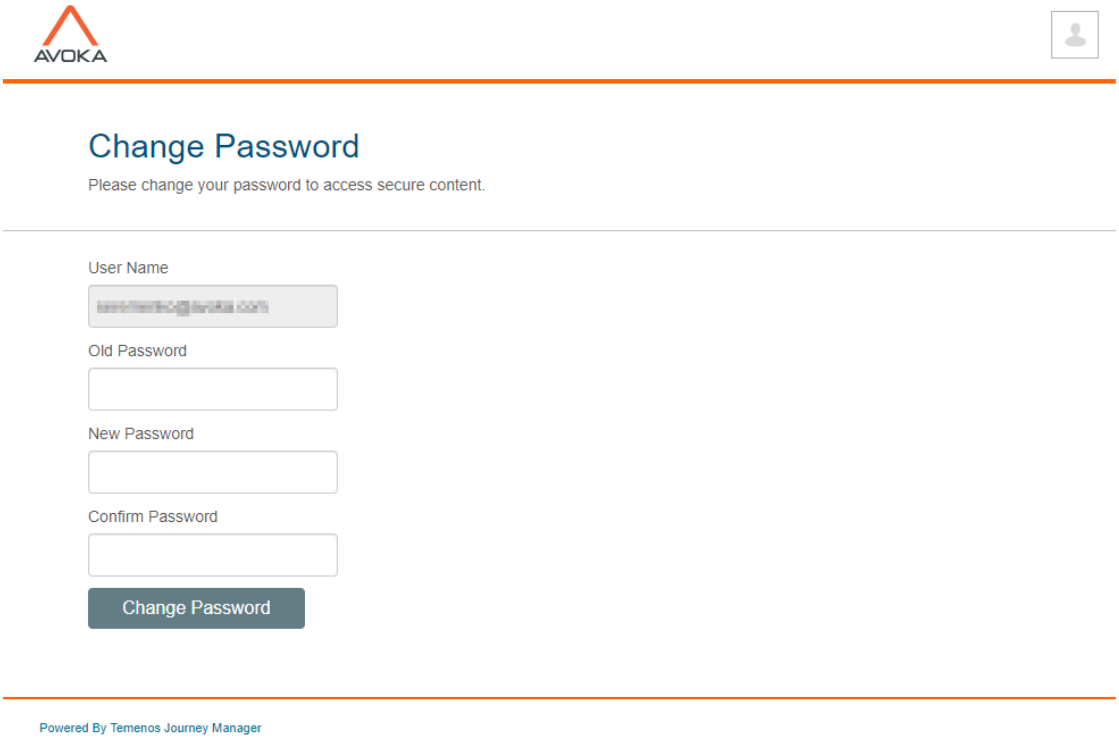
3. Provide your user name in the Login Name field.
4. Enter a password in the Password field.
5. Provide a unique one-time code in the Verification Code field. If this is your first login and you haven't set up your Authentication App, you don't need to enter the code.
6. Click Login. Manager checks your credentials and, if they are correct, redirects you to the home page. If your system is in the process of setting up 2FA, you may be taken to the enrollment page where you can start the 2FA enrollment process.

Next, learn how to change a password.

Journey Space Change Password

Journey Space can be configured to authenticate users with a user name and a password. As there are situations, such as a request to change a password on a first login or when your password was compromised, Journey Space allows you to do so.
To change a password:

1. Click a profile icon or the Account tab.

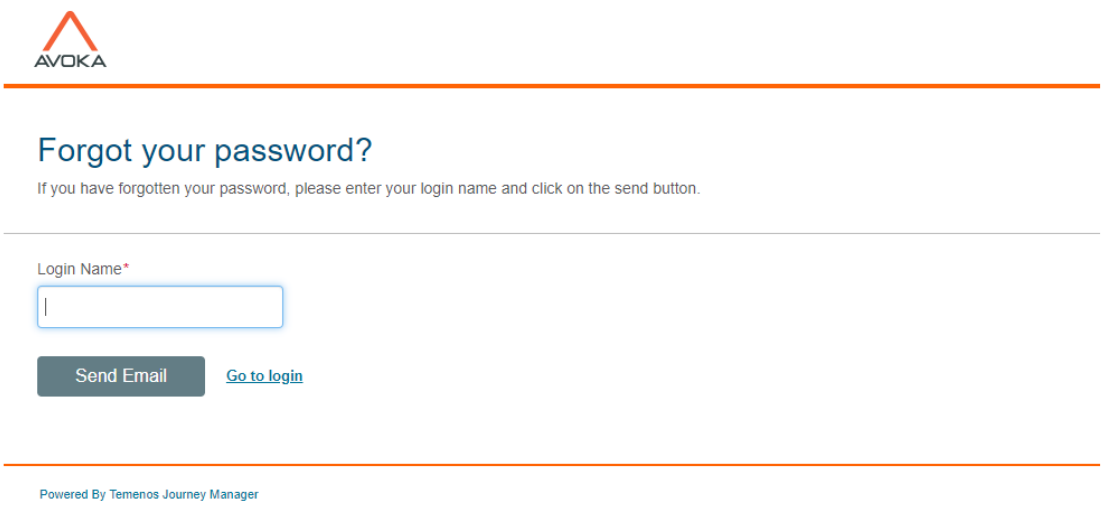


2. Provide the old password in the Old Password field.
3. Enter a new password in the New Password field and confirm it in the Confirm Password field.
4. Click Change Password.

If you have an existing Journey Space account but can't log in anymore, you must request a new temporary password.

To request a new temporary password:

1. Click Forgot Password on the login screen.



2. Provide your login name in the Login Name field.
3. Click Send Email. Manager will send a new temporary password to an email address configured in your user account.
4. Try to log in again and you will be asked to change your password.

If you still can't access Journey Space with no clear indication why, try the following actions to resolve it:

1. Check Login History to see any login failures and lockouts. Account unlocked and failures continue.
2. Click Reset 2FA Key to request a new verification code. Check there is not error message displayed when you request the reset.
3. Check what form spaces, the user account a part of, use a security manager with 2FA.
4. Revoke all form spaces from the user account, but the one with 2FA.
5. Click Reset 2FA Key again to generate a new verification code.
6. Check your email box and follow the instructions install and configure the Authenticator App. You may need to re-sync your Google Authenticator App.
7. Log in to Journey Space again using the new code.

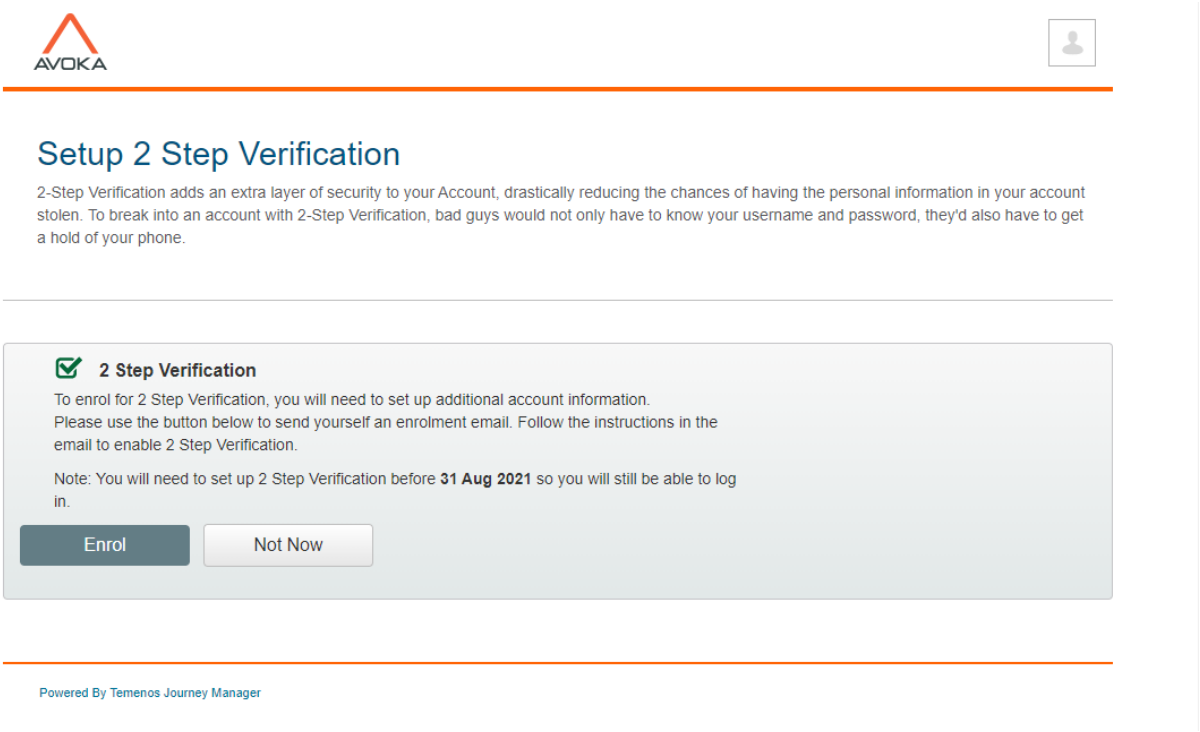
Next, learn about the Home tab.

Journey Space 2 Factor Authentication

Journey Space can be configured to authenticate users using a user name, a password, and an additional verification code, which is 2FA[1]. In this case, on your first login, you are asked to set up 2 step verification.

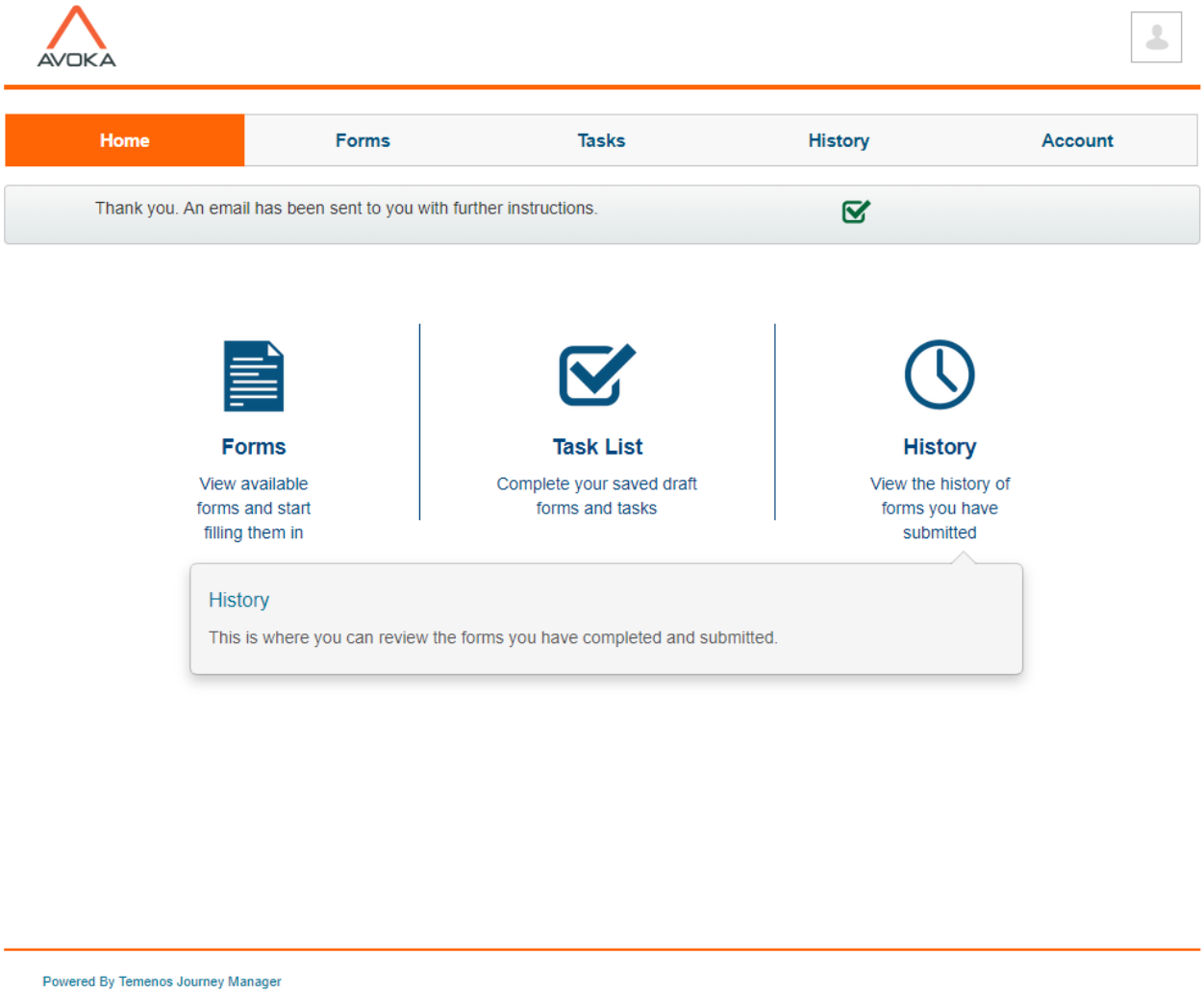
To set up 2 step verification in Journey Space:

1. Log in to Journey Space using your user name and password.
2. If your system (a form space that manages your forms) is in the process of setting up 2FA, you may be taken to the enrollment page where you can start the 2FA enrollment process.



[1]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.

3. Click Enrol to configure it now. Manager sends you an enrollment email with 2 Factor Authentication instruction. You need to complete the instructions before you will be able to login again.



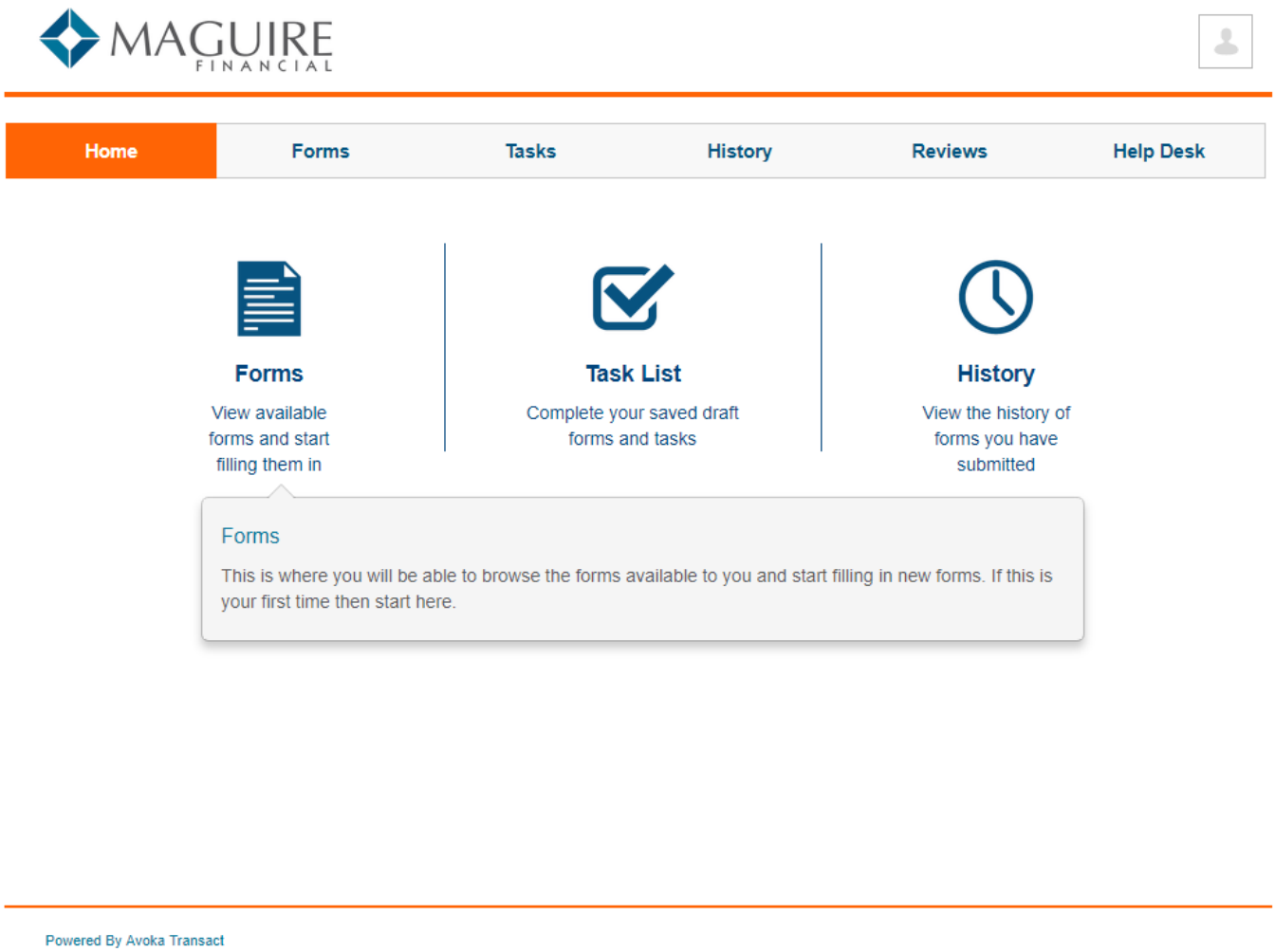
Otherwise, click Not Now to ignore it for now - you will be prompted to set up your 2FA next time to log in to Journey Space. Manager redirects you to the home page.

4. Open an email sent to you by Manager and follow the instruction to install and configure an Authenticator App.
5. Log out of Journey Space and log in again using your credentials and a verification code from the Authentication App.

Next, learn about the Home tab.

Journey Space Home

Journey Space comes with the Home tab, which is the dashboard of this application, that allows users to view relevant form and task details and navigate between the tabs.



The form is hosted as an unsecured page on the form space. An anonymous user must provide an email address when filling the form. Manager relies on this email to send status updates and saved transactions, and assign tasks back to the user. Additionally, one of the form fields can be used to provide a security challenge question, which the user must enter to open a saved form or anonymous task created by a collaboration job.

Journey Space provides users with the following views:

- Home is the Journey Space dashboard.
- Forms allows you to find and select from the list of available forms.
- Tasks allows you to complete your outstanding forms and tasks.

- History provides you with the history view of forms you have submitted and their processing status.
- Reviews allows you to search for review and approval collaboration jobs.
- Help Desk allows you to search for forms saved or completed by your customers.
- Account allows you to edit your account settings and change your password.

NOTE

Some views are only shown to users with certain roles and privileges. For more information, see permissions.

Journey Space is generally used for dealing with authenticated user transactions, but it also supports unauthenticated, which are anonymous, transactions. These permissions must be configured in Manager. You should configure the group's permission to edit tasks, claim, assign and return them to a group in order to start using Journey Space. Manager provides several authenticated user roles for the Journey Space portal.

NOTE

You should use the Workspaces instead.

Next, learn how to configure Journey Space's properties.

Journey Space Forms

Journey Space provides the Forms tab that lists all forms available to a user to start working on.

To filter or search forms:

1. Enter a search string in the search text field.
2. Click Go.
3. Click a form from the list to start working on.

Manager uses form groups to restrict the list of forms that are shown to the user. By default, a form is not assigned to a form group so it will appear in the list. When a form group is assigned to a form, and the form group has the New Form permission set, then it will be only visible to members of the form group.

Next, learn about Journey Space tasks.

Journey Space Tasks

Journey Space has the Tasks tab that lists saved forms, assigned tasks and submitted forms available to a user to start working on. Here, the submitted forms are ones that require additional attachments or payments and are not used the most. However, most collaboration jobs use the in-form transaction management features of the form space, which means the submitted forms will be empty.

To filter or search tasks:

- Enter a search string, for example, a job name, in the search text field.
- Select the time interval, when collaboration jobs were started, from the Last dropdown list The default is 90 days.
- Select a collaboration job status from the Status dropdown list, which can be assigned, saved or submitted.
- Select a form group from the Groups dropdown list, which can be:
  - Blank - for any tasks that are assigned to the user account that you are logged in with.
  - All Groups - for all form groups.
  - A chosen form group.

    NOTE

    This selection is only available if a form space has the Search Filter Group Select checkbox selected. For more information, see configure a form space properties.

- Select the Group Items checkbox to show group items only .

A collaboration job creates tasks and assigns them to an authenticated or anonymous user, or a group. An assigned task can be one of the following service types:

- Form Task - the form XML can be pre-filled. Assigned to a user or a group and displayed to secure users in the Task tab.
- Review Task - the review task copies the form XML and attachments from a previous submission. Assigned to a user or a group and displayed to secure users in the Task tab.
- Anonymous Task - the task can only be assigned to an email. The anonymous form XML can be pre-filled.

Next, learn about Journey Space history.

Journey Space History

Journey Space comes with the History tab that shows a list of all user's submissions. A submission that starts a collaboration job can contain a processing status that is updated as the collaboration job progresses through its steps.

The user can filter the result using a search and selecting all group submissions checkbox.

Next, learn about Journey Space reviews.

Journey Space Reviews

Journey Space has the Reviews tab that allows job coordinators to list all reviews and approvals of collaboration jobs and also search for a particular collaboration job to view the following information:

- The collaboration job's details.
- The current step of the collaboration job.
- The current collaboration job's task and its status.
- The collaboration job's history, including previous submissions and receipts.

To filter or search the collaboration jobs by one or more criteria, specify the following settings and click Go:

- Enter a job number, tracking code, or contact email in the search text field.
- Select the period of time from the Last dropdown list.

  NOTE

  Only last 90 days are listed by default.

- Select a status of collaboration jobs from the Status dropdown list to filter all available collaboration jobs, which can be one of the following:
  - In Progress: to see only collaboration jobs for review and approval with the In Progress status. This is the default option.
  - Blank: to see all collaboration jobs.
- Select a group from the Job Groups dropdown list, which can be one of the following:
  - [All Groups]: to see all collaboration jobs with all job groups that you are responsible for. This is the default option.
  - [No Groups]: to see collaboration jobs that are not allocated to any job group. This option is required because jobs not allocated to any job group do not appear when [All Groups] is selected. If your company allocates all collaboration jobs to job groups, this option will allow for the changeover period where you can still see jobs before the job groups were introduced.

A user can click on the heading of completed forms to download a task's receipt.

A coordinator can perform one of the following actions for tasks that are Assigned or In Progress:

- Claim a task
- Claim and open a task

- Assign to and return to the group (if claimed first)

Next, learn about Journey Space help desk.

Journey Space Help Desk

Journey Space comes with the Help Desk tab that allows you to search for forms that have been saved, completed, or abandoned by users.

By default, the Help Desk is not shown to authenticated users and doesn't display any forms. You have to configure the following role permissions in Manager to allow users to use it:

1. Assign the Work Space Staff role to the users so they can see the Help Desk tab in their work space.
2. Make sure they have the Help Desk View permission so they can view the Help Desk tab.
3. Assign the Help Desk Authenticated Edit permission to the users so they are able to work with customers' transactions.

You can customize the help desk page according to customer's requirements. It is often the case for companies with large transaction volumes, when special configuration needs to be performed to be able to filter form submissions within their form space help desk page:

- The user can filter the help desk list by a date range using the Last dropdown. | 18.05 This feature was introduced in 18.05.
- The user can filter the help desk list by form.
- The user can filter the help desk list by form group, for example, to include authenticated users.
- The Help Desk page doesn't automatically load form submissions when it's displayed first time, so the user can revise the filters first and then click Go to fetch the required transactions. This is particularly useful for system with large volumes of transactions.

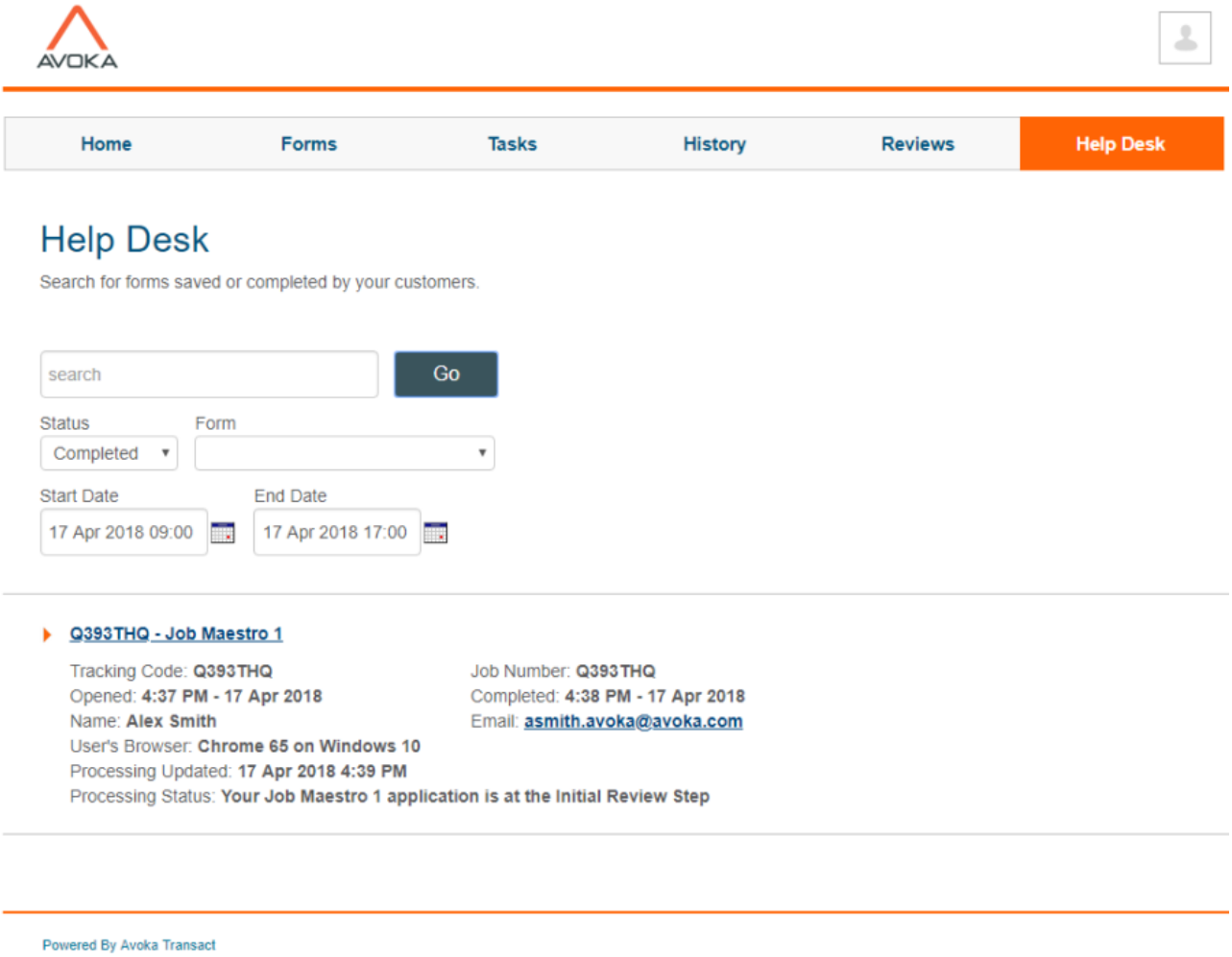
To demonstrate the recommended configuration, consider the following examples for anonymous and authenticated user form submissions.

Anonymous Users Only

Help desk users often check specific transactions and enter their tracking code in the search box and the click Go. To limit the volume of displayed transactions, you can use these filtering options:

1. Enter a complete transaction's job number, tracking code or one of the data extract fields in the search text field. By default, any saved forms are listed.
2. Select a transaction status from the Status dropdown list to see the forms that were saved, completed or abandoned.

3. Specify a date range from the Start Date and End Date. A Start Date must be selected before any submissions are displayed.
4. Select a form name from the Form dropdown list.
5. Click Go to fetch the list of transactions, as shown below.

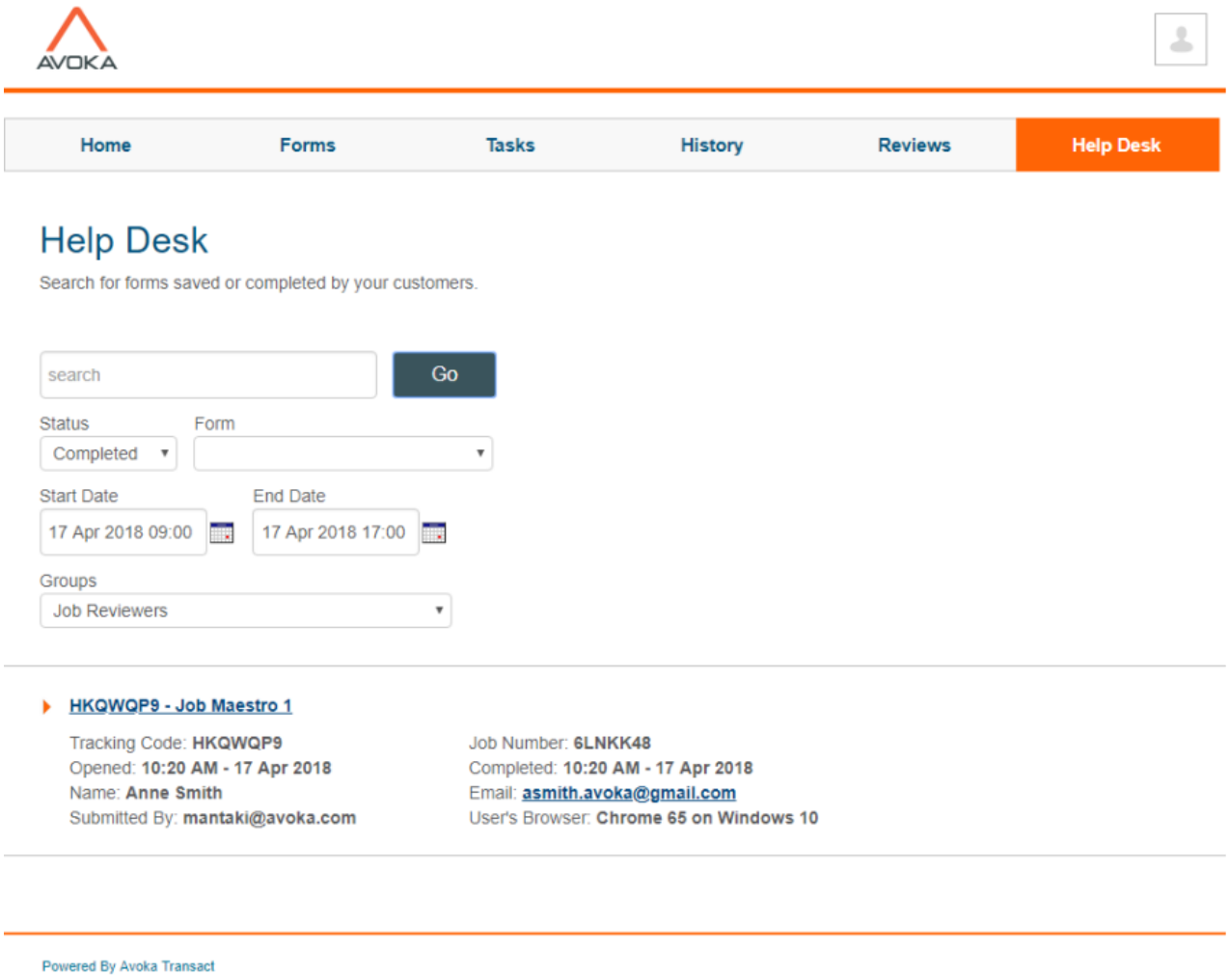


Anonymous and Authenticated Users

You can configure the help desk page to include submissions from authenticated users. This allows help desk users to narrow down their searches with the Groups filter. To limit the volume of displayed transactions, you can use the filtering options explained above with this additional option:

1. Select a group from the Groups dropdown list, which has the following options:
    - Blank to search for forms that are not assigned to any group. This is the default option.
    - [All Groups] to search for forms that belong to all form groups.
    - An individual form group name for forms that belong to a specific form group.

2. Click Go to fetch the list of transactions, as shown below.



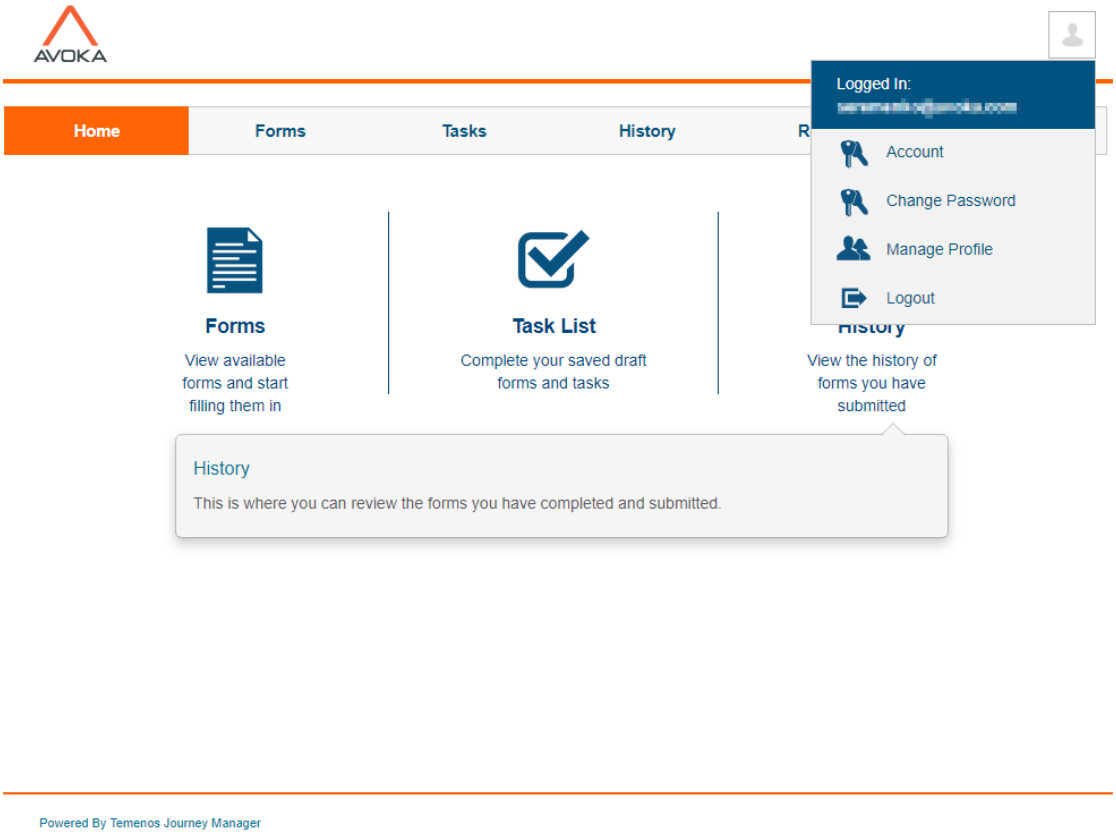
Next, learn how to update account settings.

Journey Space Account

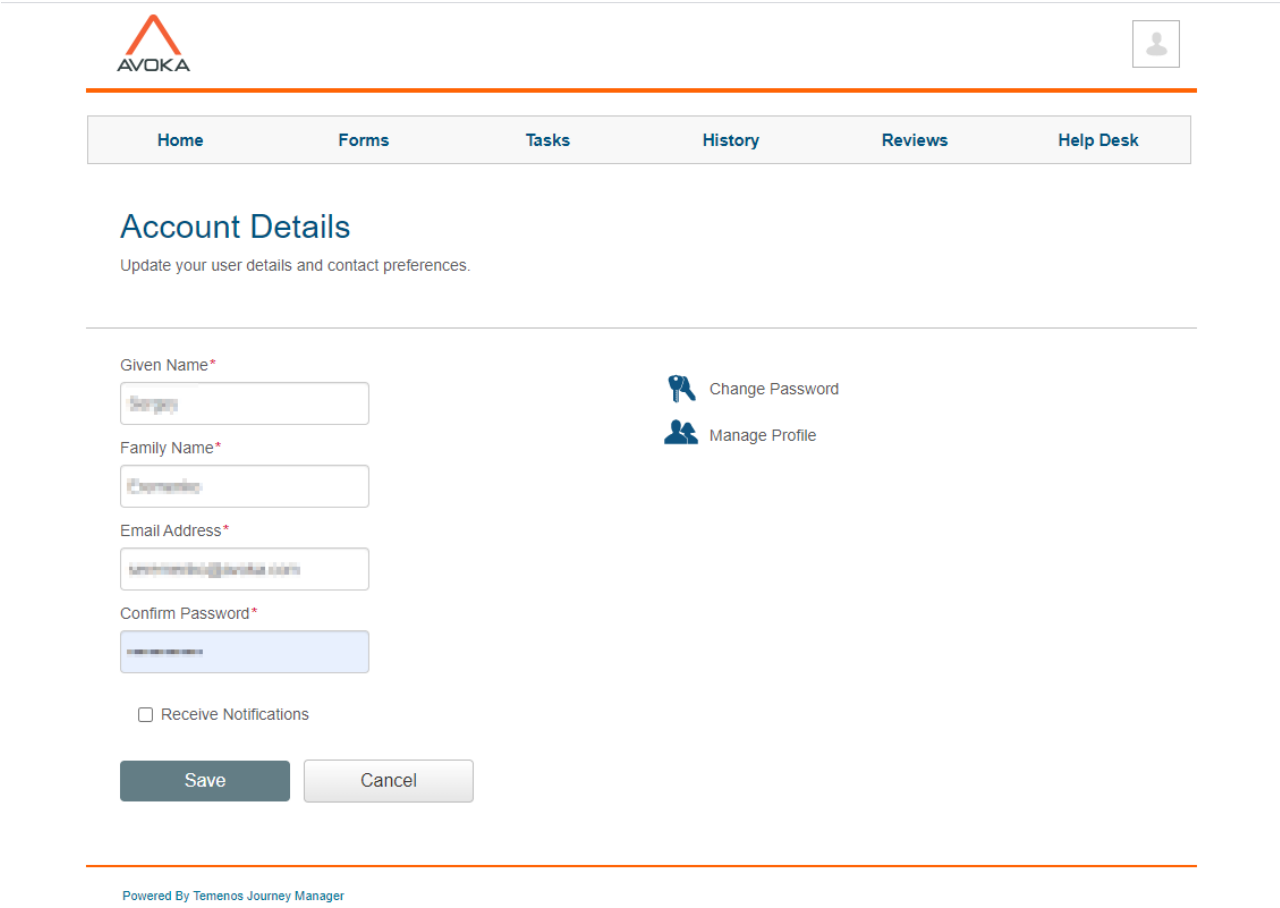
Journey Space allows you to manage the user account and profiles, as well as change a password.

To configure an account:

1. Click a profile icon or the Account tab.

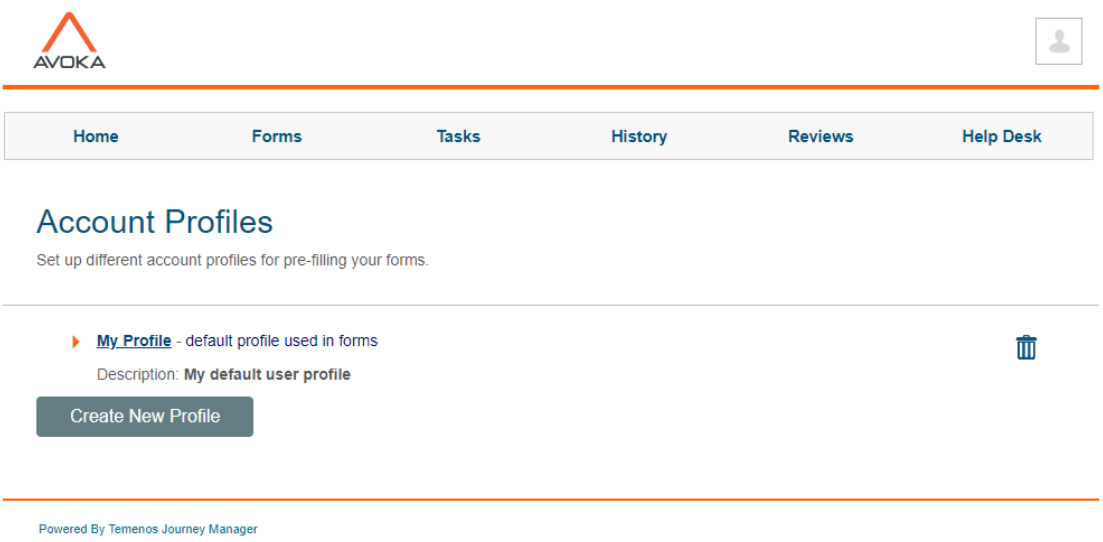


2. Select Account to update your details.

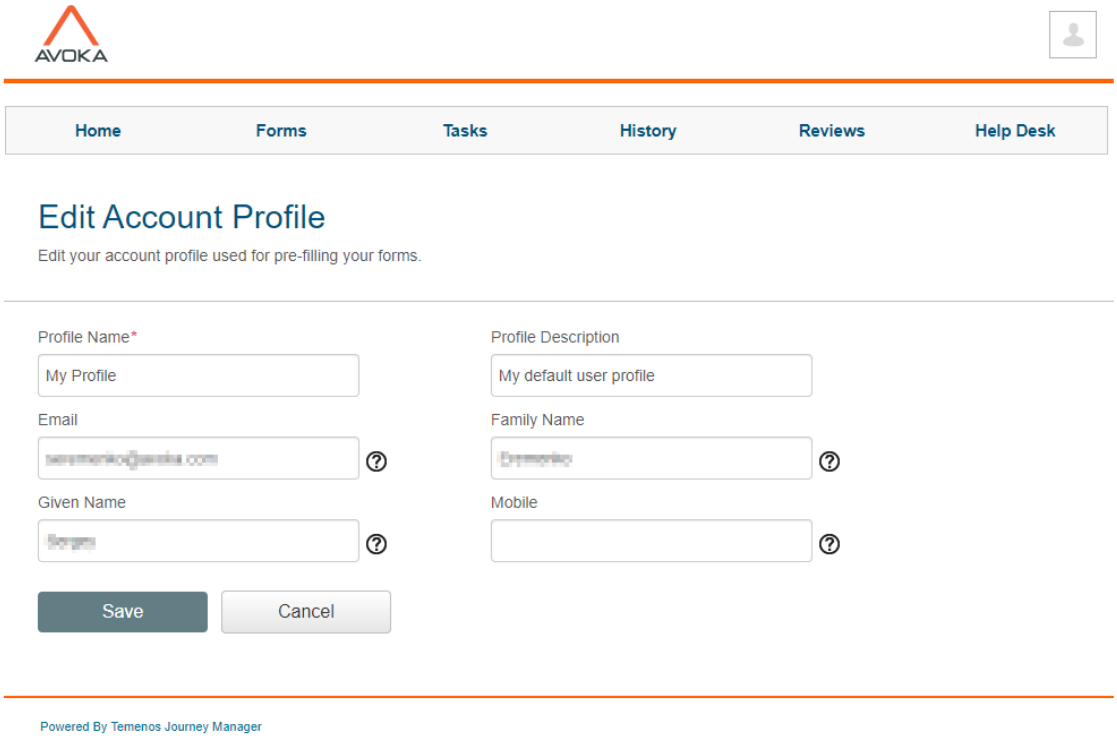


3. Check and update your details and select the Receive Notifications checkbox to allow Manager to send you email notification on various activities, such as tasks and reviews.
4. Select Change Password to change your password.

5. Select Manage Profile to create new or update an existing profile.

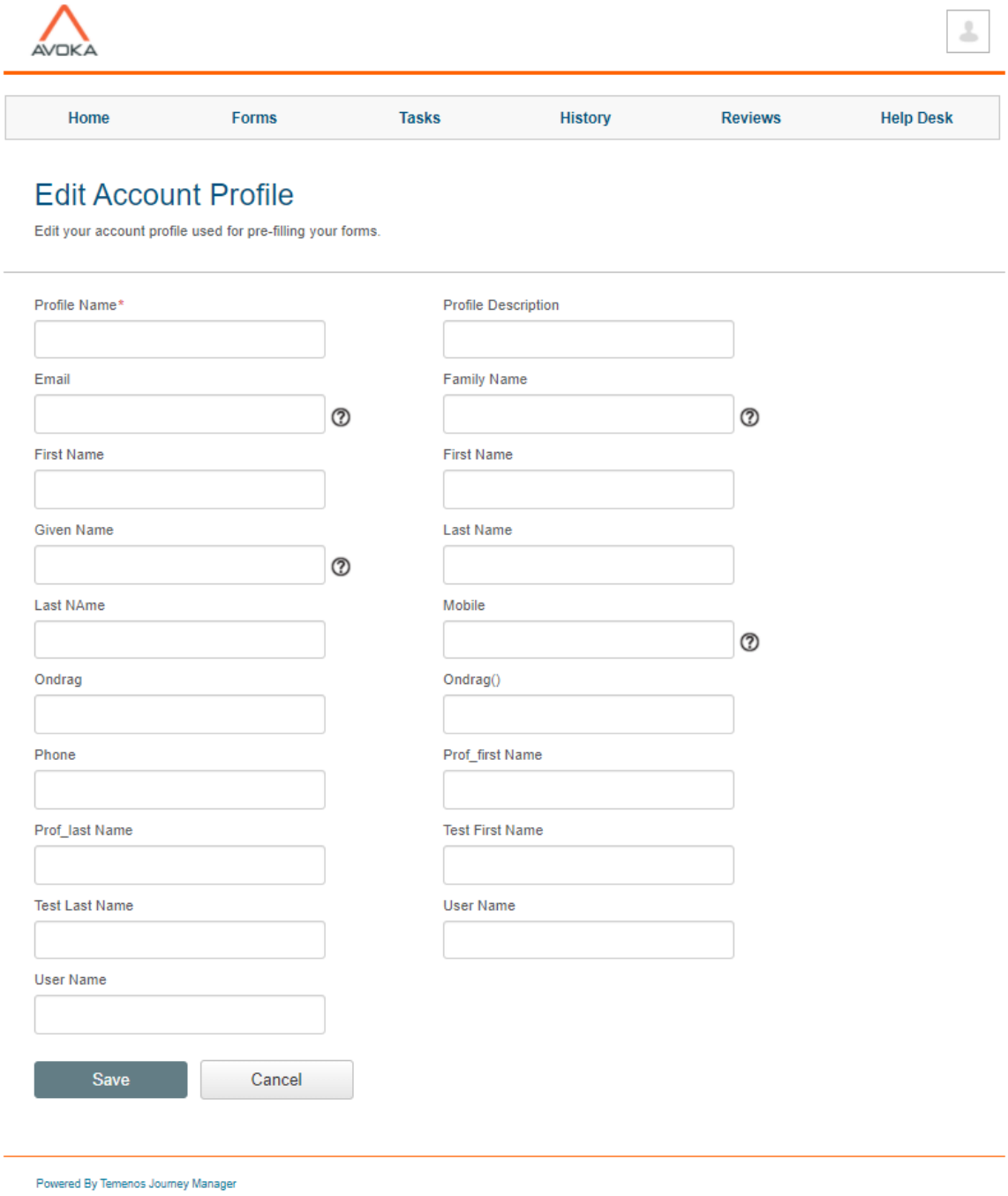


6. Select an existing user profile to view or edit it or click Delete Profile to remove it.



7. Provide a profile name and description, complete all personal details. Click Save to update the changes.

8. Click Create New Profile to create a new profile.



9. Provide all required information and click Save to create a new profile.
10. Select Logout to exit Journey Space. You will be redirected to the home page.

Next, learn about the Home tab.

Configure Journey Space Properties

Manager allows you to configure various properties of your Journey Space pages.

> NOTE
>
> You must be a Journey Manager administrator to create or configure this functionality.

To configure the page's properties:

1. Select Forms > Form Spaces.
2. Locate the Journey Space and click Edit.
3. Select the Properties Edit tab to view the following properties:
    - Search Consistent Read
    - Search Data Extracts
    - Search Data Load
    - Search Fetch Limit
    - Search Filter Group Select
    - Search Filter Jobs By User Orgs
    - Search Period Help Desk
    - Search Period History
    - Search Period Reviews
    - Search Period Tasks
    - Search Task Attributes
4. Click Edit to and update each property value as required.
5. Click Save to update the changes.

Searching and filtering options of Journey Space also depend on Manager's version, individual form space configuration and specific pages' customization. You should consider the following:

- Remove the Last dropdown from the page, because of the new, more specific date range, such as Start Date and End Date pickers. Otherwise, it can become confusing to provide both.
- A form space can be configured in such a way that when switching to the Journey Space page, the form submissions are not initially loaded. The user can select filter options and then click Go. This option is particularly useful for the help desk page where performance issues may arise from large volumes of transactions that need to be searched.
- A form space can be configured to allow a user to filter by form groups. This is only relevant to display authenticated form user transactions.

The Journey Space configuration varies for anonymous and authenticated users.

To include the anonymous user transactions only, configure the help desk markup with the following filter options:

- Search box
- Status dropdown
- Form dropdown
- Start Date picker
- End Date picker

To include the anonymous and authenticated user transactions:

1. Configure the help desk markup with the following filter options:
    - Search box
    - Status dropdown
    - Form dropdown
    - Start Date picker
    - End Date picker
    - Groups dropdown
2. Configure the work space staff user role:
    - Assign the Journey Space Staff role to the users so they can see the Help Desk tab in their Journey Space.
    - Make sure they have the Help Desk View permission so they can view the Help Desk tab.
    - Assign the Help Desk Authenticated Edit permission to the users so they are able to work with customers' transactions.
3. Configure a Journey Space property to allow the help desk user to filter by form groups by selecting the Search Filter Group Select checkbox. This will display the Groups dropdown list for help desk users that are assigned to at least one form group.

Next, learn how to change Journey Space page markup.

Configure Journey Space Page Markup

Manager allows you to configure specific properties of your Journey Space pages and customize their markup.

> NOTE
>
> You must be a Journey Manager administrator to create or configure this functionality.

To customize the pages' markup:

1. Remove the Last dropdown list by finding and deleting these lines:

```
<span class="searchSelect smallDivider">
 Last
 <br/>
 $form.fields.startDate
 </span>
```

2. Add the Form dropdown list and the Start Date and End Date pickers as shown below:

```
## A new field for form select has been added  <span class="searchSelect smallDi-
vider"> Form <br/> $form.fields.formSelect </span>
## A new field for start date has been added <span class="searchSelect smallDi-
vider"> Start Date <br/> $form.fields.startDateField </span>
## A new field for end date has been added <span class="searchSelect smallDivider">
End Date <br/> $form.fields.endDateField </span>
```

3. Add the Groups dropdown list to include the authenticated user forms as shown below:

```
#if ($form.fields.groupSelect)
 <span class="searchSelect smallDivider">
 Groups <br/>
 $form.fields.groupSelect
 </span>
 #end
```

The complete example of the help desk page markup is shown below.

```
<style type="text/css">
.searchSelect {
    display: inline-block;
    margin-right:10px;
}
#form_startDate {
    width: 110px;
}
#form_formStatus {
    width: 110px;
}
#form_startDateField {
    width: 120px;
}
#form_endDateField {
    width: 120px;
}
#form_groupSelect {
    width: 330px;
}
#form_formSelect {
    width: 240px;
}
.calendar td {
    min-width: 0px;
}
</style>

## If using the startDateField or endDateField, ensure the two lines below are not com-
mented out
<script src="$context/click/jscalendar/calendar.js" type="text/javascript"></script>
<script src="$context/click/jscalendar/calendar-en.js" charset="UTF-8" type-
e="text/javascript"></script>
<script src="$context/resources/js/searchTable.js" type="text/javascript"></script>
<!-- find-form.htm-->
<div class="row" id="findform">
    <!-- Header -->
    <div class="span12 topSpace divider">
        <h1>Help Desk</h1>
        <p>Search for forms saved or completed by your customers.</p>
    </div>
```

```
    <!-- Search panel -->
    <div class="row">
        $form.startTag()
        <div class="row span6">
            <span class="searchField smallDivider">
                <br>
                $form.fields.search
            </span>
            <span class="searchButton smallDivider">
                $form.fields.go
            </span>
            <span class="searchSelect smallDivider">
                Status
                <br/>
                $form.fields.formStatus
            </span>

## A new field for form select has been added
            <span class="searchSelect smallDivider">
                Form
                <br/>
                $form.fields.formSelect
            </span>

## A new field for start date has been added
            <span class="searchSelect smallDivider">
                Start Date
                <br/>
                $form.fields.startDateField
            </span>

## A new field for end date has been added
            <span class="searchSelect smallDivider">
                End Date
                <br/>
                $form.fields.endDateField
            </span>

## New fields for group select and group filter have been added
            #if ($form.fields.groupSelect)
            <span class="searchSelect smallDivider">
```

```
                Groups <br/>
                $form.fields.groupSelect
            </span>
            #end
            #if ($form.fields.groupFilter)
            <span class="searchSelect smallDivider">
                Group Items <br/>
                $form.fields.groupFilter
            </span>
            #end

        </div>
        $form.endTag()
    </div>
</div>
<div class="row"><hr></div>

## See template in: /resources/includes/account/tile.htm
<div id="tableHolder" class="row">
    $tileList
</div>
```

Next, learn how to configure Journey Space page properties.

Forms Overview

Journey Manager allows you to host Maestro forms so they are available to form users and applicants. Forms are exposed to users by being hosted on a user space. Each form has at least one form version, an entity that wraps the set of form-related data to be versioned. If multiple form versions exist, one of them is distinguished as the current form version, and is used when rendering the form. There is some data shared by all form versions, but most settings are maintained separately for each form version.

Manager recognizes forms of different form types, such as dynamic PDF forms or Composer SmartForms. Each form version has a form type and the artifacts needed to render the form. The form template file is used to render the form and possibly the form receipt. The receipt template file can be used to render the receipt instead. The form data configuration file defines the data structure underlying the form, and is vital for dynamic SmartForms hosted on Manager.

You create a form in Maestro using the Maestro editor. This form is called a Maestro form. Then you deploy it in Manager and then configure it to implement a user journey functionality, such as a back-end connectivity, transactions or receipts delivery. However, you are not limited to the Maestro tools to create a form and can use other alternatives.

Manager defines various form events, which you can use to trigger a function implementing some specific business requirements.

NOTE

A Manager's form is represented differently in Maestro. For more information, see Maestro's form overview.

Global Forms

WARNING

Global Forms is an experimental feature which may be subject to improvements and breaking changes in upcoming releases. If you have questions not covered by the documentation, contact your Client Services Team for assistance.

WARNING

The experimental Global Forms feature has been temporarily disabled while we continue improving it. This feature will be disabled in releases starting from 24.10 and going back to 23.10.

Journey Manager allows you to host a special type of Maestro forms - global forms - that can be used across multiple organizations. This is achieved by dynamically assigning different organizations to these forms depending on which client is using them. A form, global or not, must have a valid organization assigned to it.

The global forms let you run Journey Manager in multi-tenanted environments, that is, global forms are used by various tenants. A tenant is our client, also known as an organization, who is identified by a Client Code. When a tenant is using a global form, a tenant's organization is automatically passed to the global form so all delivery channels and services get the correct organization, as well as all transactions and form submissions are linked to that organization. This is a crucial difference between global and standard forms, where standard forms are statically assigned to one organization.

Let's look how you set up a form to be global.

You design and build a global form in a same fashion as you do a standard form using your favorite tool, for example, the Maestro editor. You may want to create a dedicated organization, for example, My Global Organization, or a project, for example, My Global Project, for such global forms, so Manager automatically deploys them to a correct place.

When the form is ready, you deploy it to Manager and change its configuration to make it global. Alternatively, you import it as a global form. You can also take any existing form and import it as a global one.

Manager allows you to change a form's behavior depending on whether it is global or not. For example, a form space now comes with the Default Client dropdown list, which you can configure to populate transactions of global forms served through the form space. Depending on your privileges as a user, you are limited to see only these form spaces that are assigned to you. This is another feature that enables Journey Manager to be used for multi-tenanted solutions, as well as improves overall security for all tenants who are using this solution.

Manager allows you to see at a glance whether a form is global or not by adding the Global column to form view interface, as well as in the form dashboard.

You can delete a global form, but additional care must be takes as the global form can be used across multiple organizations.

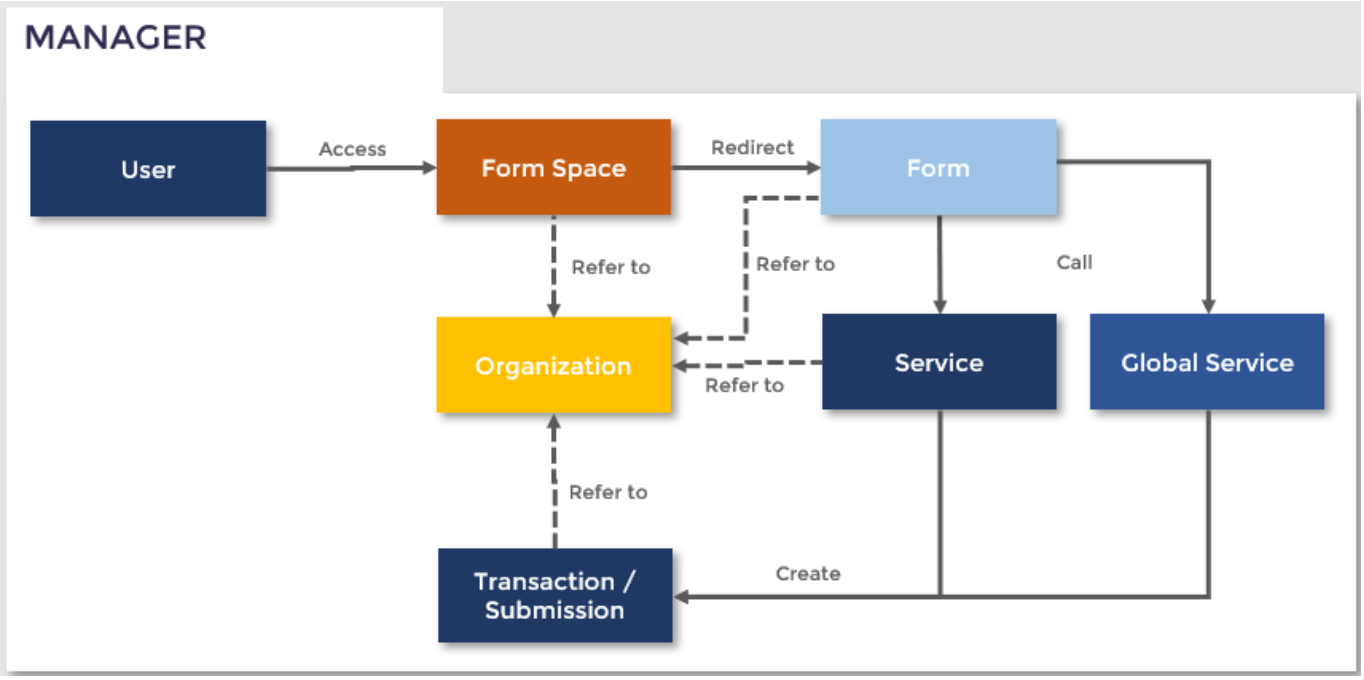
A global form can use both standard organization-specific and global services. However, we allow only the following 3 types of services to be global:

- Fluent Function
- Delivery Function
- Fluent Job Action

Further to this, when you assign services to a global form version, Manager checks whether these services are allowed to be global or not. If a service is incorrectly assigned, it throws an Illegal Argument Exception (`java.lang.IllegalArgumentException`) when you try to render the form. If a service and a form are under the same organization, the form will render fine. However, if they are from different organizations - an exception is thrown.

For a collaboration job, a delivery channel belonging to an organization is used.

The diagram below shows that an organization is central to all interactions during user onboarding journey and how an organization is used and referred to parts of the system to identify every transaction between a user and a form.

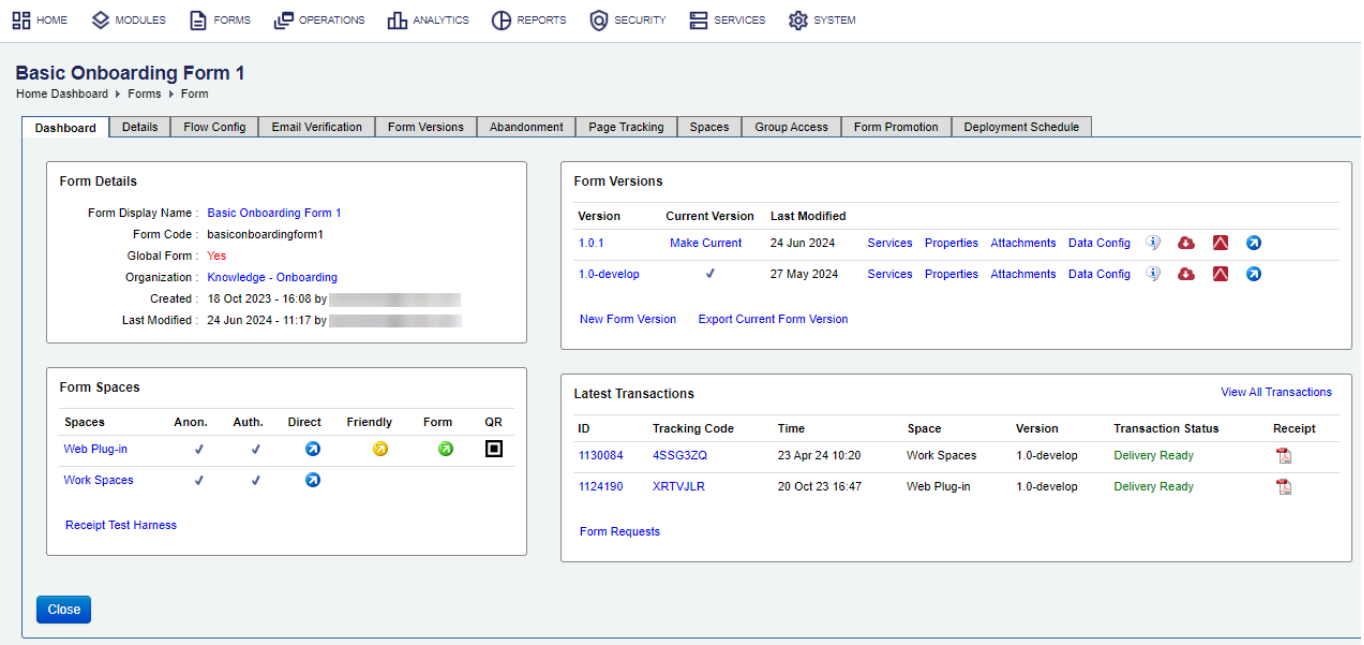


Next, learn about the form life cycle.

Form Dashboard

Manager comes with the Form allowing you to easily and quickly perform day-to day tasks for a selected form.

To navigate to the Form Dashboard, locate the form and click Edit.



The Dashboard consists of the following sections:

- Form Details
- Form Versions
- Form Spaces
- Latest Transactions

Form Details

This section displays the following basic form details:

- Form's display name
- Form code
- Global form status | 23.10 This feature was introduced in 23.10

> NOTE
>
> The global form status is shown only if the form is set as global.

- Organization

Form Versions

This section lists all available versions of this form. The current version of the form, which is the version that will be rendered when users request the form (unless A/B testing is enabled), is identified by a check-mark.

You can perform the following actions from this section of the Dashboard:

- Click Version of a selected form version to view its configuration.
- Click Make Current to promote a selected form version to be the current form version of a form, so it will be rendered when the form is requested by a user.
- Click Functions to list functions configured for this form version. | 17.10 This feature was introduced in 17.10.

  > NOTE
  >
  > The Functions link only appears when the Use Transact Functions checkbox is selected from the Build Options in Maestro. | 17.10 This feature was introduced in 17.10.

- Click Services to list services associated with this form version.
- Click Properties to list properties configured for this form version.
- Click Attachment to list attachments and their rules configured for this form version.
- Click Data Config to show data options configured for this form version. The options include form XML data, property prefill mapping, request parameter prefill mapping, and form data extract mapping.
- Click Form Design to displays a form design and audit details.
- Click Update Design from Maestro to update the selected form version based on any changes made to the original form version in Maestro.

  Only the same user who updated the selected form version last time can use this button to update the form. All other users will see this error message: Could not load latest published Maestro Form Design Version - error: 401:Unauthorized. If there have been no changes to the form, the following message is displayed: No Maestro Form Design changes to update.

  > NOTE
  >
  > This icon appears only if the form has been designed in Maestro.

- Click Open Form Design to open the related form version in Maestro. | 17.10 This feature was introduced in 17.10.

NOTE

This icon appears only if the form has been designed in Maestro.

- Click From Test Render to render this form version in the default form space.
- Click New Form Version to create a new version of this form.
- Click Export Current Form Version to export the current form version (version 3 in this case) on your local PC.

Form Spaces

This section lists form spaces where the form is hosted on and authentication mode, Anon. or Auth., users are required to access the form. If the form is configured with the Auth. mode only, users will need to log in first to access the form.

You can perform the following actions from this section of the Dashboard:

- Click Direct to access a form via the selected form space's URL. You should use this URL to open a form from your website.
- Click Friendly to open a form's landing page using a user-friendly URL format. You can use this functionality to test the form.
- Click Landing to open a form's landing page. This is the page where users can log in.
  | 22.04 This feature was removed in 22.04
- Click Form to open a form page in the selected form space where it will be rendered.
- Click QR Code to this form's QR code[1] to scan a form's QR code[2] using a QR scanner app.
- Click Add Form Space to assign a form space to this form.
- Click Receipt Test Harness to generate and test a receipt for this form as a PDF or HTML file.

---

[1]QR Code (Quick Response Code) is a machine-readable code consisting of an array of black and white squares, typically used for storing URLs or other information for reading by the camera on a smartphone.

[2]QR Code (Quick Response Code) is a machine-readable code consisting of an array of black and white squares, typically used for storing URLs or other information for reading by the camera on a smartphone.

NOTE

The form is not rendered and an Illegal Argument Exception (`java.lang.IllegalArgumentException`) is shown, if the form is not configured correctly. For example, when a global form is set up with services that are not allowed.

Latest Transactions

This section lists all latest transactions of user submissions of this form.

You can perform the following actions from this section of the Dashboard:

- Click ID or Tracking Code to see the form version's transaction details.
- Click View PDF Receipt to view and download a PDF version of the form version.
- Click Form Requests to view all form requests or search for specific form requests.

Next, learn about form details.

Deploy Forms

The Temenos Journey Manager platform allows you to deploy forms created in various tools. Maestro is our preferred tool to create your forms and components.

To deploy a Maestro form to Manager:

1. Build a form in Maestro.
2. Download the built form as a form version ZIP file from Maestro.
3. Import the form version ZIP file into Manager.
4. Configure the form in Manager, for example, delivery channels, receipts and data retention.

After the form is deployed into Manager, you can still update it in Maestro and sync up the form changes by clicking Update Design from Maestro from the form dashboard.

Caching of resources in deployed forms is straight forward with Manager, all resources bundles with the deployed form in the form archive are cachable but only in the context of the specified form. This means that the first time a user loads a form version they will have to download all the relevant bundled resources referenced by the form, excluding any external resources. This is to prevent issues with deploying a form that contains a new image or JavaScript file and due to browser caching getting the old version as it is still valid according to the cache. This is done by using the `<base href=""/>` element in the HTML `<head>`. Republishing a Maestro form of the same version will result in a different value being used for this base hef.

> NOTE
>
> The initial release of Maestro was generally available in August 2016, along with Manager 5.0, which is the first fully supported version of Manager. Publishing forms to any version prior to Manager version 5.0 is not supported. Many features may work but troubleshooting and support will not be provided for Manager versions prior to 5.0.

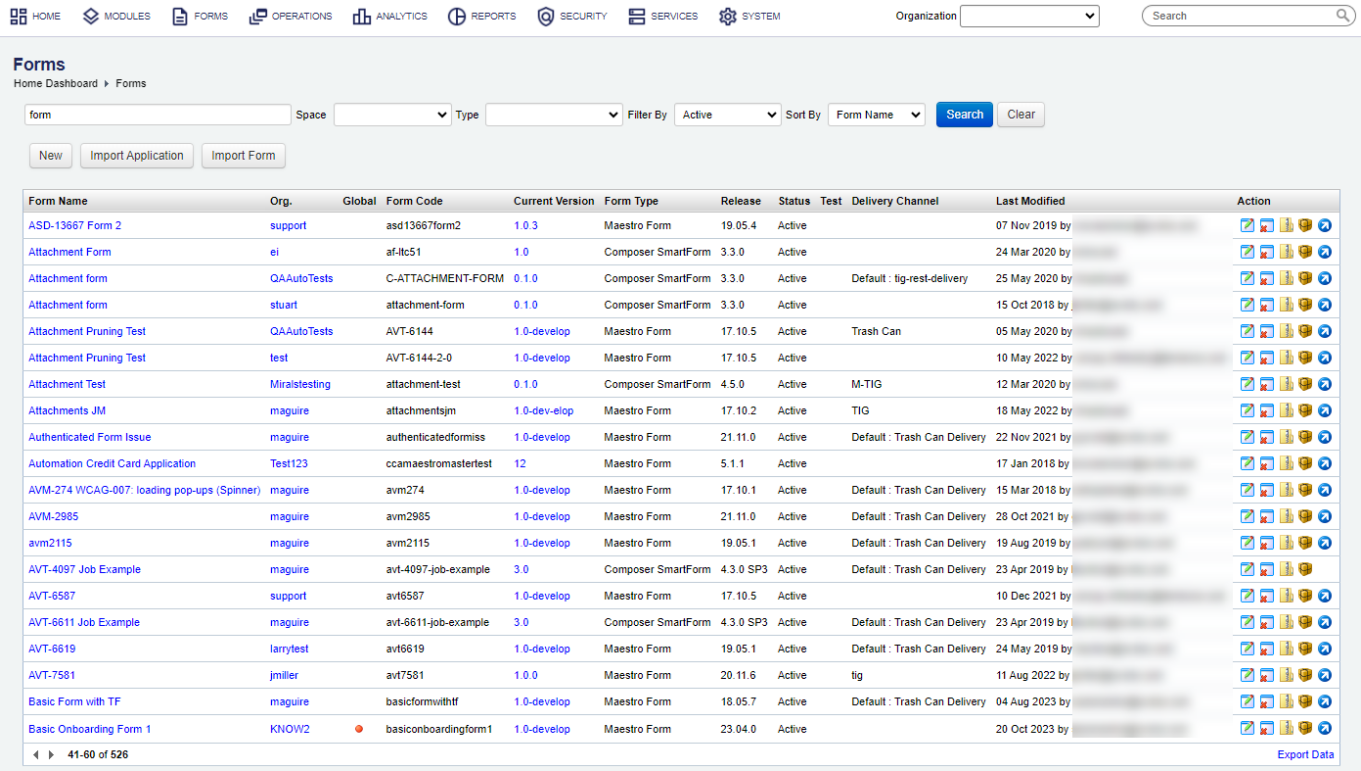
Next, learn how to view forms.

View Forms

Manager allows you to view all available forms, so you can easily find, manage, and update them.

To view all forms, select Forms > Forms.

Forms are shown page by page, with the maximum number of records per page configured in the users preferences.



The list displays the following details:

- Form Name: a form display name, which must be unique within a specific organization for a standard form and across all organizations for a Global Form. This name is used in the long form URL - Form Friendly URL.
- Organization: an organization a form is assigned to. It could be empty for a global form.
- Global: a status showing with a red dot whether a form is global or not. | 23.10 This feature was introduced in 23.10

> WARNING
>
> The experimental Global Forms feature has been temporarily disabled while we continue improving it. This feature will be disabled in releases starting from 24.10 and going back to 23.10.

- Form Code: an alphanumeric string uniquely identifying a form across all organizations.
- Current Version: a version of a form set as a default.
- Form Type: a type of a form, which can be one of the following:
  - Composer SmartForm
  - Maestro Form
  - JS App - refers to an Open UX form
  - Dynamic PDF Form
  - Static PDF Form
  - Other Document
- Release: a release of a form editor that was used to build a form.
- Status: a status of a form, which can be one of the following: | 21.11 This feature was introduced in 21.11.
  - Active Non-Test - it's an active form except that of in Test Mode.
  - Active: it is in use and readily available to all users. All functionality is allowed.
  - Offline: it can't be used at all. No functionality is available, including DDS, Delivery and so on.
  - Inactive: similar to Offline.
  - NoRender: it doesn't allow for new or saved and resumed forms, but ongoing open sessions can be completed. DDS, submits, saves, and delivery are allowed.
- Test: a status indicating whether a form is in Test Mode or not. For more information, see Configure Form Test Mode.
- Delivery Channel: a delivery channel configured for a form.
- Last Modified: a date when a form was last modified and a user account who modified the form.

To filter or search the forms by one or more criteria, specify the following settings and click Search:

- Organization
- Form name or part of a name
- Form space
- Form type
- Status | 21.11 This feature was introduced in 21.11.

To change the form list order, select one of the following from the Sort By dropdown list:

- Form Name
- Created By
- Created Date
- Modified By
- Modified Date

Click Clear to reset the search criteria.

You can also perform the following actions:

- Create a form
- Configure a form
- Remove a form
- Export forms
- Render a form
- Import forms
- Import application

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to configure a form.

Configure a Form

After you have built a form in Maestro, you have to configure it in Manager.

To configure a form in Manager:

1. Select Forms > Forms.
2. Locate a form and click Edit to display the form's dashboard.
3. Configure form details such as delivery channels.
4. Configure form flow.
5. Configure email verification.
6. Configure form abandonment to override the organization's abandonment settings.
7. Configure form page tracking for third-party page tracking and analytics.
8. Assign form spaces where the form will be hosted.
9. Configure group access to enlist groups that can access this form in case it has some restrictions.

Next, learn how to remove a form.

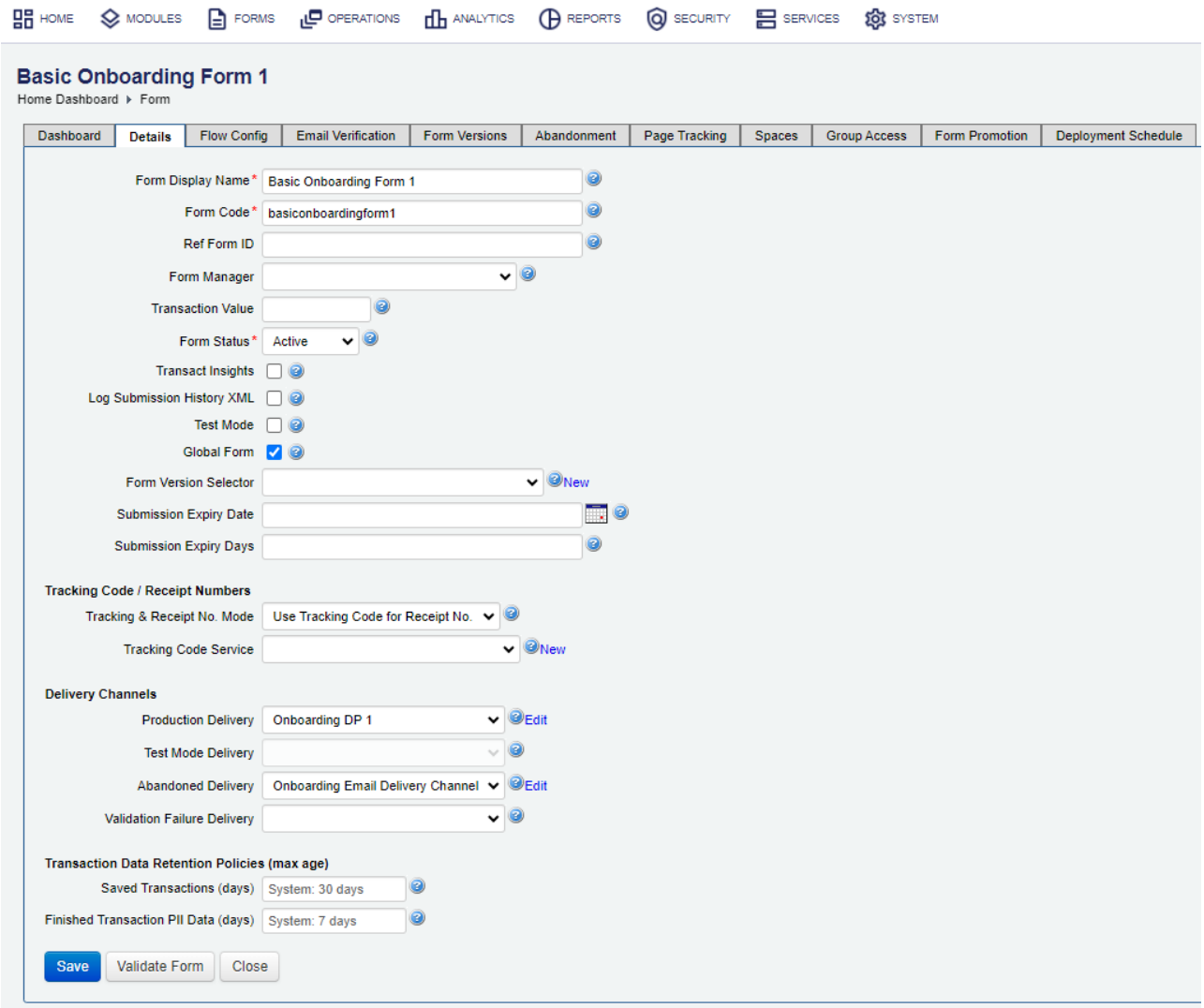
Configure Form Details

Manager allows you to fine-tune a form by configure the following details:

- General form information
- Tracking codes or receipt numbers
- Delivery channels
- Transaction data retention policies

To configure form details:

1. Select Forms > Forms.
2. Locate a form and click the Details tab to edit the form configuration.



3. Edit the name of the form that is displayed to the user in the Form Display Name field. The name uniquely identifies the form within a specific Organization. It will also be used in the long form URL (Form Friendly URL).
4. Edit a form code in the Form Code field. This is a short alphanumeric string, which uniquely identifies the form across all organizations. It is generally not displayed to users.
5. Edit a ref form ID in the Ref Form ID field. This is the identifier used internally by the organization and will be passed through during various web service calls such as prefill and receipt generation. It is optional, only used if there is an identifier that is used internally by the organization that the form version is associated with.
6. Select a form manager from the Form Manager dropdown, who will be notified via email whenever the form's status changes or a form version is promoted. The Form Manager user is associated with a form's organization.
7. Provide the form's transaction value to the business in dollars in the Transaction Value field. This value is used for reporting purposes.
8. Select a status from the Form Status dropdown list, which can be one of the following: | 21.11 This feature was introduced in 21.11.
   - Active: it is in use and readily available to all users. All functionality is allowed.
   - Offline: it can't be used at all. No functionality is available, including DDS, Delivery and so on.
   - Inactive: similar to Offline.
   - NoRender: it doesn't allow for new or saved and resumed forms, but ongoing open sessions can be completed. DDS, submits, saves, and delivery are allowed.

   NOTE

   Select the Inactive checkbox to retire a form without deleting it.

9. Select the Transact Insights checkbox to enable this form with Journey Analytics. For more information, see how to enable applications to collect analytics
10. Select the Log Submission History XML checkbox to log all form XML changes so you can view transaction history and transaction XML difference. This is handy for transaction troubleshooting. If a form uses Transact functions, this option creates a new submission history record for the `formUpdate` operation.

NOTE

Excessive logging can degrade performance of Manager applications with high throughput, so ensure that the form doesn't have the Log Submission History XML selected. Otherwise, every time the form does a `formUpdate` or background save operation, Manager stores a full copy of the submission XML which can result in large growth of the submission history XML data table on high throughput systems with modern form applications .

11. Select the Test Mode checkbox to mark the form to be in test mode, which is useful when you develop and test the form. Forms in Test Mode use the Test Mode Delivery method and are not available in TransactField App, Web Portal search pages, as well as no transaction analytics and reports are created.

    By default, the form is in production mode (the Test Mode checkbox unselected), so the Production Delivery method will be used to deliver submissions and they will be marked as live submissions. For more information, see Configure Form Test Mode.

12. Select the Global Form checkbox to make the form global.

    WARNING

    The experimental Global Forms feature has been temporarily disabled while we continue improving it. This feature will be disabled in releases starting from 24.10 and going back to 23.10.

13. Select a service responsible for selecting the version of the form to be rendered at request time from the Form Version Selector dropdown list. If there is no appropriate service, click New to create one. If left blank, the system default Form Version Selector service, such as Current Form Version, is used. You may want to use this option for form A/B testing.
14. Select a date from Submission Expiry Date that is used to ensure a form can't be submitted on or after this date. If the form is submitted after this date, it'll be rejected by a user defined as the Form Manager. The Submission Expiry Date is populated to a form's XML element `//SystemProfile/SubmissionExpiryDate` for reference purposes. The Transaction Processor abandons these submissions if the expiry date is passed and the form is not submitted, that is, it has either the Open or Saved status.

NOTE

You can also set this value using the `submission.timeTxnExpiry` API

15. Specify a number of days in the Submission Expiry Days field that are used to ensure a form can only be submitted a specified number of days after the form is requested. If a form is submitted after this time, it'll be rejected by SmartForm Manager. The form submission expiry time is calculated as `time the form is requested + submission expiry days`. This value is populated to a form's XML element `//SystemProfile/SubmissionExpiryDate` for reference purposes.
16. Select how to generate tracking and receipt codes from the Tracking & Receipt No. Mode dropdown list. You have the following options:
    - Use Tracking Code for Receipt No
    - Use Receipt Number for Tracking Code

NOTE

We recommend using Tracking Code to improve performance and scalability. For more information, see configure receipt numbers and tracking codes.

17. Select a service to generate form tracking codes from the Tracking Code Service dropdown list. You have the following options:
    - Blank - the default Tracking Code service is used
    - Fluent Tracking Number
    - Random Tracking Number
18. Select a delivery channel from the Production Delivery dropdown list to send successful submissions through. This is enabled by default.

NOTE

This dropdown is disabled if the form is in Test Mode, that is the Test Mode checkbox is selected.

19. Select a delivery channel from the Test Mode Delivery dropdown to send successful test submissions through.

> NOTE
>
> This dropdown is enabled if the form is in Test Mode, that is the Test Mode check-box is selected.

20. Select a delivery channel from the Abandoned Delivery dropdown list to send abandoned submissions through.
21. Select a delivery channel from the Validation Failure Delivery dropdown list to send submissions with data validation errors through.
22. Select the maximum age of user saved transactions, after which they will be automatically abandoned, from the Saved Transactions (days) dropdown list.
23. Select the amount of time to keep PII transaction data for after the transaction has finished from the Finished Transaction PII Data (days) dropdown list.

> NOTE
>
> For more information on data retention configuration, see how to configure form data retention.

24. Click Validate Form to check whether the form is active, has no invalid versions or expired dates and a few other parameters using values defined in the Test Import Tag metadata tag.
25. Click Save to update the changes.

Next, you may want to configure form flow.

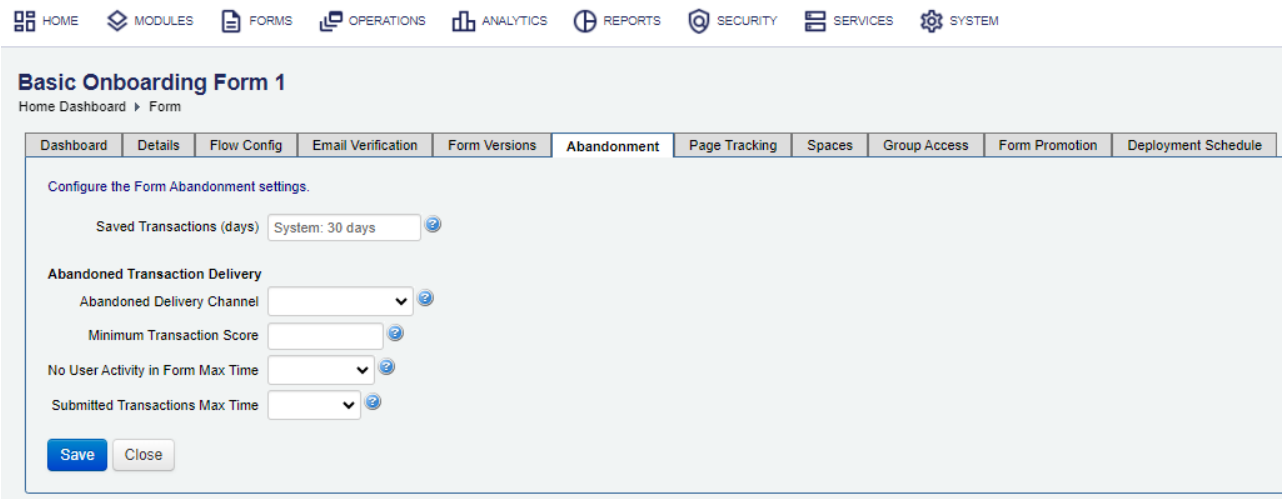
Configure Form Abandonment

Transactions can become abandoned either actively due to a user action, such as canceling a transaction, or passively based on the form's abandonment settings. You can use abandonment configuration to define when a form is considered as abandoned[1] after it has been saved, and what happens after a form transaction is abandoned. Manager allows you to configure abandonment rules of form's transactions at a global, organization, or form level. These configurations depend on:

- Customer requirements
- Server performance
- Specific data retention policies

To configure form abandonment:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Click the Abandonment tab.



4. Specify the maximum age of user saved transactions after which they will be automatically abandoned (marked as Abandoned) in the Saved Transaction (days) field. The default is 30 days, which is either the organization data retention setting, if exists, or the system data retention setting. When a saved transaction reaches this maximum age, it is marked as

[1]Any transaction that has been deemed as Abandoned in Manager. When a user leaves the application without explicitly canceling it, the transaction lives in Started status for a period defined in Manager, after which it is considered as an Abandoned transaction.

Abandoned by the system. The maximum age is based on the number of days a form has been untouched since it was last saved.

5. Select the organization delivery channel for delivering abandoned submissions from the Abandoned Delivery Channel dropdown list. The next three fields are enabled when a delivery channel is selected.

   NOTE

   Manager version 17.10 and higher has the following fields available for Composer forms only. However they are still displayed for Maestro forms.

6. Edit the Minimum Transaction Score to specify the transaction score that a form transaction must reach for it to be delivered to the Abandoned Delivery Channel. A transaction score is a numerical value calculated by the form, generally based on how much of the form was filled in. When left blank, all abandoned form transactions are delivered. Otherwise, only those that have been “adequately” filled in, meaning they have reached the Minimum Transaction Score, are delivered. Minimum Transaction Score specifies a minimum value representing how much of the form must be filled in for it to be delivered to the abandoned delivery channel. That is, a transaction score is a numerical value calculated by the form to indicate how much of the form was filled in. If you would like only submissions that exceed a certain transaction score to be delivered, enter the minimum transaction score into this field.
7. Select a time from the No User Activity in Form Max Time dropdown list to specify a time frame for allowing any form transaction by an anonymous user to remain untouched before marking it as abandoned. This is the period of time after which anonymous transactions are marked as abandoned. These transactions have been saved in the background and untouched by a user.

   NOTE

   A form transaction must also have been background saved for this setting to apply.

8. Select a time from the Submitted Transactions Max Time dropdown list to specify a time frame for allowing any form transaction that has been submitted by a user to remain incomplete before marking it as abandoned. This is the maximum age before a submitted but incomplete, for example missing attachments, transaction are marked as abandoned.

NOTE

A form transaction may be incomplete because it’s waiting for a post-submission requirement such as an attachment to be provided by the user.

9. Click Save to update the changes.

NOTE

To troubleshoot user form interactions, you may want to view abandoned transactions.

Next, learn hot to configure form details.

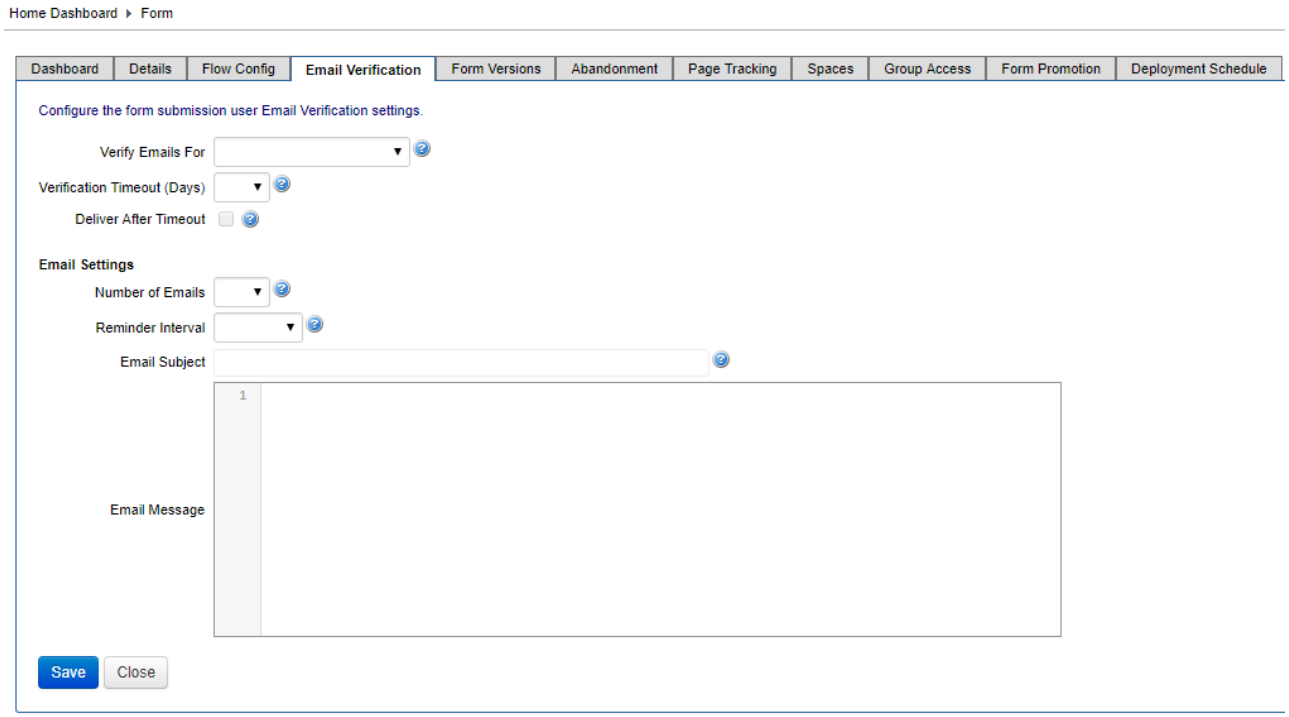
Configure Form Email Verification

A verification email is used to confirm the email address of a form user. Manager allows you to configure these verification emails for selected forms. Once configured, these emails are sent when a user submits a form. Commonly, a verification email includes a link that the user must click to verify their email address. A form does not send a verification email by default, so you have to enable this option manually.

> NOTE
>
> For submission email verification to work, a contact email address must be stored for the submission. For more information, see configuration mapping.

To configure a form with a verification email:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Click the Email Verification tab.



4. Select Verify Emails For to specify when to send an Email Verification to the form user.
   - Blank - to disable email verifications. This is the default option.
   - Anonymous Submissions - to enable email verification for submissions from an

anonymous user only.
- All Submissions - to enable email verification for submissions from all users.

> NOTE
>
> If the Email Verification is selected, you can configure the following options.

5. Select Verification Timeout to specify how many days (at maximum) you want to give a user to verify their email address. After verification timeout has been reached, the form submission will be abandoned by default unless opting to deliver submission anyway (Deliver After Timeout is selected).
6. Select Deliver After Timeout to indicate once a user misses the due date as per the Verification Timeout, whether to deliver the submission anyway (selected) or to abandon the submission (unselected is the default).

> NOTE
>
> Select the Deliver After Timeout checkbox only when email verification is not mandatory.

7. Select Number of Emails to specify how many emails in total will be sent to a user as a reminder to verify their email address.
8. Select Reminder Interval to specify the time (hours or days) to wait before sending each reminder email. For example, if Number of Emails is 3 and Reminder Interval is 7 days, then a user that hasn't verified their email address will receive up to 3 verification emails every week.
9. Edit Email Subject to specify the subject text of the Verification Email. It is predefined as Please Verify Your Email.
10. Edit Email Message to specify the message defined using basic HTML mark-up. It is predefined with a message. When making changes to the Email Message HTML, ensure the message always contains the URL that the user needs to access to complete submission email verification (the placeholder is `${verificationUrl}`).
11. Click Save to update the changes.

Next, learn how to view form versions.

Configure a Form Flow

Manager allows you to configure a user onboarding flow for each form by specifying pages a form user will go through, for example, user's redirect to a custom URL when resuming a saved form. The form flow is configured on a form level and is therefore not versioned.

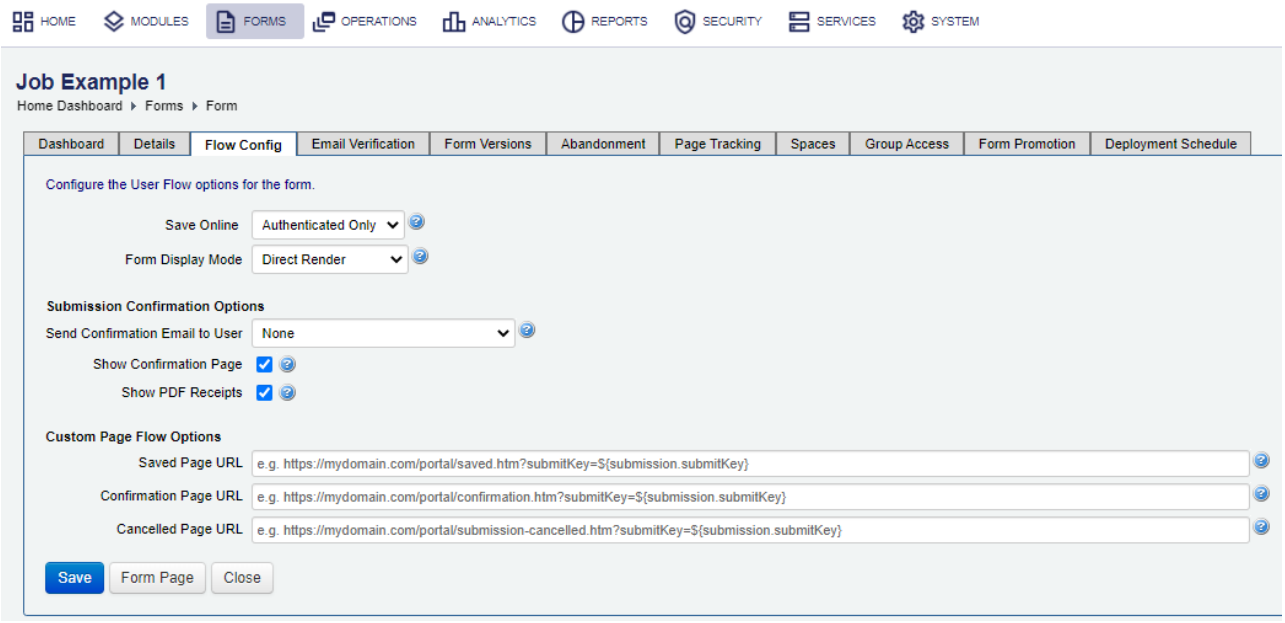
Depending on a form's configuration, an applicant, who is filling in the form, can be presented with different steps and options required for successful application submission. These steps and options represent a form flow, which may vary significantly from form to form. For example, a simple form may launch from a landing page and show a confirmation page after submission. A more complex form may have additional steps and options, both prior to a user filling in the form, such as a Terms And Conditions page, a user authentication page, as well as after the form is filled, such as saving and returning to form drafts, payment, attachments, signatures, availability of a receipt.

NOTE

As part of the ongoing modernization of the platform, since version 22.04, we have deprecated a number of Work Space pages: the Wet Signature, Landing, and Terms And Conditions pages. These pages are part of the old Non-FTX submission life cycle that is now handled within form design itself.

To configure a form flow:

1. Select Forms > Forms.
2. Locate a form and select the Flow Config tab.



> NOTE
>
> The options are listed in chronological order from a user's perspective when going through the application process. If a custom user form space is being used to host this form, not all form flow options may be supported. Here we assume that a fully features user form space is deployed.

Configure the User Flow options for the form

Manager supports legacy Composer SmartForms, so you will see the following additional options when configuring the flow of a Composer SmartForm:

3. Select the Show Landing Page checkbox to display the Form's landing page. Otherwise, the form user is directed straight to the rendered form. | 22.04 This feature was removed in 22.04
4. Select how to use the save and resume functionality, allowing to complete a form later, from the Save Online dropdown list, which can be one of the following:
   - Disabled: no users can save the form.
   - Enabled: all users, including authenticated users or user who are not logged to Manager when filling in the form, can save the form. Manager allows submission sharing

between anonymous and registered users. If an anonymous user saves the form online, they will need to pass a security challenge to return to their form.
    - Authenticated Only: only authenticated users, who are logged to Manager when filling in the form, can save a form. Otherwise, Manager will reject user's attempts to save the form online if the user is not authenticated. If an authenticated user saves a form online, they are shown a page confirming that the form is saved, and can access it again by logging on to Manager.
5. Select the Show Terms and Conditions checkbox to display a Terms and Conditions page before the form user can access the form. | 22.04 This feature was removed in 22.04
6. Select a display mode from the Form Display Mode dropdown list:
    - Direct Render: to display the form directly from the servlet.
    - Embed with DIV: to display the Form inside a `<div>` section of an HTML page.
    - Embed with iFrame: to display the form inside of an `<iFrame>` section of an HTML page.
7. Select one of the following signature options from the Form Signature Required dropdown list determines:
    - Wet Signature: to ask users to physically sign a form. | 22.04 This feature was removed in 22.04
    - eSignature: to ask for an electronic signature by confirming a receipt, which is presented to the user after the form is submitted.

Submission Confirmation Options

Manager allows you to customize what happens when the user completes a submission for this form.

8. Select an option from the Send Confirmation Email to User dropdown to determine whether a confirmation email should be automatically sent to a user who submits the form and what is to be included:
    - None - to never send a confirmation email
    - Confirmation - to send the confirmation email only
    - Confirmation and Receipt Attachment - to send the confirmation email along with an attached PDF receipt of the form
    - Confirmation and Receipt Link - to send the confirmation email along with a secure URL for the form user to download the PDF form receipt

NOTE

You can also configure form submission options in Maestro by selecting Form Options > User Experience > Auto-Send Confirmation Emails. For more information, see user experience.

9. Select the Show Confirmation Page checkbox to display the form space's confirmation page when a user has successfully submitted the form.
10. Select the Show PDF Receipts checkbox to offer PDF receipts to a user after submission. This is the default option.

Custom Page Flow Options

Manager handles the form flow when a user saves, submits or cancels a form. However, you can redirect the user to a custom location using the configuration below. You can use the following Velocity variables when constructing your custom URLs:

- $submission: the submission object. For example, `${submission.submitKey}` will be substituted with the submit key (a GUID)
- $formDataMap: the map of submission data extract values. For example, `${formDataMap.firstName}` will be substituted by the value in the map for the key `firstName`
- $request: the HTTP servlet request

11. Enter a custom URL in the Saved Page URL field to where the form user is redirected after saving the form entry. For example:
    - A current form version redirect URL, which uses all propertied configured for the current form version:

      ```
      ${submission.portal.contextPath}submission-saved.ht-
      n?submitKey=${submission.submitKey}
      ```

    - A non-current form version redirect URL, which uses all propertied configured for the specific form version:

      ```
      ${submission.portal.contextPath}submission-saved.ht-
      n?sub-
      nitKey-
      y=${submission.submitKey}&tmFormVersion=${submission.version.versionNumber}
      ```

> NOTE
>
> If you don't specify `tmFormVersion=${submission.version.versionNumber}`, the current form version's properties will be used.

12. Enter a custom URL in the Confirmation Page URL field to where the form user is redirected after submitting the form entry.
13. Enter a custom URL in the Cancelled Page URL field to where the form user is redirected after canceling the form entry.
14. Click Save to update the changes. If you have entered custom page URLs, Manager will use them instead of redirecting to the corresponding form space page.
15. Click Form Page to test how the form is rendered when accessed via the form's landing page.

Next, learn how to configure email verification.

Configure Form Group Access

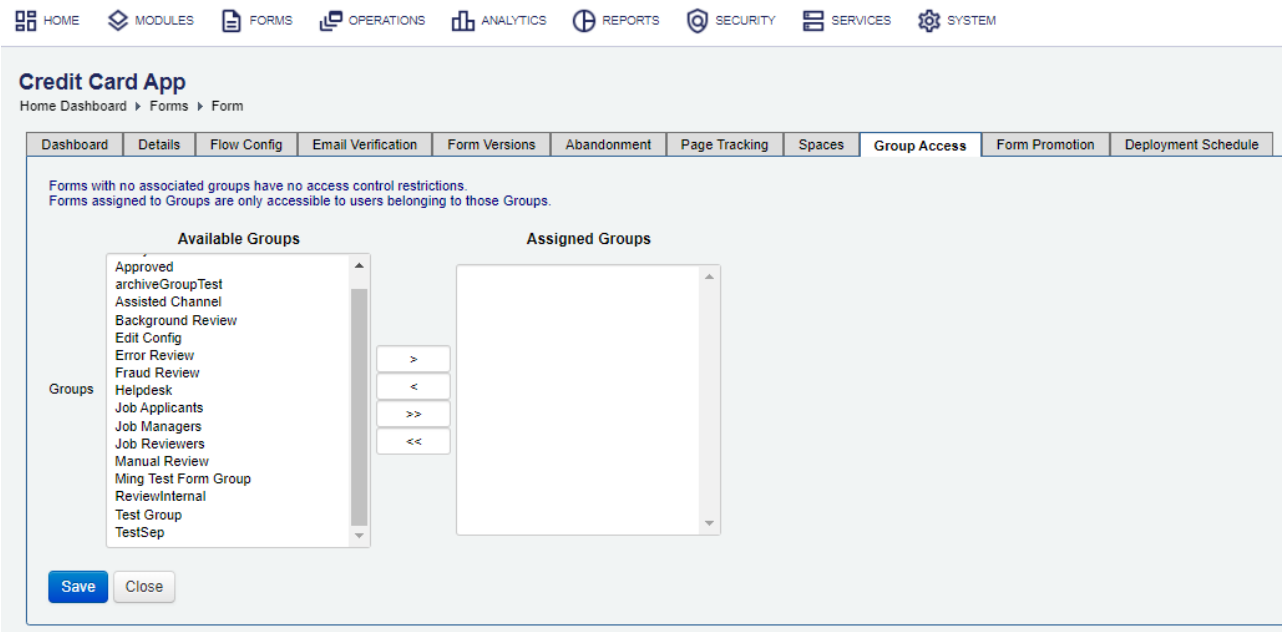
Manager allows you to restrict access to a form, so users belonging to the assigned groups can view and select this form when logged into a form space. Forms with no associated groups have no access control restrictions.

> NOTE
>
> A farm is restricted when it is assigned to at least one group.

In most cases, forms are unrestricted and available to all users accessing the form space, so for a typical form, the Group Access tab should not have any assigned groups.

To assign groups to a form:

1. Select Forms > Forms.
2. Locate a form.
3. Select the Group Access tab.



4. Select one or more groups from the Available Groups list that you want to assign to the form (on Windows, use the Shift and Ctrl keys to select multiple groups).
5. Click > to move the selected groups from Available Groups to Assigned Groups.
6. To revoke any assigned groups, select the groups from Assigned Groups and click <.
7. To assign all groups, click >>.

8. To revoke all groups, click <<.
9. Click Save to update the changes.

NOTE

A form’s assigned groups can be changed at any stage.

Next, learn how to promote a form.

Configure Form Page Tracking

Manager allows you to add third-party page tracking scripts to forms so you can track them in web analytics services, such as Google Analytics, Marketo and NetResults. You are not limited to these services only - you can add any other page tracking tags as long as you provide a valid custom JavaScript.

You can inject submission's properties into an HTML page to apply branding of a form earlier during the rendering process. | 24.10 This feature was introduced in 24.10

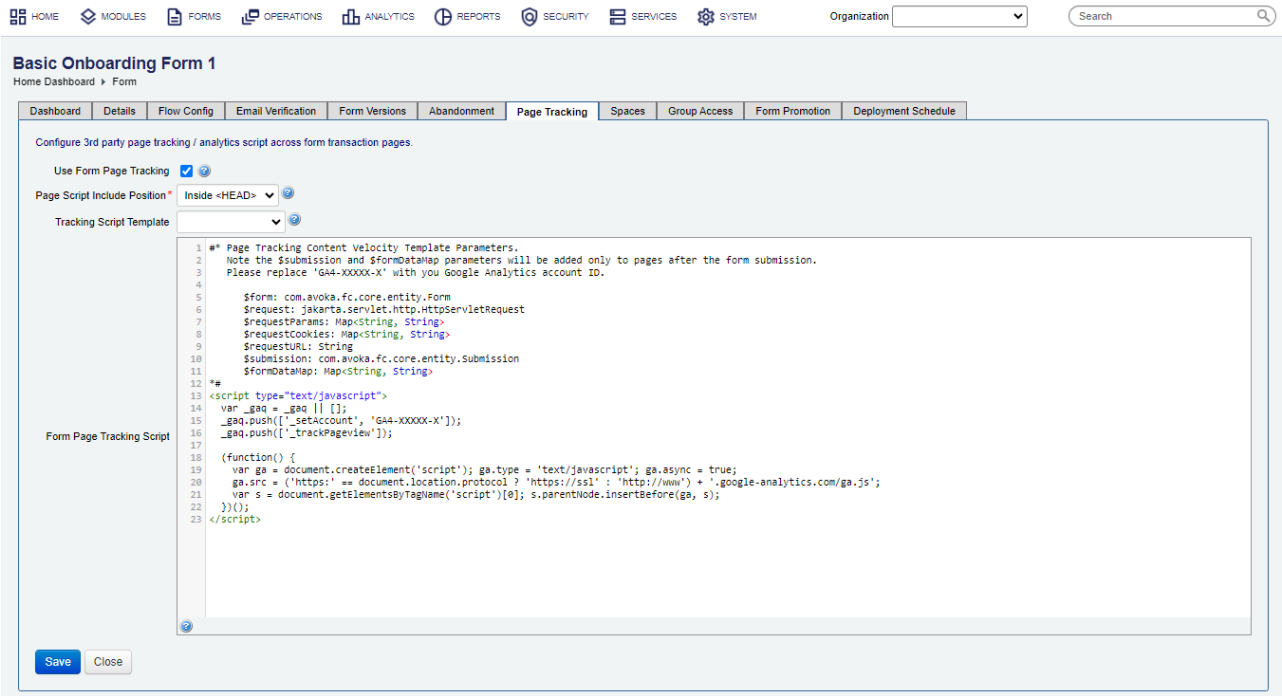
You can set up a form page tracking using one of following options:

- UI
- SDK
- API

Configure Using UI

To configure page tracking for a form:

1. Select Forms > Forms
2. Locate a form and click Edit.
3. Select the Page Tracking tab.



4. Select the Use Form Page Tracking checkbox if you want this form to use third-party page tracking.
5. Select an HTML page script include position from the Page Script Include Position drop-down list, which can be one of the followings:
    - Inside <HEAD>: to place the script inside the head of the HTML mark-up.
    - After <HTML>: to place the script outside or after the HTML mark-up.
6. Select the third-party page tracking system template from the Tracking Script Template dropdown list, which will be populated to the Form Page Tracking Script text field. The available templates are:
    - Google Analytics[1]: replace `GA4-XXXXX-X` with your Google Analytics account ID.
    - Marketo: replace `MUNCHKIN_ACCOUNT_ID` with your Marketo account ID.
    - NetResults: use as is.
    - Blank: for a custom script where you provide a script in the Form Page Tracking Script text field described below.
7. Edit the populated template script or add your own script in the Form Page Tracking Script field. The script format is show below:

```
<!-- Google Tag Manager -->
<noscript/noscript>
< script >(function(w,d,s,l,i){...});</script>
<!-- End Google Tag Manager -->
```

For example, you can add the following Google Analytics scrip to track your Maestro forms:

[1]Google Tag Manager (GTM) is a tag management system (TMS) that allows you to quickly and easily update tracking codes and related code fragments collectively known as tags on your website or mobile app.

```
<!-- Google Tag Manager -->
<noscript/noscript>
<script>window.addEventListener('DOMContentLoaded', function(){
(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-XXX');
});</script>
<!-- End Google Tag Manager -->
```

However, you can't add an input tag HTML element within the body tag, such as the one shown below:

```
<input type="hidden" name="formCode" id="submission_formCode" value-
e="$form.formCode"/>
```

Although, you can add it into the actual Maestro form as the HTML element into a static text component label.

8. Click Save to update the changes.
9. Configure the form's Analytics options in the Maestro editor.

For more information, see how to add Google Tag Manager support.

Also, in some instances, you can use this functionality to load external JavaScript libraries in a form.

Configure Using SDK

The Journey Manager SDK provides the `pageTrackingIncludePosition` and `pageTrackingScriptFile` attributes, which you define in the `form-def.json` file, to configure page tracking in a form.

For example, add the `"pageTrackingIncludePosition": "Inside HEAD"` and `"pageTrackingScriptFile": "./page-tracking.jsp"` to place the script defined in `./page-tracking.jsp` inside the `<head>` element of the HTML markup. For more information, see Form JSON Definitions.

Configure Using API

The Journey Manager allows you to configure page tracking in a form using the Configuration Service. For each form, identified with `formCode`, add the `pageTrackingEnabledFlag`,

`pageTrackingIncludePosition`, and `pageTrackingContent` attributes to the `configuration-service.json` file. The `pageTrackingContent` attribute holds the content of the JavaScript file specified in `pageTrackingScriptFile`.

The snippet of the JSON file is shown below:

```
"formDetails" : [
          {
            "formCode":"MyForm",
            "pageTrackingEnabledFlag": "true",
            "pageTrackingIncludePosition": "Inside HEAD",
            "pageTrackingContent": "#* Page Tracking Content Velocity Template Para-
meters.\n   Note the $submission and $formDataMap parameters will be added only to pages
after the form submission. \n   Please replace 'GA4-XXXXX-X' with you Google Analytics
account ID.\n \n      $form: com.avoka.fc.core.entity.Form\n      $request: jakarta.ser-
vlet.http.HttpServletRequest\n      $requestParams: Map<String, String>\n      $re-
questCookies: Map<String, String>\n      $requestURL: String\n      $submission:
com.avoka.fc.core.entity.Submission\n      $formDataMap: Map<String, String>\n*#\n<script
type=\"text/javascript\">\n  var _gaq = _gaq || [];\n  _gaq.push(['_setAccount', 'GA4-
XXXXX-X']);\n  _gaq.push(['_trackPageview']);\n\n  (function() {\n    var ga = doc-
ument.createElement('script'); ga.type = 'text/javascript'; ga.async = true;\n    ga.src
= ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-ana-
lytics.com/ga.js';\n    var s = document.getElementsByTagName('script')[0]; s.par-
entNode.insertBefore(ga, s);\n  })();\n</script>"
          },
          {...}
],
```

Next, learn how to assign form spaces to a form.

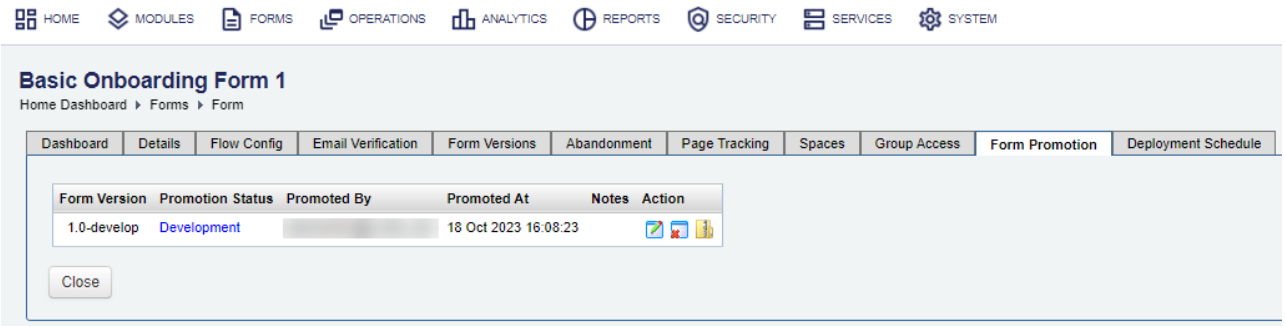
Configure Form Promotion

Manager allows you to promote or demote a form version to progress it from a development to production readiness, as per your software development cycle process. This functionality gives you the ability to add notes and set up different users with different permissions. For example, developers may be allowed to send a form version to test, but not promote it any further than that. Promotion applies to a particular form version. If a form has multiple versions, they are promoted independently of each other.

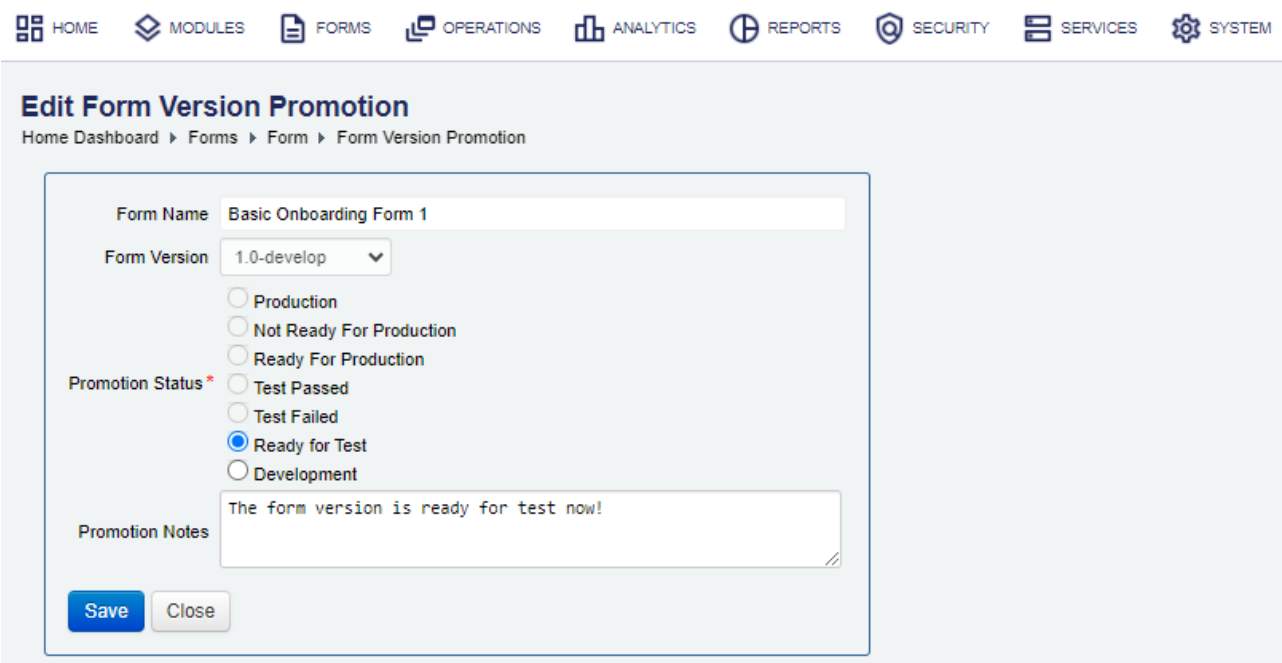
> NOTE
>
> This feature has been deprecated, but you can enable it, if required.

To promote or denote a form version:

1. Select Forms > Forms.
2. Locate a form and select the Form Promotion tab to check the latest promotions. For more information, see View Form Version Promotions.



3. Click Edit.



4. Select a form version you want to promote from the Form Version dropdown.
5. Select an appropriate promotion status from the Promotion Status list, which can be one of the following:
   - Development
   - Ready For Test
   - Test Failed
   - Test Passed
   - Ready For Production
   - Not Ready For Production
   - Production

> NOTE
>
> You can promote a form version only one step at a time, which helps set up a permission system with several interacting roles, such as developer, tester and project manager.

> NOTE
>
> You demote a form version by setting its Promotion Status back to a lower level.

6. Provide a meaningful information in the Promotion Notes field.

7. Click Save to update the changes.

A successful promotion is indicated with the message The Promotion Log has been successfully saved and you can see it in the Form Promotion tab.

After a form version has been promoted, notification emails will be automatically sent out to all organization and global administrators who have the permissions to promote the form to the next level. For example, if a form is promoted to Test Passed, all administrators who can promote this form to Ready For Production will be sent an email. Requirements for receiving promotion emails are being a member of the Receive Promotion Alerts group as well as having an email address set up in the administrator details.

You can also set a dedicated form manager for each form. This administrator will receive a notification whenever the promotion status changes.

Next, learn about deployment schedule.

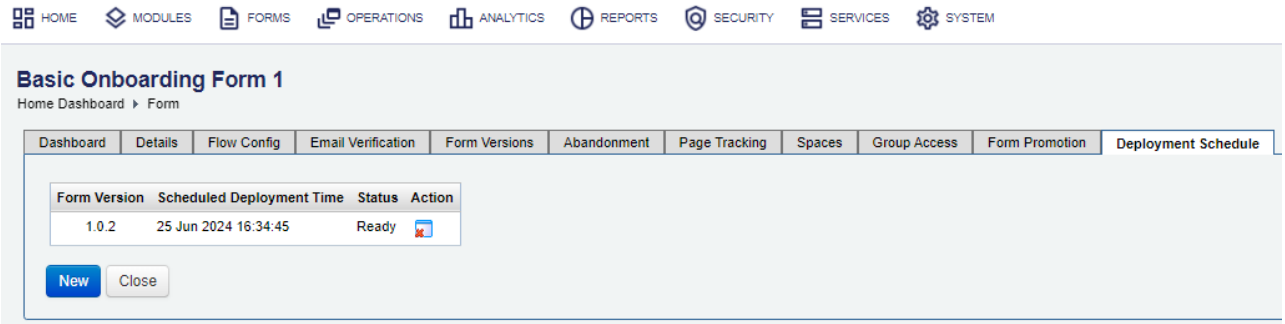
Schedule a Form Deployment

Manager allows you to specify which form version of a form is current so it will be rendered when the form is requested by a user. In addition to the manual change, which you can do using the Form Dashboard and Form Versions UI, you can schedule the change to happen automatically at a required time. This is handy when you wish to roll out a new release of a form or roll back to an older version.

Manager provides the comprehensive UI to view and configure form deployment schedules.

To schedule a form version change:

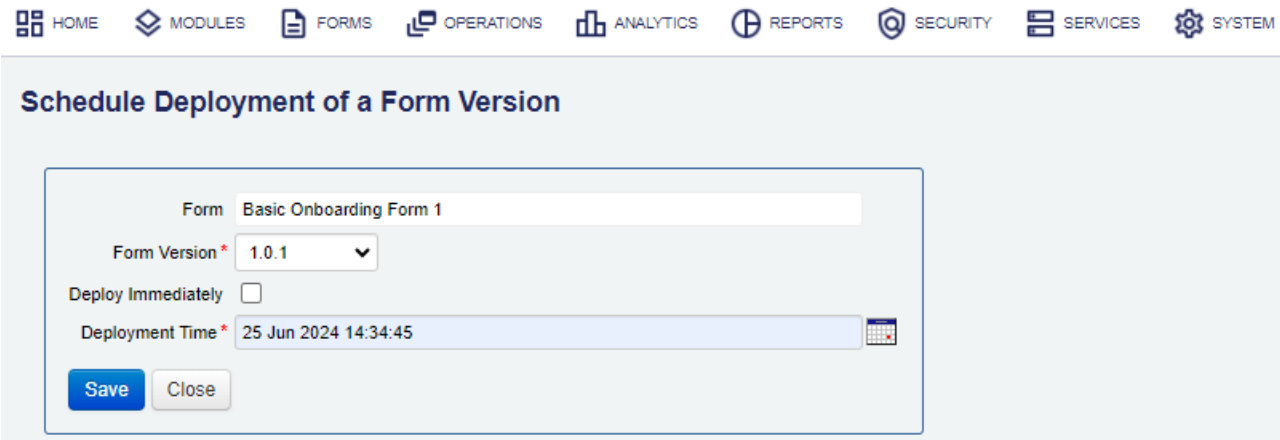
1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select the Deployment Schedule tab to see the current schedule for this form, showing all deployments that are scheduled for the future as well. If there are existing entries in the schedule, make sure that your changes will not conflict with them.



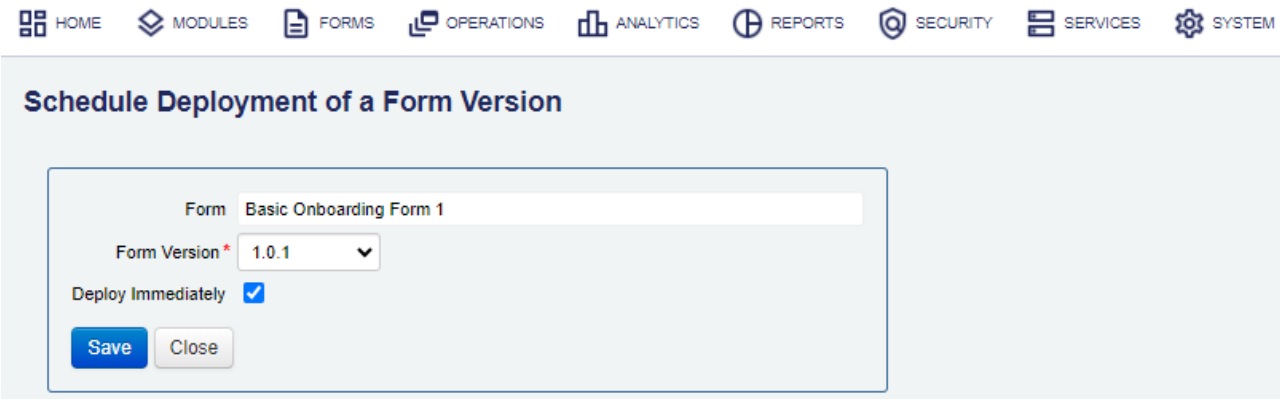
The list displays the following details:

- Form Version: a version of a form to be deployed.
- Scheduled Deployment Time: a day and time of the change to happen.
- Status: a status of the deployment, which can be one of the following:
  - Ready: ready to be deployed.
- Action: available actions, such as:
  - Remove: deletes the selected scheduled deployment.

4. Click New to create a new deployment schedule.



5. Select a new form version to be deployed from the Form Version dropdown.
6. Select the Deploy Immediately checkbox to change to the new version immediately. The form will be configured to use the new form version straight away and no schedule record will be created.



7. Enter the desired changeover date and time into the Deployment Time field.

> NOTE
>
> Do not schedule multiple deployments on the same form for the exact same time (down to the minute), because this functionality is not supported.

8. Click Save to update the changes. A schedule record will be created and the change will be carried out by the Template Version Deployment scheduled job.

NOTE

Ensure this job is configured correctly and is not paused; otherwise scheduled form version changes will not be processed.

Next, learn about form dashboard.

Configure Form Test Mode

Manager provides you with the Form Test Mode configuration so you can test a form before it is deployed in production where form applicants will access it. If you do not have multiple environments, such as test and UAT, the Form Test Mode is a great way of artificiality creating a software development cycle for a form, so developers can test a form's functionality.

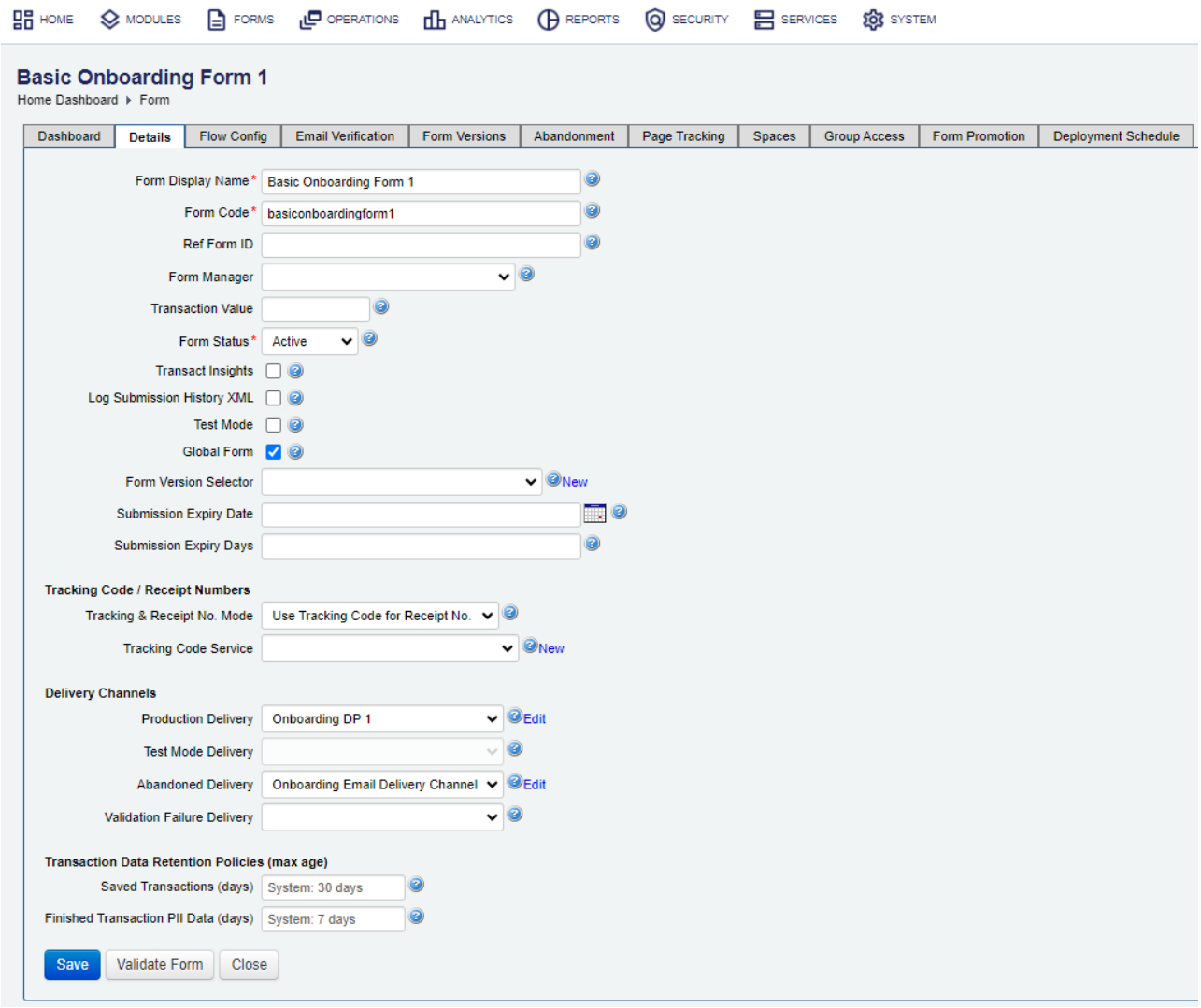
Manager uses the Test Delivery Channel to implement the Form Test Mode functionality, which imposes the following limitations:

- Forms in Test Mode are not available in TransactField App and Web Portal search pages.
- No transaction analytics and reports are created.
- Submissions don't add to an Organization's transaction count.

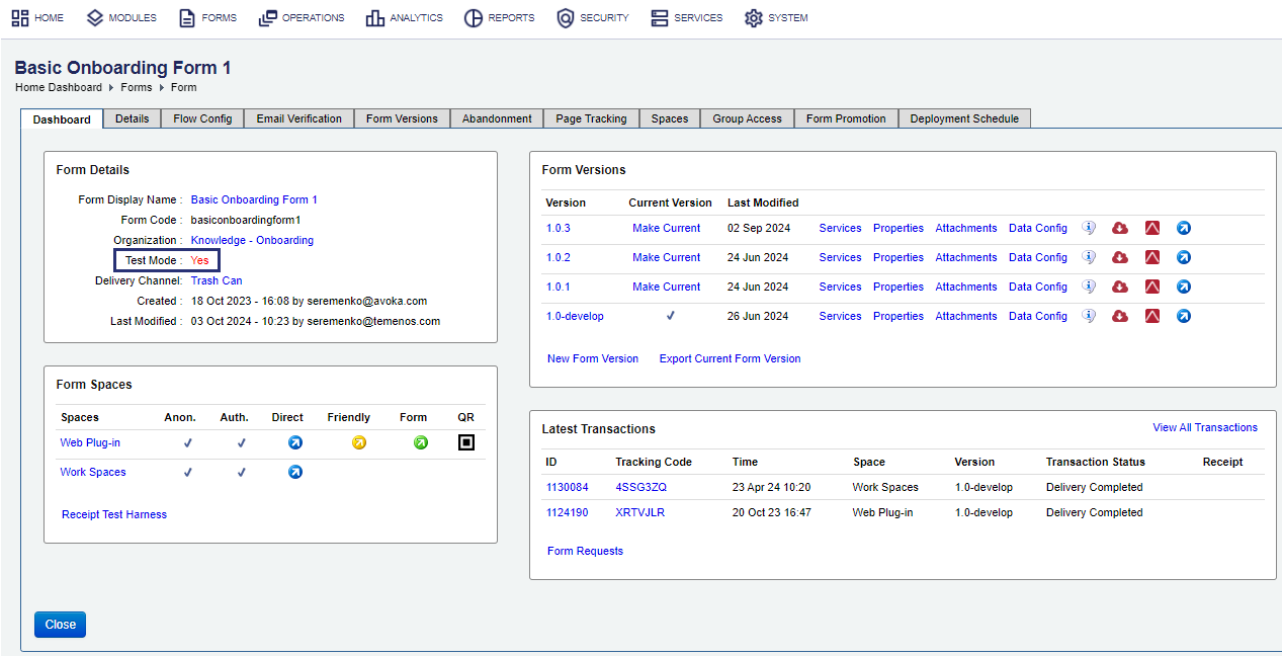
To configure the Test Mode for a Form:

1. Select Forms > Forms.
2. Select a form and then click Edit.

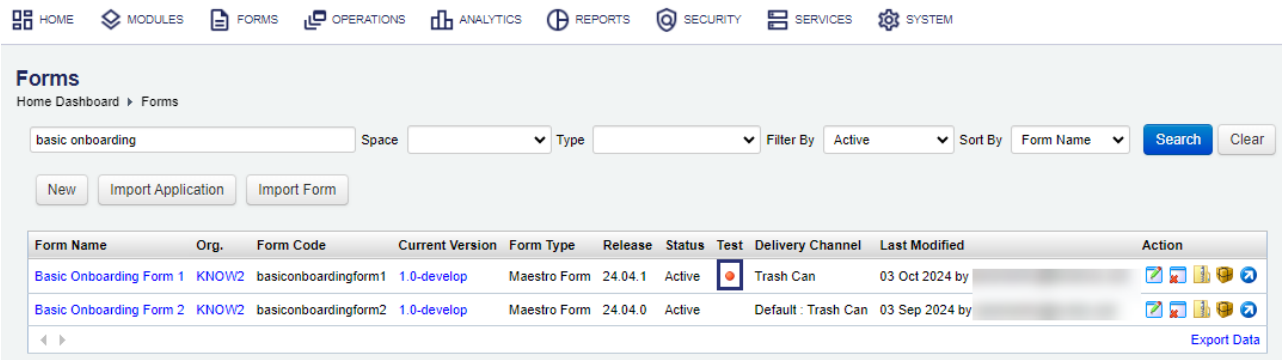
3. Click the Details tab and select the Test Mode checkbox.



4. Click Save to update the changes. This will allow you to select the Test Mode Delivery channel, which is otherwise disabled.
5. Select an organization's delivery channel from the Test Mode Delivery dropdown menu, which will be used to deliver submissions when the selected form is in Test Mode, for example, the Trash Can Delivery channel, so no submissions will be sent out.
6. Click Save to update the changes.
7. Open the Form Dashboard to verify that the form is in the Test Mode by checking whether the word Yes is displayed next to Test Mode.



Alternatively, view the form list and check if a form has a red dot is displayed in the Test column, meaning the form is in Test Mode.



Next, learn how to render a form.

Render a Form

Manager allows you to render a form without opening it in the Maestro editor, so you can check how the form will appear to a form user. The form is rendered using its configuration, such as organization and form properties, attachments and so on, but no data prefill is performed.

If you want to test form's behavior, you need to use the form test functionality, which gives you a form usability experience identical to that of a real form applicant.

To render a form:

1. Select Forms > Forms.
2. Locate a form and click Form Test Render. The form will be loaded in a new browser tab where you can fill it in normally to validate its functionality.

Next, learn how to view all forms.

Test a Form

Manager allows you to test a form without opening it in the Maestro editor, so you can check how the form will appear to a form user. The form is rendered using its configuration, such as organization and form properties, attachments and user data prefill.

Testing a form is similar to rendering it, however the form request is treated as if a form user had requested a new form. In which case, you are directed to the landing page, where you may get additional options depending on the user's form space, for example, the login. The landing page takes you to the form, which you can then fill in, submit and make payments or attach supporting documentation as needed.

Usually, you test a form to validate its behavior from the Form Dashboard, which allows you to test the form in both the production and test modes. You can also test the form by finding and opening them in the user's form space. However, this may show you only active forms in production mode, depending on how the form space is configured.

The Dashboard allows you to use both the Landing and Friendly URLs to the forms:

- Landing URL - `http://www.host.com/tmportal/landing.htm?formCode=lf-exp`
- Friendly URL - `http://www.host.com/tmportal/exp/local-form/` , which uses the client code (`exp`) and the form name (`local-form`) to form the URL. When creating the friendly URL, Manager converts the client code and form name to lower case and replace spaces with dashes (`-`), and then remove duplicate dashes. In addition, ampersands (`&`) are converted to (`and`), and all other non-alphanumeric characters are stripped out.

To test a form:

1. Open a form in the Form Dashboard.
2. Click the Landing Page link or the Friendly URL link next to the form space where you want to test the form. The landing page for the form will be loaded in a new browser tab where you can fill it in normally to validate its functionality.

Next, learn how to configure Form Test Mode.

Import Forms

Manager allows you to export and import a current form version or all versions of a selected form. After you build a form in Maestro, you usually deploy it to Manager using this import functionality.

You can also use this functionality to:

- Rollback one or several forms
- Re-create one or several forms in another Manager
- Promote one or several forms to other environments
- Migrate forms from one environment to another

Manager allows you to import a form as an application. The main difference between these imports is an archive file format that contains application XML configuration for applications or form XML configuration for forms.

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

You can import a form using one of following options:

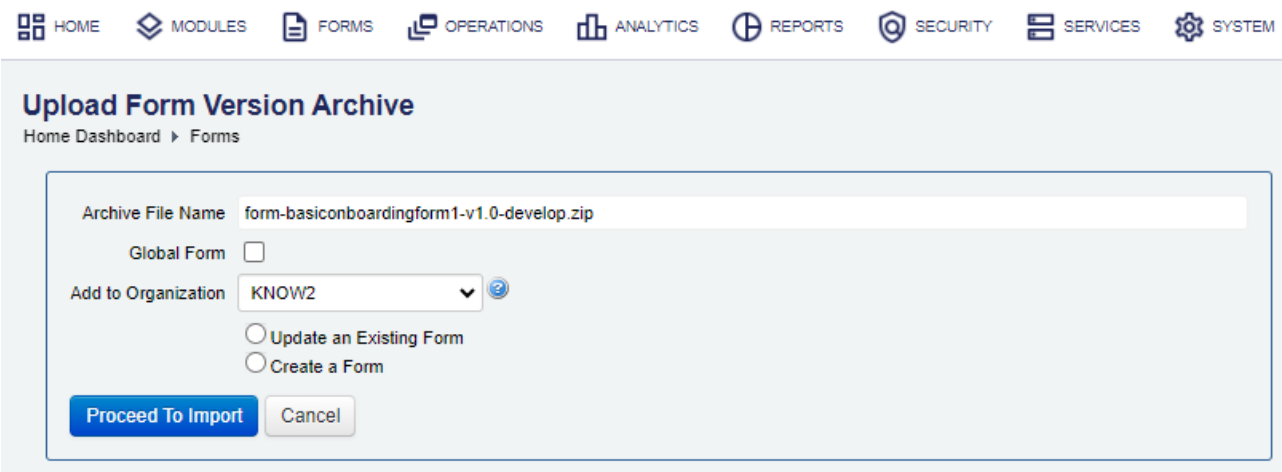
- UI
- SDK
- API

Import Using UI

To import a form:

1. Select Forms > Forms, and then click Import Form.
2. Click Choose File and choose a form archive ZIP file.

3. Click Upload to upload the file.



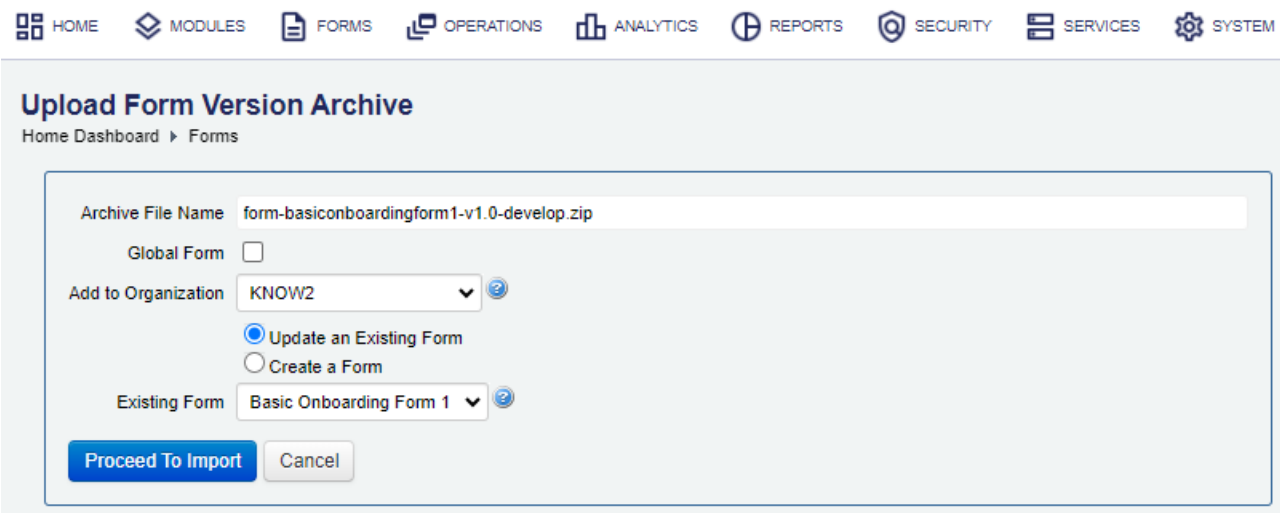
4. Select the Global Form checkbox if you want to import a form as a global form. | 23.10 This feature was introduced in 23.10

> WARNING
>
> The experimental Global Forms feature has been temporarily disabled while we continue improving it. This feature will be disabled in releases starting from 24.10 and going back to 23.10.

5. Select an organization to add the form to from the Add to Organization dropdown list, or leave it blank.

> NOTE
>
> The Add to Organization dropdown list is shown only if you have access to more than one organization.

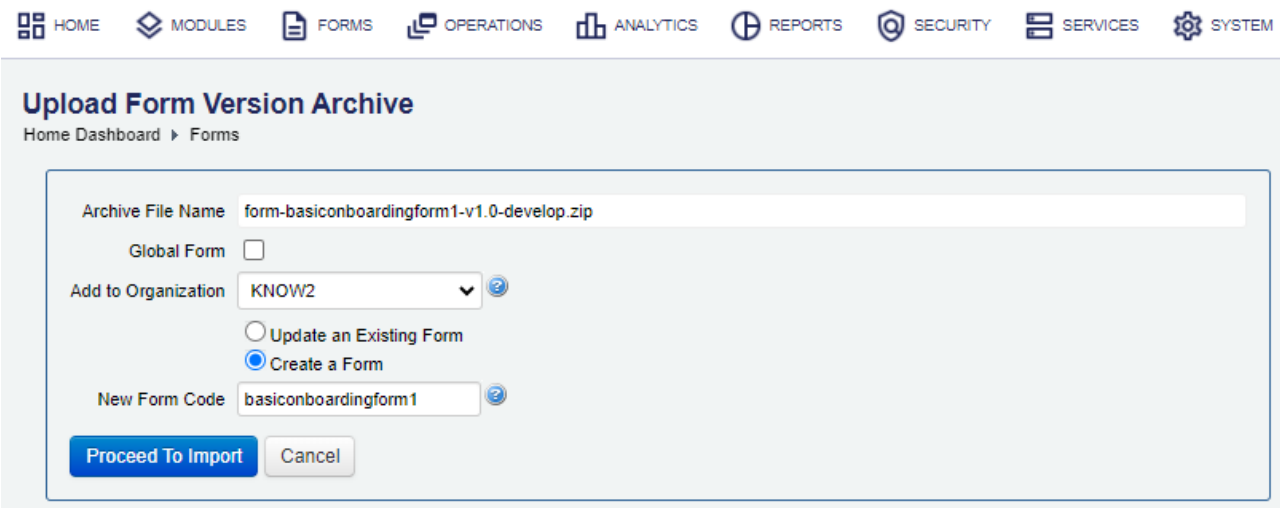
6. Select the Update an Existing Form checkbox, if the form already exists in Manager and you want to update it with the new version.



The additional field is shown where you need to:

- Select a form name you want to update from the Existing Form dropdown list.

7. Select the Create a Form checkbox if the form doesn't exist in Manager, so it will be created.

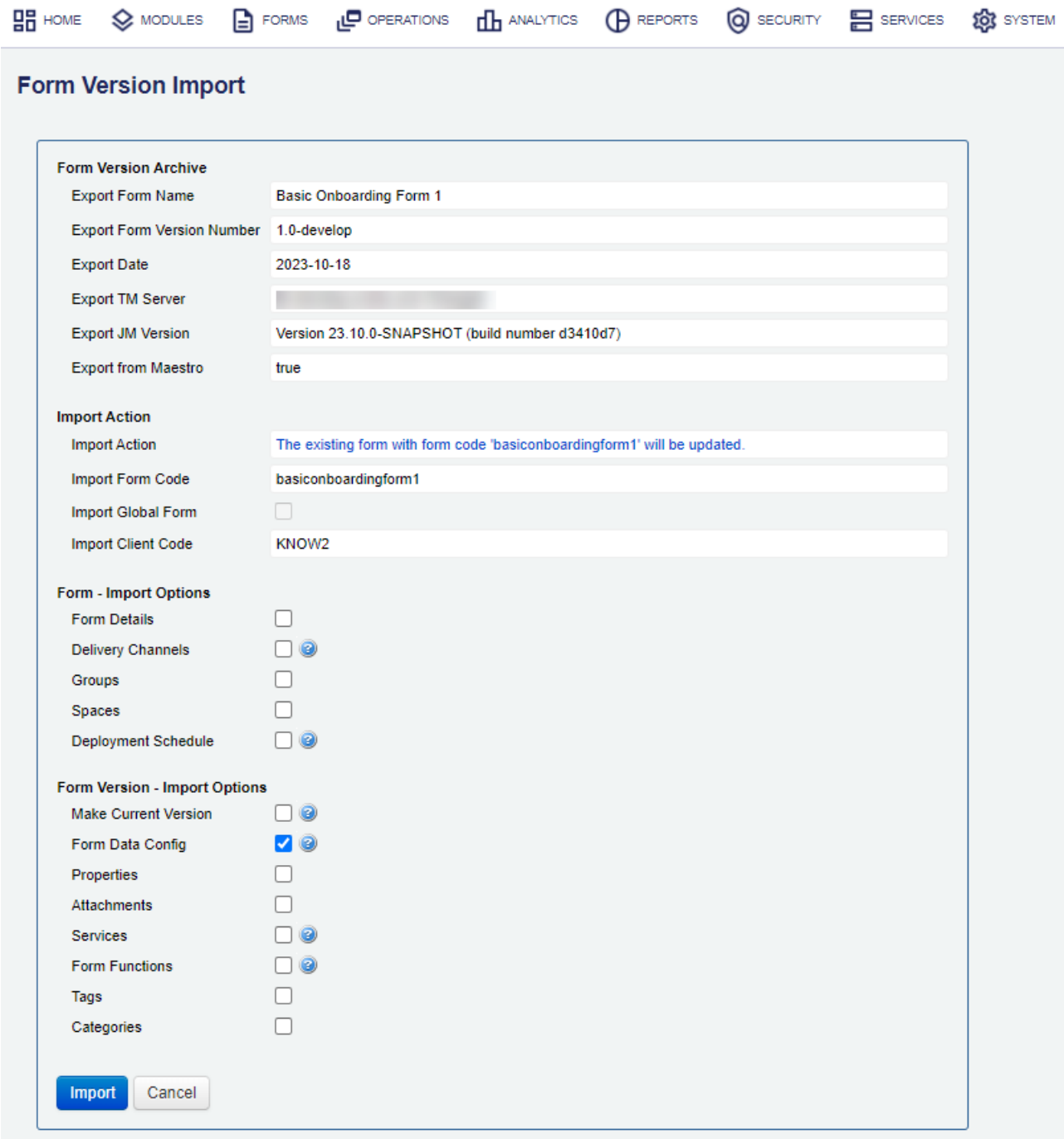


The additional field is shown where you need to:

- Update a form code in the New Form Code text field, which uniquely identifies the form across all the organizations in the current server environment. Manager automatically prefills the form code.

> NOTE
>
> The form code is an alphanumerical string that can't include spaces.

8. Click Proceed To Import to review and adjust the import options before you actually import the form.



9. Select the appropriate Form – Import Options, which are associated with all versions of the form. They are all selected by default.
10. Select the Form Details checkbox to import form details.
11. Select the Delivery Channels checkbox to associate form's delivery channels with any existing organization's delivery channels with the same name.

12. Select the Groups checkbox to import the list of groups the form allows access from.
13. Select the Spaces checkbox to import the list of form spaces assigned to the form.

    WARNING

    If you import a form with form spaces assigned to it, which don't exist in a target environment, these form spaces won't be automatically assigned to the form and will be ignored. In this case, you will need to import the missing form spaces and assign them to the form manually.

14. Select the Deployment Schedule checkbox to remove all currently scheduled form version deployments and replace them with the deployment schedule included in the import archive file. The history of the past deployments will be preserved.
15. Select the appropriate Form Version – Import Options, which are applied to a selected version of the form, normally the current version.
16. Select the Make Current Versioncheckbox to set this form as the current version of the form.
17. Select the Form Data Config checkbox to import all form versions form data configuration.
18. Select the Properties checkbox to import form properties.
19. Select the Attachments checkbox to import attachments.
20. Select the Services checkbox to import services, which will override the services used by this form version in the target system.

    NOTE

    The services are referenced by a name and a version and must exist in the target system prior to the import, otherwise they can't be imported.

21. Select the Form Functions checkbox to import form functions used by the form version. The existing form functions on the target server will be updated with that of from the imported archive. | 19.11 This feature was introduced in 19.11.

    NOTE

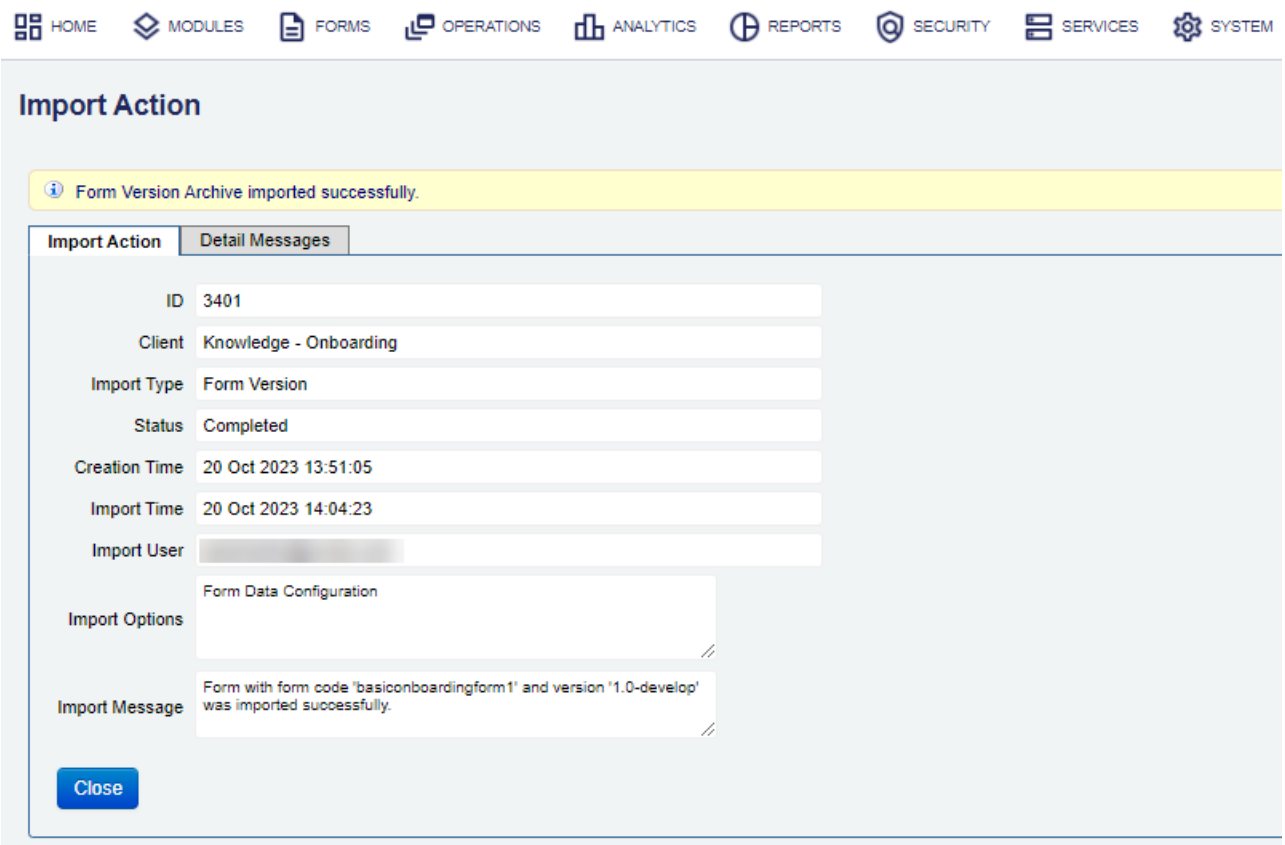
    The form functions are referenced by a name and a version and must exist on the target server prior to the import, otherwise they can't be imported.

22. Select the Tags checkbox to import tags.
23. Select the Categories checkbox to import categories.

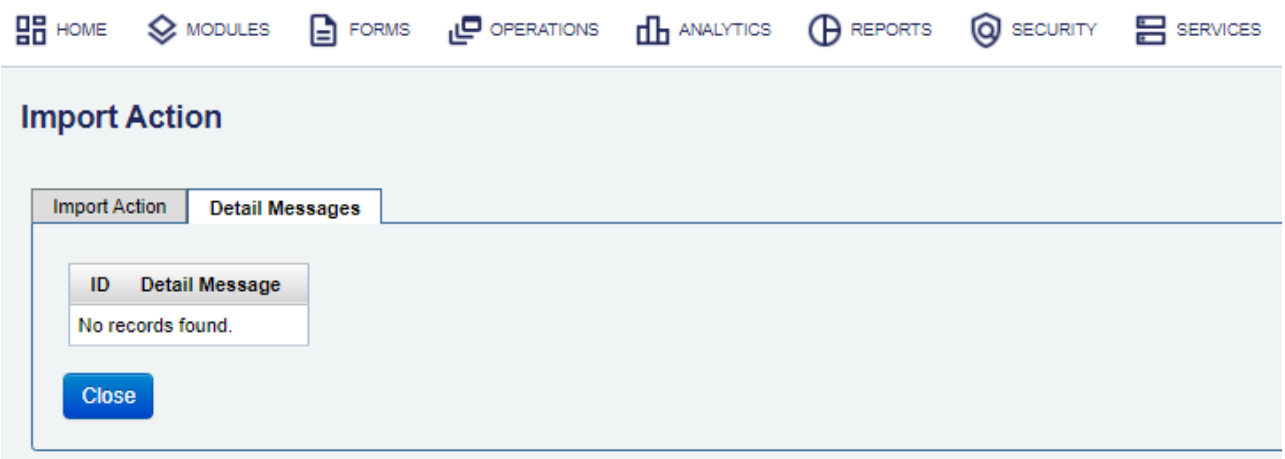
24. Click Import to complete the process.

> NOTE
>
> If the Create a New Form option was selected during the import process, and there was a form naming conflict, that is a form with the same name already existing on the server, the form will be renamed, and a message will be displayed in the Detail Message tab.

25. Check the import status and message in the Import Action tab.



26. Select the Detailed Messages to see whether there were any errors or warnings during the import.



If errors are sever, you may want to update Manager configuration to fix the error and import the form again.

27. Click Close.

You should check the list of forms to verify the imported form has been added, as well as the form's configuration is correct, for example, all form spaces are assigned to the form.

> NOTE
>
> You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

> NOTE
>
> You can also import Maestro forms using the Maestro dashboard.

When importing a form that has come from a more recent version of Manager, you may get an IllegalArgumentException error. To resolve this error:

1. Log onto Manager you have imported the form into.
2. Select Forms > Forms and locate the form.
3. Click the Current Version of the form and change the Form Type to Composer Smart Form . This allow you to render this form correctly now.

Import Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:archive-deploy goal to import a form.

Import Using API

The Journey Manager REST API provides the endpoint URL to import a form.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s)://<JM_SERVER>/manager/secure/rest/archive/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as Base64[1].

`--form ‘archiveFile=@“/Users/my-form.zip”’` - a path to an archive file to be imported.

`--form ‘importOptions=“FORM_DEFINITION,DELIVERY_DETAILS,FORM_SPACES”’` - a comma-separated list of import options to perform. The option names are the same as described in the Import Using UI section but they use an underscore, for example, `FORM_SPACES` for Spaces.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

Example

```
curl --location --request PUT ‘https://<JM_SERVER>/manager/secure/rest/archive/v1/’
\
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
--form ‘archiveFile=@“/Users/my-client.zip”’ \
--form ‘importOptions=“FORM_DEFINITION,DELIVERY_DETAILS,FORM_FUNCTIONS,FORM_
GROUPS,FORM_SPACES,CURRENT_VERSION,\
DEPLOYMENT_PLAN, FORM_DATA_CONFIGURATION,ATTACHMENTS,PROPERTIES,VERSION_
SERVICES,METADATA,CATEGORIES,\
COPY_FROM_LATEST_VERSION”’
```

Next, learn how to export forms.

[1]Base64 is a binary to a text encoding scheme that represents binary data in an American Standard Code for Information Interchange (ASCII) string format.

Import a Form Application

Manager allows you to import a form as an application package.

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

To import a form as an application, see Import an Application Package.

Check the list of forms to verify the imported forms have been added.

> NOTE
>
> You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

Next, learn how to import a form.

Export Forms

Manager allows you to export and import a current form version or all versions of a selected form. This process is commonly used to migrate forms from one environment to another. You can use this functionality to:

- Backup or archive forms.
- Re-create or migrate forms into another Manager.

To export one or several form versions:

1. Select Forms > Forms.
2. Click Export Current Form Version to export one form version or click Export All Form Versions to export all form versions.

Manager exports form versions into a form archive ZIP file, which will be downloaded to your PC.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

> NOTE
>
> You can also export Maestro forms using the Maestro dashboard.

When exporting a form from a more recent version of Manager to an older version, you may receive the Could not set property `formDataEncryptFlag` on entity TemplateVersion error message. However, as we're exporting between two different versions of Manager, the original form has this flag set on it but it doesn't exist in the new environment so it can't be set. This will not stop the form from functioning correctly.

Next, learn how to import forms.

Remove a Form

Manager allows you to remove a form, which also deletes all form versions, transactions and submissions associated with the form.

> NOTE
>
> We recommend making a form inactive rather than remove it, as it may impact form users.

Make a Form Inactive

To make a form inactive:

1. Select Forms > Forms and locate a form.
2. Select the Details tab to open form's details.
3. Select Inactive from the Form Status dropdown list.
4. Click Save to update the changes.

> NOTE
>
> If a form has form versions, you can mark a particular form version as Inactive instead of disabling the whole form. For more information, see Configure a Form Version.

Remove a Form

To delete a form:

1. Select Forms > Forms and locate a form.
2. Click Remove and click OK confirm your decision.

Next, learn how to view all forms.

Migrate Forms

Manager provides a mechanism to migrate forms from one environment to another. It allows you to develop and test forms in different environments before migrating them over to the production environment.

To migrate one or several forms:

1. Export forms from one environment.
2. Import forms to another Manager environment.

> NOTE
>
> You can use the same process to perform a form rollback.

Next, learn how to view forms.

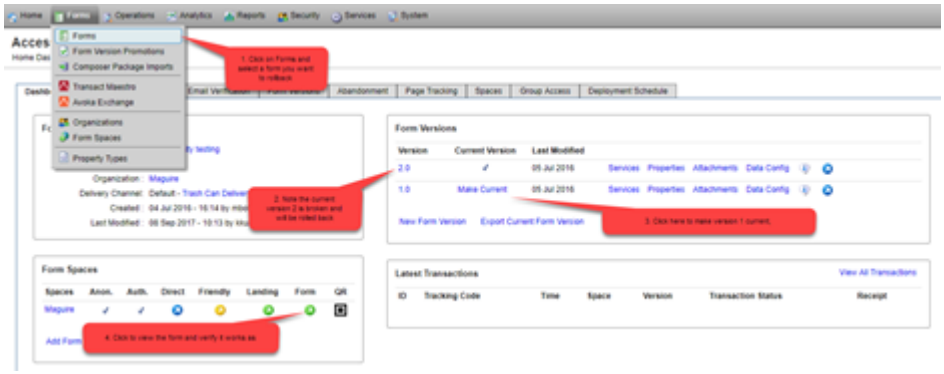
Rollback Forms

Manager provides a mechanism to rollback forms so you can restore your environment to correct form deployment failure or an error. Forms, installed as application packages, can be easily rolled back to previous versions in. However, you must:

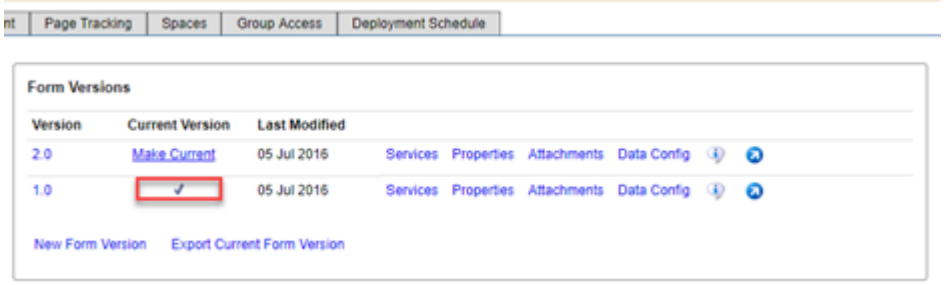
- Understand all components involved into the workflow
- Have backups of the current and previous versions
- Maintain your change logs records, so you know the latest working versions of applications

To rollback a form:

1. Locate a form you want to rollback.
2. Click the Make Current link of the previous form you want to rollback to, for example the version 1.



3. Confirm that the current version is what you have rolled back to, for example 1.



4. View the form in the browser and verify it works as expected. Your rollback is complete.

NOTE

If you have only one application deployed, you can not roll it back as shown above. However, and you can restore it from a backup using form import, or contact our Customer Care team for help.

NOTE

Be careful of having versions of forms or services which are incompatible with each other, as it will result in a error and inability to rollback to your previous forms.

Next, learn how to view forms.

Save Challenge

Journey Manager comes with a Save Challenge feature that allows users to resume their form applications, so they can complete or cancel them. This is an important step in the onboarding process when a user needs to save a form application to get some required information or a web browser is closed or crashed. Therefore, the Save Challenge provides a secure mechanism to return to the form.

To enable a Save Challenge:

1. Select Forms > Forms.
2. Locate a form and select the Flow Config tab.
3. Select Enabled from the Save Online dropdown list. For more information, see Configure the User Flow options for the form
4. Provide a valid path in the Save Challenge XPath field in Form Configuration Mapping, for example, `//DateOfBirth`.
5. Check the `supportsSaveChallenge` is set to `true` in the SDK `form-def.json` file.

   > NOTE
   >
   > The `application-package-config.xml` is updated with this option after running the `app-package` command.

6. Check the Supports Save Challenge checkbox is selected in a form version.
7. Check the Save Challenge Data checkbox is selected in Transaction Status.

Next, learn abut other Manager's features.

Maestro Form Static Content Externalization

Manager includes a new Journey Maestro form rendering mode, Static Content Externalization[1] (SCE), which significantly reduces the time it takes to load large forms. This is important because long application loading times can significantly increase the bounce rate with users closing the browser window before starting the application.

Using this new Maestro SCE rendering mode option, Manager will externalize 85-95% of the Maestro HTML form content, which can then be served in parallel by Content Distribution Networks[2] (CDNs). This reduces the amount of time it takes for a form to initialize as the initial HTML page is downloaded quickly and the larger static images, JSON[3] and JavaScript[4] content can be downloaded in parallel by separate browser threads. By serving this static content out of CDN, the content is more likely to be close to the users device, reducing the network latency. The content can also be cached by the users browser.

It's possible to achieve a 2x reduction in form load times for large Maestro applications.

There are also benefits of using Maestro SCE rendering mode option for the Manager server as we have observed a 20% throughput improvement under performance load tests

The SCE mode works with older Maestro release forms and does not require any changes to the Maestro forms to use it. This feature doesn't change the structure of the form but simply externalizes static resources, such as images, JSON and JavaScript content. The form is still a single page form with static resources have been separated out.

Next, learn how to enable the Maestro SCE Mode opt in feature.

---

[1]Static Content Externalization (SCE).
[2]A content delivery network or Content Distribution Network (CDN) is a system of distributed servers (network) that deliver pages and other Web content to a user, based on the geographic locations of the user, the origin of the webpage and the content delivery server.
[3]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.
[4]Javascript (JS) is a scripting languages, primarily used on the Web. It is used to enhance HTML pages and is commonly found embedded in HTML code. JavaScript is an interpreted language. Thus, it doesn't need to be compiled.

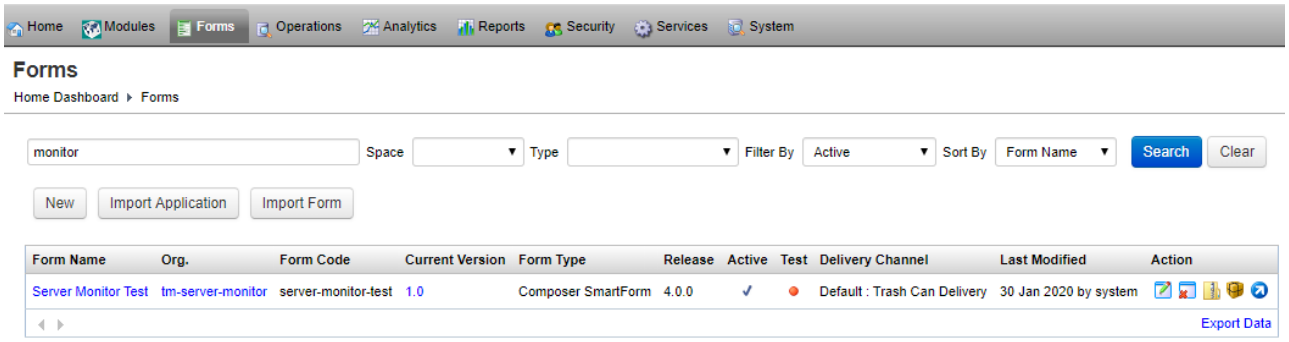
Server Monitor Test Form

Manager comes with a Server Monitor Test form, which you can use to check Manager server heartbeat to make sure it's up and running. This is often done for the cloud based installations.

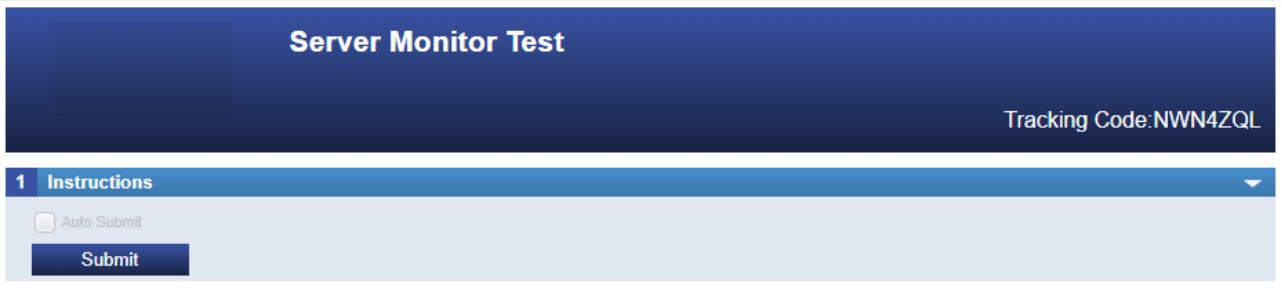
> NOTE
>
> The Server Monitor Test form is only available if the Deploy Monitoring Form checkbox is selected during Journey Manager installation.

To run the Server Monitor Test:

1. Select Forms > Forms.
2. Search for the Server Monitor Test form.



3. Click Form Test Render.
4. Click Submit.



5. Check the submission was successful.
6. Type your email address and click Send Now to send the PDF copy to validate the delivery.

Next, learn how to view all forms.

Integrate Forms to Existing Websites

Manager allows you to integrate its forms to existing websites. The modern trend is for launching forms from existing websites is to open a new browser tab or window. This has a few important advantages:

- Works better on mobile devices where screen real-estate is limited
- Prevents the user from moving away from the data entry experience by clicking on links in the surrounding page
- Is simpler from an implementation perspective

You can find several integration strategies listed below:

iFrame Embedding

Form Display Mode is Embed with iFrame.

Pros

- Simple
- No "leakage" of JavaScript or CSS from parent page to form page

Cons

- Pages although appear to be integrated are actually separated causing a number of issues.
- Refreshing of the page is problematic.
- Back button does not work properly.
- Difficulties in maintaining styling with significant effort needed on BT to maintain.
- Difficult to pass control between the pages.
- Complexities in the JavaScript as the parent control needs to be passed to Manager.
- Potential responsive design issues, especially on smaller devices.

<DIV> Embedding

Form Display Mode is Embed with a DIV.

Pros

- Solves some of the problems associated with iFrames.

Cons

- Complexity around the sharing of JavaScript used to render the contents of the form into the DIV element.
- Potential for code bleed or CSS clashes
- We don't recommend this integration

Launch a new tab or window

Form Display Mode is Direct Render.

Pros

- Clean code separation
- Clear boundary of ownership
- Control easily and intuitively passed between NextGen UI and the Manager on-boarding pages
- Form access is controlled at the URL level
- Recommended by Temenos as their preferred approach with their other customers

Cons

- The form must be built with a simple header that includes a small amount of branding from the web site
- Visual design - need to be simplified in terms of shared header

Configure the Form Display Mode

The Form Display Mode is the process of embedding the form, either with a DIV or iFrame, to display the form inside a portal page, where that page typically provides the header, footer and sidebars. Direct Render displays the form using a Servlet and does not include any elements outside the form itself. For more information on the Form Display Mode setting, see form flow configuration.

Form URLs

These different methods of displaying the form mean that different URLs are used by Transact, as follows.

URL	Usage
<portal>/form.htm?formCode=<form code>	Form Page which will embed a new HTML or PDF form in the page using an IFRAME

	or DIV tag. The form is identified by the formCode parameter.
<portal>/secure/form.htm?formCode=<form code>	Same as above but under a secure path, which requires the user to be authenticated
<portal>/form.htm?submitKey=<submit key >	Form Page which will embed a saved HTML or PDF form in the page using an IFRAME or DIV tag. The form submission is identified by the submitKey parameter.
<portal>/form.htm?taskKey=<task key >	Form Page which will embed an assigned HTML or PDF form task in the page using an IFRAME or DIV tag. The form task is identified by the taskKey parameter.
<portal>/servlet/SmartForm.html?...	Same as above options but doing a direct render , where by the form is not being embedded in another HTML page

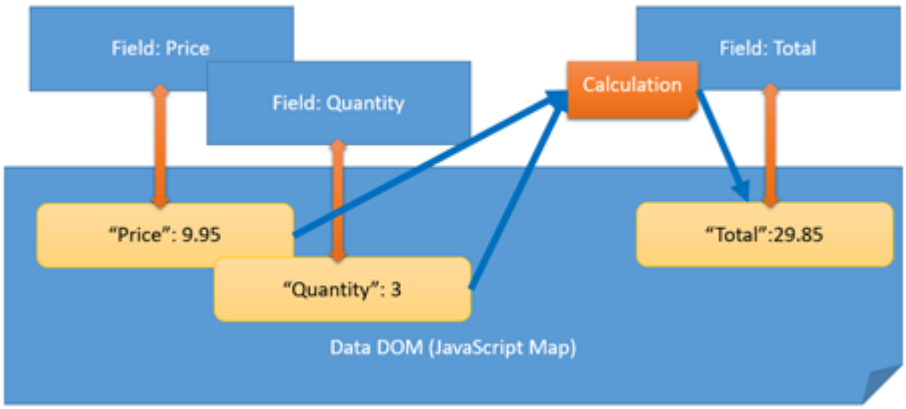
When a HTML form is being rendered on a mobile/tablet device (iPad, phone, etc.) it will automatically redirect to the /servlet/SmartForm.html path, even if the form is configured to be embedded in the form page. This is because embedding HTML forms does not always work well on mobile/tablet devices.

Next, learn how to view all forms.

Maestro Data Model

The Maestro data model is a powerful yet simple abstraction that allows you to use much of the Journey Maestro functionality when you are developing a form, so it is important to understand it.

When you type a piece of data into a component in Journey Maestro, the data is not only displayed in the component itself but is also copied to an internal data structure known as the data model, or the data DOM[1]. The component and its value are sometimes known as the view. These terms both derive from well-known programming pattern Model-View-Controller. In our case, the controller is any business rule that is defined in Maestro, as shown below.



Maestro uses a unique field ID to identify both the component and its value in the data model. Internally, the data model is implemented as a JavaScript map. You can think of the data model as being a spreadsheet with two columns: the ID of the component and its value. This is illustrated with a sample table.

Field ID	Value
"Price"	9.95
"Quantity"	3
"Total"	29.85

[1]The Document Object Model (DOM) is a cross-platform and language-independent application programming interface that treats an HTML, XHTML, or XML document as a tree structure wherein each node is an object representing a part of the document. The objects can be manipulated programmatically and any visible changes occurring as a result may then be reflected in the display of the document.

The component's value and the value in the data model are automatically synchronized in both directions. Therefore, when you are programming in Maestro, it is usually much easier to modify the data in the data model and let the automatic synchronization take care of updating the visible values in the components themselves. If you refer to the above diagram, you will see that the calculations refer to the values in the data model, rather than referring directly to the components themselves.

For example, when you are in the Script Editor, and double-click on the price component, what is inserted into your script is "data.price". This is not actually referring to the component itself, but rather referring directly to the component's value in the data model. The "data." is the clue that you are referring to the data model. The values in the data model can be both read and written, and the synchronization will take care of updating all the components in the view automatically.

To learn how to use Data Mapper in Journey Maestro, see data mapper.

It is important to ratify your data model upfront as much as possible before setting out - and do not assume that a model which represents a downstream integration is a good way to operate.

For example, if you are integrating with Vendor A, and Vendor A requires the following format of output for "pets":

```
<Classification>
   <PartA>
      <Pets>
         <Pet>
            <Name/>
            <Type/>
         <Pet>
      </Pets>
   </PartA>
<Classification/>
```

It may be tempting to include this structure as part of your form. This has the benefit of making the payload meet the requirements for integration with little work. However, the forms will suffer as small tweaks to the schema for output require republishing every form.

Consider a client data dictionary, as it has several benefits, including:

- Ownership - is in the client's estate, they have to agree on data definitions up front and it removes uncertainty.

- Integration – switching vendors in and out becomes less problematic as you have one working model and various translation layers.
- Certainty – over time, model ownership synthesizes to good working practice, extensibility, and surety of operation.

When working with complex data structures, you can't set complex JSON (arrays, nested structures) into a data field bound to the payload XML (saved data to Manager) and expect it being prefilled back on resume as a JSON object value. This is not supported in Maestro. Any value stored in an XML bound data field must be of a primitive type which can be easily serialized as a string (int, float, string, boolean). If you do store a complex JSON to a bound data field, Maestro will try to recreate the JSON structure as XML nodes with elements structure. It will be still saved to Manager as form XML but when resuming a form, Maestro won't recreate the JSON object from the XML data.

To go around this, you can choose one of the following recommendations:

- Try to use repeats which mimic your JSON structure using Maestro widgets and bindings for data storage and have unbound data field as the data source for your data drive implementing calculate rules from the data in the repeat.
- Store the JSON to a bound data field as a serialized JSON string via JSON.stringify() but you would need to decentralize it back from the bound data field into another one used as the data source for you data driven widget on form load.

The Maestro editor comes with the Integration pane so you can easily configure component's data extracts and integration with Manager.

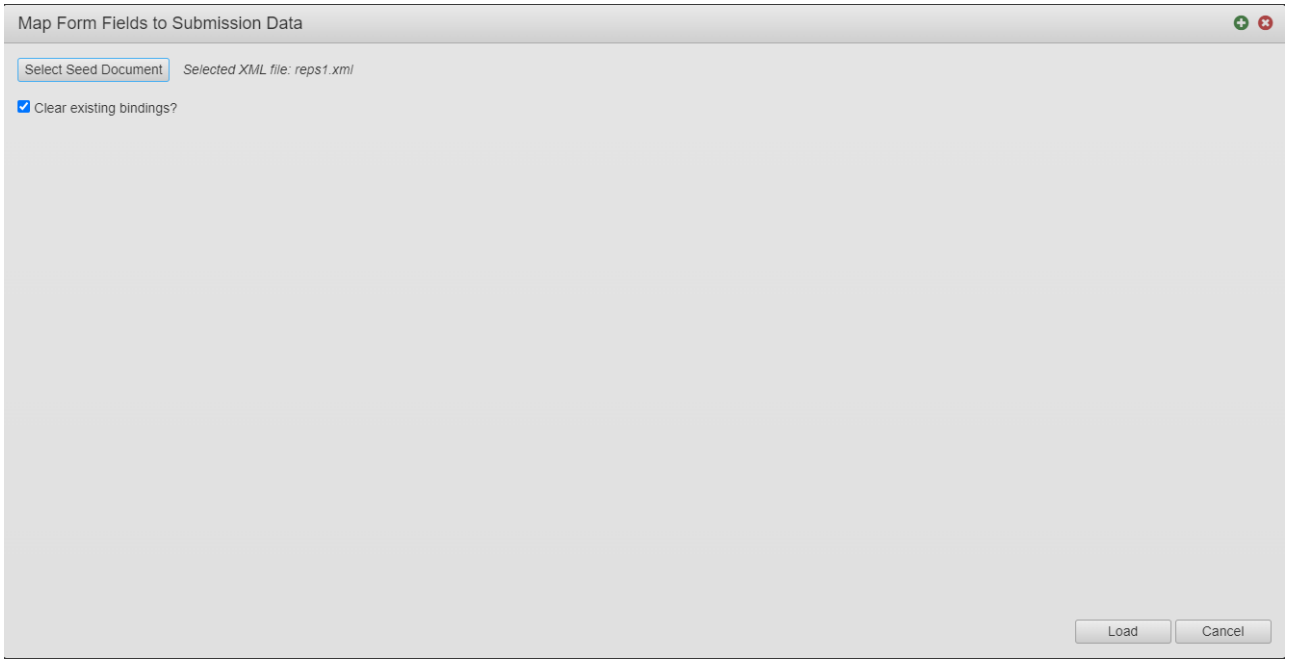
Next, learn about the unified application data document model.

Data Mapper

The Maestro editor allows you to map an existing client schema, represented with an XML file, into a form. This is a quick and easy way to set up a form compatible the desired submission output. For more information, see form data XML.

To map form fields to submission data:

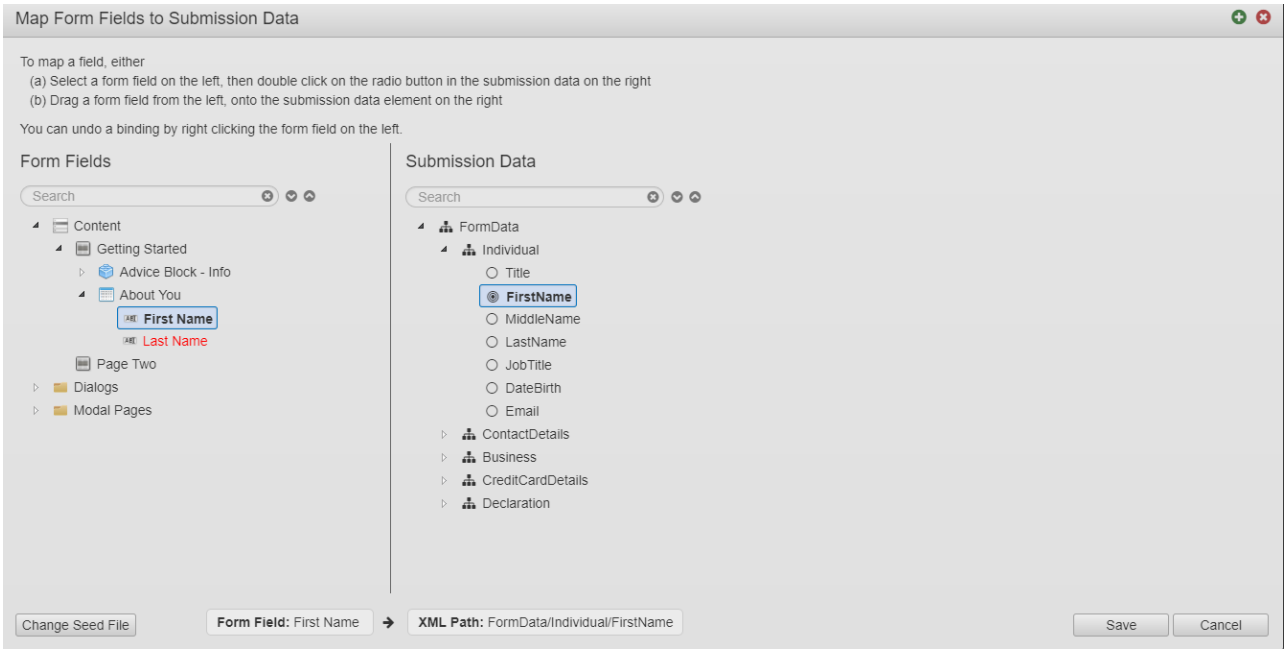
1. Open a form in the Maestro editor and click Settings Menu.
2. Select Open Data Mapper to map form fields to submission data.



3. Click Select Seed Document to choose an XML file to upload from your PC. A file content can be like one shown below:

```
<root>
        <firstName/>
        <lastName/>
</root>
```

4. Select the Clear existing bindings checkbox to clear data bindings.
5. Click Load to upload the file.



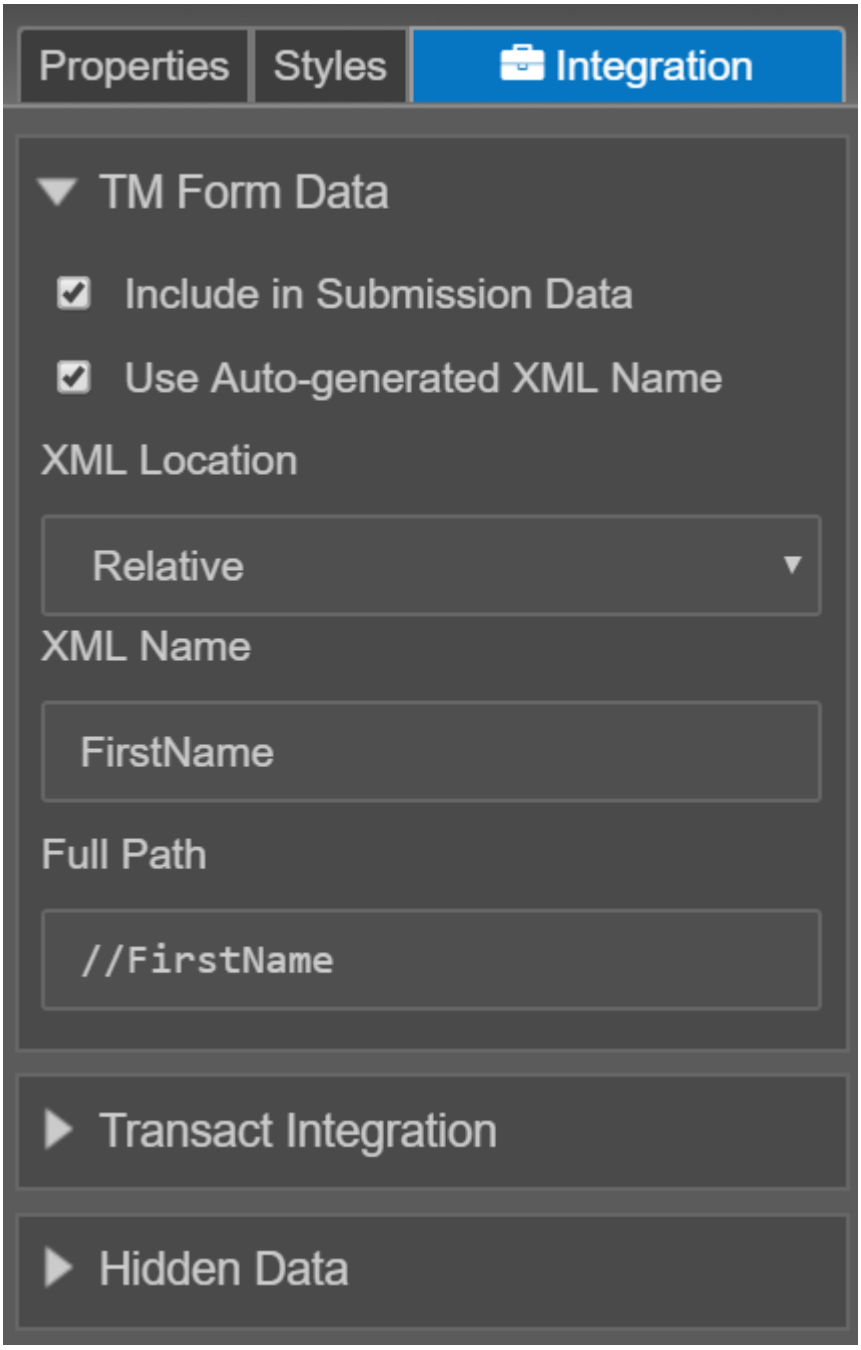
6. Map the components from the Form Fields column to the Submission Data column by dragging it across.
7. Click Save.
8. Click Build and Render Form to see how this mapping affects the form

Next, learn about form XML data payload.

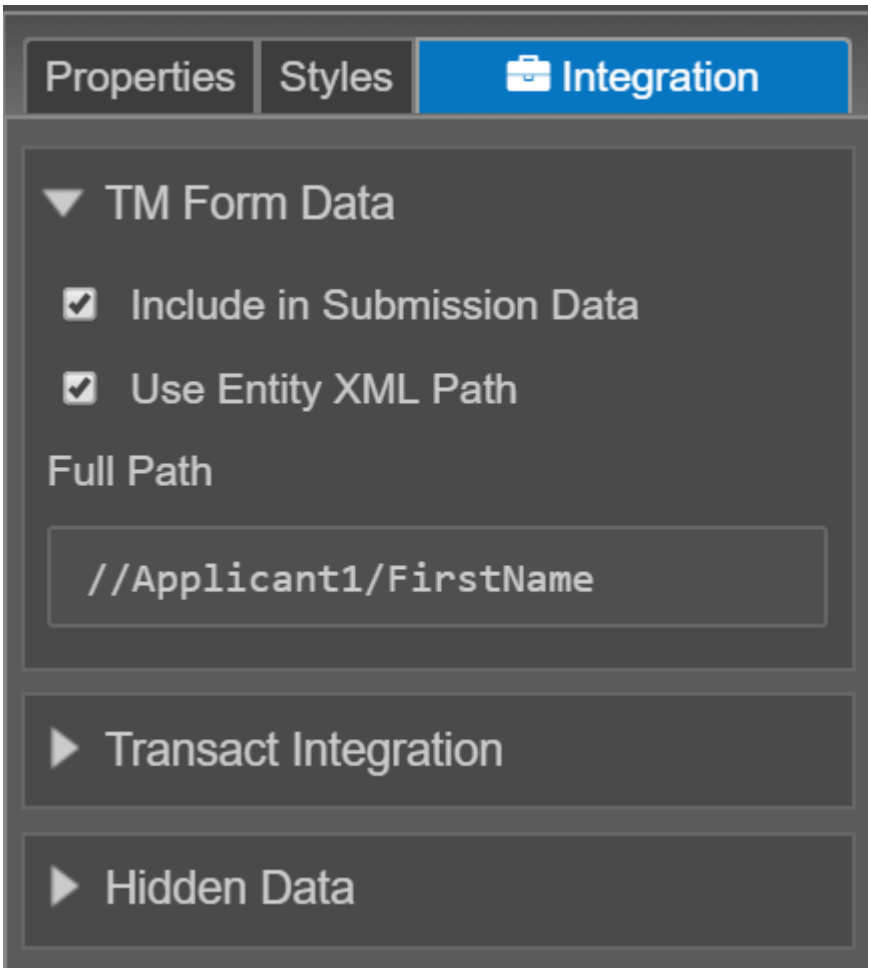
Component XML Name and Entity XML Path

Journey Maestro's Integration pane shows a component's XML Name and Full Path properties. By default, XML Name and Full Path are populated based on the component ID value. You can see the difference between an ID without an entity and one with an entity below:

ID without an Entity



ID with an Entity



If Use Auto-generated XML Name is not selected, you can change the XML Location, and XML Name and Full Path are not automatically updated when the component ID changes.

If Use Entity XML Path is not selected, you can change the Full Path and it is not automatically updated based on the component ID changes.

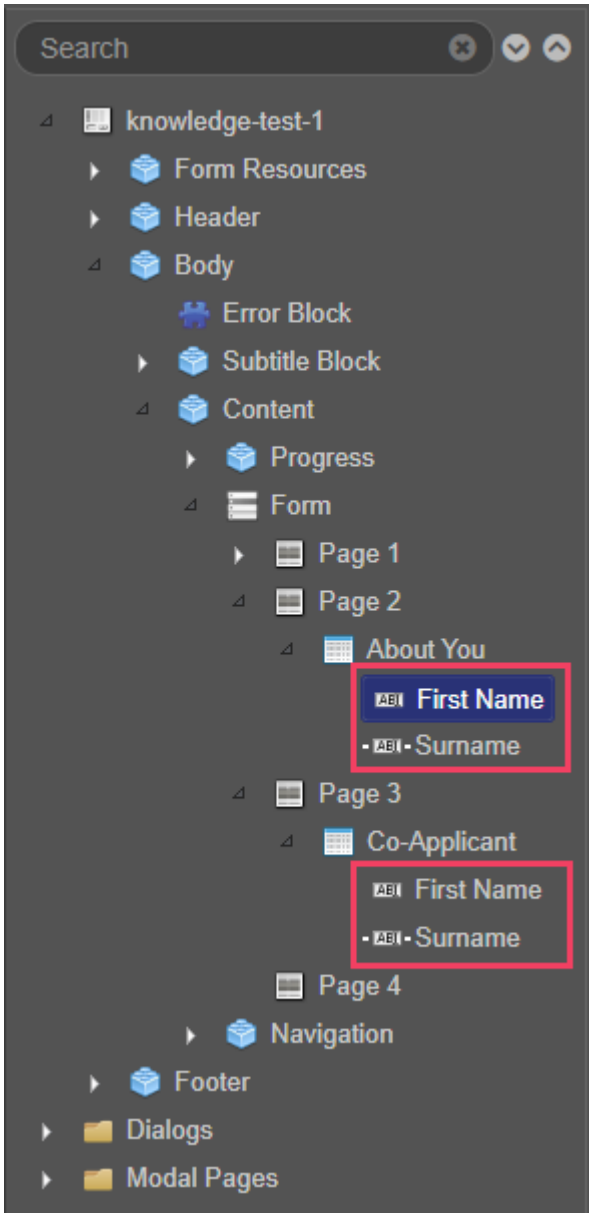
Next, learn how Journey Maestro data model.

Form XML Data Payload

Each Journey Maestro form has an XML data payload, which is the XML layer that represents data pertinent to a form as well as user's input into the form. This XML is used to pass data between Manager and a form application and for external systems integrations. The XML layer is created when you build a form when most components are added to the XML layer by default, except for pages, sections, blocks, and data fields. For more information, see data model overview.

The form XML data payload is an XML document with a flat structure that includes only components, but not the hierarchy they belong to. That is, when you add a component to a form, its ID is auto-generated in a way it's unique across the form. When you add another component, even with the same Label, its generated item's ID will be different. This is reflected in the form XML data payload.

Let's illustrate it with a form which has 2 TextFields - First Name and Surname - that are used in 2 different sections of the form - About You and Co-Applicant.



The default, an XML tag is based on the item's ID, which can't be the same in two different items. When adding the second First Name item, its ID is automatically set to `firstName_1`, which means its XML tag is `//FirstName_1`. The resulting form XML data payload looks like this:

```
<AvokaSmartForm>
   ...
   <FirstName/>
   <Surname/>
   <FirstName_1/>
   <Surname_1/>
   ...
</AvokaSmartForm>
```

Even though the Maestro editor never allows two items in the same form to have the same ID, the user can manually change the binding path for any item. However, care should be taken not to duplicate binding paths, as this can lead to data loss. For example, if you decide to change the binding path of the second item to `//FirstName` in the Integration pane, this will result in the duplicate bindings error message while building the form.

> NOTE
>
> The Maestro editor allows you to check for duplicate data bindings as well as resolve them while in design mode.

The structure of the form XML data changes depending whether you set blocks to Include in Submission Data or not, which adds layers to the structure. In this example, we selected the Include in Submission Data checkbox of the About You and Co-Applicant Section components, so the new structure looks like this:

```
<AvokaSmartForm>
   ...
   <AboutYou>
       <FirstName/>
       <Surname/>
    </AboutYou>
    <Coapplicant>
        <FirstName_1/>
        <Surname_1/>
    </Coapplicant>
   ...
</AvokaSmartForm>
```

If you would like to have the second elements as FirstName and Surname as well, they just have to be in different parents inside the XML, namely a binding path of `//AboutYou/FirstName` and `//Coapplicant/FirstName`. In this case, the XML structure is shown below:

```
<AvokaSmartForm>
   ...
   <AboutYou>
      <FirstName/>
      <Surname/>
   </AboutYou>
   <Coapplicant>
      <FirstName/>
      <Surname/>
   </Coapplicant>
   ...
</AvokaSmartForm>
```

Also, you can easily achieve this by using entities so the item's ID are like these `AboutYou_FirstName` and `Coapplicant_FirstName`, which is the easiest way to handle this binding.

Next, learn about component data binding.

Data Extract Elements

Manager provides the set of elements to access underlying data model, such as form data extracts, which you can use in forms and collaboration jobs to get elements runtime values.

> NOTE
>
> We use of camel case notation when referring to the data extract references, for example, `$formDataMap.firstName` for the First Name.

The main data extract elements are listed below.

$formDataMap

This element holds all the form data extracts, so you can use `$formDataMap.firstName` to access the first name of a user entered into a form. A collaboration job processing automatically adds a Route Name, so it's available in the data extracts as `$formDataMap.routeName`. Where a submission is not defined for an action service, it uses the data extracts from the previous or the start submission. A typical usage is shown below.

```
{ "name": "Task Subject", "value": "Initial review of ${job.referenceNumber} from
${formDataMap.firstName} ${formDataMap.lastName}" }
{ "name": "Task Subject", "value": "Job Number: $job.referenceNumber $sub-
mission.form.formName From $formDataMap.firstName $formDataMap.lastName" }
{ "name": "Task Subject", "value": "Job Number: $job.referenceNumber $sub-
mission.form.formName From $formDataMap.firstName $formDataMap.lastName" }
{ "name": "Task Subject", "value": "Job Number: $job.referenceNumber From $formDataMap.e-
mail" }
```

$job

This element holds the job entity instance where:

- `${job.name}` is the job name
- `${job.referenceNumber}` is the job number
- $`{job.submission.formName}` gets the form name of the first submission.

A typical usage is shown below.

```
{ "name": "Process Message Text", "value": "Your ${job.submission.formName} application
is at the Initial Review Step" },
{ "name": "Task Message", "value": "Review of the ${job.referenceNumber} from
${formDataMap.firstName} ${formDataMap.lastName}." }
```

$jobAction

This element holds the current collaboration job action entity instance. You can get the current step's name by using `$jobAction.step.name`., as shown below.

```
{ "name": "Task Subject", "value": "${jobAction.name} of ${job.referenceNumber} from
${formDataMap.firstName} ${formDataMap.lastName}" }
```

$func

This element provides various functions for there is a list JobFunctions in the API .

- `$func.formProperty('Initial Review')` retrieves the form's or organization's property value named Initial Review.
- `$func.invoke('Groovy Service Name', arg1, arg2)` invokes a Groovy[1] service.

A typical usage is shown below.

```
{ "name": "Task Form Code", "value": "$func.startFormCode()" }
{ "name": "Task Assign Group", "value": "$func.formProperty('Initial Review')" }
```

Next, learn about the submission data extract.

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

Submission Data Extract Overview

Manager comes with the concept of submission's data extracts that allow a form builder to configure certain Maestro components so their user input can be extracted from the XML[1] payload and then saved in an easy to read format, such as an Excel spreadsheet. This data extract can then be used for integration with external systems or distributed via emails for further processing. The Maestro editor comes with the Integration pane so you can easily configure component's data extracts and integration with Manager.

Submission data extracts usually represent only a subset of the form data, not all of it. To select which components are included in the data extract, you need to configure them either in Maestro or Manager.

Submission data extracts can also be used in:

- Collaboration jobs. You can pass information from a form to a collaboration job, for example, to customize email messages by taking the user's first name and adding it to the email's greeting.
- Overlay receipts. Forms can be implemented as pixel-perfect receipts by using data extract values in AcroForms[2].
- Card. A card is a repeating element in the form space, which can be customized from data in the form using data extracts.
- Form spaces' searches. You can use data extracts when searching form spaces. For example, you can search for a task that has been assigned to you by using the data extracts and not the entire XML file.

Form data extracts are a great way to quickly inspect key data elements in Manager and allow you to make business decisions or meet reporting requirements.

There are some best practices when using form data extracts:

---

[1]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
[2]AcroForms are the original PDF forms technology that accepts input in both Forms Data Format (FDF) and XML Forms Data Format (XFDF). Many 3rd party vendors support AcroForms.

- Do not extract Personally Identifiable Information[1], because it might be downloaded as a bulk CSV[2] file using Manager. Using data extracts in this way is not the intended design, rather it is used for reporting purposes for generalizing on a particular set of data, or testing. It is best to keep any PII out of the data extracts, and rather send this data to your secure back-end systems for processing.
- Do not extract every single piece of data in your form using data extracts. There are many other ways of integrating data into systems and sending your form data to your processing areas. Extracting all the data in your form using submission data extracts adds a level of development and ongoing maintenance overhead, which is not necessary given the many other options to retrieve your data.

Next, learn how to view submission data extracts.

Last updated on : 09 September 2019

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

[2]CSV is a comma-separated values fil, which s a delimited text file that uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

View Form Submission Data Extract

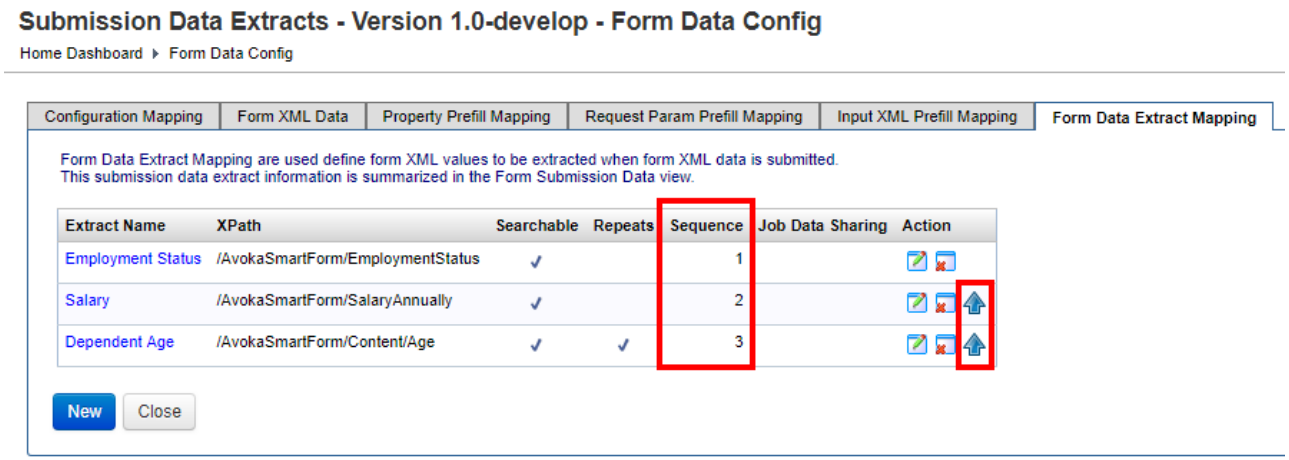
Manager allows you to view data extracts of any submitted form.

To view submission data extracts:

1. Go to the Home Dashboard.
2. Locate a form transaction from the Latest Transactions section and click ID.
3. Select the Form Data Extract tab. For more information, see view form data extract.

To configure and rearrange data extract mapping:

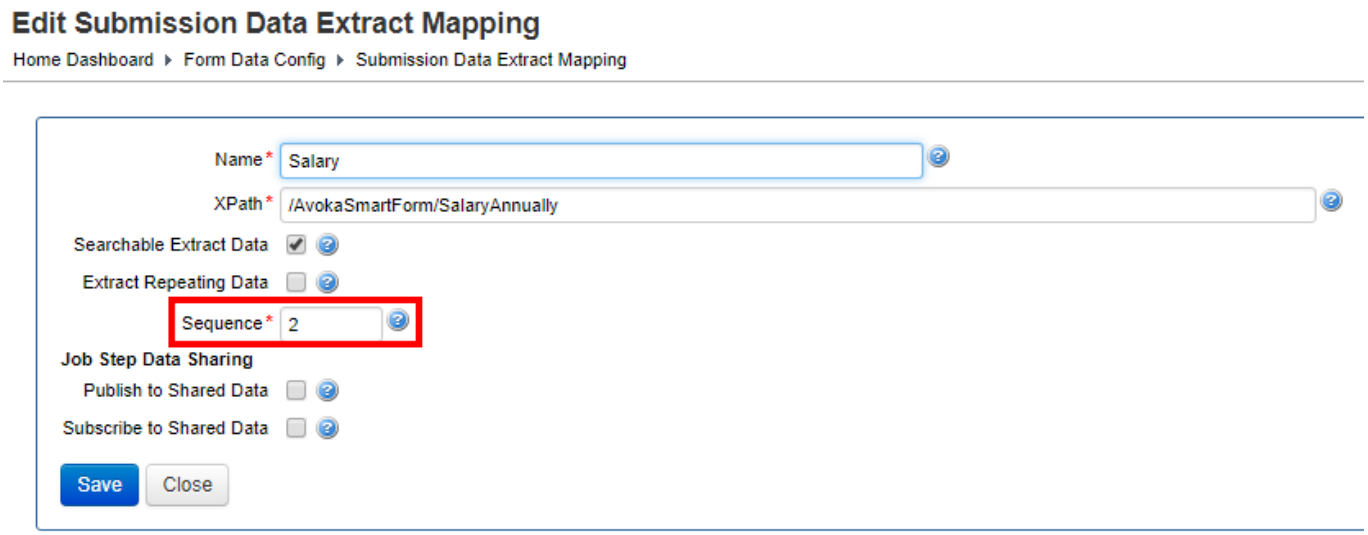
1. From Form Versions section, select a form version and then select Data Config.
2. Select the Form Data Extract Mapping tab.



3. To change the order that the extracted submission data appear in the table in Form Data Extract tab, use the arrows to move data items, one at a time, up in the sequence. This is the easiest way to reorder the data extracts.

The sequence can also be changed in each submission data extract mapping. Select Extract Name, update Sequence (but must be unique) and click Save.

This order only applies to the data extracted and does not rearrange the component order on the Maestro form as displayed to the user.



Next, learn about form prefill options.

Form Data Overview

Manager uses XML[1] format to store and represent form's data, such as user's input and prefilled information, for example, a current location. Manager automatically generates a default XML data structure to handle the data passed between the server and a form.

Each Maestro form that you deploy into Manager contains a `formseed.xml` archive file describing all relevant form data in an XML format. If you have specific data schema requirements, you can edit the form's XML data structure or upload a form seed file to update the form XML data. When you export a form from Manager, the archive file also contains a `formseed.xml` file so you can check and modify the XML data structure as required.

The form XML data structure varies depending on a form content, but it usually includes the following elements:

- `AvokaSmartForm` - the root element that incloses all other tags. The example is shown below:

  ```
  <AvokaSmartForm>
  ...
  </AvokaSmartForm>
  ```

  NOTE

  If a form is configured to uses Unified App Data, then `<Root>` is the root element of the form XML data.

- `SFMData` - Smart Form Data that contains transaction data, such as `formCode`, `TemplateVersionNumber`, `ReceiptNumber`, `PaymentDetails`, and so on. The example is shown below:

[1]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

```
<SFMData>
    <SystemProfile>
      <DisplayMode/>
      ...
    </SystemProfile>
    <PaymentDetails>
      <Payment>
        <SubTotal/>
        <GST/>
        <ServiceFee/>
        <Total/>
      </Payment>
      <PaymentType/>
      ...
 </PaymentDetails>
</SFMData>
```

- `SystemProfile` - system specific properties of the form. The example is shown below:

```
<SystemProfile>
        <FormCode>test</FormCode>
        <TemplateVersionNumber>1.0</TemplateVersionNumber>
        <UpdateUserProfile>false</UpdateUserProfile>
        <HostContext>Page</HostContext>
        <SubmissionType>Submitted</SubmissionType>
        <OnlineSaveEnabledFlag>true</OnlineSaveEnabledFlag>
        <Referer><![CDATA[...]]></Referer>
        <RequestLogKey>...</RequestLogKey>
        <TestMode>false</TestMode>
        <SubmissionNumber>
        </SubmissionNumber>
        <DisplayMode>Entry</DisplayMode>
        <ShareForm></ShareForm>
        <ComposerDataVersion>21.05.4</ComposerDataVersion>
        <ServerBuildNumber>22.4.0</ServerBuildNumber>
        <RevisionNumber>0</RevisionNumber>
        <FormDataServiceURL>...</FormDataServiceURL>
        <OfflineSubmissionId>..</OfflineSubmissionId>
        <TrackingCode>...</TrackingCode>
        <AbandonmentReason>...</AbandonmentReason>
        <AbandonmentComments>...</AbandonmentComments>
</SystemProfile>
```

NOTE

If you build a form using the Use server side Transact Functions (requires TM 17.10 +) option, a number of `SystemProfile` properties, such as `<TestMode>...</TestMode>`, are removed from the form and from the submission XML data.

So, in the above example, the reduced set of form's properties will be like this:

```
<SystemProfile>
        <FormCode>test</FormCode>
        <RequestLogKey>...</RequestLogKey>
        <DisplayMode>Entry</DisplayMode>
        <ServerBuildNumber>22.4.0</ServerBuildNumber>
        <RevisionNumber>0</RevisionNumber>
        <FormDataServiceURL>...</FormDataServiceURL>
        <TrackingCode>...</TrackingCode>
        <AbandonmentReason>...</AbandonmentReason>
        <AbandonmentComments>...</AbandonmentComments>
</SystemProfile>
```

- Form's specific elements, such as `FirstName`, `LastName`, `EmailAddress`, and so on. The example is shown below:

```
<FirstName/>
<LastName/>
<DateOfBirthDropdown>
    <Day/>
    <Month/>
    <Year/>
    <Date/>
</DateOfBirthDropdown>
<EmailAddress/><AddressAustralia>
    <AddressLine1/>
    <AddressLine2/>
    <City/>
    <State/>
    <PostCode/>
</AddressAustralia>
```

NOTE

You can download the complete example of form XML data here.

When a user selects a form, Manager starts to controls the data flow that can be one of the following:

- Standard default form processing
- Customized form processing. This is the most common method to process a form.

Form data configuration addresses one of a common requirements - prefill of a form with user's details.

Before proceeding with data configuration in Manager, there are several data settings that should be configured in Maestro. Once the Maestro form version is built and published, these data configuration is transferred to Manager.

The list below identifies details on how to customize forms in Manager for the following data requirements:

- Prefill a form with user details using one of the supported prefill methods:
  - Property Prefill mapping
  - Request Parameter Prefill mapping
  - Input XML Prefill mapping
- Capture and store additional details that are passed along with the form submission that may be required for handling attachments, receipts, and anonymous submissions. For more information, see configuration mapping.
- Provide an alternative data structure, according to your enterprise's requirements, to the default XML. For more information, see form XML data.
- Extract some details for easy access for monitoring and managing form submissions. Data extracts are also required for some collaboration job features.

Form Builders customize these data mappings using Manager or Maestro UI, so that individual data elements can be externally defined and mapped to your form's XML data structure when the form is rendered.

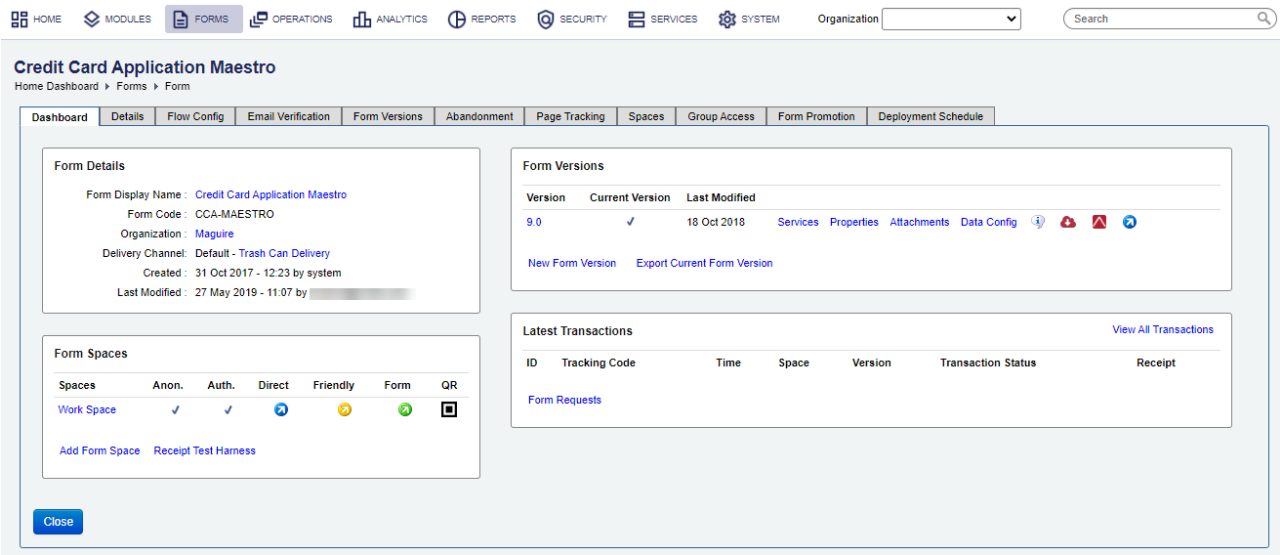
Next, learn how to view form data configuration.

View Form Data Configuration

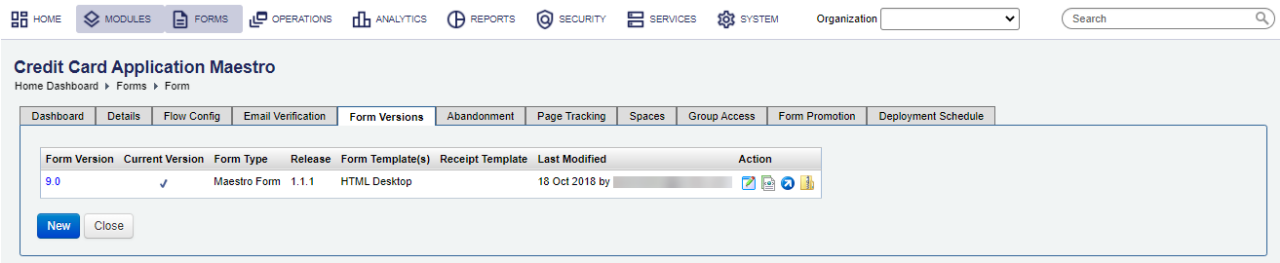
Manager allows you to view data configuration for each form version.

To view data configuration of a form version:

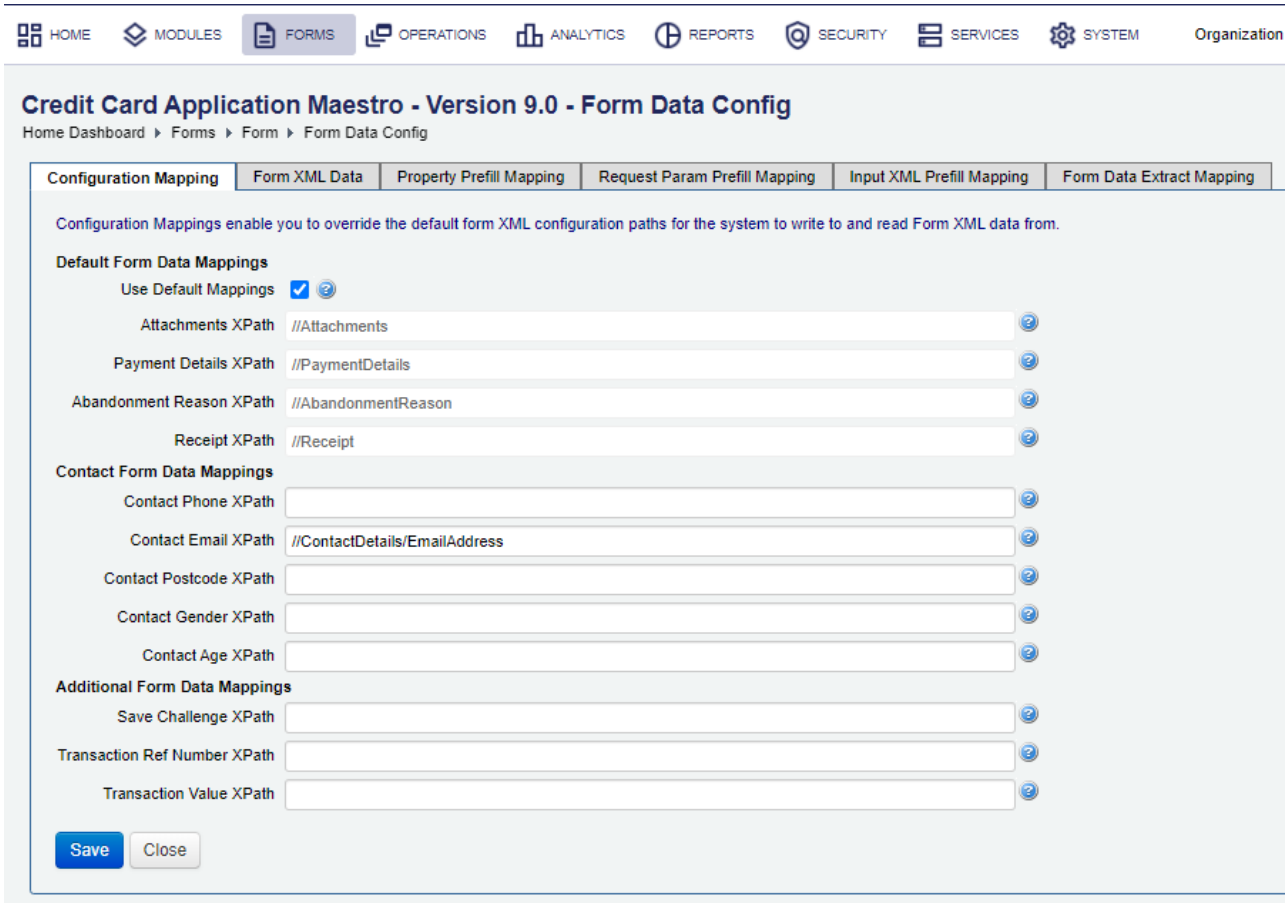
1. Select Forms > Forms.
2. Locate a form and click Edit or a form name.
3. Click Data Config in the Form Versions section.



Alternatively, select the Form Versions tab and click Form Data Configuration .



4. Check and update form data configuration.



Next, learn about form configuration mapping.

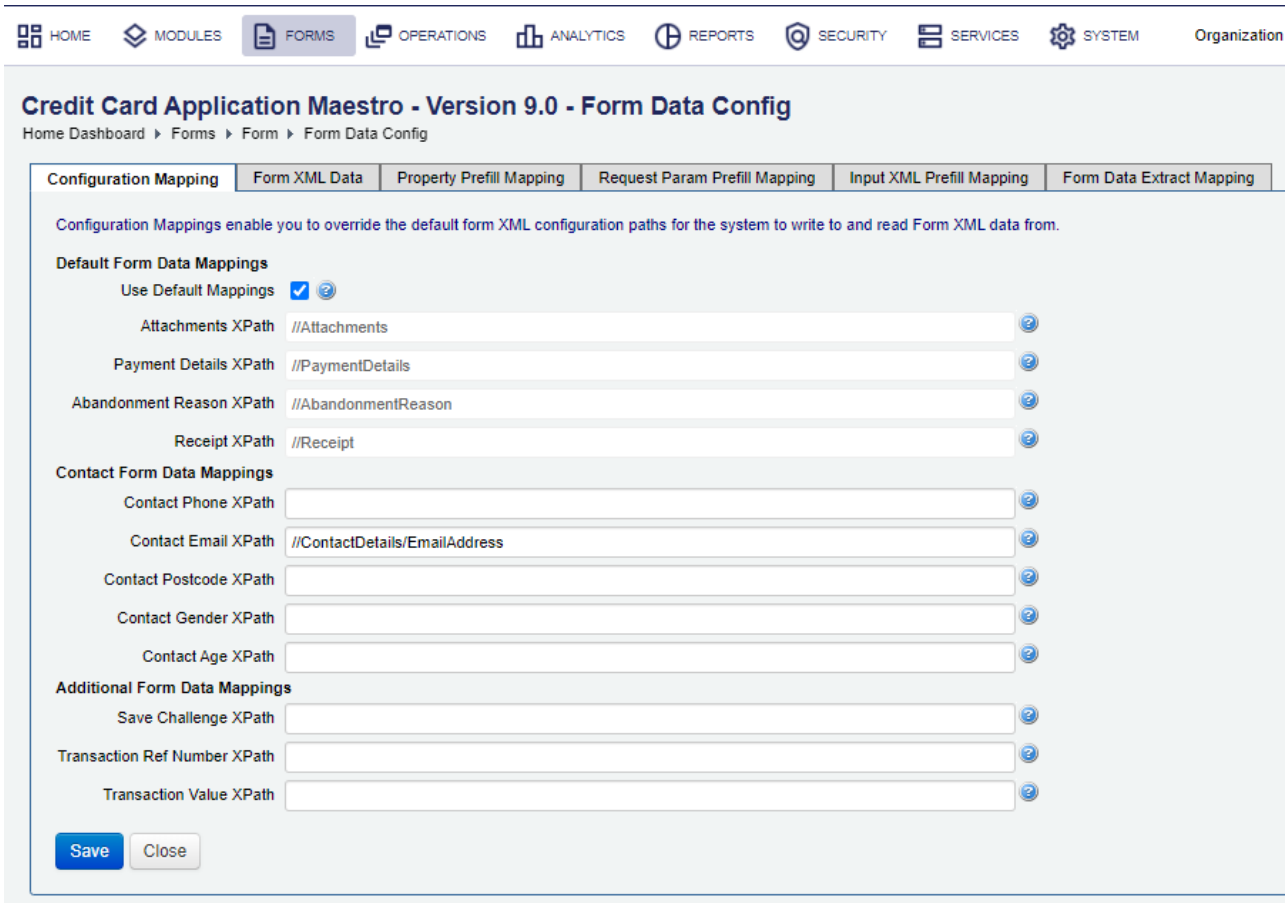
Form Configuration Mapping

Manager allows you to override the default form XML configuration paths for the system to write to and read from XML data form. This, in turn, let you pass some key data into and out of a user's form session. You can modify the following form's operations:

- Attachments
- Payments
- Signatures
- Email verifications
- Form saving and retrieval
- Anonymous submissions

To configure form version's configuration mapping:

1. Select Forms > Forms.
2. Select a form and then click Edit.
3. Select a form version and click Edit Form Data Config.
4. Click the Configuration Mapping tab.



5. Select the Default Mappings checkbox to use default Configuration Mappings for attachments, payment details, receipts, and signatures. Clear this checkbox to change the defaults for the following five XPath[1] locations.

> NOTE
>
> If Use Default Mappings is not selected, you can change the default XPath in the Form XML Data.

6. Select the Use Default Mappings checkbox to uses default Configuration Mappings for Attachments, Payment Details, Receipts, and Signatures. Otherwise, clear the Use Default Mappings checkbox to configure the following XPath locations manually.
7. Attachments XPath - edit the default XPath of the `Attachments` element in the Form XML Data.
8. Payment Details XPath - edit the default XPath of the `PaymentDetails` element in the Form XML Data.
9. Abandonment Reason XPath - edit the default XPath of the `AbandonmentReason` element in the Form XML Data.
10. Receipt XPath - edit the default XPath of the `Receipt` element in the Form XML Data.
11. Contact Phone XPath - The XPath that points to the element in the Form XML Data where the user's contact telephone number is stored. This may also be defined in Maestro.
12. Contact Email XPath - The XPath that points to the element in the Form XML Data where the user's contact email address is stored. This may also be defined in Maestro.
13. Contact Postcode XPath - The XPath that points to the element in the Form XML Data where the user's contact postcode is stored.
14. Contact Gender XPath - The XPath that points to the element in the Form XML Data where the user's contact gender is stored.
15. Contact Age XPath - The XPath that points to the element in the Form XML Data where the user's contact age is stored.
16. Enter an XPath to an element containing the answer to the challenge question users will need to answer to return to an anonymously saved form in the Save Challenge XPath field. The element in question must not be blank when the form is saved.

---

[1]XPath (XML Path Language) is a query language for selecting nodes from an XML document. In addition, XPath may be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document. XPath was defined by the World Wide Web Consortium (W3C).

NOTE

You can also configure a save challenge for a form in Maestro.

17. Transaction Ref Number XPath - The XPath in the Form XML Data that points to the element that contains the transaction reference number which will be stored with the submission. This is provided for backward compatibility.
18. Transaction Value XPath - The XPath in the Form XML Data pointing to an element that contains the numeric value of this transaction.
19. Old Wet Signatures Required XPath - The XPath in the Form XML Data pointing to an element that contains true if the form requires wet signatures.
20. Click Save to update the changes.

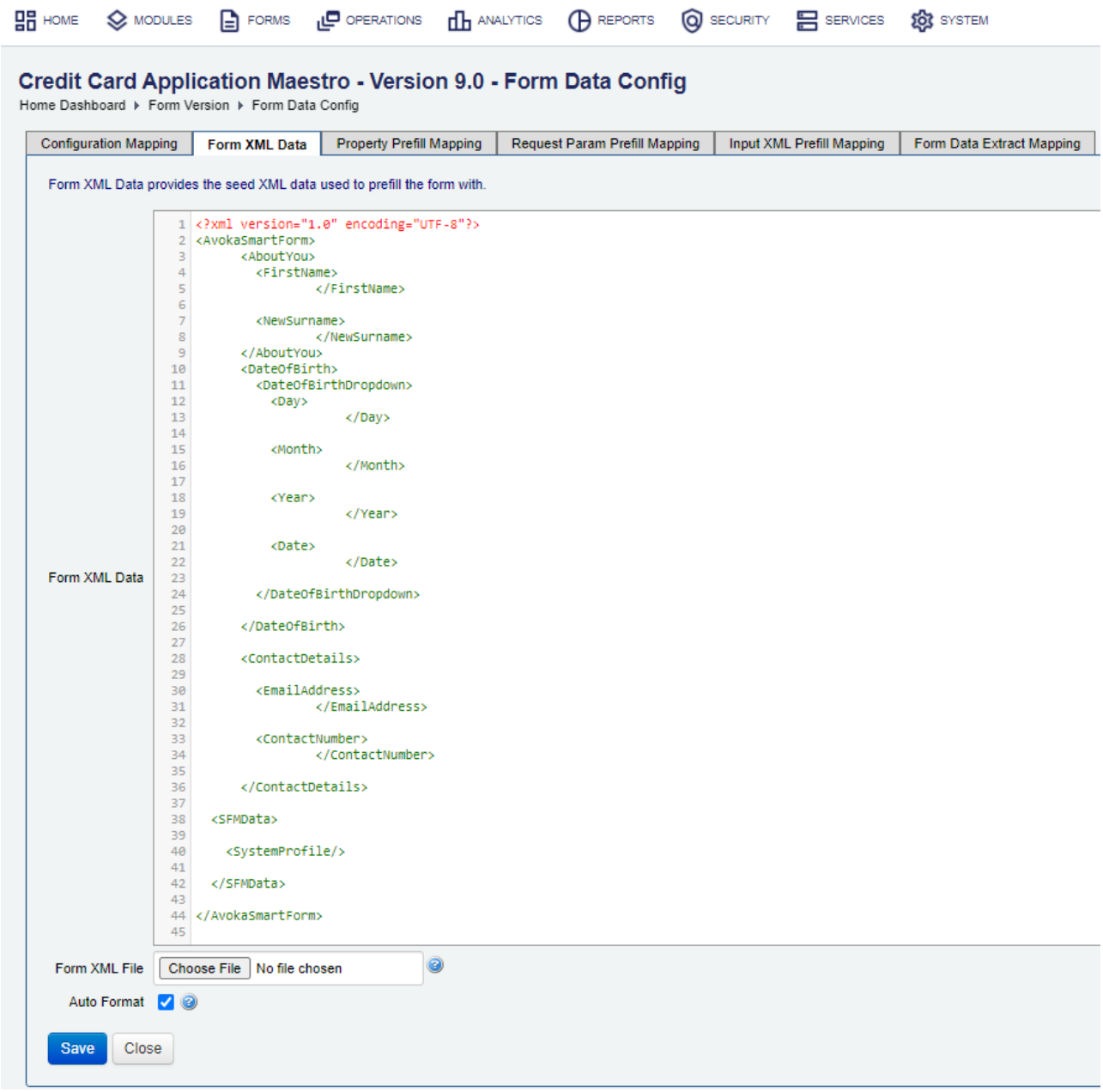
Next, learn how to view form XML data.

View Form XML Data

Manager allows you to view and edit form's data as form XML data, which represents all data pertinent to a form itself as well as user's input provided into the form.

To view or edit form XML data:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select a form version and click Edit Form Data Config.
4. Click the Form XML Data tab to view the XML data.



5. Edit the XML content directly in the Form XML Data tab or click Choose File to upload a seed XML file.
6. Select the Auto Format checkbox to automatically format the XML data. Otherwise, it retains the original XML format when saving it.
7. Click Save. Manager will validated the updated XML file for any syntax errors before saving it.

View Form XML Data in a Browser

When you test or troubleshoot a form, you may want to view and even modify form XML data sent from a client's web browser to Manager during POST requests. You can easily do this using browser's tools, for example, Google Chrome's Developer tools.

To view form XML data:

1. Locate a form you need to test or troubleshoot.
2. Open a relevant form version of this form.
3. Clear the Form Data Encryption checkbox to disable HTML client-side form XML data obfuscation.

   WARNING

   You should run such tests in the test environments only, such as SIT and UAT. Never run them in the production environments.

4. Click Form Test Render to open the form version in a new web browser tab.
5. Open the browser's Developer tools where the form version was rendered and select Elements (if you are using Google Chrome).
6. Expand the `<body>` element of the form and look for `script[@id='sfc-prepop']`, which is the XML content of the form XML data that is embedded within the HTML of the initial form load.
7. Expand `<script id="sfc-prepop">` to view all elements of the form XML data.
8. Perform your tests.
9. Select the Form Data Encryption checkbox to enable HTML client-side form XML data obfuscation.
10. Verify that you can see form XML data as obfuscated data.

Next, learn about form property prefill mapping.

Form Seed

Manager generates a form seed XML file to help you create and edit form XML data files for your form applications. We don't use XML Schema, RelaxNG or DTD for this purpose, because they are rather difficult to read and interpret by humans and by computers

The XML seed file introduces some simple “annotations” that allow you to create rather complex XML data structures. The seed file advantages are listed below:

- There are actually very few constraints we require, such as, max length, data type, regular expression, max-occurs, enumeration
- This isn’t industry standard, but it’s simple
- You can write a simple processor to process XSD and spit out the annotated seed file
- It gets around the complexities of multiple file imports and namespace issues
- We support two styles, storing the restrictions, in either attributes or data

Here’s the simple seed file with data

```
<root>
  <first_name>John</first_name>
  <last_name>Citizen</last_name>
  <age>61</age>
  <tfn>123456</tfn>
  <children>
    <child>
      <name>Jack</name>
      <birth>2000-01-01</birth>
    </child>
  </children>
</root>
```

Here’s an augmented version:

```
<root>
  <first_name>maxLength="20" type="xs:string"<first_name>  <!-- restrictions in the data
-->
<last_name maxLength="20" type="xs:string"></last_name> <!-- restrictions in the attrib-
utes -->
<age type="xs:integer" minInclusive="18"></age>
<tfn>pattern= "[a-zA-Z0-9]{8}"</tfn>
<children>
  <child max_occurs="5">
    <name>maxLength="20"</name>
    <birth>type="xs:date"</birth>
  </child>
</children>
</root>
```

Next, learn about form property prefill mapping.

Form Data Extract Mappings

Journey Manager supports a data extract feature to extract form data and store it in the external data elements against form submissions. Whenever a user submits a form with data extracts, this data is stored against each form's submission as a convenient way of viewing and accessing essential data from within Manager. Data extracts may assist in transaction monitoring and reporting, especially for operational support teams, and are required for supporting collaboration jobs.

Manager allows you to define data extracts mappings for form versions and automatically populates them.

> NOTE
>
> We recommend to configure data extracts in Journey Maestro instead of Manager before proceeding. Do not configure data extracts in both Journey Maestro and Manager.

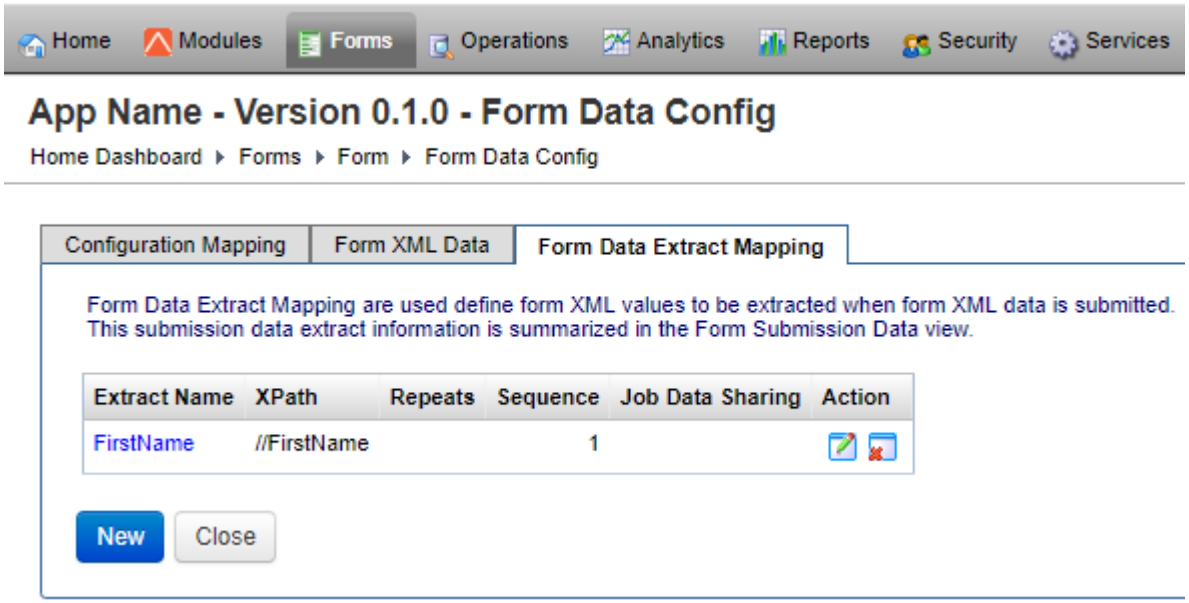
A data extract mapping specifies where to find the data to extract it from the form's XML[1], using an XPath[2], and how to name the extracted value, which will be stored in Manager.

To configure data extract mappings:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select a form version and click Edit Form Data Config.
4. Click the Form Data Extract Mapping tab to see the existing mappings.

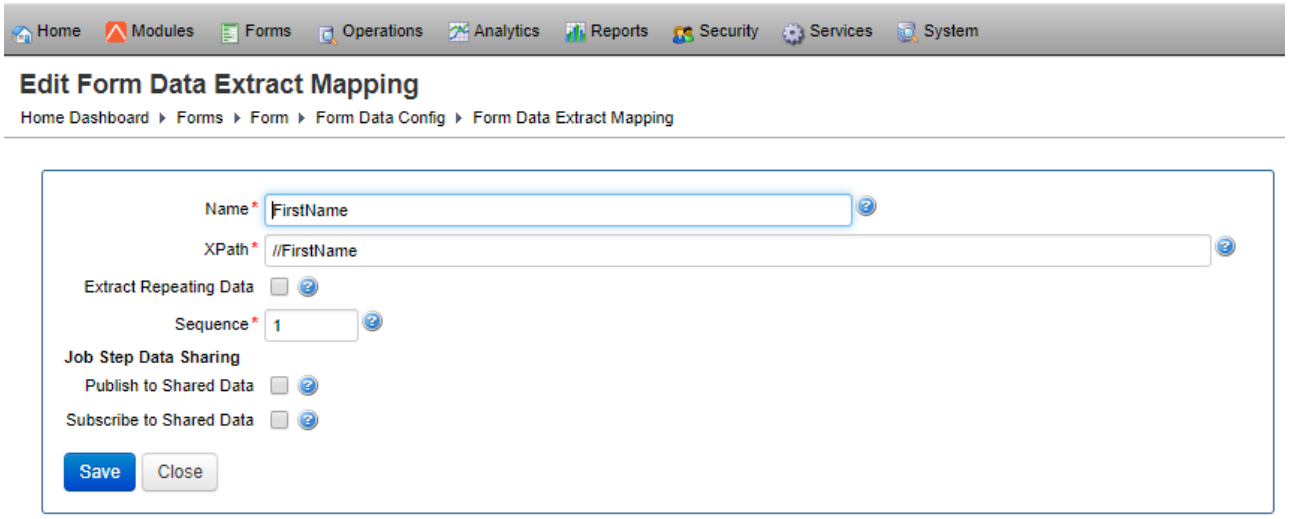
---

[1]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
[2]XPath (XML Path Language) is a query language for selecting nodes from an XML document. In addition, XPath may be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document. XPath was defined by the World Wide Web Consortium (W3C).



Here, the extract field name is FirstName, so the extract reference used in the action property value will be `$formDataMap.firstName`, and the action property used in the job definition's will be `{ "name": "Task Subject", "value": "Job Number: <b>$job.referenceNumber</b> $submission.form.formName From $formDataMap.firstName $formDataMap.lastName" }`

5. Click New to create a new data extract mapping or click Edit to update the existing one.



6. Edit the label for the extracted, displayed or downloaded data item in the Name field.
7. Edit the XPath location in the `formXml` where the data item can be found in the XPath field. This is used in the Publish / Subscribe to update the `formXml`.
8. Select the Searchable Extracted Data checkbox to enable searching on encrypted data. For searching, the user must enter the full string, which is not case sensitive. | 18.11 This feature was removed in 18.11

NOTE

Manager 18.11+ is able to search encrypted PII data, so you don't need to mark form data manually whether it is as searchable or not.

9. Select the Extract Repeating Data checkbox to automatically extract all repeating XML elements in the form submission data extracts. This is only relevant for a data item extracted from a repeating component.
10. Edit the Sequence, which is the order that the extracted data item will display against the form submission. With the first data extract, it automatically starts at 1 and increments with each new data extract. This may be adjusted but must be a unique value. It is better to reorder using the Reorder icon from the FormData Extract Mapping tab.
11. Select the Publish to Shared Data checkbox to publish shared collaboration job step extract data to other step submissions. This is used by collaboration jobs in conjunction with the Subscribe to Shared Data, which is described below. It is configured in the Job Definition JSON in the Step by setting the `shareExtractData` attribute to true. Publish Subscribe is an alternative mechanism to share data between tasks in a form bundle's collaboration job step, which is made up of two or more separate tasks (submissions). When one task is submitted, Manager processes all the data extracts marked as Publish and puts the values in a map. It then goes through all the other form tasks in the step. Where the data extract has a value that subscribes, the data extract value will be overwritten with the published value. The `formXml` on the subscribing form will also be updated.

    NOTE

    This is relevant for form bundles.

12. Select the Subscribe to Shared Data checkbox to subscribe to shared collaboration job step extract data changes. This is used by collaboration jobs in conjunction with the Publish to Shared Data configuration, which is described above. The subscribe consumes the corresponding Publish event using the published value. To use Subscribe, the XPath location needs to be set to a single node. You won't be able to use the double slash `//FirstName` to set the value using XPath. Data extracts specified when designing your form don't have the full XPath location.

> NOTE
>
> This is relevant for form bundles.

13. Select the Encrypt Extracted Data checkbox to automatically encrypt the extracted data. This means that the value is unencrypted when accessed via the Manager console, rest API, as well as from entities within Groovy scripts such as delivery services. | 17.10 This feature was removed in 17.10
14. Click Save to update the changes.

You can also perform the following actions:

- Delete a data extract mapping.
- Reorder the sequence by moving the mapping up one position.

Collaboration Job Contex

Manager passes a collaboration job's context to a form using the `SFMData.SystemProfile.Job` XML node. This is done during the form rendering. The snippet below highlights the `<Job>` XML node:

```
<?xml version="1.0" encoding="UTF-8"?>
<AvokaSmartForm>
    <SFMData>
        <SystemProfile>
            <DisplayMode>Receipt</DisplayMode>
            <HostContext>Page</HostContext>
            <ReceiptNumber>fct-3097-composer-3</ReceiptNumber>
            <SubmitDateString>12 Feb 2015 5:38:54 PM</SubmitDateString>
            <SubmissionMessage/>
            <FormDataSer-
viceURL>http://localhost:9080/maguire/servlet/FormDynamicDataServlet</FormDataServiceURL>
            <RequestLogKey>9c1b9756b3958671c9b2471e23f91d08</RequestLogKey>
            <TrackingCode>NVKMQ9</TrackingCode>
            <Job>
                <AvailableRoutes/>
                <Assignee/>
                <AssignRepeatIndex/>
                <AssignRepeatItem/>
                <ReferenceNumber/>
                <StepName/>
                <RouteName/>
            </Job>
            <FormCode>fct-3097-composer</FormCode>
....
```

The `<Job>` node has the following attributes:

AvailableRoutes

A collaboration job passes the `AvailableRoutes` property to the tasks created at a particular collaboration job step. The job definition below shows the 2 Step Review Job.

```
{
  "name": "Initial Review",
  "type": "",
  "actions": [
     {
       "name": "Create Task",
       "type": "Job Task Assign",
       "properties": [
         ...
       ]
     },
     {
       "name": "Handle Submission",
       "type": "Job Task Wait",
     }
  ],
  "routes": [
     { "name": "Approve",  "nextStep": "Application Delivery" },
     { "name": "Decline",  "nextStep": "Applicant Update" },
     { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display": "false"
},
     { "name": "Terminate",  "nextStep": "Terminated Initial" }
  ]
}
```

When the collaboration job moves to the Initial Review step, it populates the form data with `<AvailableRoutes>Approve|Decline|Terminate</AvailableRoutes>`. The `Exceeds Threshold` route is not included due to the `display` item. This is used in a form as the read-only value.

ReferenceNumber

A `ReferenceNumber` is a Job Number field, which is displayed on forms with the Step Number as the heading. This is used in a form as the read-only value. For more information, see Composer documentation.

StepName

A `StepName` is a name of the current step that the collaboration job is currently at. The form can use this value to hide, show or set editability of fields and form sections. This is used in a form as the read-only value.

RouteName

A form user can set a new route in the form using a dropdown list or a radio button. If this is not set, the collaboration job uses the default route. This attribute is set by a form.

Next, learn how to test a new form data extract mapping.

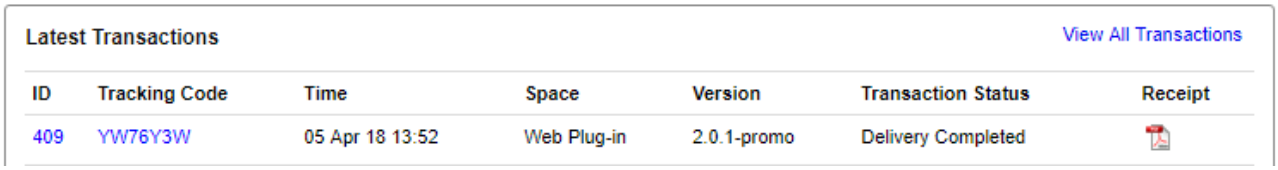
Last updated on : 18 October 2019

Test a Form Data Extract Mapping

Manager allows you to test a form data extract mapping.

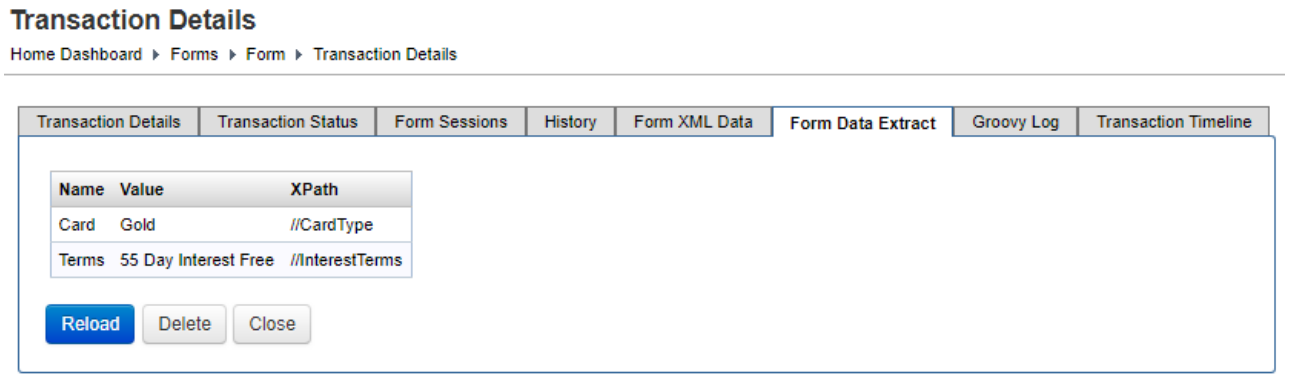
To test a data extract:

1. Render a form version.
2. Complete and submit the form
3. Select a credit card type and interest terms.
4. Locate the data extracts for this form submission.
5. Navigate to the Latest Transactions section and click ID.



> NOTE
>
> You can view the same information from Operations > Form submission Data.

6. Click the Form Data Extract tab to view available form data extracts. In this example, a user of this form submission entered Gold for their Card and 55 Day Interest-Free for their Terms.



> NOTE
>
> In Manager prior to v17.10, you can click the Form Submission Data icon from a form Dashboard to view form submissions. This functionality has been removed.

Form Prefill Options

Manager offers several options to prefill a form with some data, including applicant data, when the form is rendered to the applicant. Form prefill is a great way to significantly improve user experience and reduce data entry errors. You need to configure or map form's data with prefill data, which will affect a form's XML at render time. The form prefill is a flexible tool that allows you to implement various customer requirements. You can even mask some certain fields, such as a date of birth, phone number or address, on a form resume to improve application security.

Manager distinguishes the following mappings:

- Property prefill
- Input XML prefill
- Request parameter prefill

Let's look at each mapping in details:

Property Prefill Mapping

You can use properties defined for an organizations, form versions, or user profiles using property scopes. The scope can be one of the following:

- Client - a property configured for an organization and its forms, so it obtains its value dynamically from an organization.
- Form - a property configured for form versions, so it obtains its value directly or dynamically from a form version. This property can override Client's scope property.
- User - a property configured for user profiles, so it obtains the value from a user's profile of a logged user.

This allows you to reuse data that is already stored in Manager:

- User data
- Organizational data - reuse across multiple forms
- Configure form specific properties without needing to customize or redeploy the form
- Can be configured to sync or update the user properties maintained in Manager

For more information, see form property prefill mapping.

Input XML Prefill Mapping

You can use an external XML file containing data, which is obtained at the time a form is rendered, that will be used to prefill a form. This provides the following benefits:

- Use 3rd-party and external data.
- Create mappings for a particular XML schema.

However, this method requires a prefill service to fetch data. For more information, see form input XML prefill mapping.

Request Parameter Prefill Mapping

You can use data passed in as HTTP request parameters, session attributes or cookies. This has the following advantages and disadvantages:

- Pros: Simplicity - data can be provided as part of an URL.
- Cons: Difficulty - create and map complex data structures can be daunting.

For more information, see form request parameter prefill mapping.

> NOTE
>
> You can also configure extracts of submission data using form data configuration.

Before a form is rendered and presented to form users, the various prefill methods are applied in the following order:

1. Form XML data
2. Property prefill - client forms
3. Request parameters
4. Property - user profiles

As a form may use multiple prefill methods, it is important to understand the way this is handled in Manager. Where multiple mappings to the same form field exist, prefill data obtained at later stages overrides earlier obtained prefill data.

> info
> Often, prefill is not carried out when someone subsequently opens a saved form. For more information, see Groovy Service API > Form Prefill.

Next, learn about form property prefill mapping.

Form Input XML Prefill Mapping

Manager allows you to prefill a form by passing an XML[1] prefill document with form details. A form should have XML input mappings configured, so you can use this functionality in several ways:

- When XML input mappings are configured, the XML is returned from a Groovy Prefill Service.
- Assign tasks in Manager, when the Form Data field is set to Input Xml Data, so the mappings are used to map data into the form seed.
- Create anonymous saved forms using the `SubmissionTaskService.createAnonymousSavedForm` and `SubmissionTaskService.createFormTask`, when the `inputXmlData` parameter is used to pass in the input XML object.

As an example, the XML prefill form passes all the details for the first page of the Credit Card Application form. This is handled by the form version prefill service which assembles the prefill details as an XML document.

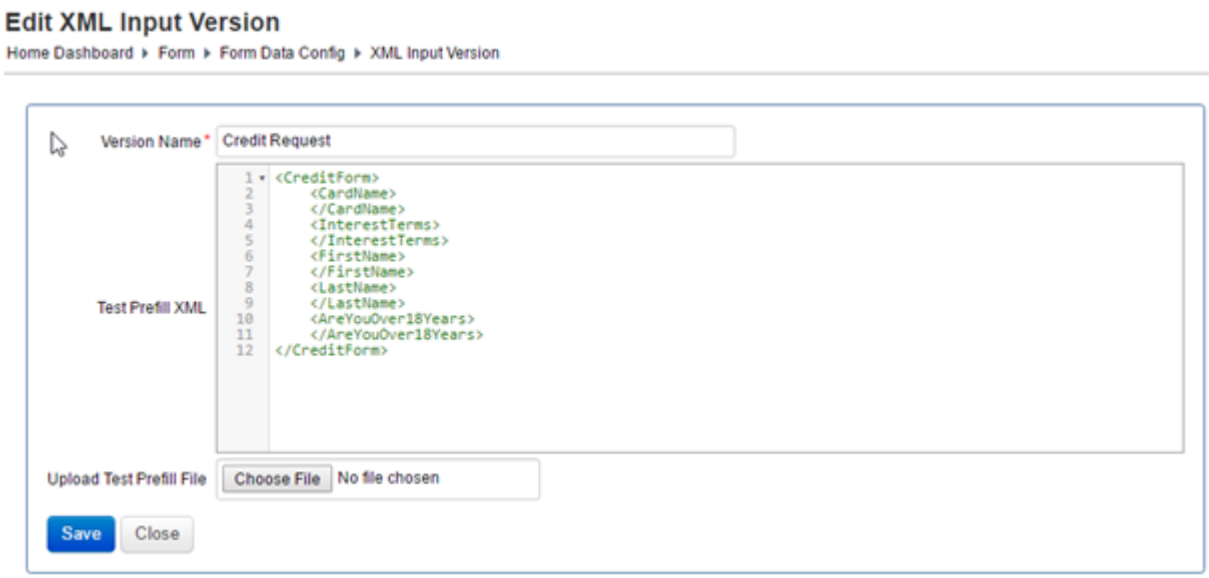
> NOTE
>
> If you don't map properties, they won't be in a XML document that Manager sends to a form for the properties prefill.

To configure a form input XML prefill mapping:

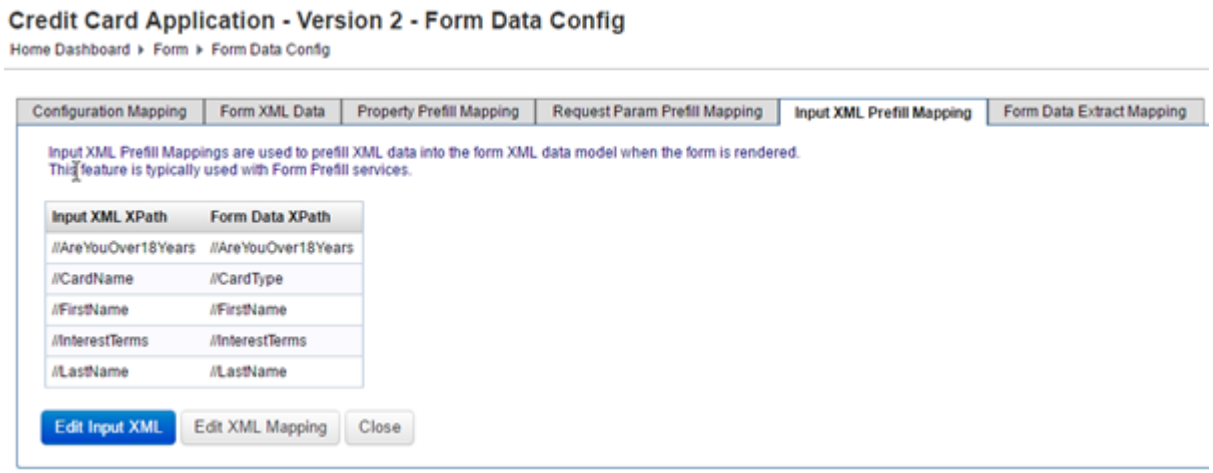
1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select a form version and click Services to configure the form version to invoke the prefill service.
4. Select a form prefill service from Form Prefill Data Service dropdown.
5. Click Save to update the changes.
6. Click the Form Data Config tab to configure the mapped fields.
7. Select the Input XML Prefill Mapping tab.
8. Click Edit Input XML.

---

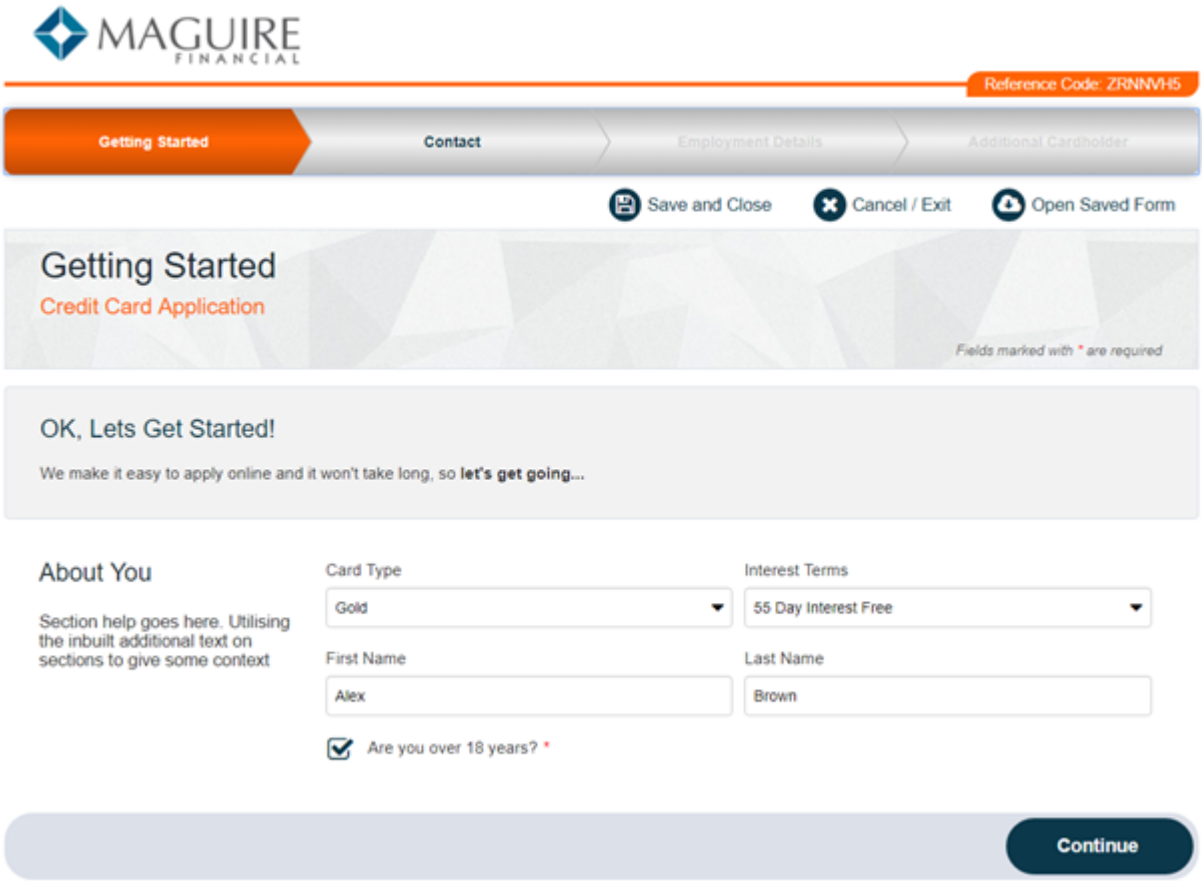
[1]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.



9. Enter a name into Version Name.
10. Enter an XML form structure of the external fields that can be used for data prefilling. It must match the XML form that is generated by your Form Prefill Data Service. Alternatively, you can upload an XML file by clicking on Choose File in Upload Test Prefill.
11. Click Save to update the changes.
12. Click Edit XML Mapping to map each prefill data element from the source to the target. You have the following options to map elements depending on your Manager version:
    - Use drag and drop.
    - Use XPath configuration. | 18.11 This feature was introduced in 18.11.
13. Click Save to update the changes.
14. Check the mapped data as shown below



15. Check the rendered form that is prefilled with data provided by the Form Data Prefill Service.



NOTE

It is also possible to use the prefill data service without mapping your fields. In this case, the exact same XML structure as the form XML must be returned from the prefill data service.

Next, learn about form data extract mapping.

Form Input XML Prefill Mapping Using Drag and Drop

NOTE

As Adobe Flash is not supported after the end of 2020, this feature is not available in Journey Manager since version 19.11.
You should use Form Input XML Prefill Mapping Using XPath instead.

Manager allows you to map each prefill data element from the source to the target using drag and drop functionality. Manager uses Adobe Flash[1] to implement drag and drop.

To map prefill data elements:

1. Click New Input XML to create a new mapping or click Edit XML Mapping to change the existing one.
2. Review a source and a target of the property mappings in the table containing the following columns:
   - Source: Prefill Data XML.
   - Prefill Data XPath.
   - Form Data XPath.
   - Target: Form Data XML.
3. Drag and drop each data element from the Source: Prefill Data XML section to the Prefill Data XPath section.
4. Drag and drop each data element from the Target: Form Data XML to the Form Data XPath.

[1]Adobe Flash is a deprecated multimedia software platform used for production of animations, rich Internet applications, desktop applications, mobile applications, mobile games and embedded web browser video players. Flash displays text, vector graphics and raster graphics to provide animations, video games and applications. It allows streaming of audio and video, and can capture mouse, keyboard, microphone and camera input. Related development platform Adobe AIR continues to be supported.

5. Check that the XPaths are mapped correctly in the middle two sections. In the case of repeats, the parent of the repeatable object must be mapped to an identically named form data field.
6. Click Save to update the changes.

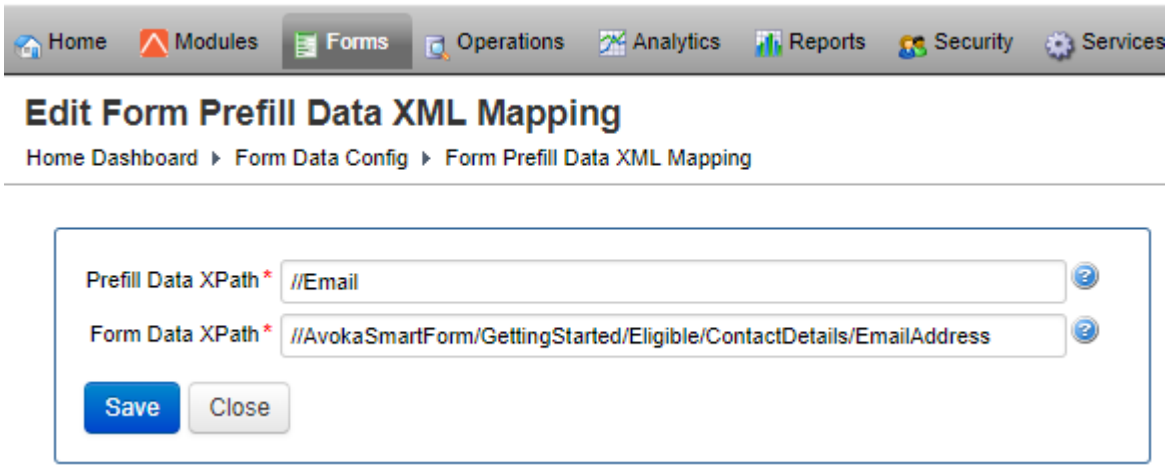
Next, go back to form input XML prefill mapping to finish the configuration.

Form Input XML Prefill Mapping Using XPath

Manager allows you to map each prefill data element from the source to the target using XPaths[1].

To map prefill data elements:

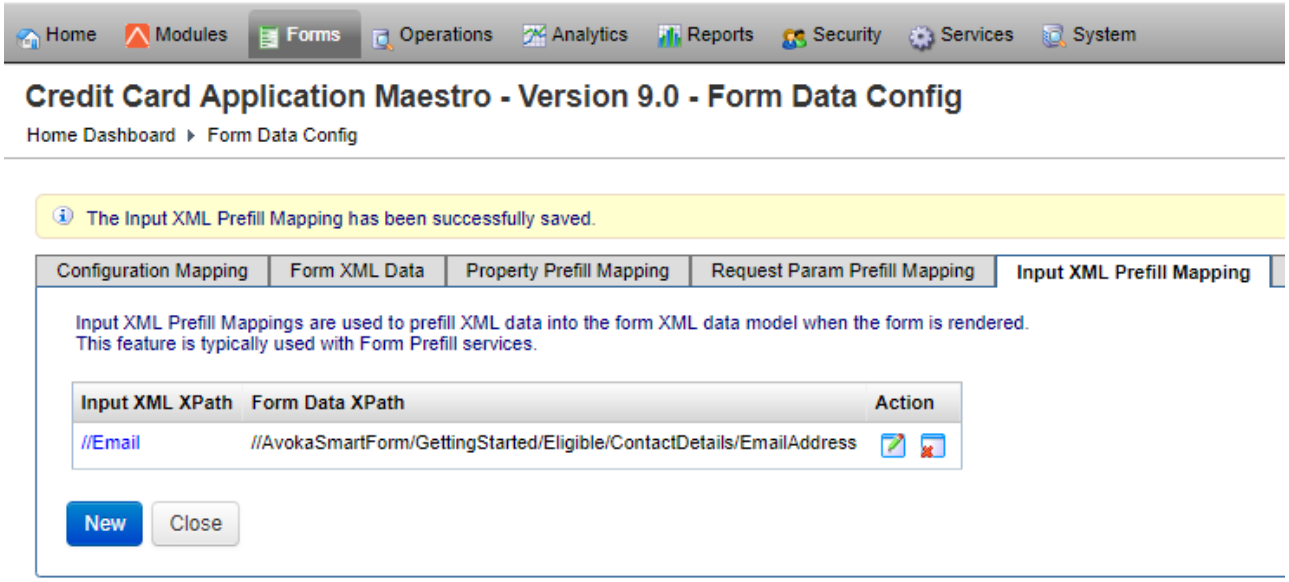
1. Click New to create a new mapping or click Edit to change the existing one.
2. Enter a valid sours XPath from the XML input version in the Prefill Data XPath field, for example, `//Email`.
3. Enter a valid target XPath from the form XML data in the Form Data XPath field, for example, `//AvokaSmartForm/GettingStarted/Eligible/ContactDetails/EmailAddress`.



4. Click Save to update the changes.

[1]XPath (XML Path Language) is a query language for selecting nodes from an XML document. In addition, XPath may be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document. XPath was defined by the World Wide Web Consortium (W3C).

5. Check that the XPaths are mapped correctly.



Next, go back to form input XML prefill mapping to finish the configuration.

Form Property Prefill Mapping

Manager allows you to map form properties to values, defined in a form seed XML file. You can use this functionality to prefill a form. For example, when a user logs in to a form space and starts to fill in an application form, user's contact details are already prefilled. This is achieved with the properties mechanism to prefill a form when it is rendered to a user. This relies on mapping a target form data field to the related user property using XPath[1] variables.

> NOTE
>
> If you don't map properties, they won't be in a XML document that Manager sends to a form for the properties prefill.

Form property can have one of the following scopes:

- Client - a property configured for an organization and its forms, so it obtains its value dynamically from an organization.
- Form - a property configured for form versions, so it obtains its value directly or dynamically from a form version. This property can override Client's scope property.
- User - a property configured for user profiles, so it obtains the value from a user's profile of a logged user.

Let's look at each scope and how it is prefilled in details:

- Client mappings prefill the form XML using the client's property value stored in the form's version properties. These properties need to be synchronized to work correctly. Once the mappings are created, the Form Versions Properties UI provides the Sync Org Properties button which will import the required Properties based on the Client Property Fill Mappings.
- Form mappings prefill the form XML using the Form's Property value stored in the form's version properties.
- User mappings prefill the form XML using the property value's stored in the logged in User's Profile. Once User Property Prefill mappings have been created for a form version, by setting the Form's System Profile flag UpdateUserProfile to true, on submission of a

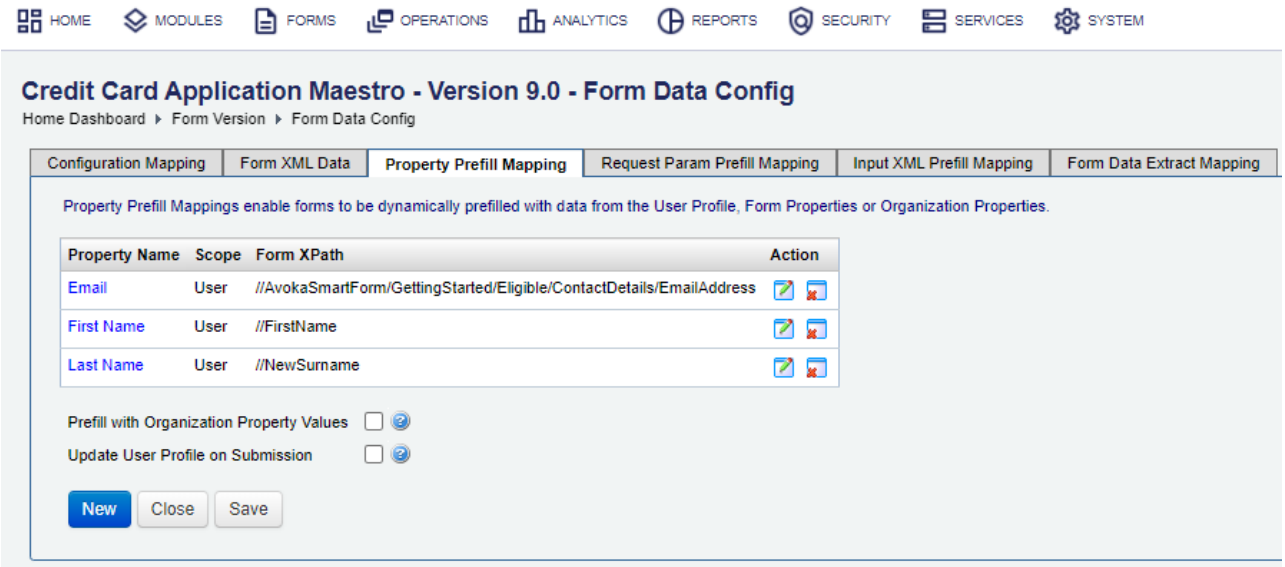
---

[1]XPath (XML Path Language) is a query language for selecting nodes from an XML document. In addition, XPath may be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document. XPath was defined by the World Wide Web Consortium (W3C).

form, the User's User Profile properties will be updated to reflect any submitted changes to the User Property prefilled fields. This affects only user properties defined in the user profile, not the User Account's details directly. Further, a User may have multiple profiles with different values. Only the profile that is currently selected for the User will be used for prefill and saving. If no User Profile is currently selected, no user data will be prefilled or saved.

To configure form property prefill mapping:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select a form version and click Edit Form Data Config.
4. Click the Property Prefill Mapping tab to view any existing mappings.



5. Click Edit to update the XPath of a property. Click New to create a new mapping. You have the following options to map elements depending on your Manager version:
   - Use drag and drop
   - Use XPath configuration. | 18.11 This feature was introduced in 18.11.

   NOTE

   You can't edit the property name or the scope of a property. If you wish to change them, you need to create a new mapping and delete the existing one.

6. Select the Prefill with Organization Property Values checkbox to use organization (Client scope) defined property values to prefill the form.

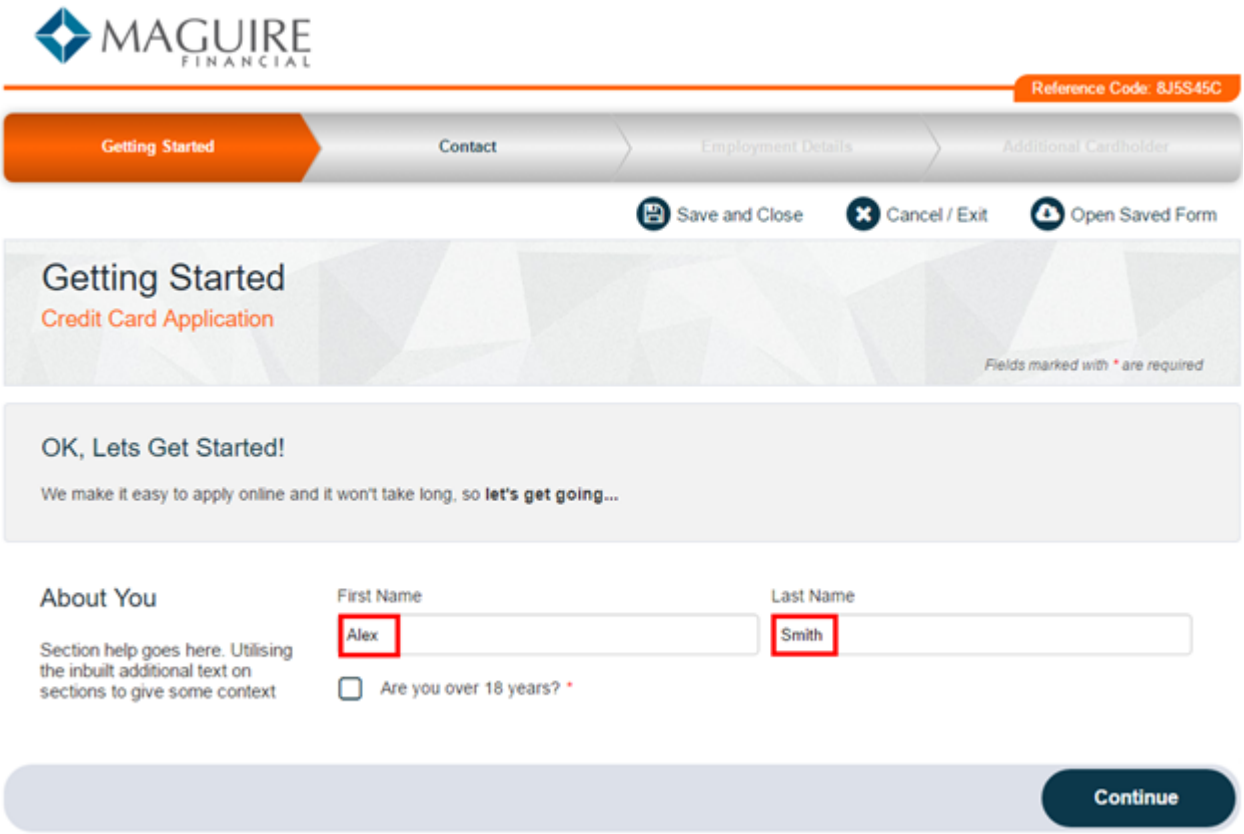
WARNING

Organizations property values will be overridden by property values defined at the form version level.

NOTE

This setting doesn't apply to user properties.

7. Select the Update User Profile on Submission checkbox to use the mapping for user properties at submission time to store data the user entered into the form back into the user profile.
8. Click Save to update the changes.
9. When a user logs onto a form and selects the first form, it is already prefilled with the user data such as First Name, Last Name, and Email from the user profile.



NOTE

Watch the Manager property prefill video to learn how to configure and use property prefill of form scope.

Next, learn about form request parameter prefill mapping.

Form Property Prefill Mapping Using Drag and Drop

Manager allows you to map form properties to user contact details to prefill a form using drag and drop functionality. Manager uses Adobe Flash[1] to implement drag and drop.

> NOTE
>
> Adobe Flash will be phased out by the end of 2020.

To map prefill data elements:

1. Select a property scope from the Property Scope dropdown list that can be:
   - Client for properties that should be set on an organization level.
   - Form for properties where values are specific to each form version.
   - User for properties sourced from the user profile.
2. Review the mappings in the property mapping table containing the following columns:
   - Property Name.
   - Scope of the property.
   - Form Data XPath for the XPath of the target data element in the form XML, which is used for prefilling the corresponding field on the form.
   - Target: Form Data XML for the target data element in the form XML.
3. Drag and drop the field from theTarget: Form Data XML column to Form Data XPath column against the related field in the Property Name column as shown below:

[1]Adobe Flash is a deprecated multimedia software platform used for production of animations, rich Internet applications, desktop applications, mobile applications, mobile games and embedded web browser video players. Flash displays text, vector graphics and raster graphics to provide animations, video games and applications. It allows streaming of audio and video, and can capture mouse, keyboard, microphone and camera input. Related development platform Adobe AIR continues to be supported.

4. Check that the XPaths are mapped correctly in the middle section.
5. Click Save to update the changes.

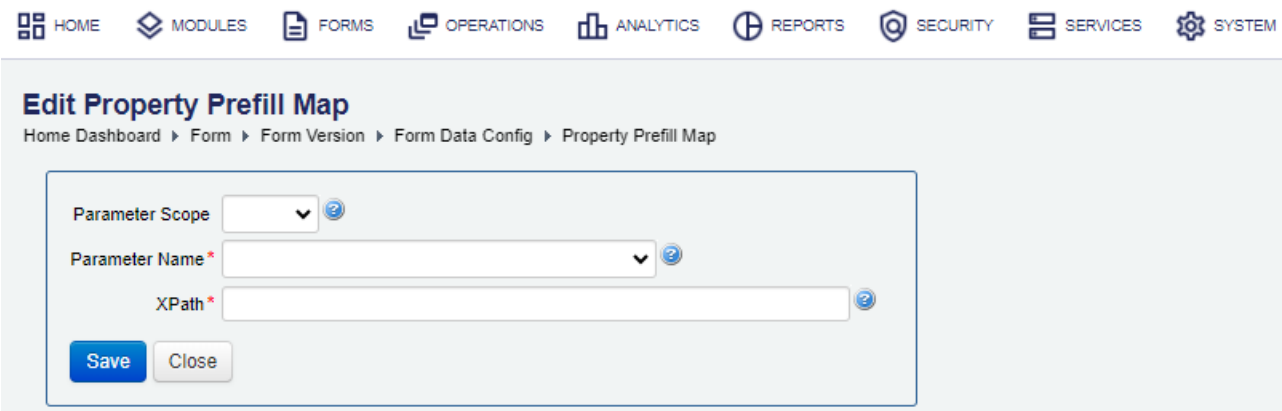
Next, go back to form property prefill mapping to finish the configuration.

Form Property Prefill Mapping Using XPath

Manager allows you to map form properties to values within the form's seed for prefill purposes using an XPaths[1].

To map prefill data elements:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select a form version and click Edit Form Data Config.
4. Click the Property Prefill Mapping tab and click New to create a new mapping.



5. Select a property scope from the Parameter Scope dropdown list that can be:
   - Client - a property configured for an organization and its forms, so it obtains its value dynamically from an organization.
   - Form - a property configured for form versions, so it obtains its value directly or dynamically from a form version. This property can override Client's scope property.
   - User - a property configured for user profiles, so it obtains the value from a user's profile of a logged user.
6. Select a parameter name from the Parameter Name dropdown list, for example, Email.
7. Enter a XPath to the valid XML element, for example, `//AvokaSmartForm/GettingStarted/Eligible/ContactDetails/EmailAddress`.

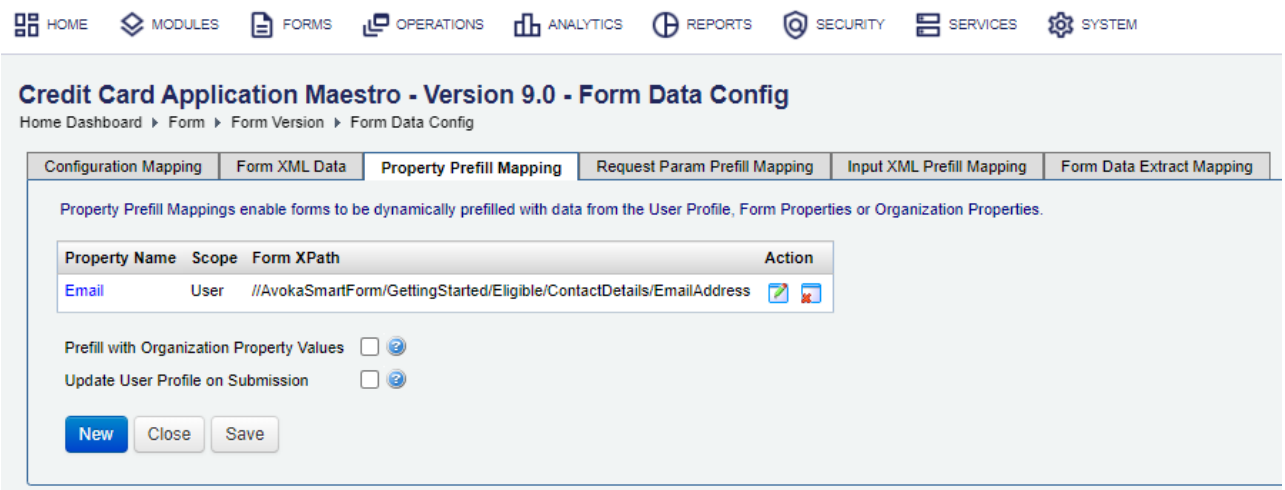
[1]XPath (XML Path Language) is a query language for selecting nodes from an XML document. In addition, XPath may be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document. XPath was defined by the World Wide Web Consortium (W3C).

8. Click Save to update the changes.

   If the XPath value is invalid, the No XML element found for XPath: //EmailAddress error message is displayed, so you need to find the correct XPath to proceed. To do so, Select the Form XML Data tab to validate that the XPath is mapped correctly by checking the form XML structure as shown below:

```
<AvokaSmartForm>
        ...
        <GettingStarted>
        ...
        <Eligible>
        ...
        <ContactDetails>
           <EmailAddress>
           </EmailAddress>
        ...
        </ContactDetails>
        ....
</AvokaSmartForm>
```

9. Check the created mapping.



10. Click Close.

Next, go back to form property prefill mapping to finish the configuration.

Form Request Parameter Prefill Mapping

As another means of form prefill, data may be obtained from an HTTP[1] request and used to prefill the form when it is rendered. Prefill data can be passed in via JSON[2] Prefill Data, request URL parameters, session values or cookie values.

When the form is rendered, Manager searches for each request parameter prefill mapping to find a value. When a match is found, the value of the entity is used as the prefill data. However, if multiple matches are found, the later entry takes precedence, for example request parameters over session attributes:

1. Set of HTTP session attributes
2. Cookies
3. Request parameters
4. Special request parameter, such as `jsonPrefillData`.

> NOTE
>
> It is not recommended to pass any PII[3] details, for example, names or contact details, as parameters.

As an example, you can pass the selected card type and interest terms as request parameters to prefill a Credit Card Application form. If a customer has already chosen the specific credit card from your website, then there is no need for them to select the credit card again when filling in the Credit Card Application form.

The Credit Card Application form must include two dropdown menus that are used to indicate the user's chosen credit card. You can add them in Manager:

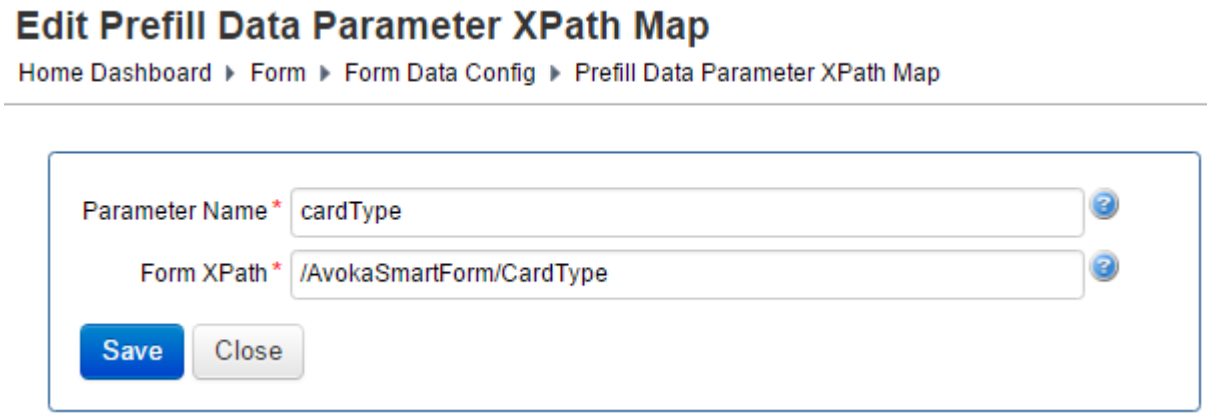
---

[1]HTTP (HyperText Transfer Protocol) is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.
[2]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.
[3]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

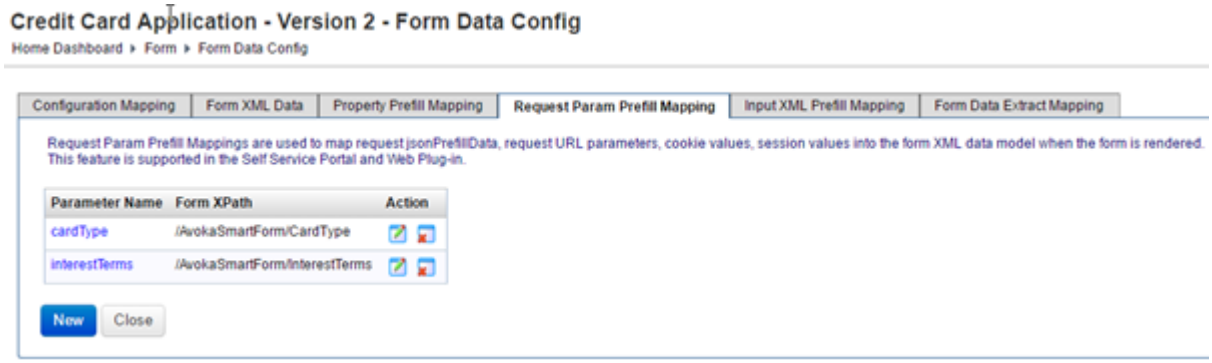
1. Add Card Type dropdown with options such as Student, Low Fees, Gold, and Platinum.
2. Add Interest Terms dropdown with options such as 55 Day Interest-Free and Low Rate.
3. Build your form.

To configure Request Param Prefill Mappings:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select a form version and click Edit Form Data Config.
4. Click the Request Param Prefill Mapping tab.
5. Click New.
6. Enter each mapping as shown below:

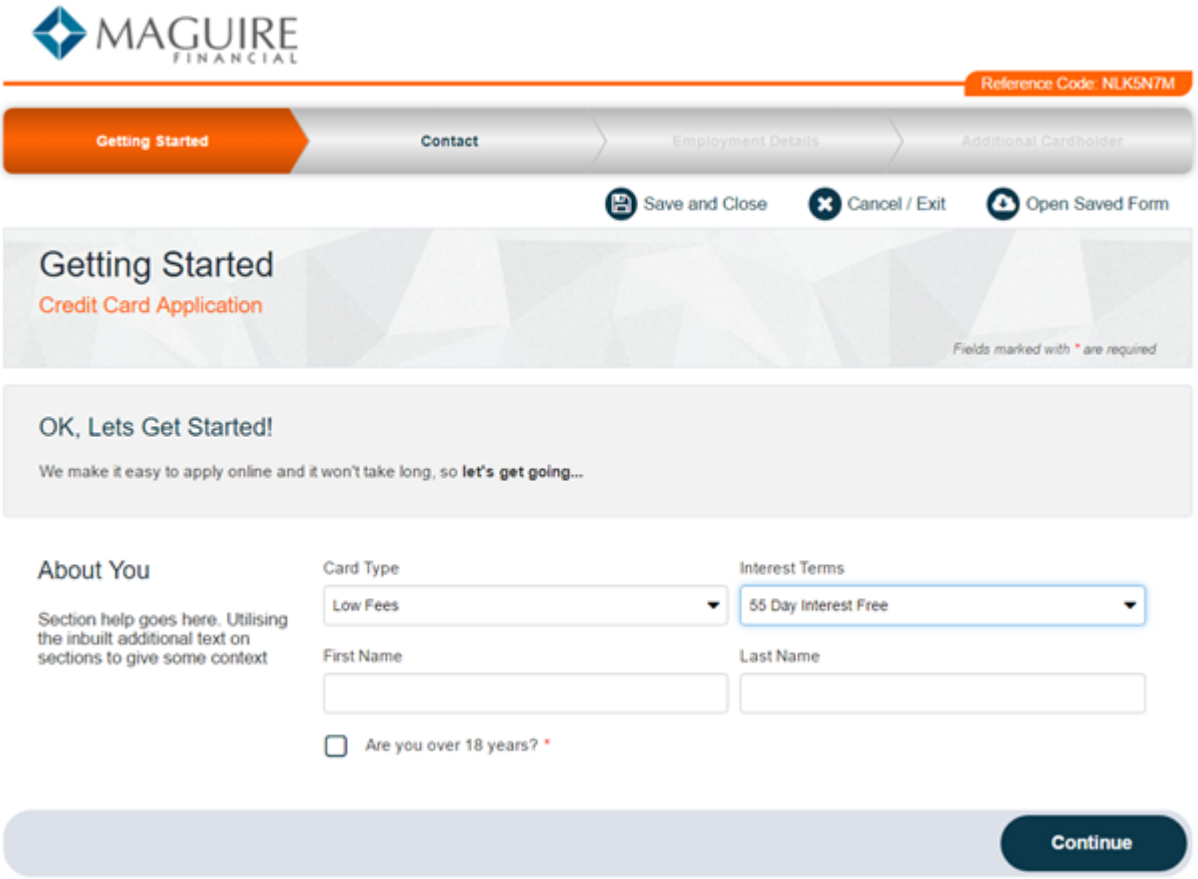


7. Click Save to update the changes.
8. Check the newly defined mappings.



To test these mappings, render your Credit Card Application form and try adding to the URL both parameters (card Type and interest Terms) with each value matching one of the related dropdown options, as defined in Maestro.

For instance, this shows that card type of Low Fees and interest terms of 55 Day Interest-Freewere passed as URL parameters to prefill the Credit Card Application form:



GET vs POST for Request Parameter Prefill Mapping

When using the Request Parameter Prefill Mapping feature, it's important to choose the appropriate HTTP method for your scenario. Manager supports both GET and POST requests with this feature. Each of these methods has its own unique behaviors. Many of these behaviors can be considered a pro or a con, depending on the situation.

GET Requests

- Easily cacheable
- Typically stored in browser history
- Can be stored in bookmarks
- Can be shared between customers
- Browser refresh function works without issue (although entered data will be lost, the same valid parameters can still be used to pre-fill the form)
- May store parameters in logging of network equipment

- Parameters visible in browser address bar
- Limited by maximum URI length

POST Requests

- Not limited by maximum URI length (can send more data)
- Parameters unlikely to be cached
- Parameters won't be stored in browser history
- Parameters won't be shared if the link is copied and shared/posted elsewhere
- Parameters won't be visible in browser address bar

Now that these differences are understood, let's consider a couple of scenarios where each of the methods would perform best.

Prefilling Product Information

In this scenario various pieces of product information need to be prefilled in order for the application form to know what product/sub-product the customer is applying for. Example parameters:

Parameter Name	Description
promoCode	A 10 character promotional code, which will update the form to display promotional rate/offers assuming the promotion is still active. (Optional)
productId	The 8 character numeric ID of the product being applied for, such as a particular type of loan or credit card. This will update the graphics and product wording in the form.

In this scenario, the parameters being prefilled are publicly available information. We're quite happy for our customers to share these links with their friends/family or on other sites such as social media. If the promo ID is invalid, appropriate messaging with be presented to the customer, so again it's OK for this information to be shared/cached/bookmarked/etc.

In this situation the best option is a GET request, which will ensure the required product ID parameter isn't lost if the link is reused.

Prefilling Customer Information

In this scenario a customer ID is being used to reduce the number of fields presented to the customer. Example parameters:

Parameter	Description

Name	
customerId	The ID our customer uses to login to online banking. This will be used to remove field-s/pages from the form when that information is already known about the customer.

In this scenario, the parameter being prefilled is more sensitive in nature. Although there is no data harvesting risk, since we're removing sections rather than pre-filling personal data, this is still something we'd rather not have stored in browser history or shared accidentally.

In this situation the best option is a POST request, which will help ensure the parameter isn't stored in history, shared or visible in the browser address bar.

Next, learn about form input XML prefill mapping.

Add Repeating Content from XML Prefill Mapping Dynamically

When you configure Input XML Prefill Mapping in Manager to prefill a form, you need to take into account how repeats will factor into the prefill XML. This is especially true when you want to provide inspectors with case history on a coversheet, or if field workers are required to check off items which are driven by ERP. Although there are other workarounds, this technique is quite elegant, as it minimizes coding, and allows for optimized dialog between form developers and integration teams. In this example, we define an array of JSON objects in a single XML value to store all the prefill data in the repeat. An array of JSON objects is the standard used by Dynamic Data. Therefore, there's some re-usability in response formats. Also, it can easily be updated with test data with a single button (good for demos or UI testing).

You need the following widgets on your form:

- A data element to store the JSON object (in this example, called JSONactionsData)
- A Repeating Block Template, customized to include whichever content you like (in this example, three fields). Ensure initial value is set to 0.
- A business rule, to set the contents of the repeat using the data element.
- A standard button, to set the contents of the data element (In this example, called "Simulate Lookup")
- Optionally, a field called "debug", to see the contents of the data element above.

You have to create the following business rule:

```
if (!sfc.isBlank({JSONactionsData}) && ({RepeatableContent}.getInstanceCount()==0)){
    //assume we only ever run once to avoid duplicates on save & resume ------^
    var obj = JSON.parse(unescape({JSONactionsData}));
    for (var i=0;i<obj.length;i++) {
        var newInstance = {RepeatableContent}.addInstance();
        sfc.setRawValue(sfc.convertToNode(sfc.normalizeUid(newInstance, "./RepeatIn-
ner/Item/_outerArea/_contentArea/RepeatingFields/_layoutAtAction/At")),obj[i].at);
        sfc.setRawValue(sfc.convertToNode(sfc.normalizeUid(newInstance, "./RepeatIn-
ner/Item/_outerArea/_contentArea/RepeatingFields/_layoutAtAction/Action")),obj[i].ac-
tion);
        sfc.setRawValue(sfc.convertToNode(sfc.normalizeUid(newInstance,
"./RepeatInner/Item/_outerArea/_contentArea/RepeatingFields/_lay-
outAtAction/readOnly")),"true");
    }
}
```

Where:

- `{JSONactionsData}` is the data element
- `{RepeatableContent}` is the Instance Manager of the repeat
- `_layoutAtAction` is the root custom field in the Repeating Template, in this case a block with three fields.

Then, you need to create the following button (Simulate Lookup):

```
var obj = [ {"at":"AAA", "action":"Check SCADA"},
            {"at":"BBB", "action":"Check 10 Open"},
            {"at":"BBB", "action":"Check 11 Open"},
            {"at":"AAA", "action":"Close BBB 30"}, ];
sfc.setRawValue( {JSONactionsData} , escape(JSON.stringify(obj)));
```

Where the only dependency is a node {JSONactionsData}.

Finally, you can use the following debug calculation:

If you want to see the data which we are storing in the data element, you will need to unescape it before displaying it.

```
if (!$ISBLANK({JSONactionsData})){
    return unescape({JSONactionsData});
} else return undefined;
```

This button sets the values. It makes a great demo, but it can also be extended to use data from a lookup or other type of prefill.

Next, learn more about form data XML.

Form Prefill for Anonymous Users

Manager offers a wide range of options to support form prefill or pre-population including anonymous users.

There are several options for different use cases, which are safe to use for Personally Identifying Information[1] data meaning they are only to be used to pass non PII data, such as a product choice on a website.

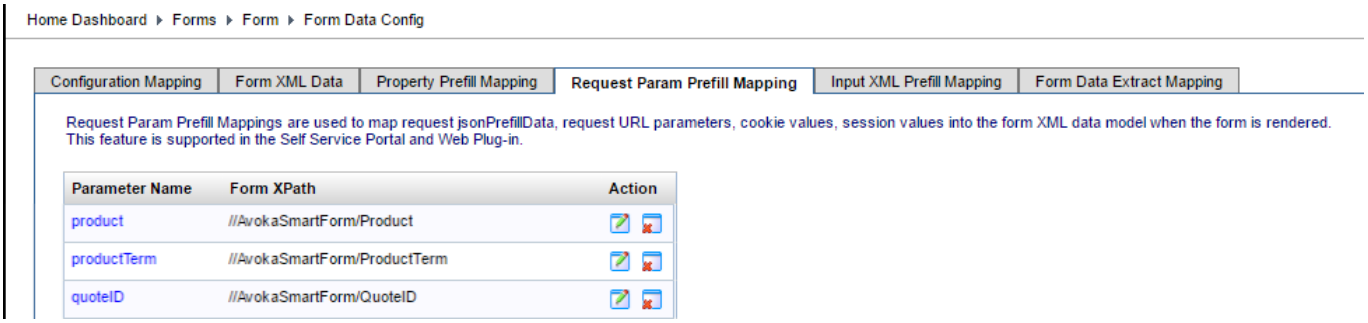
It should be mentioned that a term Anonymous is actually a bit misleading. The user might be authenticated in another system, they are just not a registered user in Manager.

You can solve this problem There are several options when tackling this problem.

- URL Parameter passing on a GET request
- HTTP request headers on a POST request
- Cookies
- Short life tokens
- Encrypted string

URL Parameter passing on a GET request

It doesn't get much simpler than this. All you need to do is add some form prefill mappings using the same name as the URL parameter you use as shown below.



Would work with a url such as https://t-m.demo.avoka.com/test/servlet/SmartForm.html?formCode=cgf-apply-ga&-product=single&productTerm=Fixed&quoteID=GTM495780

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

Pros

- Very simple
- Out of the box

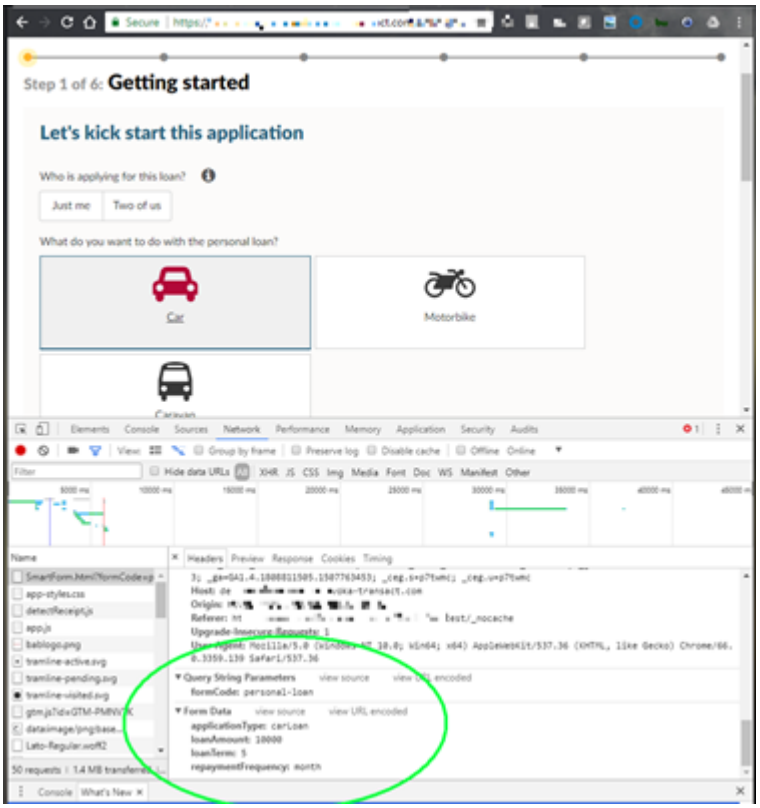
Cons

- It should only be used for non sensitive or PI data i.e. you should not be passing data such as the customer name

HTTP request headers on a POST request

This is pretty much the same mechanism as the above method but the data name/value pairs our provided as header attributes on the POST request. This implementation is very common in our solutions.

The example is shown below



The Manager config is exactly the same.

Pros

- Simple
- Out of the box

- Slightly more sophisticated than URL parameters

Cons

- It should only be used for non sensitive or PI data i.e. you should not be passing data such as the customer name

Encrypted Cookies

This is not an out of the box option and requires the creation of a custom form component and two TM groovy services.

The service requires three components:

- Cookie Manager (Composer/Maestro widget) - added to the form
- XXXCLIENT Cookie Decryption (Transact Dynamic Data service) - called by the Cookie Manager widget
- XXXCLIENT Cookie Encryption (Transact Submission Completed Processor service) - associated with the form

How it operates:

- XXXCLIENT Cookie Encryption: extracts form values from the "XXXCLIENTData" node and populates an encrypted cookie named "XXXCLIENTData".
- Cookie Manager: holds data fields populated from the encrypted "XXXCLIENTData" cookie. Also hold a business rule that invokes the "XXXCLIENT Cookie Decryption" dynamic data service to decrypt the contents of the cookie. The business rule is only invoked once during the form's initialization phase.
- XXXCLIENT Cookie Decryption: Dynamic data service invoked via form business rule within the Cookie Manager widget.

On submission the data from the form is encrypted and made available to any subsequent form in the solution that the user might access.

Pros

- Fairly secure

Cons

- Relies on the next form being opened on the same device

Short life tokens (SLT)

There are often several client systems from which secure Personally identifiable information[1] (PII) can be sourced. For this reason, Manager provides a simple reusable prefill implementation that can be consumed by several “client system” sources, such as:

- Online Banking
- Customer portals
- CRM

Manager exposes a RESTful service to enable the “client system” to push customer data to Manager when a customer requests the form from within the “client system” application. Manager passes back a unique identifier (UUID) that will enable the client system to build a URL request that will open the form with the prefilled data.

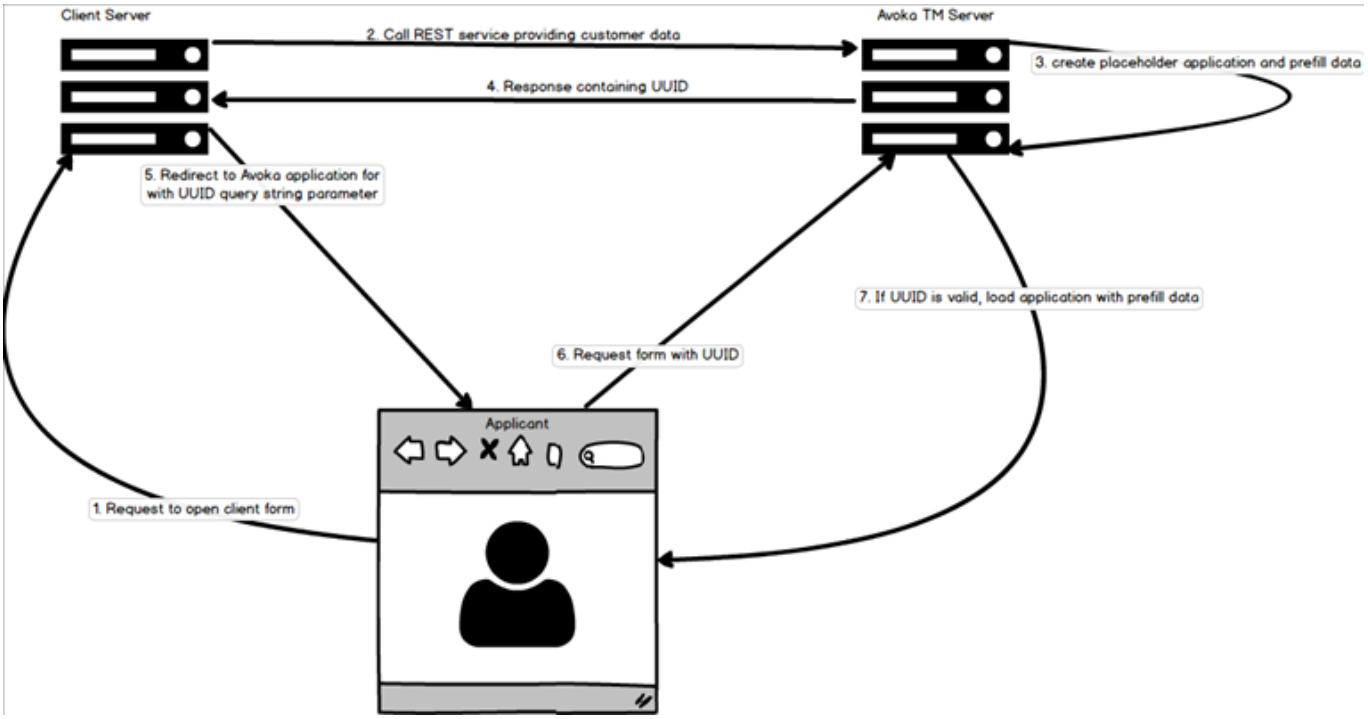
Prior to Manager 5.0, implementations of a "container" submission are used to store the data. Since Manager 5.0, storing the temp prefill data in MemCache for the very short time is possible and is a neater design than creating a temporary submission object to hold the prefill data for a short time frame.

The token will be short lived, if the UUID (submitkey) passed is for a container submission older than 30 seconds (configurable), Manager loads an empty form. The container will be removed.

If the form prefill is successful, the form will load, and the container will be removed.

A tidy up task will be scheduled to remove any containers older than 1 minute.

[1]Personally identifiable information (PII) is any data that could potentially identify a specific individual. Any information that can be used to distinguish one person from another and can be used for de-anonymizing anonymous data can be considered PII.



If the RESTful customer data service (2) is unsuccessful the UUID will not be returned.

If the form prefill service in Manager fails for one of the reasons below, it will load an empty form.

For the prefill service call (2), IP whitelisting and basic auth. will be used to secure access.

Several URL parameters will need to be passed with the Request (5):

Example prefill data provided could be:

- Salutation
- First Name
- Last name
- Date of birth
- Address
- Mobile
- Email

Pros

- secure option
- single prefill service required in Avoka
- securely pass data from multiple sources

Cons

- does require some work by the client

Encrypted String

This is an enhancement on the prefill mapping GET or POST request, rather than sending name value pairs in clear text you encrypt all your prefill data into a single string. This is not a common implementation.

```
Generally the outline is as follows:
 # A public/private key pair is generated for the TM instance. The encryption (public)
key is shared with the external calling site/portal.
 # The external (referring) website/portal generates the secure token and adds this to
the HTTP request, either in the payload data or as a request cookie.
 # The TM prefill service parses the token from the request, decrypts and prefills the
form per BAU.

The encryption / decryption code is largely boiler-plate Java code and easily trans-
portable, e.g.:
/**
 * Encrypts or decrypts the specified byte value
 * Presumes that the decryption value has been Base64 decoded already, e.g.:
 * def URLCodec urlcodec = new URLCodec()
 * def byte[] value = urlcodec.decode(aString).decodeBase64()
 * Presumes that the encrypting value is in UTF-8 encoding, e.g.:
 * byte[] value = aString.getBytes("UTF-8") *
 * @param value
 * @param encrypt
 * @param keystore
 * @param keystorePass
 * @param keyAlias
 * @param aliasPass
 * @return byte[]
 */
private byte[] encryptDecrypt(byte[] value, boolean encrypt, byte[] keystore, String key-
storePass, String keyAlias, String aliasPass) {
 // access the keystore and load the decryption key
 KeyStore keyStore = KeyStore.getInstance(KeyStore.defaultType)
 ByteArrayInputStream bais = new ByteArrayInputStream(keystore)
 keyStore.load(bais, keystorePass.toCharArray())
 Key key = keyStore.getKey(keyAlias, aliasPass.toCharArray())
 int keyBuffer = key.getModulus().bitLength() / Byte.SIZE
 Certificate certificate = keyStore.getCertificate(keyAlias)
 Cipher cipher = Cipher.getInstance(key.algorithm)
 // encryption
 if(encrypt) {
 cipher.init(Cipher.ENCRYPT_MODE, certificate.publicKey)
 ArrayList encryptedArray = []
 value.toList().collate(keyBuffer-11).each() { bytes ->
 def encryptedData = cipher.doFinal((byte[])bytes)
```

```
  encryptedArray.addAll(encryptedData)
  }
  return (encryptedArray as byte[])
  }
  // decryptior
  else {
  cipher.init(Cipher.DECRYPT_MODE, key)
  StringBuffer decryptedStr = "" << ""
  value.toList().collate(keyBuffer).each() { bytes ->
  def decryptedData = cipher.doFinal((byte[])bytes)
  decryptedStr << new String(decryptedData)
  }
  return decryptedStr.toString().getBytes("UTF-8")
  }
 }
```

Pros

- Fairly secure
- Prefill data is encrypted.
- It can be a little more flexible to a roll-you-own solution without resorting to more structured models (like SAML).

Cons

- For larger payloads, encrypted data will need to be sent as a POST request.
- Encryption keys need to be maintained (on both sides).
- Prefill data is encrypted. Debugging and operational support becomes more difficult.

Next, learn about form property prefill mapping.

Launch a Form From Another Form

Manager allows you to launch one form from another one using prepop.

Sometimes you may have an existing database of information that you want to use as a basis for pre-populating a form. This data is often known as Context Data.

For example, you may have an existing customer database (or list of properties or contracts or medical records or accounts or property locations or whatever), and you want to be able to:

- search for a customer by various criteria
- view summary information for all matching customers
- click a button to open a form on behalf of that customer (there could be multiple forms to choose from)
- have that form open in new window, and be pre-populated with the customer information.
- The form submits in the usual way.

Technique

1. Search form

Build a search form. This will use a Dynamic data lookup to populate a table or a repeat with information from the database. (This tech note does not cover how to build this search form using dynamic data, that is covered elsewhere.)

There are two alternatives.

1. If there is just a small amount of data, pass all the data in request parameters to the new form, and use pre-fill mappings to map the request parameters into form field data.
2. If there is a larger amount of data, pass a unique identifier in a request parameter, and use a form pre-fill service to populate the form.

This article will look at the second technique which is slightly more complicated - the first technique is very similar

In the result row, including a button with the following script:

```
var param = {MyUniqueIdField}; // This parameter is the unique id that comes from the
repeating sectior
var formUrl = "https://t-
m.demo.avoka.com/finance/servlet/SmartForm.html?formCode=myformcode";
// Append a parameter to the end of the URL
window.open(formUrl + '&myparam=' + param,'_blank');
```

This script will launch the form named "myformcode" with an extra parameter.

Note that the Search form does not have a submit button - it is used only to retrieve data, and launch

Main form

The Main form will use a pre-pop script in order to fetch prefill data. There are several different ways to do the actual prepop, but the key issue is getting hold of the URL parameter that was passed from the Search form.

If you create a new service of type Groovy pre-fill script, it accepts several parameters, including:

```
formRequestParams : Map<String, String>
```

In order to retrieve our parameter, we would use something like this:

```
var myParam = formRequestParams['myparam'];
```

Thereafter, use this parameter to retrieve data from the backend system, and generate the XML pre-pop data.

Please see the following page for more information, and also refer to the Transaction Manager documentation.

Alternatives

- Another way to do this is to directly add a similar button into the native application that you use for your customers. However, sometimes this is not possible to do, so only building the Search form may be useful.
- This sample hard-codes the location of the server ('formUrl" in the JavaScript code above). You can build up the URL dynamically by using some of the values injected into the form by Transaction Manager - you will need to use the DynamicDataServiceURL and strip out some of it in order to find the base URL for forms. This has not been included in this sample to keep it simple.

Restrictions

This technique cannot be used in the Transact Field App.

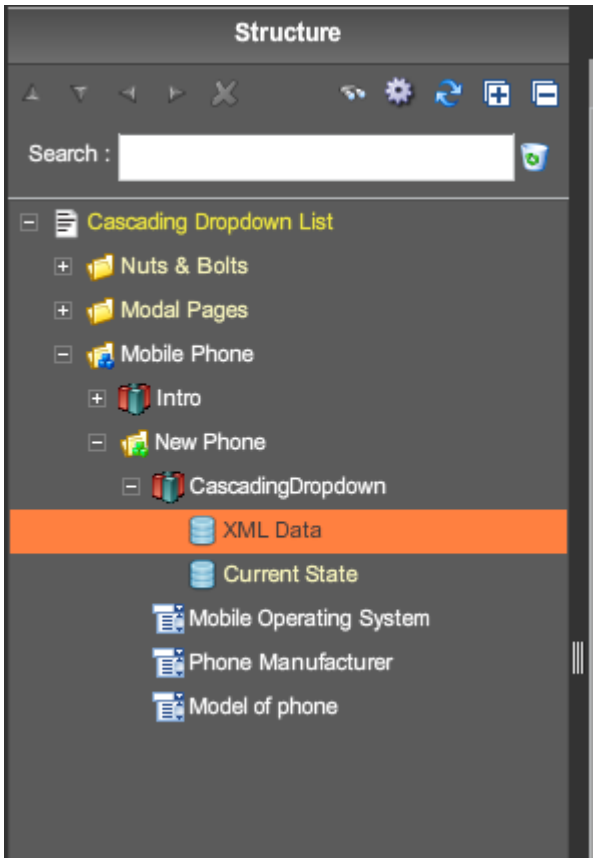
Next, learn about form property prefill mapping.

Prefill a Cascading Dropdown in Composer

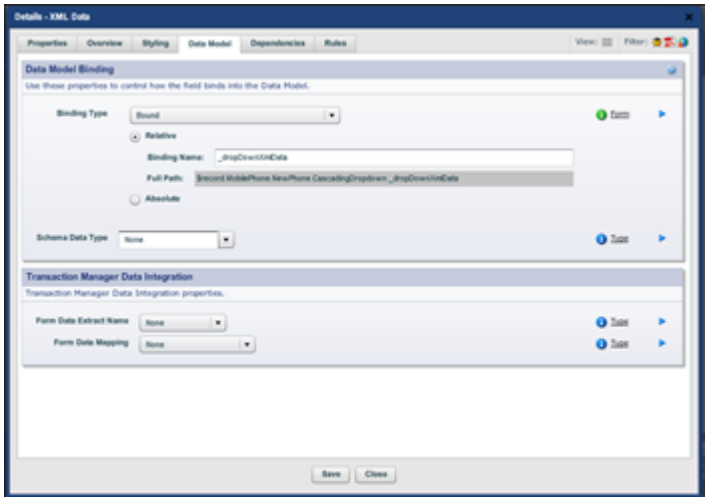
Manager allows you to update XML data that defines a cascading dropdown in a Composer form from the form properties defined within Manager.

To prefill a cascading dropdown:

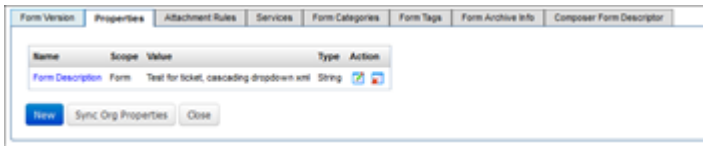
1. Open the structure panel in Composer and navigate to the XML data element to adjust the setting for the XML data model bindings in the form before publishing it.



2. Open the properties for the xml data. Navigate to the data model tab and change the binding type to Bound. Note the path as you may need it later.



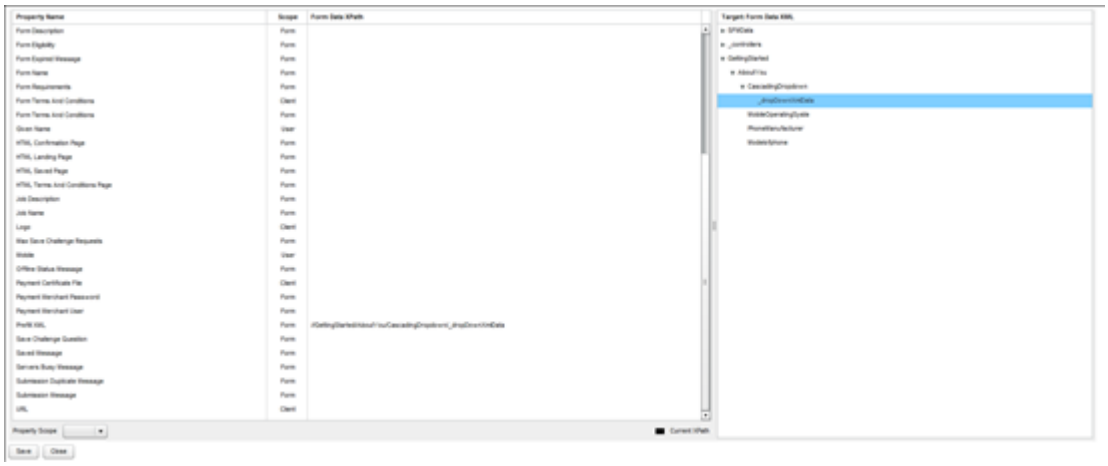
3. Publish the form to Manager.
4. Navigate to your form in Transaction Manager and click on properties in the Form Versions block for the form you are making these changes to. Here we are going to add a new form property with some modified XML that will be used to update the form in Manager.



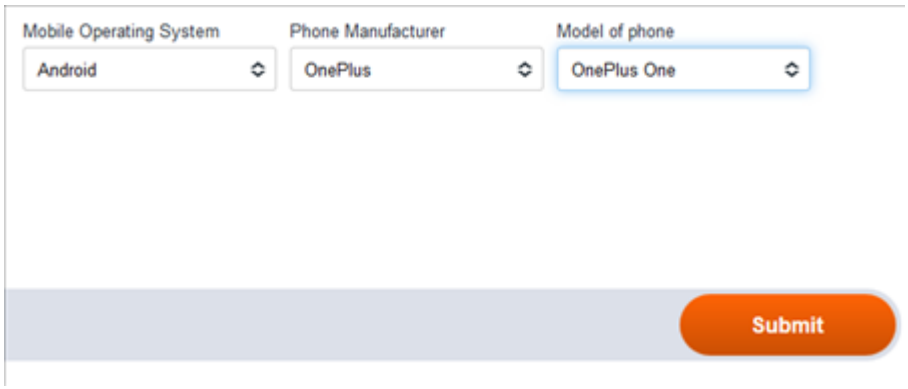
5. Click new and fill in the Name and set the data type to String. A new field will become visible; this is where you add the modified xml that will be used to update the form. For this example, another manufacturer was added to the XML from the original example used.

```
<root>
<os name="Android" value="Android">
  <manufacturer name="HTC" value="HTC">
    <model name="One" value="One"/>
    <model name="One X" value="One X"/>
  </manufacturer>
  <manufacturer name="Samsung" value="Samsung">
    <model name="Galaxy S3" value="Galaxy S3"/>
    <model name="Galaxy Note 3" value="Galaxy Note 3"/>
    <model name="Galaxy S4" value="Galaxy S4"/>
    <model name="Galaxy Express" value="Galaxy Express"/>
    <model name="Galaxy XCover 2" value="Galaxy XCover 2"/>
    <model name="Galaxy Fame" value="Galaxy Fame"/>
  </manufacturer>
  <manufacturer name="OnePlus" value="OnePlus">
    <model name="OnePlus One" value="OnePlus One"/>
  </manufacturer>
</os>
<os name="Windows Phone 7" value="Windows Phone 7">
   <manufacturer name="Nokia" value="Nokia">
      <model name="Lumia" value="Lumia" />
      <model name="Lumia 625" value="Lumia 625" />
   </manufacturer>
   <manufacturer name="Samsung" value="Samsung">
      <model name="OMNIA 7" value="OMNIA 7" />
   </manufacturer>
</os>
<os name="iOS" value="iOS">
  <manufacturer name="Apple" value="Apple">
    <model name="iPhone 5" value="iPhone 5"/>
    <model name="iPhone 5S" value="iPhone 5S"/>
    <model name="iPhone 5C Green" value="iPhone 5C Green"/>
    <model name="iPhone 5C Blue" value="iPhone 5C Blue"/>
    <model name="iPhone 5C Yellow" value="iPhone 5C Yellow"/>
    <model name="iPhone 5C Pink" value="iPhone 5C Pink"/>
    <model name="iPhone 5C White" value="iPhone 5C White"/>
  </manufacturer>
</os>
</root>
```

6. Click Save and navigate back to the starting form page. Now click Data Config, located in the same field as Properties from earlier. In data config navigate to the Property Prefill Mapping tab. Now click the edit button.
7. On this page you will see a list of properties in a large table. Scroll down till you see the one you created earlier. On the right hand side of the page you will see the Targets pane click through the links there to open the path you saw when you changed the bindings for the Xml data in composer it should be something like the picture below.



8. Drag `_dropDownXmlData` over to the property you created to make the link. Click Save and then save again when it returns to the data config page. Return to the form page. Now when you load the form in Manager, it will reflect the changes made to the XML.



Next, learn about [form property prefill mapping](#).

Form Versions Overview

Manager offers a flexible mechanism to import, create and manage multiple versions of a form, which benefits in the following:

- Allows users to switch between different form versions.
- Allows controlled releases of new versions into a Manager environment.
- Provides the ability to roll back a form version if it has a problem or a bug. When developing and deploying forms, occasionally challenges will arise making it necessary to revert to a previous version of a form. Form versions are also required to support the form A/B testing feature in Manager.

Although Manager provides the functionality to create a new form version based on the existing form, we recommend using Maestro for this purpose, because it provides manageable and clear approach to keep form versions in sync across both tools. Let's look at this in more detail.

You create a new version of a form in Maestro, edit it to add features and functionality and then build it in the Maestro editor to generate a JM Form Version ZIP file. This file is a new form version of the original form and you need to deploy it to Manager using the Import Form functionality.

We don't recommend creating many versions of a single form. Ideally, there should not be more than 10 form versions. This is not to clutter the environment.

Even though there can be many form versions of a form, there is always only one current version, which is used to server user requests for this form. You can't delete a current version of a form, but you can nominate another form version to be the current one by using the Make Current functionality. Manager also allows you to change the current form version automatically at a scheduled time.

After you have imported a new form version, you need to apply the following configurations to start using it:

- Form version details
- Form version functions
- Form version services
- Form version attachment rules
- Form categories
- Form design info

- Job properties
- Job info

COMPOSER

It is not possible to create and maintain form versions in Transact Composer. Instead, Transact Composer users may choose to import the Transact Composer package, which is the published form, as a new version.
Form versions contain the actual form template file, the form data configuration, property and metadata values, attachments and other information.

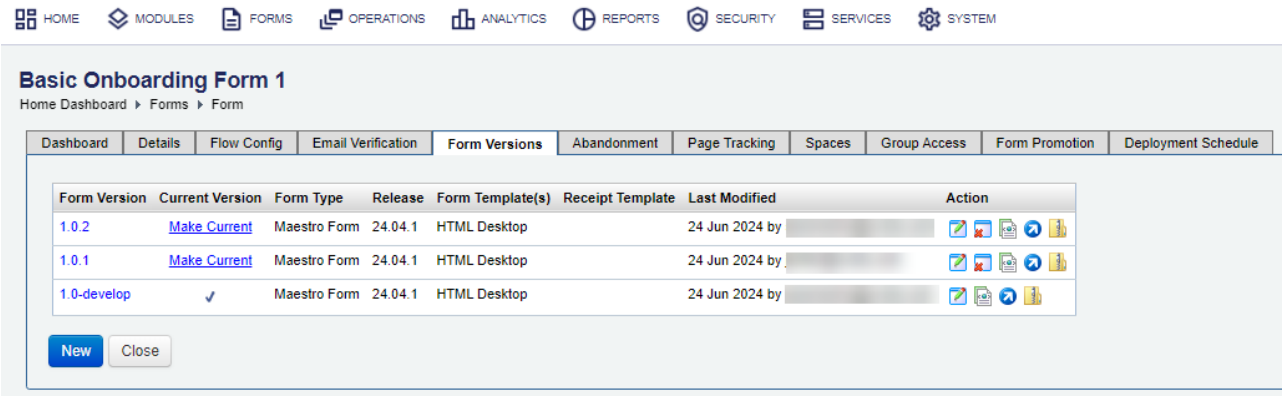
Next, learn how to configure a form version.

View Form Versions

Manager allows you to view all available form versions that have been imported or created for an application form, so you can easily manage them.

To view form versions of a form:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select the Form Versions tab to see the following details.



The list displays the following details, sorted by last modified date of a form version:

- Form Version: a form version that corresponds to the version of a form in Maestro.
- Current Version: marks the current form version with . Click Make Current to promote a selected form version to be current.
- Form Type: a type of a form that identifies how the form was created. A type can be one of the following:
  - Composer SmartForm: a Composer FAR file that is created when you use Publish as ZIP file option in Composer.
  - Maestro Form: a Maestro file that is created when you download a form version after publishing.
- Release: a release of Maestro or Composer the Form Version was published or built in.
- Form Template(s): a template of the form, which can be one of the following:
  - HTML Desktop: for Maestro and Composer forms.
- Receipt Template: a receipt template or blank if it doesn't exist.
- Last Modified: when and who modified the form version in Maestro or Manager last.

You can also perform the following actions:

- Click Edit to view and update a form version.
- Click Make Current to promote a selected form version to be current.
- Click Form Data Configuration to view and update form data configuration.
- Click Download to download the form or receipt template as an XDP[1] file.
- Click Form Test Render to render the form version in a new web browser tab.
- Click Export Archive to download a form version archive file.
- Click Remove to delete a form version.

> NOTE
>
> You can't remove the current form version.

Next, learn how to configure form abandonment.

[1]XML Data Package (XDP) format is a file format developed by Adobe Systems for packaging PDF data into XML files; contains the entire PDF document contents including form and template data; may also include character-encoded sections for binary content.

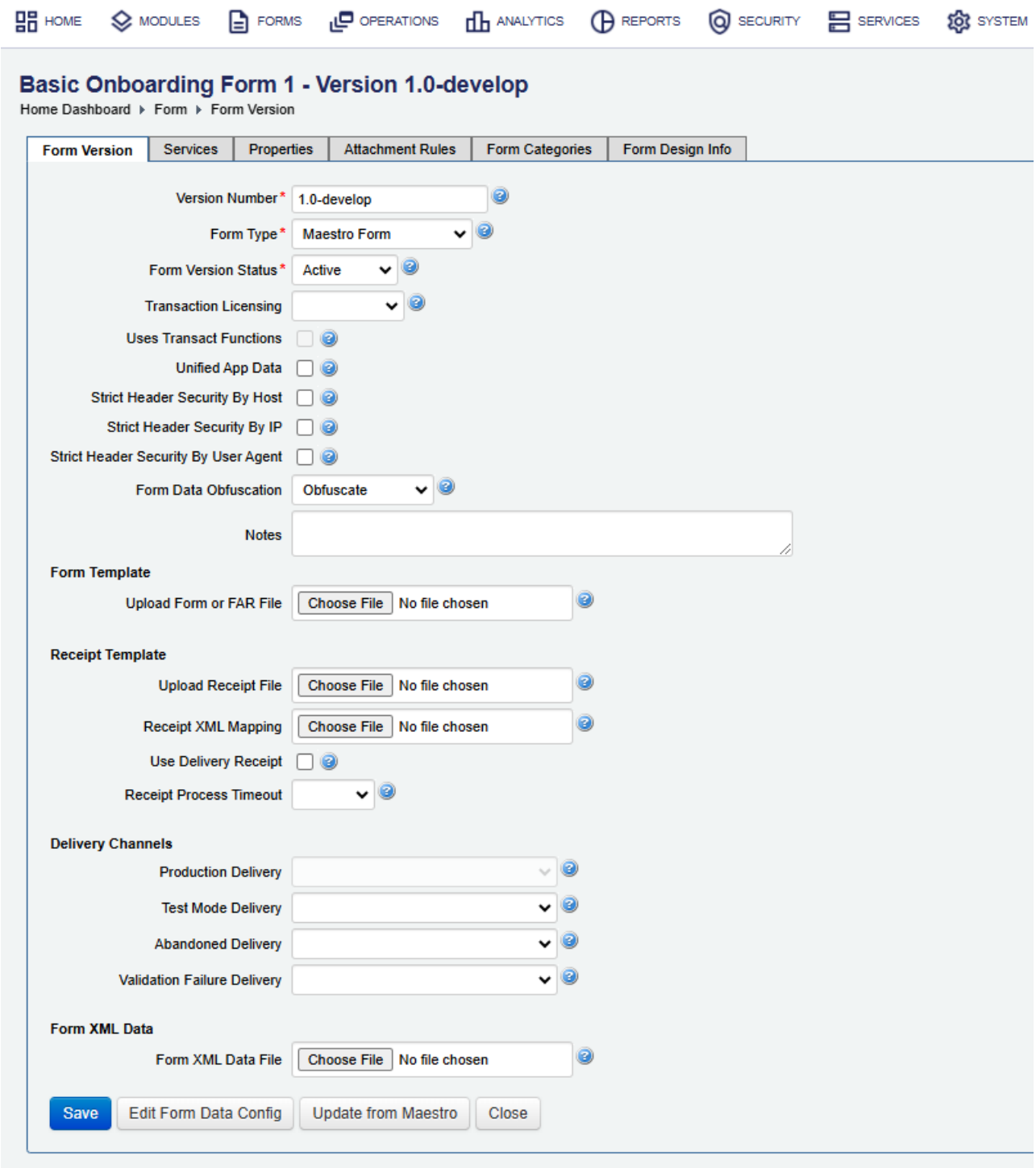
Configure a Form Version

Manager allows you to view and update configuration of a form version. A form version contains the set of versioned information for a form. To allow flexible switching between versions, almost all data related to a form is versioned. A form data configuration defines how prefill data are mapped into the form's XML schema and identifies special sections in the form. Form data configurations are crucial for pre-populating forms with property values such as a logo, attachments and contact details.

To configure a form version:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select a form version and click Edit.

4. Select the Form Version tab to view and modify the configuration.



5. Update a version number of the form, which is generated by Maestro during form build, in the Version Number field.

NOTE

We don't recommend updating the version number manually. It should be populated when you import a form version.

6. Select a type of form deployed on the server from the Form Type dropdown list.
   - Composer SmartForm
   - Maestro Form
   - JS App - refers to an Open UX form
   - Dynamic PDF Form
   - Static PDF Form
   - Other Document
7. Select a status from the Form Version Status dropdown list. A form version status can be one of the following: | 21.11 This feature was introduced in 21.11.
   - Active: it is in use and readily available to all users. All functionality is allowed.
   - Offline: it can't be used at all. No functionality is available, including DDS, Delivery and so on.
   - Inactive: similar to Offline.
   - NoRender: it doesn't allow for new or saved and resumed forms, but ongoing open sessions can be completed. DDS, submits, saves, and delivery are allowed.

NOTE

Select the Inactive checkbox to retire a form version without deleting it.

8. Select a licensing type of the form version from the Transaction Licensing dropdown list, which can be one of the following: | 22.10 This feature was introduced in 22.10.
   - Billable: a form version is licensing for billable transactions.
   - Non-Billable: a form version is licensing for non-billable transactions.
   - Blank: both of licensing types. This is the default.

NOTE

Knowing the licensing type of a transaction helps you with the tracking of billable transactions. For more information, see transaction licensing report.

9. The Uses Transact Functions checkbox specifies whether the Transact Function API programming model is available on the server. Transact Functions allow you to apply various business rules at crucial form events, such as form opening, saving or resuming, which

gives a standardized approach to handling the events. | 17.10 This feature was introduced in 17.10.

If this checkbox is not selected, the Functions tab is not shown so you can't view and configure form functions. To enable this checkbox and thus have ability to use and edit form functions, you must build a form in Maestro with the Use server side Transact Functions (requires TM 17.10 +) checkbox selected.

Conversely, if you don't require form functions in your form, you must rebuild the form with this option unselected. However, we recommend building the first form version with the Use server side Transact Functions (requires TM 17.10 +) option always selected so you can decide later whether you need it or not. This is because if you decide to add form functions to a form and import it as a new version, you may need to re-import all versions of the form because of version conflicting configurations or import the form with a new name.

10. Select the Unified App Data checkbox to enable the Unified Application Data Document Model in this form version. It provides you with the fine-grained control of the output data by decoupling the client-side data from the server side. For example, you may decide not to send sensitive fields to the user's browser. | 17.10 This feature was introduced in 17.10.
11. Select the Strict Header Security By Host checkbox to enforce strict matching of user session and request HTTP headers (`Referer`). It is not selected by default. | 24.10 This feature was introduced in 24.10
12. Select the Strict Header Security By IP checkbox to enforce strict matching of user session and request HTTP headers (`IP Address`). It is not selected by default. | 24.10 This feature was introduced in 24.10
13. Select the Strict Header Security By User Agent checkbox to enforce strict matching of user session and request HTTP headers (`User-Agent`). It is not selected by default. | 24.10 This feature was introduced in 24.10

    Manager checks that requests are coming from the same User Agent and IP Address that were used in the user's last login session to a form space. Manager also check the Referer header to ensure it refers to the server instance it is accessing. This hardens the system security against man-in-the-middle attacks.

    NOTE

    This configuration overwrites the Strict Header Security settings of a security manager > Configure a Security Manager.

14. Select an obfuscation mode of HTML client-side form XML data from the Form Data Obfuscation dropdown list, which can be one of the following:
    - None: no obfuscation is performed. Choose this option if you need to run automated tests that require viewing and modifying the `formData` payloads sent from the web browser to Manager during POST requests. The `formData` parameter is the XML form data starting at the `<AvokaSmartForm>` tag.

      > WARNING
      >
      > You should run such tests in the test environments only, such as SIT and UAT. Never run them in the production environments.

    - Obfuscate: This is the default option, so we strongly recommend keeping this option enabled, especially in the production environment. For more information, see Security Managers Overview.
    - Force Obfuscate: to obfuscate the form data regardless of the client agent type. This is important because some clients, such as Postman or embedded browsers, receive the data in clear text format, so it must be obfuscated to make it secure. | 24.10 This feature was introduced in 24.10

    This obfuscation is an additional security hardening to the standard TLS[1] transport encryption. For more information, see Data Protection.

    > NOTE
    >
    > Prior to Manager 24.10, you select the Form Data Encryption checkbox to obfuscate HTML client-side form XML data, which is the same as the Obfuscate option in the dropdown list above.

15. Type the comments relevant to this form version in the Notes field.
16. Click Upload Form or FAR File to deploy a form template file in one of the following formats: PDF, XDP[2], and FAR.

---

[1]Transport Layer Security (TLS) is a cryptographic protocol designed to provide communications security over a computer network.
[2]XML Data Package (XDP) format is a file format developed by Adobe Systems for packaging PDF data into XML files; contains the entire PDF document contents including form and template data; may also include character-encoded sections for binary content.

17. Click Upload Receipt File to deploy the receipt templates file in one of the following formats: PDF, XDP, and FAR.
18. Click Upload Signature File to deploy the template file (PDF or XDP) to be presented to users for signatures. If not specified, the receipt file is used.
19. Click Receipt XML Mapping shows how receipt XML data is mapped to AcroForm[1] fields using XQueries[2]. You can use this in overlay receipts.
20. Select the Use Delivery Receipt checkbox to specify a separate delivery of a PDF receipt where ReceiptMode is set to Delivery. This enables delivery PDF receipt to display additional Office Use Only data, which is not printed on the user's PDF copy of the receipt.
21. Select a timeout for the PDF receipt rendering from the Receipt Process Timeout dropdown list. By default, no timeout is enforced.
22. Select a delivery channel from the Production Delivery dropdown list that will be used to deliver successful submissions for this form version if the form version is not in the test mode.
23. Select a delivery channel from the Test Mode Delivery dropdown that will be used to deliver successful submissions for this form version if the form version is in the test mode.

    NOTE

    This dropdown is enabled if the form is in test mode, that is the Test Mode checkbox is selected.

24. Select a delivery channel from the Abandoned Delivery dropdown list to deliver abandoned submissions.
25. Select a delivery channel from the Validation Failure Delivery dropdown list to deliver submissions with data validation errors.
26. Click Form XML Data File to upload a form XML seed data file, containing the data structure and the XML schema, to generate a blank seed XML file to be used in the form.
27. Click Save to update the changes.
28. Click Edit Form Data Config to configure form version data mapping.
29. Click Update from Maestro to update the published form version using Maestro.

---

[1]AcroForms are the original PDF forms technology that accepts input in both Forms Data Format (FDF) and XML Forms Data Format (XFDF). Many 3rd party vendors support AcroForms.
[2]XQuery is a specification for a query language that allows a user or programmer to extract information from an Extensible Markup Language (XML) file or any collection of data that can be XML-like.

30. Click Save to update the changes.

Next, learn how to configure functions for a form version.

Create a Form Version

Manager allows you to create a new form version using its functionality. However, we recommend creating a new version of a form in Maestro and deploying it to Manager using the Import Form functionality.

Let's look at this process in more detail. Presume, you have just finished developing a Maestro form Basic Onboarding Form 1 and saved it as the initial form version `1.0-develop`. What you wish to do next is to create this form version in Manager. The easiest way to do this is to:

1. Open a form in Maestro and build the form version as a JM Form Version archive file, for example, `form-basiconboardingform1-v1.0-develop.zip`.
2. Log onto Manager and import the JM Form Version archive file - `form-basiconboardingform1-v1.0-develop.zip`.

This creates the initial form version `1.0-develop` of the Basic Onboarding Form 1 form in Manager.

Now, you want to modify the form version by adding new components and reflect these changes by creating a new form version in Manager. Here is what you should do:

1. Open the form version in Maestro and modify it as required.
2. Create a new form version to track the changes.
3. Build the form version as `form-basiconboardingform1-v1.0.1.zip`.
4. Log onto Manager and import the JM Form Version archive file - `form-basiconboardingform1-v1.0.1.zip`.

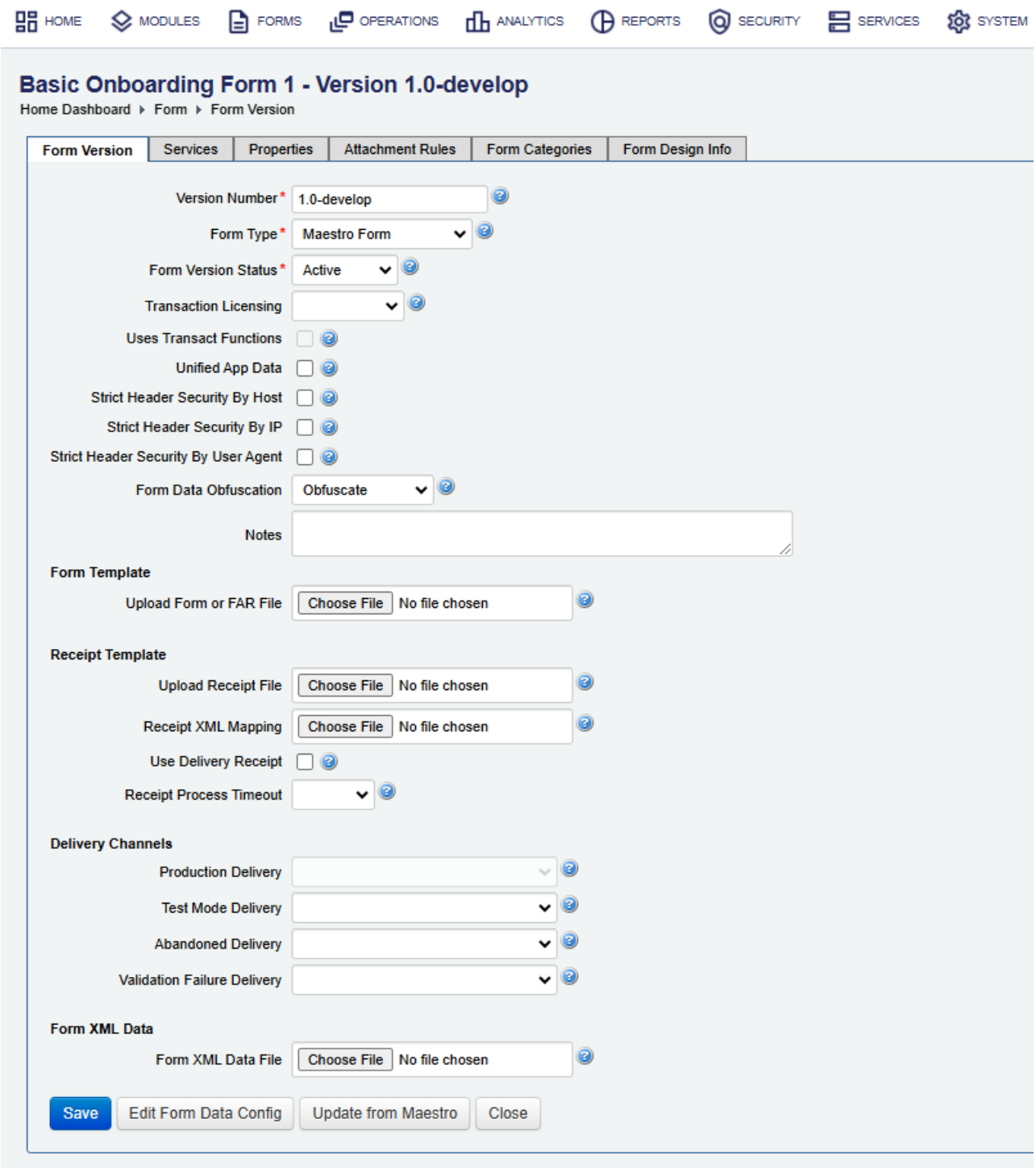
Now you have the `1.0.1` form version created in Manager.

Having said that, you can still create a new form version using Manager's functionality. One example why you would need to do so is when you wish to have a copy of an existing form version to make some small configuration changes to the form without effecting the live form version. In this case, you have to remember that this form version doesn't exist in Maestro.

To create a new form version in Manager:

1. Select Forms > Forms.
2. Locate a form and click Edit to open the form in the Dashboard.

3. Click New Form Version under the Form Version section to create a new form version.



4. Modify the form version properties as required.
5. Click Save to update the changes.

Next, learn how to configure form flow.

Make Current Form Version

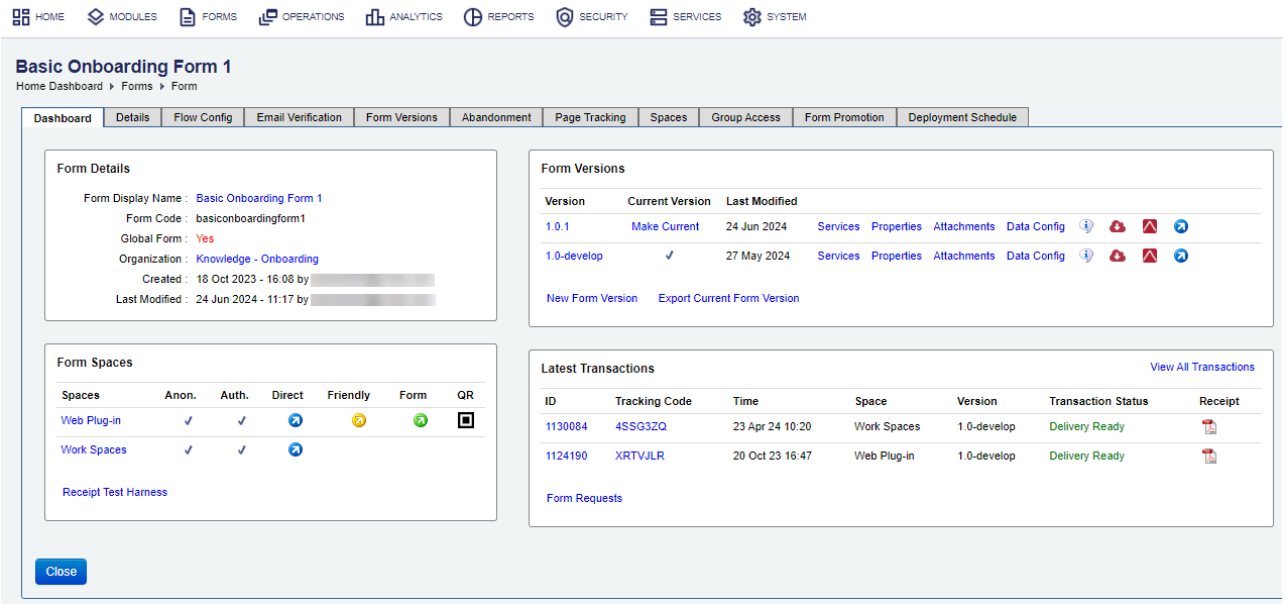
Manager allows you to specify which form version of a form should be rendered when the form is requested by a user, that is, when the user opens an application form. If a form has only one form version, it becomes the current form version by default, so this version is served when a request comes from the server. Because of this fact, you can't delete a current version of a form.

However, a form can have several form versions and you may want to change the current form version to any one that suits your needs or requirements. This is called a form version promotion and it is often used when you introduce new changes to the form or perform a rollback activity.

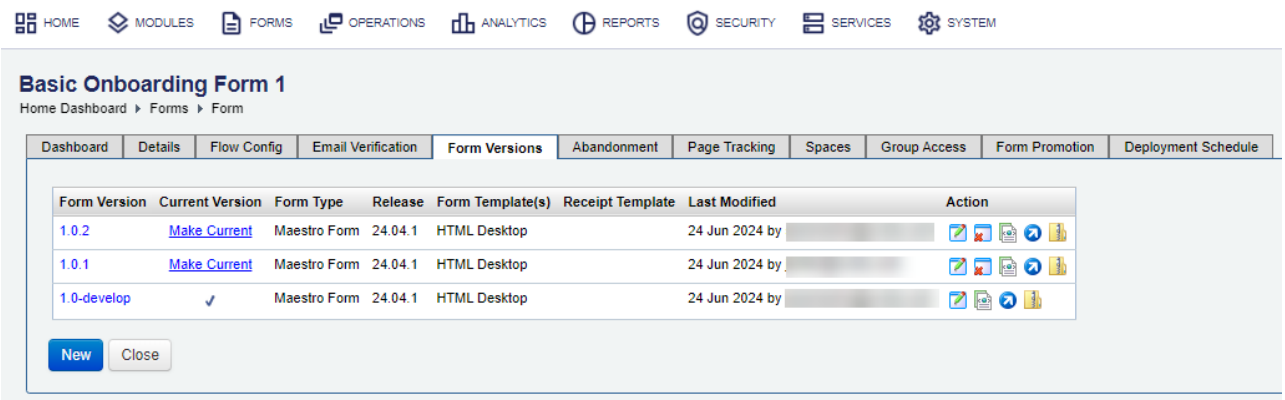
The current form version handles a request by default, so you don't need to supply a version number to render it. For example, `https://my-domain.com/web-plugin/form.htm?formCode=basiconboardingform1` will render the current version of the Basic Onboarding Form 1 application form. You can still request a specific form version by explicitly appending its number to the request. For instance, `https://my-domain.com/web-plugin/servlet/SmartForm.html?formCode=basiconboardingform1&tmFormVersion=1.0.1` will render the `1.0.1` version of the Basic Onboarding Form 1 application form.

To make a form version current:

1. Select Forms > Forms.
2. Locate a form and click Edit to open the form in the Form Dashboard. All available versions of a form are listed in the Form Versions tile.



Alternatively, select the Form Versions tab to view all available form versions.



The current form version is marked with in the Current Version column.

3. Click Make Current to nominated another form version as the current one.
4. Click OK to confirm.

Manager also allows you to change the current form version automatically at a scheduled time.

> NOTE
>
> If a form is configured for A/B Testing, it is controlled by the A/B Testing workflow rather than the current version. Only when A/B Testing is done, the current version will be enforced.

Next, learn how to configure a form version.

Configure a Form Version Metadata

Manager allows you to view and update metadata of a form version. Metadata can be used to run queries on the set of forms defined in Manager using a Web Service API. Each form version maintains its own set of organization and form metadata independently.

To add a metadata value:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select a form version.
4. Select the Form Tags tab and click New.
5. Choose a metadata tag to assign a value to from the Tag Name dropdown. If no suitable metadata tag exists in the system, you can either create one first, or leave the Tag Name dropdown blank.
6. Provide a name in the Name field.
7. Select a data type from the Data Type dropdown list.
8. Select a scope from the Scope dropdown list.
9. Set a value for this metadata for this particular form version in the Value field.
10. Click Save to update the changes. You will see that the metadata value has been added to the "Form Tags" tab.

To configure the form version metadata:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select a form version.
4. Select the Form Tags tab to view and update tags.
5. Click Save to update the changes.

Click Sync Org Tags to synchronize organization metadata values with the current values stored for the organization.

Click Validate Tags to validate the metadata values for the form version. Any errors, such as missing values for required metadata tags, will be reported so you can correct them.

Next, learn how to configure services for a form version.

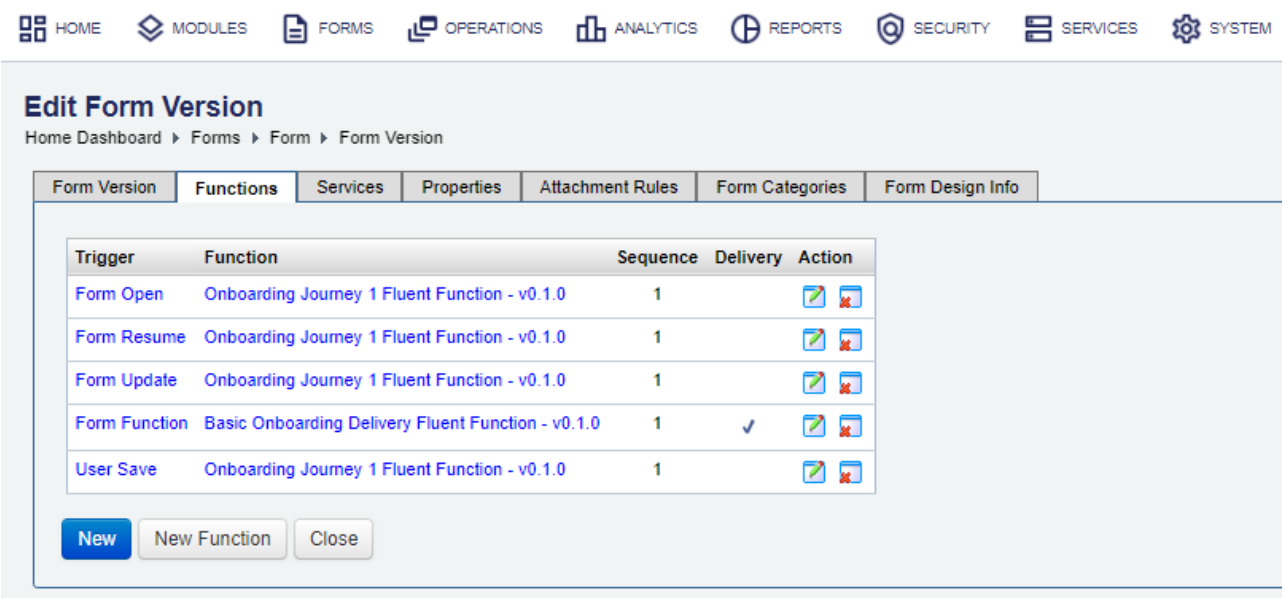
Form Version Functions

Manager allows you to view, create new and edit existing triggers to invoke form functions for each form version.

To view form functions:

1. Select Forms > Forms.
2. Locate a form and select the Functions tab to view all available triggers and form functions invoked by the triggers.

> NOTE
>
> The Functions tab is available only when the Use Transact Functions checkbox is selected from the Build Options when building a form in Maestro. If you still can't see the tab, check the Uses Transact Functions checkbox is selected. For more information, see Configure a Form Version.



The list displays the following details:

- Trigger: a name of a trigger, which can be one of the following:
  - Form Open is called when new form transaction is created by Manager, immaculately before it is rendered to the browser.
  - Form Resume is called by Manager when a saved form transaction is resumed, immediately before it is rendered to the browser.
  - Form Save Challenge is called when a save challenge is performed.

    - Form Update is called when the form makes an background update operation in a browser.
    - Form Ineligible is called when the form determines the user is ineligible to complete the application.
    - Form Function is called by Maestro form business rule scripts generally to get dynamic data from the server and perform any transaction updates.
    - User Save is called when the user explicitly saves and closes the form in a browser.
    - User Submit is called when the user explicitly submits a completed form application in a browser.
    - User Cancel is called when the user explicitly cancels and closes a form application in a browser.
    - Background Delivery is called to run a Delivery Function in a background, for example, when the user submits a form but the PDF receipt document will not be available yet. For more information, see Delivery Function.
  - Function: a name of a form function invoked by a trigger.
  - Sequence: an order that functions with the same trigger are executed.
  - Delivery: whether a form function is of the Delivery Function type or not.

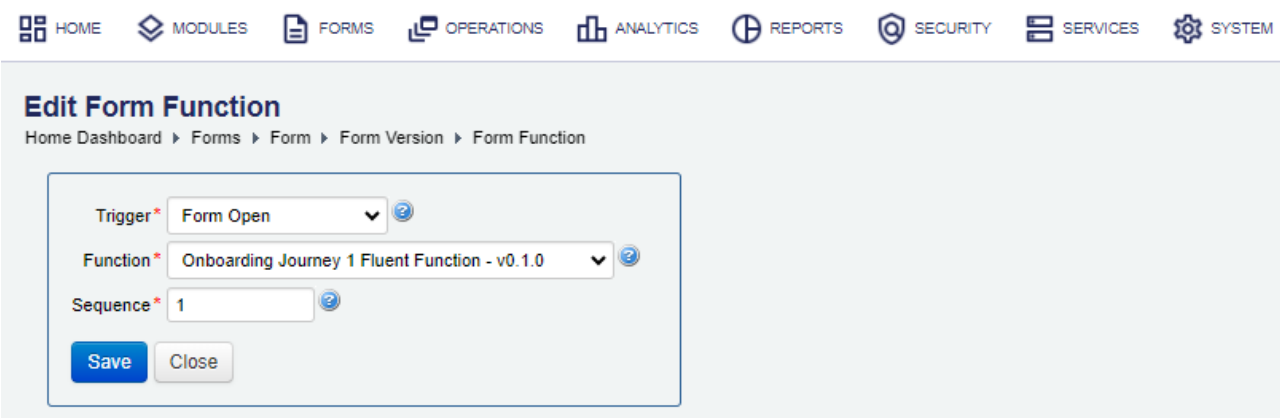
You can also perform the following actions:

- Click Edit to update a selected trigger.
- Click Edit to update a selected form function.
- Click Remove to delete a selected form function.
- Click New to create a new trigger to form function mapping.
- Click New Function to create a new Fluent function, which you can use to create a trigger to form function mapping.

Create a Trigger

To create a trigger for a form function:

1. Select Forms > Forms.
2. Locate a form and select the Functions tab
3. Click New to create a new trigger to form function mapping or click Edit to update the existing one.



4. Select a trigger from the Trigger dropdown list to define when a Transact function is called. For more information, see Function Triggers.
5. Select a Transact function, which will be invoked on the form trigger event, from the Function dropdown list.
6. Specify the order of functions with the same trigger to be called in the Sequence field.
7. Click Save to update the changes.

NOTE

You can also edit a trigger while configuring a Transact Function.

Next, learn how to configure services for a form version.

Form Version Services

Journey Manager comes with a set of standard services, invoked at different stages of form processing, which you can assign and customize for each form version. Manager uses system default services for any services that are not assigned. It is recommended to assign services to a form version only when necessary.

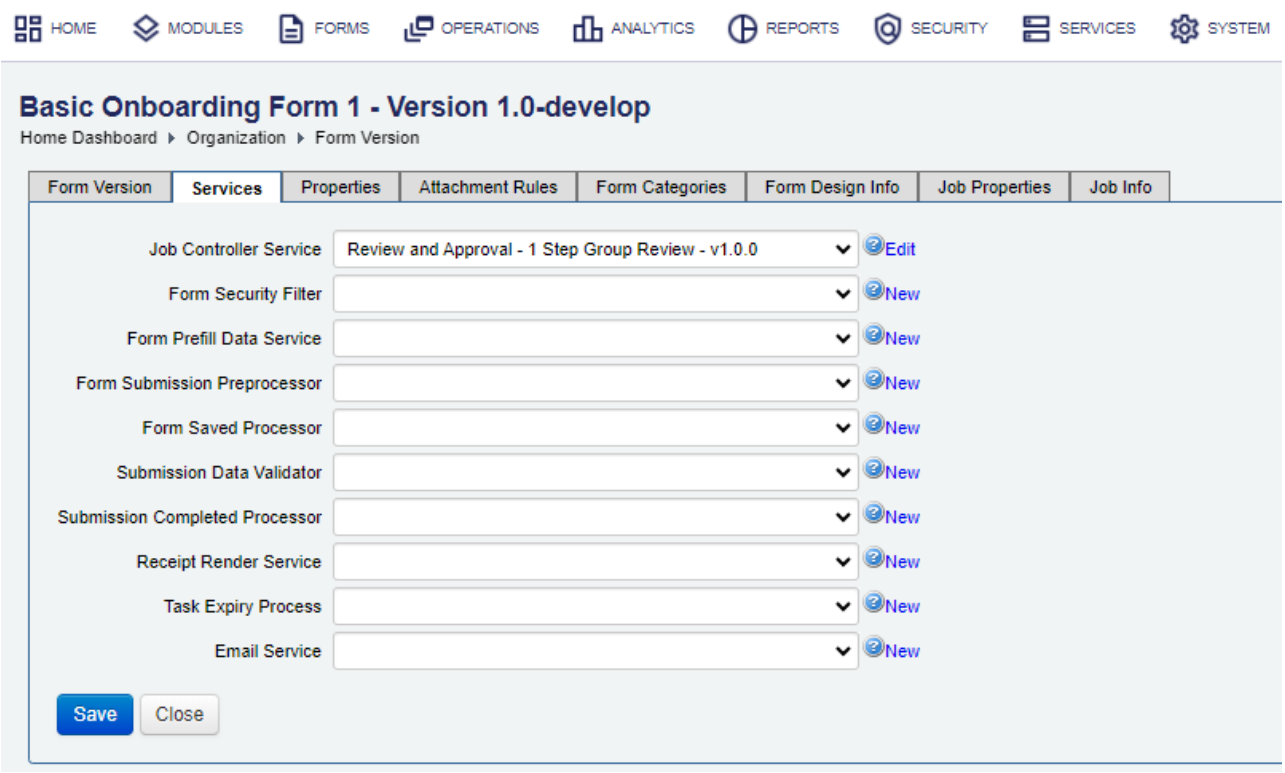
You can also create and customize optional services for a form version; for example, when a form version is configured with a collaboration job. There is no default service for an optional service when not selected. For example, if a form version uses a collaboration job, then you need to select the applicable job controller service. Otherwise, this form is a standard form and no system default service will be used for the job controller service.

Another common example where you need form version services is receipt customization, especially, the headers and footers of a receipt. For more information, see Receipt Headers and Footers Customization.

To view and assign services to a form version:

1. Select Forms > Forms.
2. Locate a form and click the Form Version tab.
3. Select a form version and click the Services tab.

> NOTE
>
> The list of available services is limited when the Use Transact Functions checkbox is selected from the Build Options in Maestro. | 17.10 This feature was introduced in 17.10.



4. For each service type, select a service from the corresponding dropdown list as required.
   - **Job Controller Service:** This service is invoked when a form submission creates a new collaboration job.
   - **Form Security Filter:** This service controls user access to the form, and is called when a form is rendered before Manager access controllers run. See also Services Guide > Form Security Filter.
   - **Form Prefill Data Service:** This service is invoked to provide XML data prefill when a form is rendered. See also .
   - **Form Render Service:** This service is called to render the form template.
   - **Form Submission Preprocessor:** This service is called when a form is submitted so the submission can be reviewed, modified or rejected before it is stored in Manager. See also Services Guide > Submission Preprocessor.
   - **Form Saved Processor:** This optional service is called when a form is saved online. See also Services Guide > Form Saved Processor.
   - **Submission Data Validator:**This service is called to validate the submitted form XML data. See also Services Guide > Submission Data Validator.
   - **Submission Completed Processor:** This service is invoked when a submission's form status changes to Completed (form submission for a collaboration job is complete) and no more user steps are required.

- **Receipt Render Service:**This service, such as the Dynamic PDF Receipt service, is called to render the form submission as a PDF receipt. See also Services Guide > Render Receipt Service.
- **Task Expiry Service:** This service is invoked when a task expires. See also Services Guide > Task Expiry Process.
- **Email Service:** This service is called to send submission related emails to users. See also Services Guide > Email Service.

5. For each pre-defined service, click New to create a new service or click Edit to configure a selected service.
6. Click Save to update the changes.

To make sure the correct services are assigned to the form version, render it by clicking Direct on the Form Dashboard.

This is especially important when you assign services to a global form version. Manager checks whether these services are allowed to be global or not. If a service is incorrectly assigned, it throws an Illegal Argument Exception (`java.lang.IllegalArgumentException`) when you try to render the form. If a service and a form are under the same organization, the form will render fine. However, if they are from different organizations, an exception is thrown.

Next, learn how to configure form version properties.

Form Version Properties

Manager allows you to create, edit and view properties of a selected form versions. You can use properties to define common reusable form elements. It can be a layout and content of a specific email message that include any required dynamic data, such as user and form details. In some cases, you need to write a script to make use of these elements. Properties are commonly used to prefill data of form versions dynamically so you don't need to update a form when data changes.

Another common example where you need form version properties is receipt customization, especially, the headers and footers of a receipt. For more information, see Receipt Headers and Footers Customization.

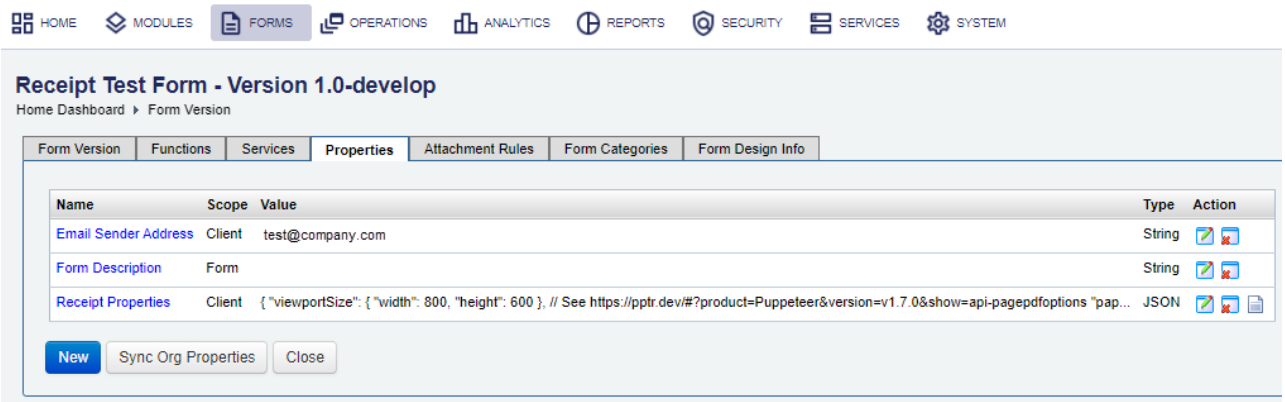
When creating a form version, Manager includes a standard set of properties for it with already prefilled default values. Form version's properties allow you to create properties to override the existing form space's and organization's properties.

Each form version maintains its own set of organization and form properties independently. This becomes important when you configure a form flow using a non-current form version as you must explicitly specify a form version number for the correct set of properties to be used. For example, `${submission.portal.contextPath}submission-saved.htm?submitKey=${submission.submitKey}&tmFormVersion=${submission.version.versionNumber}`. Otherwise, the current form version's properties is used.

View Form Version Properties

To view form version properties:

1. Select Forms > Forms.
2. Locate a form version and click Edit.
3. Select the Properties tab to see the following details.



- Name - a name of the property, which is based on a property type of this property
- Scope - a scope of the property, which can be one of the following:
  - Client - a property configured for an organization and its forms, so it obtains its value dynamically from an organization.
  - Form - a property configured for form versions, so it obtains its value directly or dynamically from a form version. This property can override Client's scope property.
  - User - a property configured for user profiles, so it obtains the value from a user's profile of a logged user.
- Value - a value of the property
- Type - a data type of the property.

You can also perform the following actions:

- Click New to create a new property.
- Click Edit to modify an existing property.
- Click Remove to remove an existing property.
- Click Sync Org Properties to synchronize a form version with the organization properties. Manager refreshes all form version properties customized where:
  - The Property Scope is Client
  - There exists a customized property of the same name for the organization of the form

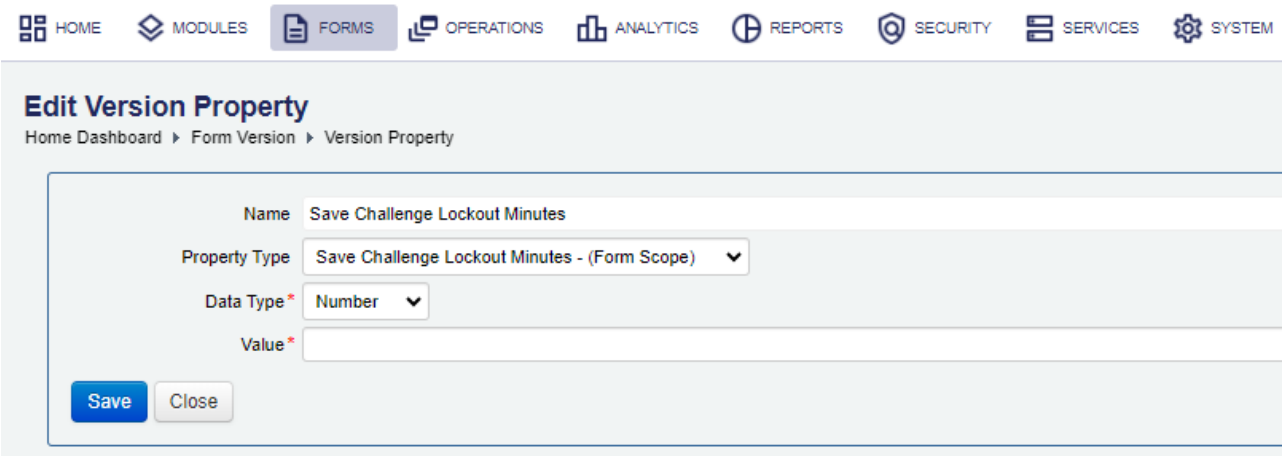
> NOTE
>
> Before proceeding with the Sync Org Properties, it's recommended backing them up, because the sync is done immediately without requesting any confirmation.

Create a Form Version Property

To create a new form version property:

1. Select Forms > Forms.
2. Locate a form version and click Edit.

3. Select the Properties tab and click New.



4. Select a desired property type from the Property Type dropdown list, which populate the Name, Data Type, and the Value fields. You can leave the property type blank when no suitable type can be found in the dropdown list. Alternatively, you may also add a new property type and then use it here.

   NOTE

   If a property of a desired property type is already defined for this form version, this property type is not shown in the dropdown list, so you can't create another property of the same property type.

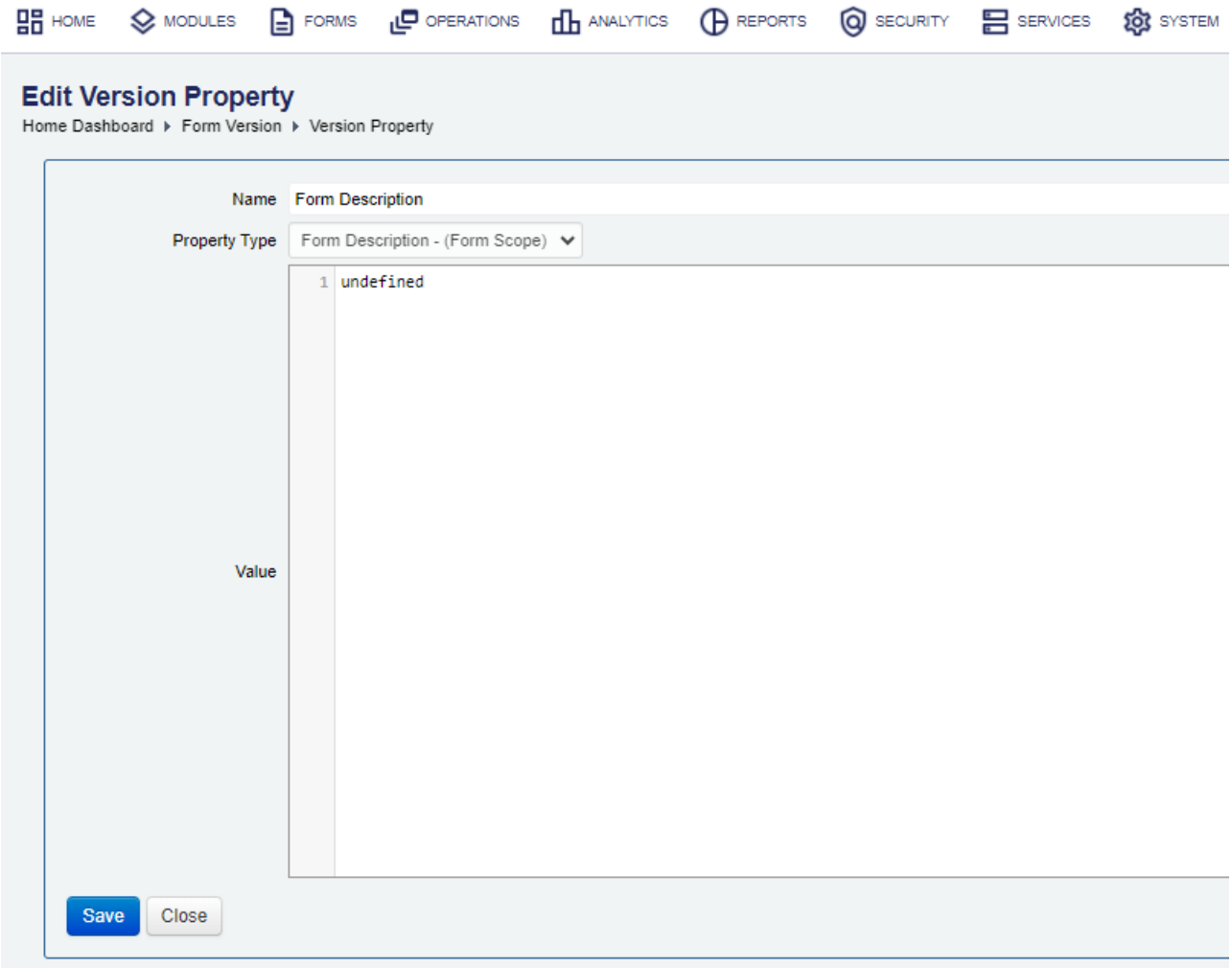
5. Provide a property name in the Name field, if the property doesn't have any property type, that is the Property Type dropdown is left blank. Otherwise, the name is populated based on the selected property type below.
6. Select a data type from the Data Type dropdown list. The field is populated depending on the property type you select above. However, you have to select an appropriate data type when the property type was left blank.
7. Provide or update a value in the Value field, which is populated based on the property type you selected. It coan be a file upload field, depending on the property type.
8. Click Save to update the changes.

Edit a Form Version Property

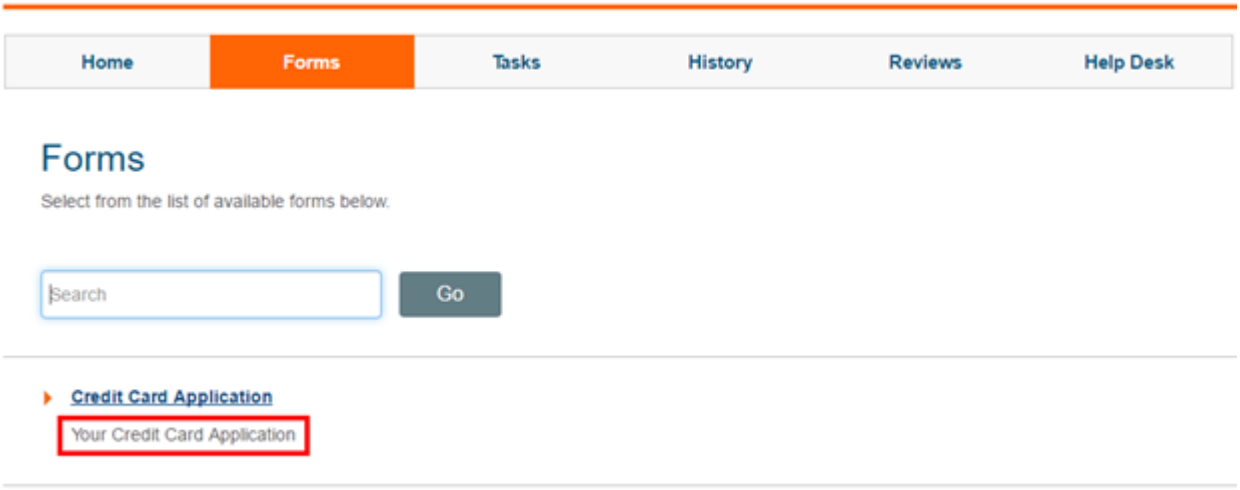
To edit a form version property:

1. Select Forms > Forms.
2. Locate a form version and click Edit.

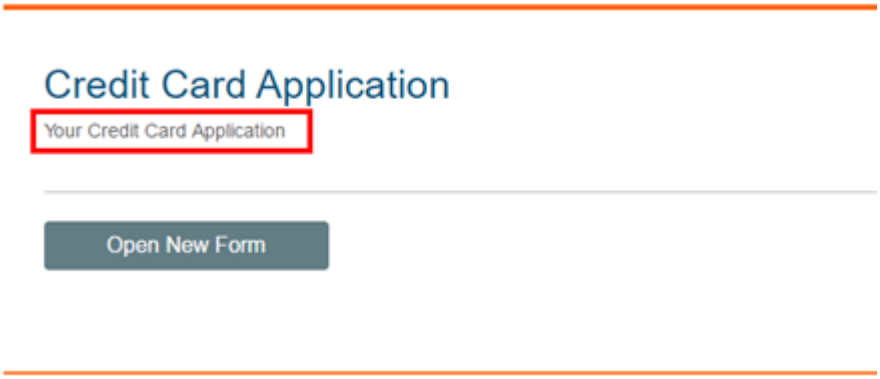
3. Select the Properties tab.
4. Locate a form version property, for example the Form Description, and click Edit..



5. Update the property with a new description for your form in the Value field to assist form users in choosing the correct form, the form description is displayed in the list of forms a user can view in the form space, as well as, on the form's landing page.
6. Click Save to update the changes.
7. Check the list of forms in the form space, with the form description that was entered in the form version property:



Then click the form to see the form's landing page, with the form description that was entered in the form version property, as shown below:



Override a Form Version Property

You may want to customize a form version property, for example, the form receipt email message by overriding the default Email Form Receipt Message property, which was initially created for a form space, as shown below:

Work Space - Property Email Form Receipt Message

Home Dashboard ▸ Form Spaces ▸ Module ▸ Portal Property

Name: Email Form Receipt Message

Value:

```
<p> Thank you for submitting ${form.formName} at ${portal.name}. </p>
<p> Your form reference code is: <b> ${submission.trackingCode} </b> </p>
#if ($hasReceiptAttachment)
<p> Please see the attached PDF for a copy of your form submission. </p>
#end
#if ($jobTasks && !$jobTasks.isEmpty())
  <h1>Next Steps</h1>
  <p> You are required to complete these additional forms to finalize your request. </p>
  <ul>
  #foreach ($task in $jobTasks)
  <li>
    #if ($task.isJobFormEditable() || !$task.isFormCompleted())
      #if ($task.taskSubject)
      <a href="${task.portal.contextPath}form.htm?submitKey=${task.submitKey}">${task.taskSubject}</a>
#if($task.isFormCompleted())- (Completed) #end
      #else
      <a href="${task.portal.contextPath}form.htm?submitKey=${task.submitKey}">${task.formName}</a>
#if($task.isFormCompleted())- (Completed) #end
      #end
    #else
      #if ($task.taskSubject)
        $task.taskSubject - (Completed)
      #else
        $task.formName - (Completed)
      #end
    #end
  </li>
  #end
  </ul>
#end
<!-- ${environmentName} -->
```

Save Close

Let's suppose you need to add contact details to the message. You locate the existing message and edit its value preserving the HTML format for the message as shown below.

Edit Version Property

Home Dashboard ▸ Forms ▸ Form ▸ Form Version ▸ Version Property

Name: Email Form Receipt Message

Property Type: Email Form Receipt Message - (Client Scope)

Data Type*: HTML

Value:

```
<span/>
<p> You recently submitted a ${form.formName} at ${portal.name}. </p>
<p> Your form reference code is: ${submission.trackingNumber} </p>
<p> For Customer Service Contact  Mon-Fri Office Hours </p>
#if ($hasReceiptAttachment)
<p> Please see the attached PDF for a copy of your form submission. </p>
#end
#if ($hasReceiptLink)
<p> To access your PDF receipt please: </p>
<p> 1.   Copy the URL below into a web browser: </p>
<p> <b style="color:navy"> ${receiptUrl} </b> </p>
<p> 2.   Enter in your Reference Code:    <b> ${submission.trackingCode} </b> </p>
#end
#if ($jobTasks && !$jobTasks.isEmpty())
<h1>Next Steps</h1>
<p> You are required to complete these additional forms to finalize your request. </p>
<ul>
#foreach ($task in $jobTasks)
<li>
#if ($task.isJobFormEditable() || !$task.isFormCompleted())
#if ($task.taskSubject)
<a href="${task.portal.contextPath}form.htm?submitKey=${task.submitKey}">${task.taskSubject}</a> #if($task.isFormCompleted())- (Completed) #end
#else
<a href="${task.portal.contextPath}form.htm?submitKey=${task.submitKey}">${task.formName}</a> #if($task.isFormCompleted())- (Completed) #end
#end
#else
#if ($task.taskSubject)
$task.taskSubject - (Completed)
#else
$task.formName - (Completed)
#end
#end
</li>
#end
</ul>
#end
<!-- ${environmentName} -->
```

add contact details

Save | Close

Now, when a form user submits this form, an email message will include the customer service contact details. However, this override doesn't affect the email receipt messages for any other form.

CREDIT CARD APPLICATION CONFIRMATION

You recently submitted a Credit Card Application at Work Space.

Your form reference code is: GLR443D

For Customer Service Contact Mon-Fri Office Hours

NOTE

Manager allows you to create a broad range of properties, not only form version specific ones, so that you can customize the whole form application including services and service connections, to name a few. For more information, see properties.

Next, learn how to configure form version attachment rules.

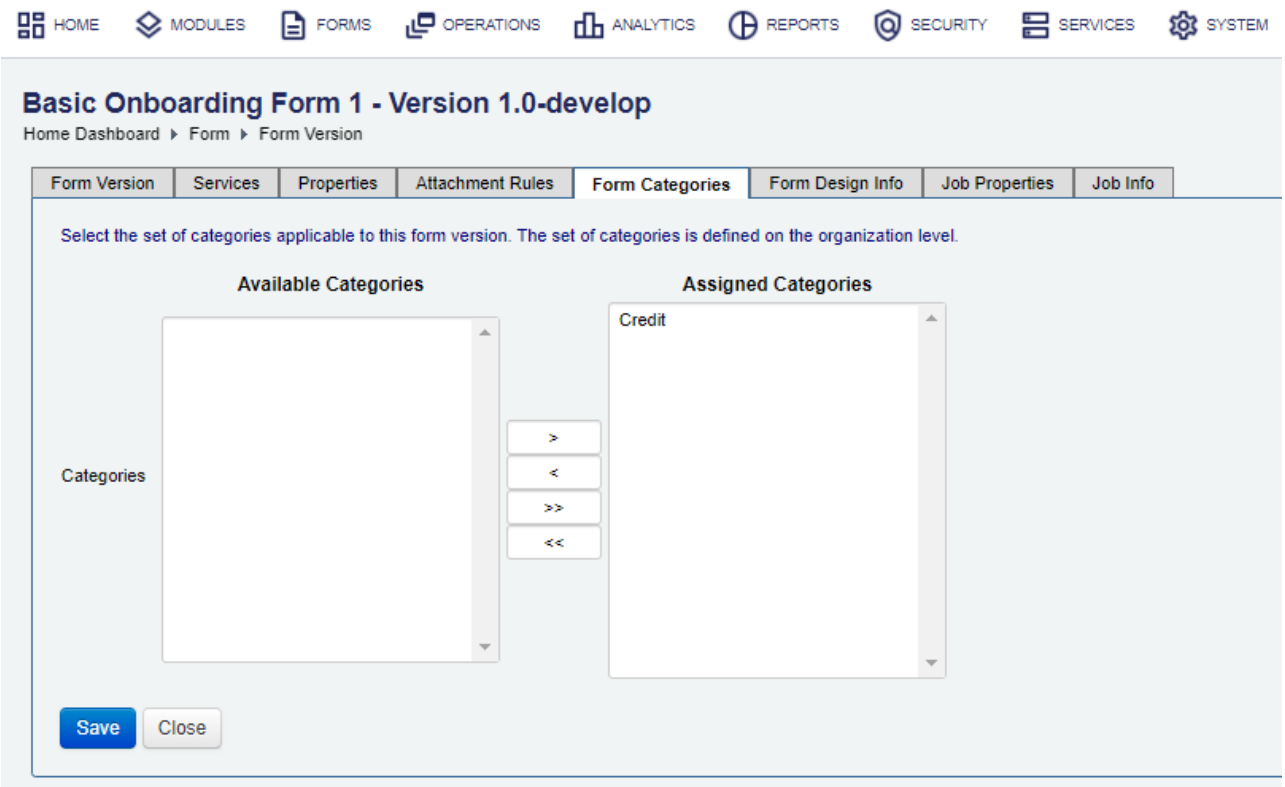
Form Categories

Manager allows you to assign forms to one or more form categories, for example, Business or Credit, to enable form space users to search forms by categories. However, the form space must support it. Form categories are especially useful if an organization has many forms.

> NOTE
>
> You must configure form categories on the organization level before you can assign them to your form versions.

To assign a form version to an organization's form category:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select a form version.
4. Select the Form Categories tab.
5. Select one of the available categories from the Available Categories and then click > to move them to the Assigned Categories.



6. Click Save to update the changes.

Now you can filter the form by categories. When at least one form version is assigned to a form category, the Category dropdown list is displayed in the list of form versions and can be used to filter the list of forms by a form category.

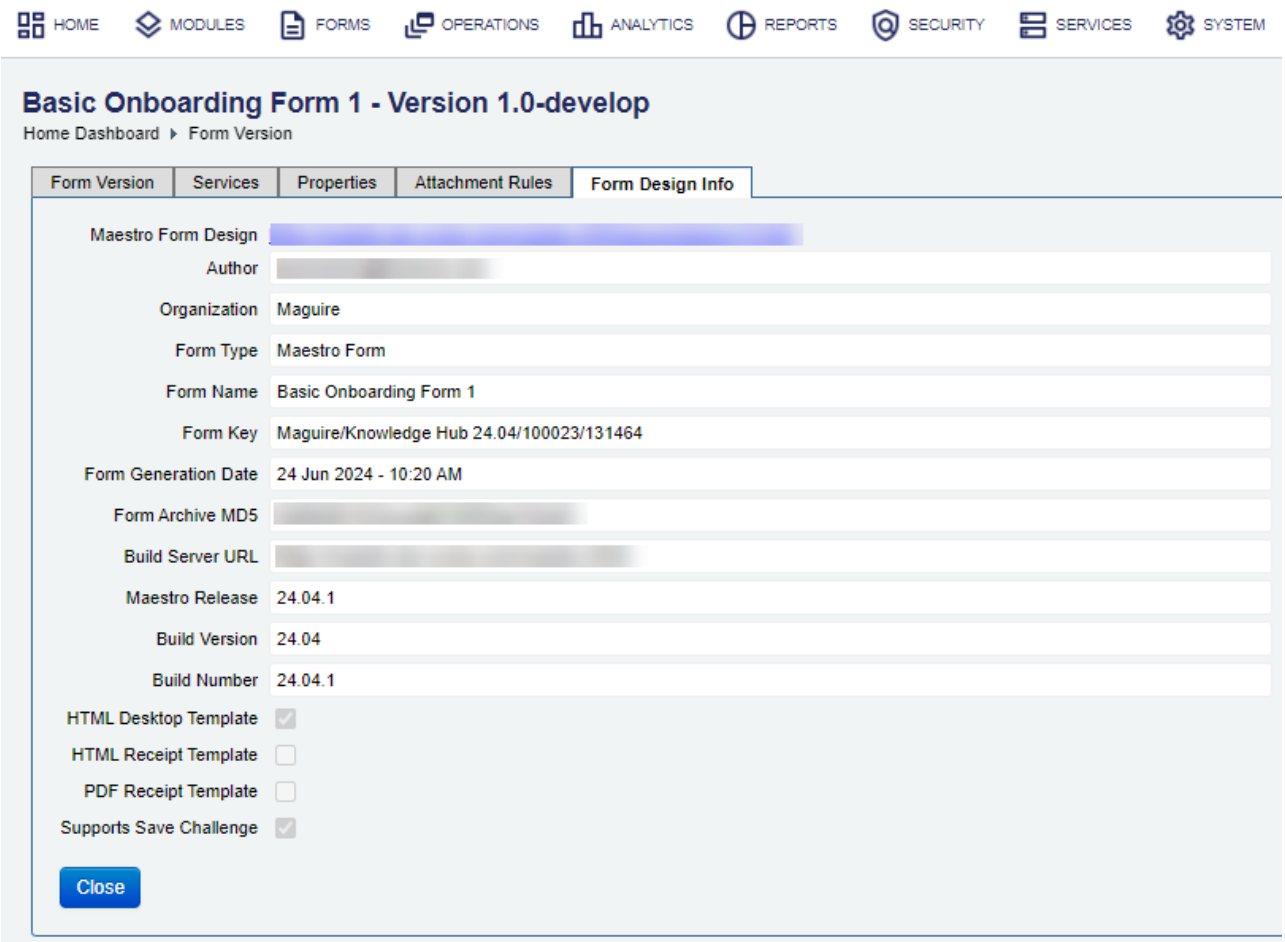
Next, learn about form design info.

Form Design Info

Manager allows you to view key information about a form design, which is useful when you want to develop and build a form version continuously. You can navigate back to the originating form version in Maestro using the Maestro Form Design url to update form elements.

To view form design information of a form version:

1. Select Forms > Forms.
2. Locate a form version.
3. Click Form Design Info to see the following details.



The list displays the following details:

- Maestro Form Design: a direct link to the corresponding form version in Maestro. Click the link to open the form version in the Maestro editor. If you are not logged in to Maestro, you will be prompted to enter your credentials.
- Author: a name or email address of a user who created the form version.
- Organization: an organization that the form belongs to.

- Form Type: the type of form created, such as Maestro or Composer.
- Form Name: a name of the form.
- Form Key: a unique key of the form.
- Form Generation Date: the date when the form was generated (built).
- Form Archive MD5: the MD5 checksum of the form archive.
- Build Server URL: the URL of the server used to build the form.
- Maestro Release: the Maestro release number that was used to build the form.
- Build Version: the version of the core Maestro platform used to build the form. A build version that is different than the Maestro Release version means that the form has not been upgraded to the new release and hence has been built to be compatible with the older version.
- Build Number: the number of the build used to build the form version.
- Build Environment - the environment used to build the form. For Maestro it is normally local,but for Composer can reflect the test and staging environments.
- HTML Desktop Template: indicates whether this form version has an HTML desktop template.
- HTML Receipt Template: indicates whether this form version has an HTML receipt template. This is for Composer use only.
- PDF Receipt Template: indicates whether this form version has a PDF receipt template. This is for Composer use only.
- Supports Save Challenge: indicates whether this form version supports the Save Challenge feature. If this option is selected, Manager takes a user, who is trying to resume a form application, to a form page with the Save Challenge dialog. Otherwise, it redirects a user to the `web-plugin/save-challenge.htm` form space page.

4. Click Close.

> NOTE
>
> For Composer forms, the details displayed on the Form Design Info tab are slightly different. These variations in the fields are applicable to Composer features including legacy ones.

Next, learn about form version details.

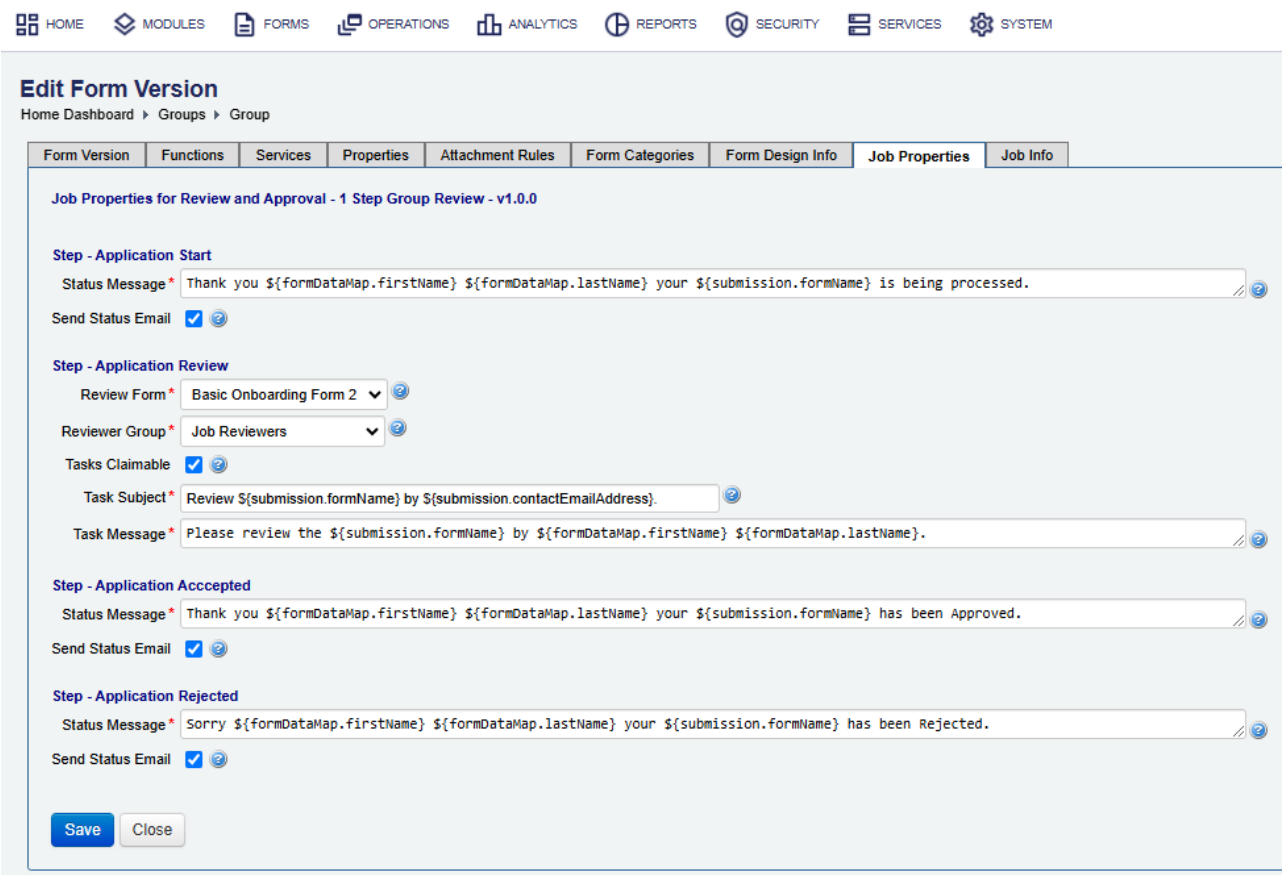
Job Properties of a Form Version

Journey Manager allows you to assign a job controller service to a form version so it becomes a part of a collaboration job. Then, you can configure job properties of this form so they are used by the collaboration job.

> NOTE
>
> The Job Properties tab is only available to a form version that is associated with a review and approval collaboration job. It isn't available to an onboarding collaboration job.

To configure collaboration job's properties:

1. Locate a form version.
2. Select the Job Properties tab to customize the job controller of a collaboration job. The set of parameters you can configure depends on the collaboration job associated with this form version. These parameters are grouped by each collaboration job's step.



3. Enter the message that will be sent to the applicant to update the status of their application in the Status Message field.

4. Select the Send Status Email checkbox to send an email to an applicant with the status update of a form application.
5. Select the form which will be used to review the application from the Review Form drop-down menu.
6. Select the staff group that will review the application from the Reviewer Group dropdown.
7. Select the Tasks Claimable checkbox to allow group reviewers to claim review tasks.
8. Enter the subject line for the review task email in the Task Subject.
9. Enter the detailed message for the review task email in the Task Message.
10. Click Save to update the changes.

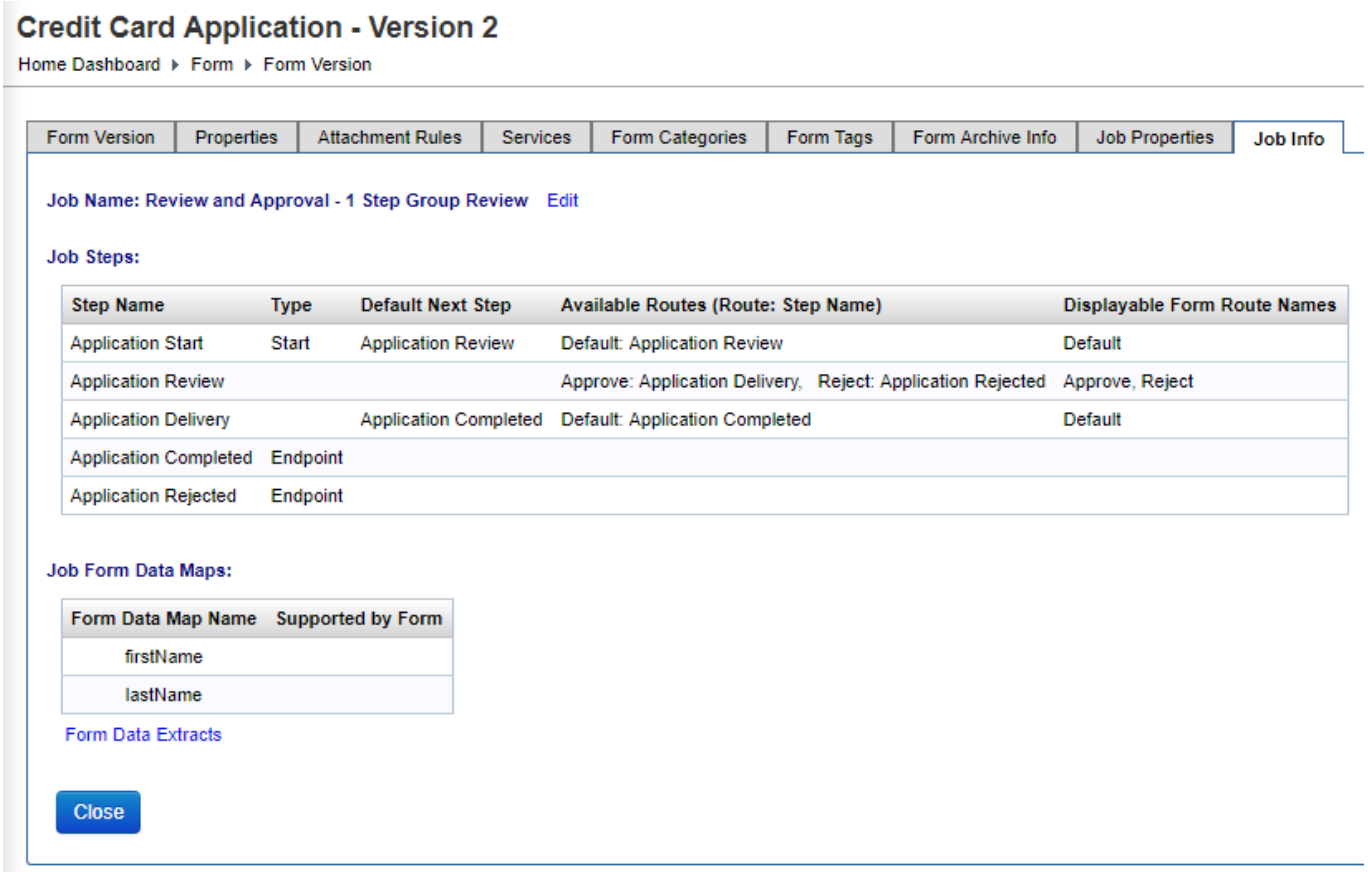
Next, learn about form version job info.

Job Information of a Form Version

Manager allows you to assigned a job controller service to a form version so the form becomes a part of a collaboration job. You can view collaboration job information related to this form version in the Job Info tab.

> NOTE
>
> The Job Info tab is only visible if a job controller service is configured for a form version.

To view collaboration job information for a form version, select the form version and click the Job Info tab.



Next, learn about form version job properties.

Delete a Form Version

Before deleting a form version in Manager, consider if it is necessary to do so. When you delete a form version, all its related entities including submissions are also deleted.

Instead of deleting a form version, you may wish to:

- Leave the form version and assign another form version as the current version.
- Retire the entire form by deactivating it.

> NOTE
>
> You can't delete the current version of a form.

The Temenos Journey Manager platform follows open source semantic versioning conventions, that is, all new form version names should start at the version `1.0-develop` until they reach a production release version.

To delete a form version:

1. Select Forms > Forms.
2. Locate a form and select **Form Version** tab.
3. Check if the form version you wish to delete is the current form version - the Current Version column is selected. If it is so, reassign the correct form version as the current form version by clicking on its Make Current link.
4. Click Delete.
5. Click OK to confirm your decision.

Next, learn how to delete Maestro form versions.

Delivery Channels Overview

With very few exceptions, Managerr is implemented as a hosted solution. It is a Platform as a Service (PaaS) solution, which means that the data collected do not reside in the client's database while applications are in progress. The Manager server lives in Temenos's cloud subscription – not the client's. For that reason, the last step of most Manager jobs is the all-important delivery step in which everything about the completed application is zipped up into a file and delivered to the client, usually by means of secure file transfer. It is critical that everything goes well with the delivery step, since the client needs to have the detailed record of the application for compliance purposes including all of the customer's sign-offs on disclosures, terms, and conditions. For this reason, Manager provides a robust delivery framework that includes retry capabilities and notifications should delivery fail even after repeated attempts.

Once a form is submitted and ready for further processing, the organization needs to be notified so they can take appropriate action. Journey Manager allows you to set up automatic email, web service or REST delivery on a per-form basis, with separate settings for test and production mode. A delivery channel sends user form submissions to a platform backend or third-party system for processing. After a user submits a form, a delivery channel determines what happens to the submitted, abandoned and invalid form transactions.

A delivery channel utilizes one of the following mechanisms:

- PDF receipt: a read-only copy of a form with user details in PDF format.
- Form XML data: a machine-readable copy form's filled details in XML[1] format.
- Attachment: a file which is a part of a form submission, for example, a copy of a driver license.

You can configure different delivery channels for:

- An organization: this is the organization level when you configure the organization-wide delivery channels, which are used by all forms and form versions if their delivery channels are not configured
- A form: this is the form level when you configure a delivery channel, different to the organization's one, for individual forms. This gives you an extra flexibility to have different delivery flows for various forms within one organization.

---

[1]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

- A form version: this is the form version level when you configure a delivery channel, different to the form's one, for individual form versions. This means you can implement requirements when each form version of a form should use various delivery channels. | 21.05 This feature was introduced in 21.05..

Manager determines a delivery channel to use in the following order:

1. The form version's delivery channel.
2. The form's delivery channel, if the above is not set.
3. The organization's delivery channel, if the above is not set.
4. The Trash Can delivery process, if the above is not set.

> NOTE
>
> Manager delivers a form submission only if all attachments are scanned for viruses and marked as free of viruses or malicious software.

A delivery channel can be one of the following:

- Delivery Process
- Email
- Email Secure
- REST Service
- Web Service
- Trash Can

Let's look at each method in details.

Delivery Process

The Delivery Process is the most widely used delivery method because it's highly customizable and secure way for immediate submission data transfer. A Delivery Process invokes a Groovy service to perform a delivery of submission details. A delivery is marked as complete if the service indicates that delivery has been successful.
If delivery has not been completed within the default 72 hours of a form been submitted, escalation emails are sent to all global and organization administrators in the Receive Delivery Escalation Alerts group.

Email

The Email method sends the submission details as attachments to one or more configured email addresses. A delivery is marked as complete as soon as the email is sent. The email can include a link to the submission attachments, if they were added by a user.

Email Secure

The Email Secure method differs from Email in that the email sent does not contain any submission details. Instead, it contains a link to a secure page in Manager where the submission details can be downloaded.

A delivery is marked complete only after a user has accessed this secure page and confirmed that the submission has been received.

If delivery has not been acknowledged within a certain period, which is 24 hours by default, reminder notifications are sent to the recipients defined in the delivery channel. This continues daily until a recipient acknowledges it.

If the delivery channel's Email Addresses field is left blank, a reminder is sent to all global and organization administrators in the Receive Delivery Escalation Alerts group.

If delivery remains unacknowledged within the default 72 hours of a form been submitted, escalation emails are sent out to all global and matching organization administrators in the Receive Delivery Escalations Alerts group.

REST Service

The REST Service method stores the submission details within Manager and offers a secure method of the enterprise retrieving these submission details, using a REST endpoint, when convenient.

For more information, see the services guide.

Web Service

The Web Service method notifies your enterprise by making a web service call, using a pre-defined API. For more information, see the Manager Web Service API documentation.

Web services may be either defined as part of the Integration Agent[1] or defined separately by a developer.

Delivery is marked as complete once the web service client confirms that it has received the submission.

If delivery has not been completed within the default 72 hours of a form been submitted, escalation emails are sent out to all global and organization administrators in the Receive Delivery Escalation Alerts group.

Trash Can

The Trash Can Delivery process is a do-nothing delivery process for form testing. This is usually configured as the default and won't be available in the organization configuration,

[1]Transact Integration Agent is a software appliance hosted inside the client organization's data center, which receives form submission deliveries and file attachments from the Journey Manager server.

but is available in the form configuration. The Trash Can Delivery process is essential in ensuring that every form has a delivery channel set up so its submissions and transactions can be purged from the system by Data Retention Management service.

In a conclusion, Journey Manager's job delivery framework is a powerful tool that ensures all the critical data collected during an application process is securely and reliably transferred to the client. By following the steps outlined above, you can set up a robust delivery process that includes retry mechanisms and notifications to handle any potential issues. Whether you're a developer or an IT administrator, understanding how to configure and use job delivery is essential for maintaining compliance and ensuring the smooth operation of your systems.

Manager automatically delivers submissions according to the delivery channel currently configured for the form, and the organization default delivery channel if the form is not associated with a delivery channel, and will automatically retry delivery if failures occur depending on the delivery method used. However, you can monitor submission delivery and change the status for a set of submissions, for example, to retry delivery after a prolonged outage.

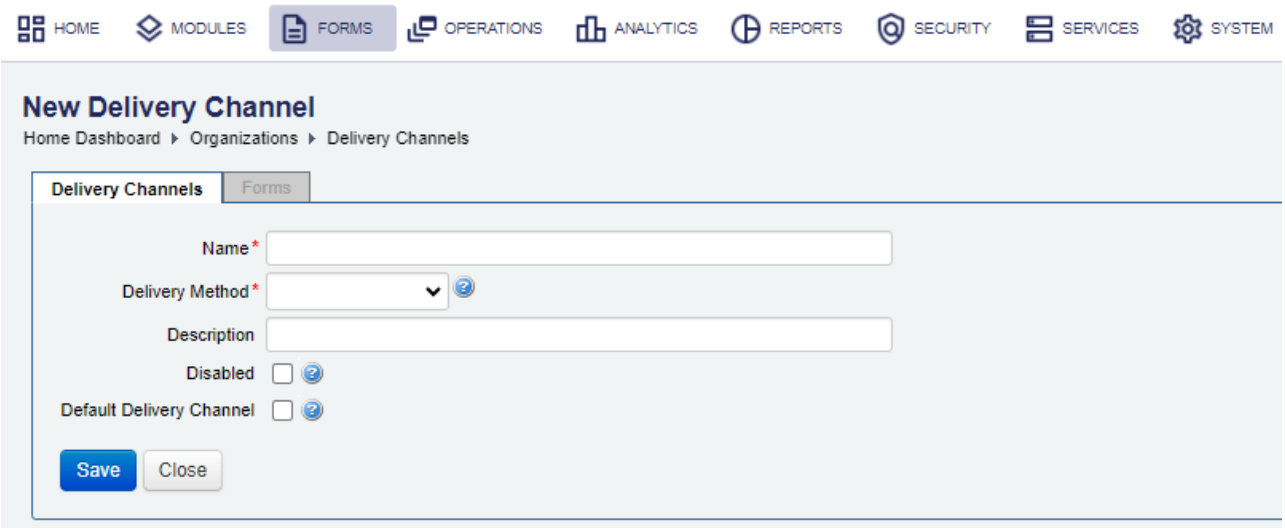
Next, learn how to create a delivery channel.

Create a Delivery Channel

Manager allows you to create and configure a delivery channel for an organization.

To create a delivery channel:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Delivery Channels tab and click New to create a new channel.



4. Enter a unique name, within your organization, of the delivery channel in the Name field.
5. Select the delivery method from the Delivery Method dropdown list, which can be one of the following:
   - Delivery Process
   - Email
   - Email Secure
   - Web Service
   - REST Service

> NOTE
>
> A delivery channel configuration is different for each delivery method.

> NOTE
>
> The default Trash Can Delivery Process is set by default, so collaboration jobs can run. However, you may want to choose user email delivery.

6. Enter a description of the delivery channel in the Description field.
7. Select the Disabled checkbox to disable the delivery channel temporarily.
8. Select the Default Delivery Channel checkbox to make it the default delivery channel for any forms within this organization that have not yet been associated with a delivery channel.
9. Click Save to update the changes.

Now, you can use this delivery channel in your applications. If you want different forms to use various delivery channels, you must assign the delivery channel to each form.

Next, learn how to create a Delivery Process delivery channel.

Create a Delivery Process Channel

Manager allows you to create and configure a delivery channel as a Delivery Process for an organization.

A delivery process can be implemented as a Job Delivery service, which is a delivery that takes place asynchronously at the completion of a collaboration job. A collaboration job is the set of steps that take place after an applicant submits a completed form. These steps can be automated or manual and they can happen in Manager or in another system. The steps can even involve the completion of additional forms by extra participants like a joint applicant or internal reviewer. In the end, all the completed steps and submitted forms make up the finished job.

The last step of most Manager jobs is the all-important delivery step in which everything about the completed application is zipped up into a file and delivered to the client, usually by means of secure file transfer. What our client does with the delivered ZIP file is up to them. In most cases, clients save the delivered file in an indexed document repository for safekeeping. They may also pull out data from the package and store them in a data warehouse for analytics and reporting.

To set up a job delivery, you need to have:

1. A delivery process.
2. A delivery channel that invokes the delivery process when required.
3. A job delivery service that references the delivery channel.
4. A collaboration job that includes a job delivery step referencing your job delivery.
5. A form that initiates the collaboration job when submitted. Collaboration jobs don't just run by themselves, so you need to configure a form with a correct Job Controller Service, such as the Onboarding Job Controller, to kick off the job when the form is submitted.

Let's look at each of these steps in more detail.

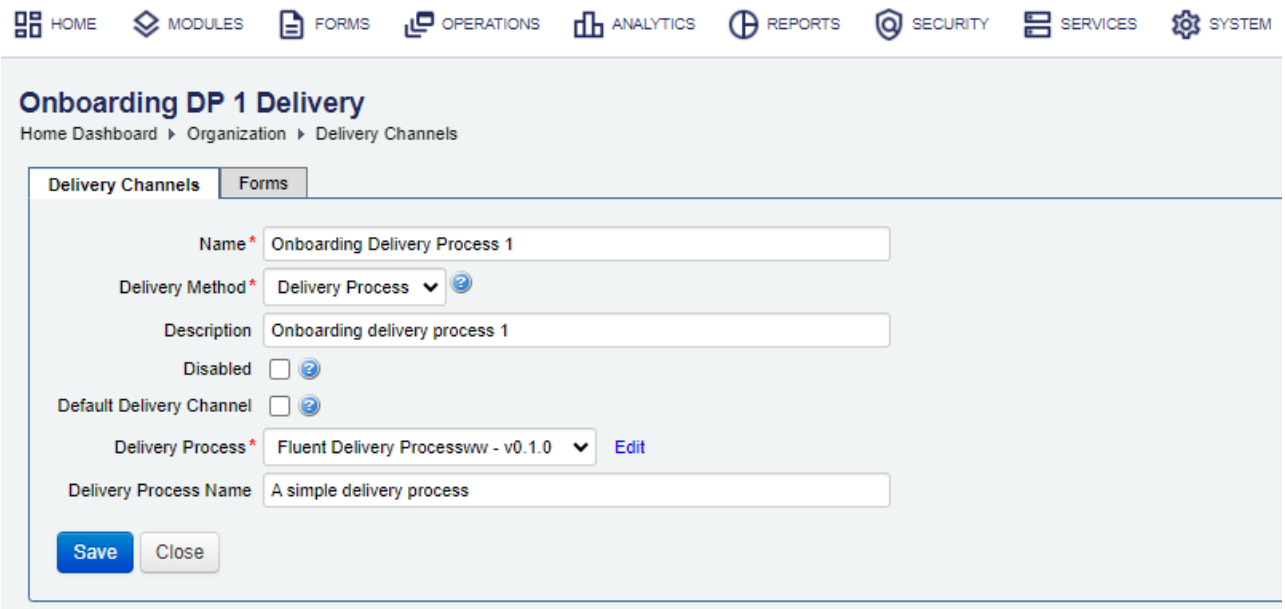
To create a delivery process:

1. Create a Groovy service to bundle up the bits of the job that you want to include in the delivery package. These may include receipts, form data, transaction properties, job properties, attachments, or history of the job and the transactions within it. Since packaging and delivering the ZIP file can take a while, delivery processes are asynchronous so that the applicant or reviewer isn't required to wait around while it happens.
2. Configure the service to specify the maximum number of retries before it finally gives up and marks the job as undeliverable. You can also specify maximum retries for your checkpoints. For more information, see Fluent Delivery Process.

Now, you have a new delivery process, but how do you invoke it? Well, you don't. Instead, you set up a delivery channel and reference your delivery process there so that the delivery channel can invoke it when the time is right to do so.

To create a delivery channel to invoke a delivery process:

1. Create a delivery channel using the Delivery Process as a delivery method.



2. Select a service to perform a delivery from the Delivery Process dropdown list or click New to create a new service.

> NOTE
>
> If you create a new Delivery Channel of the type Groovy Delivery Process you have the option to set it up with a default template for a Fluent, Fluent with Checkpoints, or a Groovy Delivery process. The actual service code can then be updated to suit the clients needs.

> NOTE
>
> Use a built-in delivery process called Trash Can Delivery Process, which dumps a transaction data without any further actions. Often, this is set as the default delivery channel, so any forms that don't specify a delivery channel will use it.

3. Enter a name of an orchestration to be invoked in the Delivery Process Name field.

4. Select the Deliver Submitted PDF checkbox to include the submitted PDF in editable form in the delivery. | 19.11 This feature was removed in 19.11
5. Click Save to update the changes.

When you have the delivery channel ready, you need to create a job delivery service which will be called as the final step of a collaboration job used in an onboarding journey application.

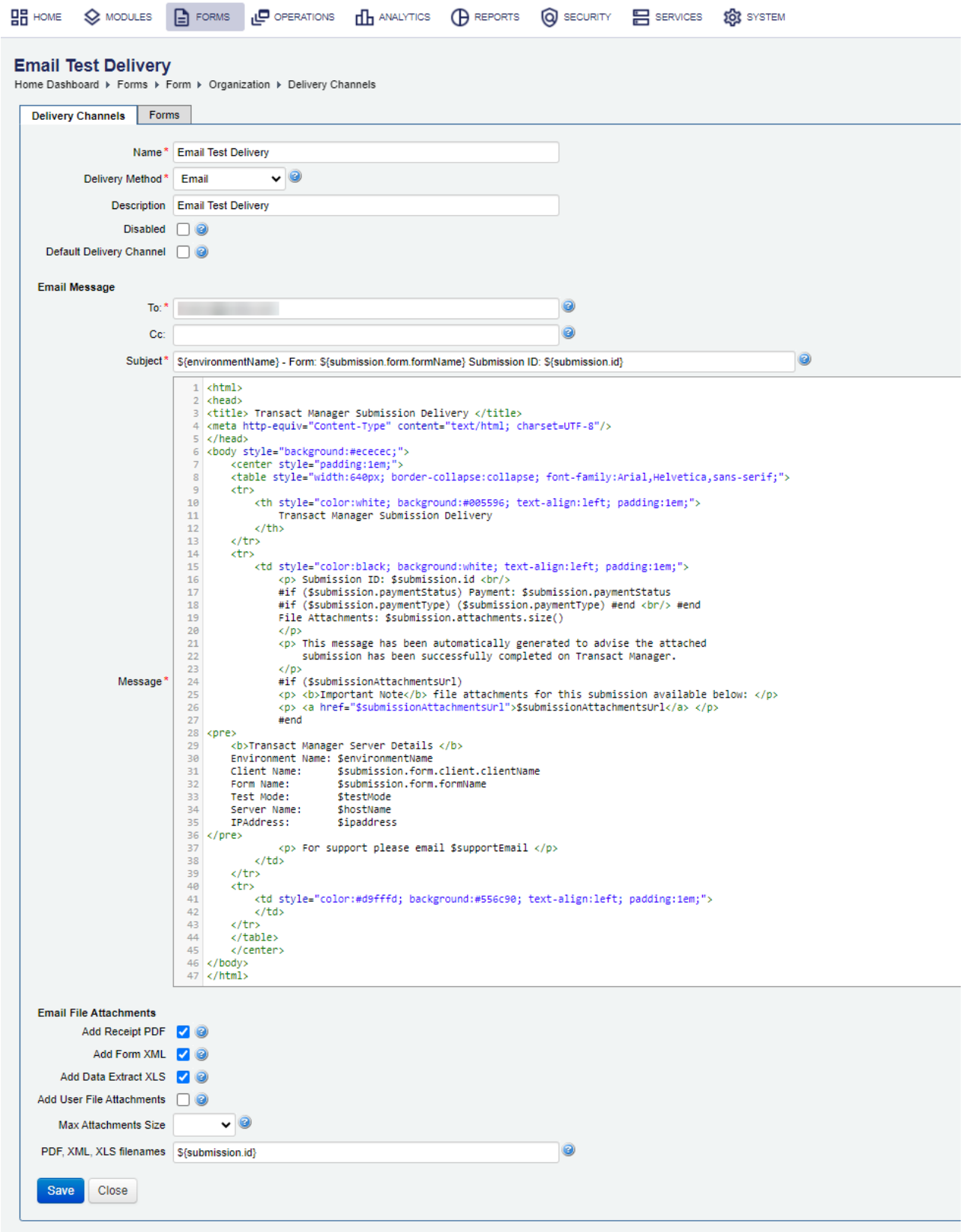
Next, learn how to create an Email delivery channel.

Create an Email Delivery Channel

Manager allows you to create and configure the Email delivery channel for an organization. The Email delivery channel works by sending out an email to a set of email addresses. The email can contain the submission data. Delivery is marked as completed as soon as the email has been sent.

To create a Delivery Process delivery channel:

1. Create a delivery channel using the Email delivery method.



2. Enter one or more email addresses of recipients, separated by commas, in the **To** field. Alternatively, you can use an email address of a distribution list configured in your email client or server.
3. Enter one or more email addresses of email recipient(s) separated by commas in the **Cc** field.
4. The **Subject** field is pre-populated with the following identifying details: an environment, form name, and submission ID. However, you can customize the subject as required.

   NOTE

   You can change the email subject at the organization level by updating a corresponding organization's property. This will affect the default subjects whenever a new email delivery channel is created.

5. The **Message** field is pre-populated with a message body content as an HTML file. You can customize this message as required using valid HTML and the Velocity Template Language (VTL) markup. For more information, see Configure a Form Space Email Template
6. Select the Add Receipt PDF checkbox to attach the PDF receipt to the email. This is selected by default.
7. Select the Add Form XML checkbox to attach the form submission XML file with the email. This is selected by default.
8. Select the Add Data Extract XLS checkbox to attach the Excel file of the form submission data extracts with the email.
   This is selected by default.
9. Select the **Add User File Attachments** checkbox If there are file attachments that a user must include with the email.
10. Select the appropriate size (in megabytes) from the **Max Attachments Size** dropdown if your email server has a limit set on the size of an email. Whenever the limit is reached, a link to the form submission is added to the email instead of an actual attachment.
11. Specify the PDF receipt file name, form XML and Excel Data Extract XLS files in the PDF, XML, and XLS filenames field, which is set to the submission ID, `${submission.id}`, by default. Manager automatically appends the correct file extensions, such as .PDF, .XML or .XLS. The file name must comprise only supported Velocity Template Language variables, such as:
    - Receipt number - `${receiptNumber}`
    - Submission - `${submission}`
    - Form - `${form}`

12. Click Save to update the changes.

Next, learn how to create Email Secure delivery channel.

Create an Email Secure Delivery Channel

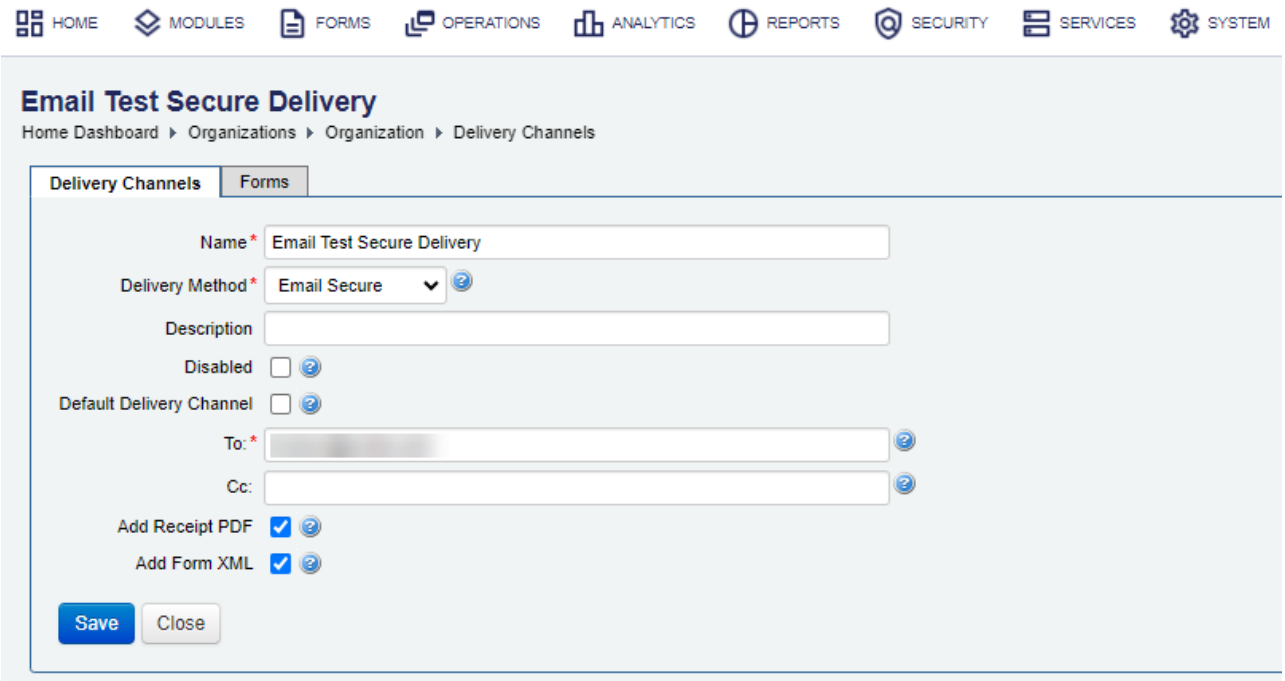
Journey Manager allows you to create and configure the Email Secure delivery channel for an organization. This is same as the standard email delivery, which sends out an email to notify the organization of a submission, but the email itself does not contain any submission data; instead, it contains a link to a secure page hosted on a Manager server so a form user can download the submission data.

The Email Secure delivery completes only after an Administrator user has accessed this page and confirmed that the submission has been received. If delivery has not been acknowledged within a certain time period, which is 24 hours by default, reminders are sent out to the recipients defined in the delivery channel. If the delivery channel's Email Addresses field is blank, the reminder is sent to all global and organization administrators defined in the Receive Delivery Escalation Alerts group.

If delivery remains unacknowledged (by default 72 hours after the form was submitted), escalation emails are sent out to all global and matching organization administrators defined in the Receive Delivery Escalations group.

To create an Email Secure delivery channel:

1. Create a delivery channel using the Email Secure delivery method.



2. Configure the required fields as described in Create an Email Delivery Channel.
3. Click Save to update the changes.

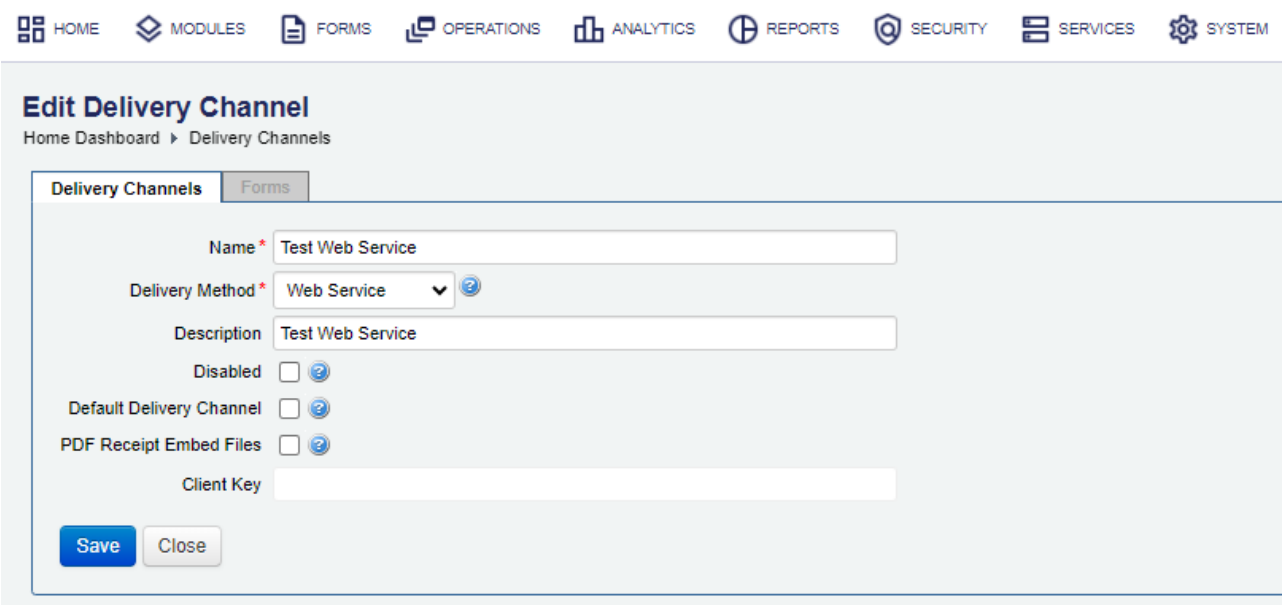
Next, learn how to create a Web Service delivery channel.

Create a Web Service Delivery Channel

Journey Manager allows you to create and configure the Web Service delivery channel for an organization. This delivery channel notifies an organization by making a web service call using a pre-defined API. If delivery has not been completed within the specified period of time, which is, by default, 72 hours after the form was submitted, escalation emails are sent out to all global and organization administrators defined in the Receive Delivery Escalation Alerts group.

To create a Web Service delivery channel:

1. Create a delivery channel using the Web Service delivery method.



2. Select the PDF Receipt Embed Files checkbox to embed a form XML and attachment files into the PDF receipt.
3. Provide a unique organization's client ID in the Client Key field. For more information, see Configure an Organization.

   NOTE

   This field is available for editing once the delivery channel has been saved.

4. Click Save to update the changes.

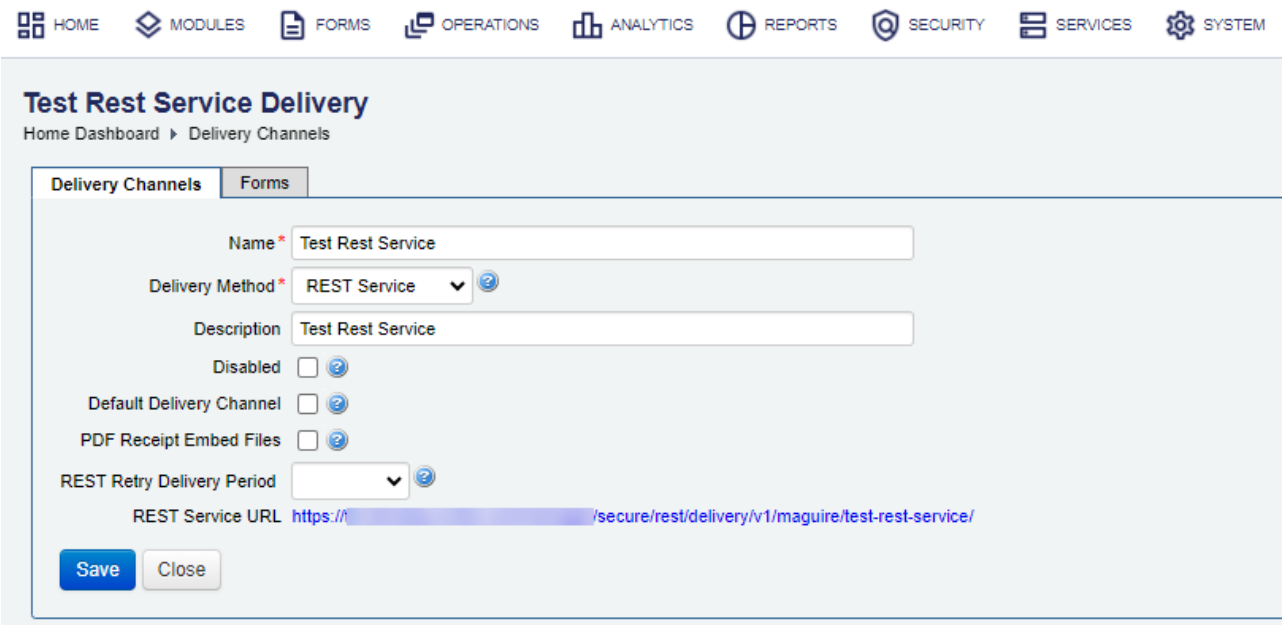
Next, learn how to create a REST delivery channel.

Create a REST Delivery Channel

Journey Manager allows you to create and configure the REST Service delivery channel for an organization. This provides a REST endpoint to retrieve submission data.

To create a REST Service delivery channel:

1. Create a delivery channel using the REST Service delivery method.



2. Select the PDF Receipt Embed Files checkbox to embed a form XML and attachment files into the PDF receipt.
3. Select a period of time from the REST Retry Delivery Period dropdown to impose a time limit for submissions that are In Progress. Once the time limit is reached a submission is reset to delivery status Ready.
4. Check the endpoint URL in the REST Service URL field. You can copy it to use in your scripts and other configurations.

   NOTE

   This field is displayed once the delivery channel has been saved.

5. Click Save to update the changes.

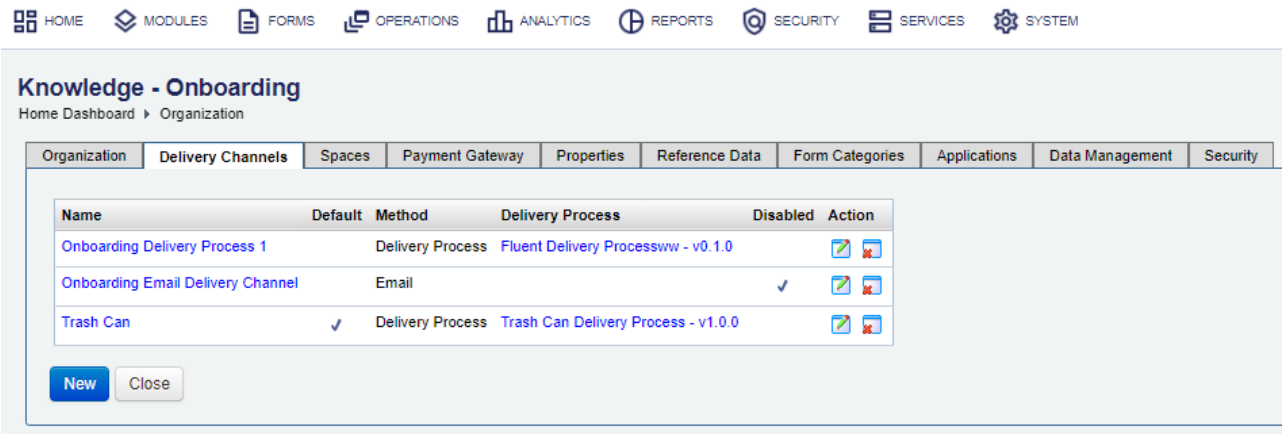
Next, learn how to view delivery channels.

View Delivery Channels

Manager allows you to view the list of delivery channels configured for a selected organization.

To view the delivery channels for an organization:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Delivery Channels tab to see all delivery channels for this organization.



The list displays the following details:

- Name: a unique name, within your organization, of a delivery channel.
- Default: whether the delivery channel is default or not. If a form doesn't have a delivery channel assigned, the default one is used. Form's delivery channel takes precedence over the organization's delivery channel. If no delivery channel is set for both the form and organization, then the delivery is moved to an error state - Delivery Not Configured.
- Method: a delivery method used to transfer submission data. This can be one of the following:
  - Delivery Process
  - Email
  - Email Secure
  - Web Service
  - REST Service
- Delivery Process: a name of a delivery process if the method is of the Delivery Process type.
- Disabled: whether the delivery channel is disabled or not. If disabled, the delivery channel can't be used.

You can also perform the following actions:

- Click New to create a delivery channel.
- Click Edit to configure a delivery channel. Each delivery channel has different set of parameters so check its documentation for details.
- Click Remove to remove a delivery channel when you don't need it any more.

> TIP
>
> We recommend disabling a delivery channel, rather than deleting it.

Next, learn how to view submission delivery.

Assign a Delivery Channel to a Form

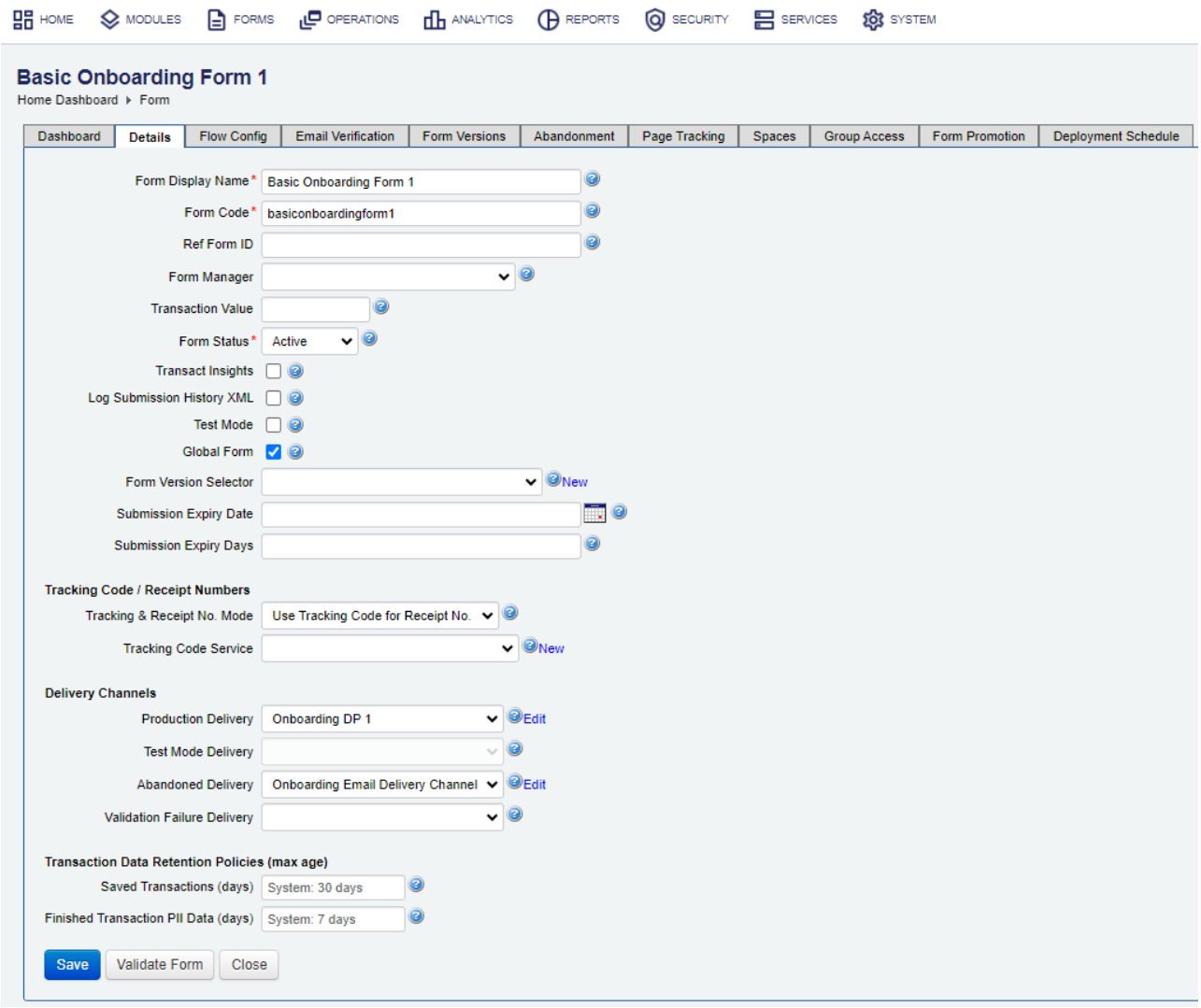
Journey Manager allows you to assign a delivery channel to a form, so when a user opens the form one of the following takes place:

- A user successfully completes and submits the form, so Manager delivers the form details to the platform back-end or to a third-party system.
- A user cancels the form, so Manager delivers the abandoned form details to the platform back-end or to a third-party system.
- A user submits the form but a form validation error occurs, so Manager delivers the failed form details to the platform back-end or to a third-party system.

To assign a delivery channel to a form:

1. Select Forms > Forms.
2. Locate a form and click Edit.

3. Select the Details tab and configure the Delivery Channels section.



4. Select a delivery channel from the Production Delivery dropdown list that will be used to deliver successful submissions for this form if the form is not in the test mode.
5. Select a delivery channel from the Test Mode Delivery dropdown that will be used to deliver successful submissions for this form if the form is in the test mode.

> NOTE
>
> This dropdown is enabled only if the form is in test mode, that is the Test Mode checkbox is selected.

6. Select a delivery channel from the Abandoned Delivery dropdown list to deliver abandoned submissions.

7. Select a delivery channel from the Validation Failure Delivery dropdown list to deliver submissions with data validation errors.
8. Click Save to update the changes.

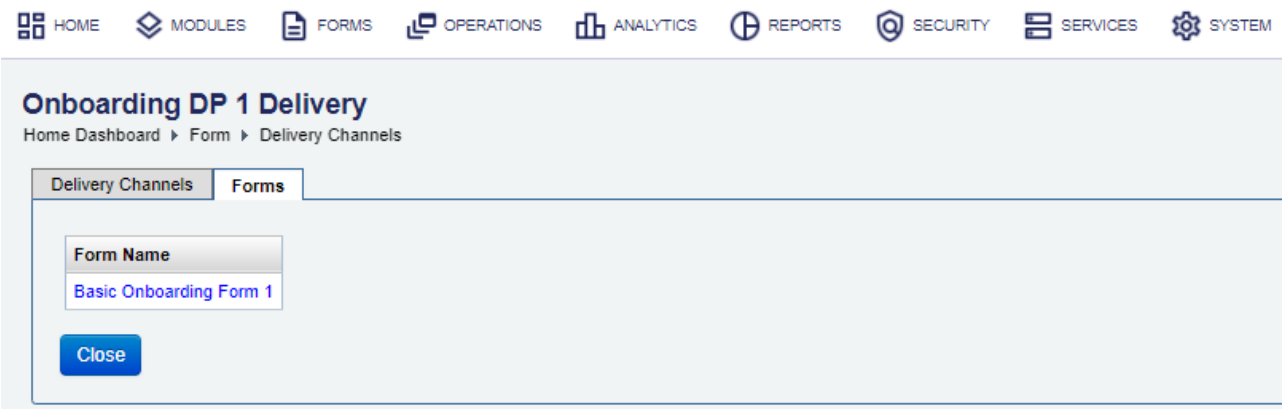
For more information on other form details, see form configuration.

View Forms Associated with a Delivery Channel

Manager lets you see what forms make use a specific delivery channel. When modifying or deleting a delivery channel, it can be helpful to know which forms are using it to avoid unwanted side effects. Also, this is handy when you troubleshoot a form submission delivery.

To view forms associated with a delivery channel:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Delivery Channels tab and locate a delivery channel you wish to check.
4. Select the Forms tab to view the details.



5. Click Close.

> NOTE
>
> This list includes only forms explicitly associated with the delivery channel. If a delivery channel is marked as the default for the current organization, it will be used for all forms not explicitly set to use other delivery channels.

Next, learn how to view transactions delivery.

Remove a Delivery Channel

Manager allows you to remove a delivery channel from an organization when it is not required any more. However, we suggest disabling a delivery channel rather than removing it. If you still decide to delete the delivery channel, you need to update forms that use this delivery channel with a new one.

Let's look this in detail.

To reassign a delivery channel:

1. Check if any forms are associated with the delivery channel.
2. Reassign each form associated with this delivery channel to a valid delivery channel.

To remove a delivery channel:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Delivery Channels tab.
4. Locate a delivery channel you wish to remove and click Remove.

Next, learn how to create a delivery channel.

View Transactions Delivery

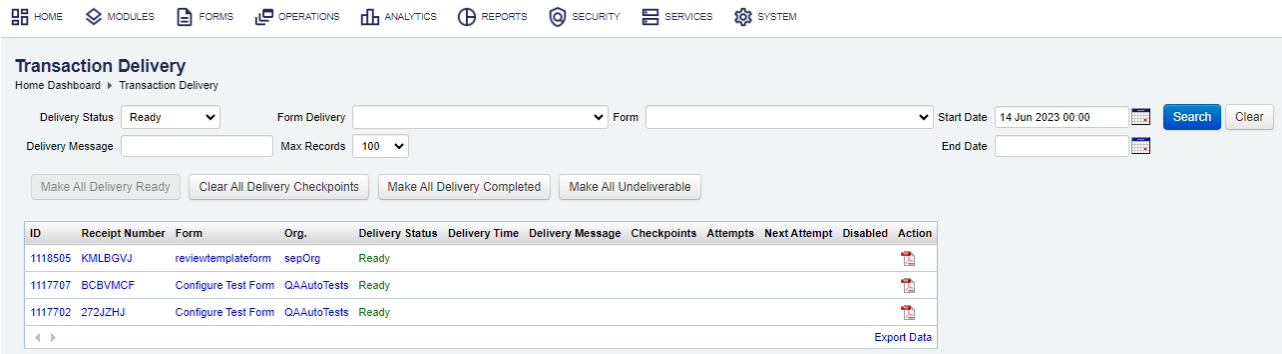
Journey Manager keeps track of each form during its life cycle, from the moment a user opens the form and until the form's data is purged from the system. It allows you to check the following information:

- User interactions with a form.
- Delivery of submissions and all relevant notifications.
- Form abandonment by users.
- Form validation errors.

If you are getting Receive Delivery Escalation Alerts email notifications for a particular submission delivery, you can use this functionality to troubleshoot this delivery failure and subsequently fix it or mark it as Not Required or Unavailable, so the alert will go away.

To view deliveries of form transactions:

1. Select Operations > Transaction Delivery.



2. To filter or search the submissions by one or more criteria, specify the following settings and click Search:
   - Delivery Status, which can be one of the following: Ready, Send Email, In Progress, Pending, Completed, Error, and Undeliverable. Use the status to narrow down on transactions you are interested it. For example, if you are look for a stuck transaction, check transactions that have statuses other then Completed or Undeliverable.
   - Delivery Message
   - Form Delivery
   - Max Records
   - Form
   - Start Date
   - End Date

Click Clear to reset the search criteria.

3. You can perform the following bulk operations to change the status of all selected submissions:
   - Click Make All Delivery Ready to reset delivery status to Ready so the delivery will be retried.
   - Click Clear All Delivery Checkpoints to clear checkpoints, if the delivery service supports delivery checkpoints.
   - Click Make All Delivery Completed to set delivery status to Completed without retrying delivery.
   - Click Make All Undeliverable to set delivery status to Undeliverable, so this delivery will not be reattempted.
4. Click on a transaction receipt number to view the details of each submission. You can change the delivery status for the individual submission, if required.
5. Click View Receipt to view and download a transaction receipt as a PDF file.

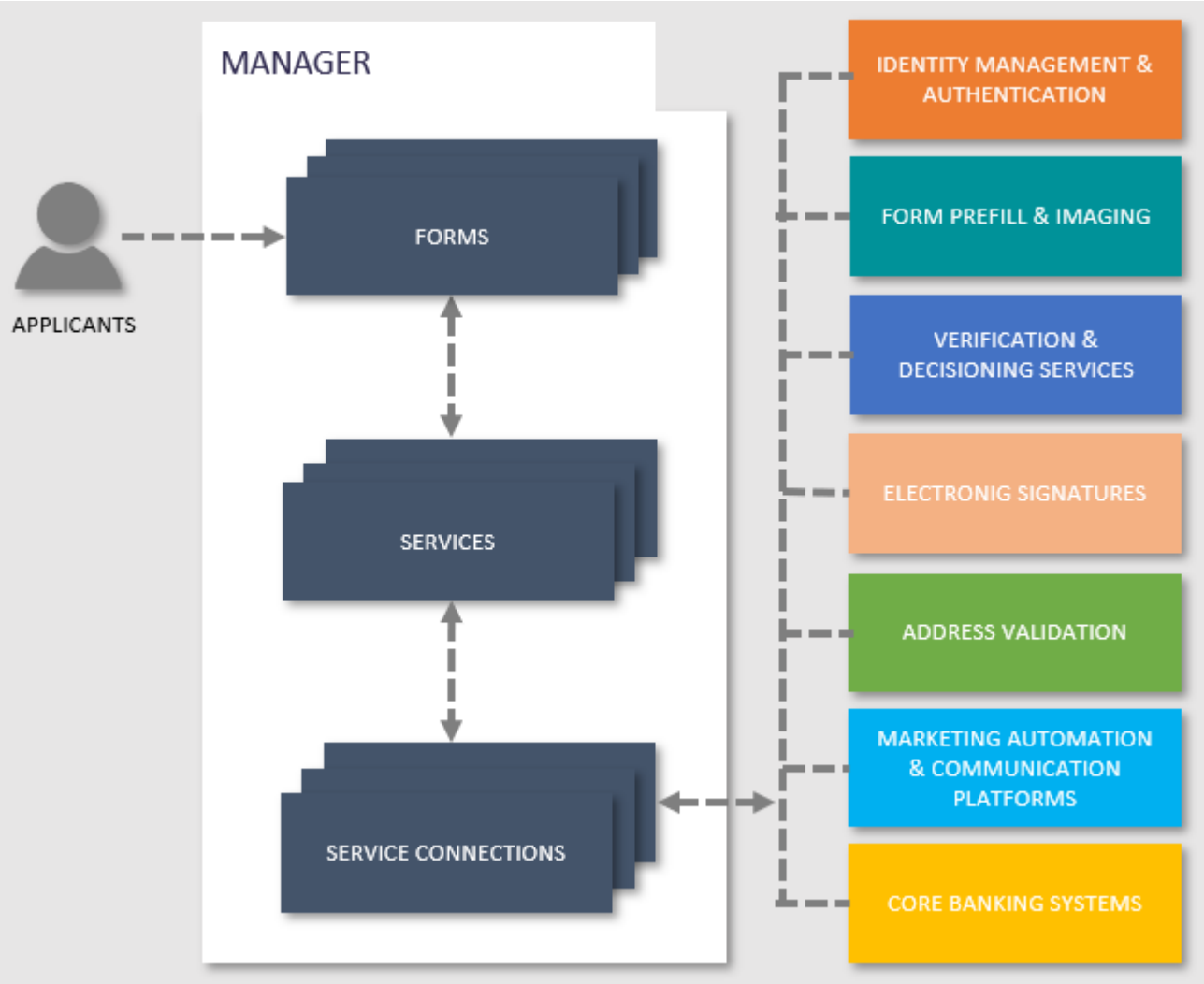
Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to create a delivery channel.

Service Connection Overview

Journey Manager comes with services that require connections to external servers to perform their functionality. These connection details are managed separately to services using service connections. This allows you to reuse service connections in different services and change their configurations without modifying services.

Manager includes pre-built connectors for best in class services such as fraud detection, ID verification, digital signatures, and over 30 different connection partners. The integration has already been done, so each connector can be added to the user experience as needed when designing the application.



Manager uses service connection types to define service connections functionality, so you can create your own service connection based on a selected service connection type. You can define a service connection that is:

- Global - it doesn't belong to an organization and thus can be used by all services.
- Organization specific - it belongs to a single organization and can be used only by services that belong to this organization.

You can also remove, import and export service connections.

Next, learn how to view all service connections.

Service Connection Types

Manager uses service connection types to define and group service connections according to their functionality. A service connection type can:

- Act as a template by providing the applicable set of parameters for each service connection type
- Determine how it is used within Manager

Manager has the following predefined service connection types:

Service Connection Type	Description	Used in
AWS KMS	Data storage	
AWS S3	Data storage	
AWS SQS	External publish queue	AWS SQS system event publisher service
AWS STS	External publish queue	
Azure Storage	External publish queue	Azure submission data storage service Azure system event publisher service
DocuSign	DocuSign service | 22.10This feature was updated in 22.10.	
DocuSign OAuth	DocuSign OAuth service | 22.10 This feature was introduced in 22.10.	
HTTP Endpoint	Generic HTTP service	
Kafka	Apache Kafka	Event-Based Architecture

	| 22.10 This feature was introduced in 22.10.
SalesForce	SalesForce CRM
Integration Gateway	TIG service

You can create new service connections based on the existing types.

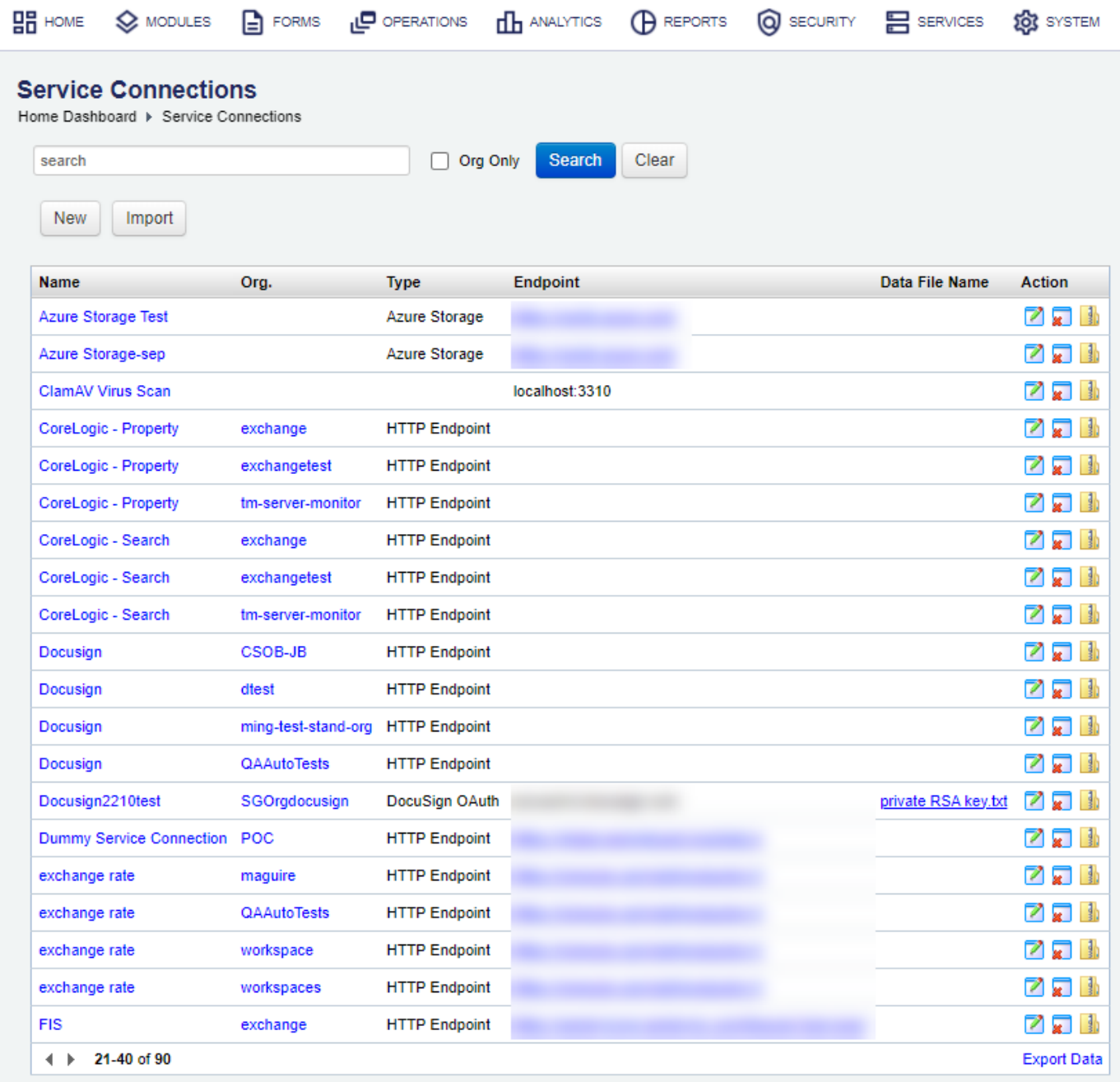
Next, learn how to view service connections.

View Service Connections

Manager allows you to view all service connections, so you can easily find one you need to use it in your service configuration.

To view service connections, select Services > Service Connections.

Service connections are shown page by page, with the maximum number of records per page configured via the user's preferences.



The service connection list displays the following details:

- Name - a service connection name, which is unique across one organization for an organization-specific service connection or across the complete environment for a global service connection
- Org - an organization a service connection belongs to, or blank for a global service connection
- Type - a service connection type or blank
- Endpoint - a URL where a service connection points to or blank for certain service connection types
- Data File Name - a link to a specific resource required by a service connection, such as a certificate key

To filter or search the services by one or more criteria, specify the following settings and click Search:

- Service connection name or part of a name, or the endpoint[1] URL
- Select the Org Only checkbox to show service belonging to an organization. This will exclude global service connections, which are available to all services.

Click Clear to reset the search criteria.

You can also perform the following actions:

- Create a service connection
- Configure a service connection
- Remove a service connection
- Export service connections
- Import service connections

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to create a service connection.

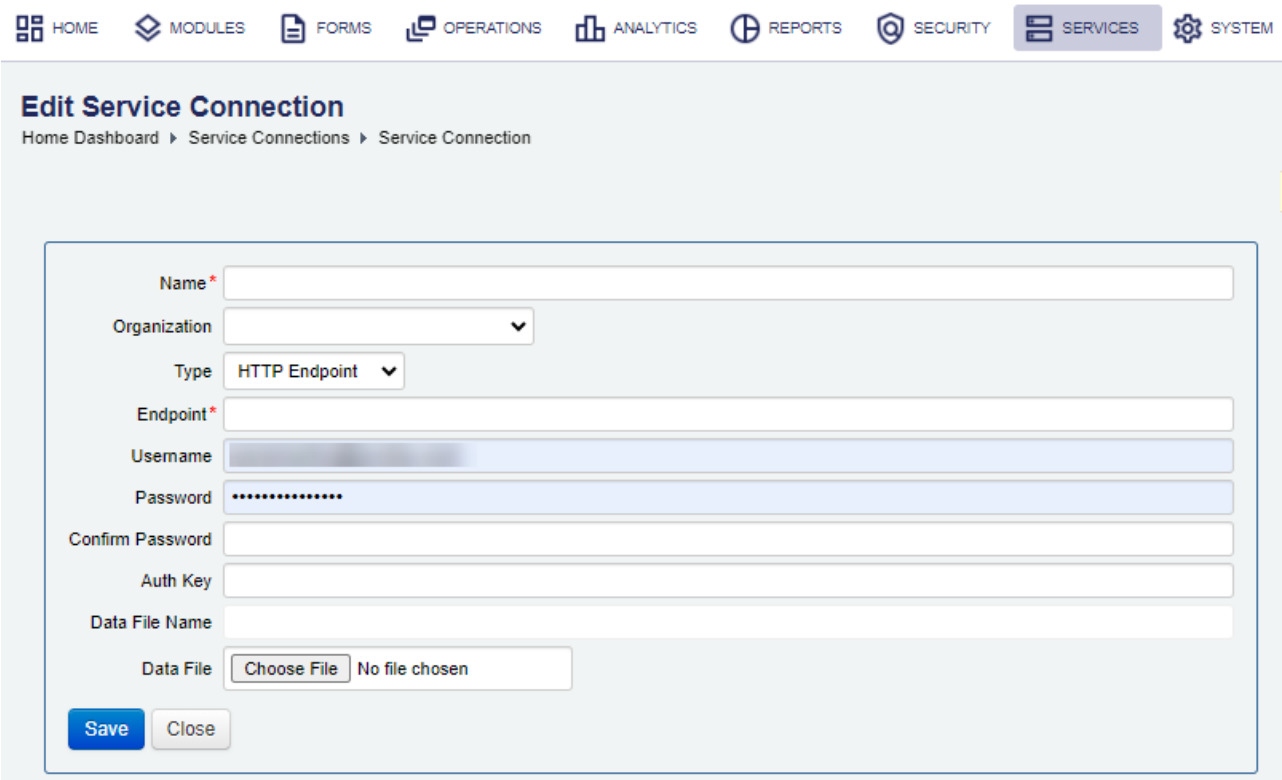
[1]Endpoint is a web address (URL) at which clients of a specific service can gain access to it. By referencing that URL, clients can get to operations provided by that service.

Create a Service Connection

Manager allows you to create service connections, so you can use them in your services to implement various client's requirements.

To create a new service connection:

1. Select Services > Service Connections.
2. Click New.



3. Provide a unique service connection name in the Name field.
4. Select an organization from the Organization dropdown list to restrict this service connection to a single organization. Leave is blank to create a global service connection that can be used by all services.

   NOTE

   After the service connection is created, you can't change the organization it belongs to, nor set it to blank.

5. Select a service connection type from the Type dropdown list to define and group the service connection.

> NOTE
>
> Service connection configuration options are different for each service connection type. Refer to a specific service connection type configuration for more details.

6. Enter the Endpoint [1] as a URL to an external service.

> NOTE
>
> The username and password fields should not contain certain HTML characters, such as `>` or `<`. If they do, you may find the service throws an exception when it's called. If this happens, check your logs for further details.

7. Configure connection specific options.
8. Click Save to update the changes.

Next, learn how to remove a service connection.

[1] Endpoint is a web address (URL) at which clients of a specific service can gain access to it. By referencing that URL, clients can get to operations provided by that service.

Configure a Service Connection

Manager allows you to create new and edit existing service connections, so you can use them in your services to implement various client's requirements.

To configure a service connection:

1. Select Services > Service Connections.
2. Locate a service connection and click Edit.
3. Update a service connection name in the Name field.
4. Check an organization in the Organization field.

   NOTE

   You can't change the organization it belongs to, nor set it to blank after the service connection has been created.

5. Select a service connection type from the Type dropdown list to define and group the service connection.

   NOTE

   Service connection configuration options are different for each service connection type. Refer to a specific service connection type configuration for more details.

6. Configure connection specific options.
7. Click Save to update the changes.

You can't always access the `param1`, `param2`, `param3`, and `param4` properties of a service connection via the Service Connection Configuration UI to update them. If you are required to use these parameters, you can do the following:

1. Export a service connection as a ZIP file
2. Open a `connection-config.xml` file, which is a part of the archive
3. Check or modify the `param1`, `param2`, `param3`, and `param4` properties as required. For more information on their definition, see JSON Definitions Reference in the Journey SDK documentation.
4. Create a new ZIP archive file containing the updated configuration
5. Import the new archive and configure the service connection

NOTE

You can also update the `param1`, `param2`, `param3`, and `param4` properties using the Fluent API `SvcConnUpdater.setParam1`, `SvcConnUpdater.setParam2`, `SvcConnUpdater.setParam3`, and `SvcConnUpdater.setParam4` methods. For more information, see Fluent API.

Next, learn how to remove a service connection.

Remove a Service Connection

Manager allows you to remove service connections if they are no longer required.

> NOTE
>
> Manager irrevocably removes all data associated with the service connection. This action cannot be undone

To remove a service connection:

1. Select Services > Service Connections.
2. Locate the service connection.
3. Update all services using this service connection so they are not left without a valid service connection. Otherwise, they will most likely fail.
4. Click Remove and then click OK to confirm.

Next, learn how to view service connections.

Migrate Service Connections

Manager provides a mechanism to migrate service connections from one environment to another. This allows you to develop and test a service connection in different environments before migrating it over to the production environment.

To migrate one or several service connections:

1. Export service connections
2. Import service connections
3. Configure service connections as some sensitive configuration, such as user names, passwords, assess keys, and so on, are tot included in the export files, so you need to re-enter it manually.

NOTE

You can use the same process to perform a service connection backup and rollback.

Next, learn how to view service connections.

Export Service Connections

Manager allows administrators to export service connections. You can use this functionality to:

- Back-up or archive service connections
- Re-create or migrate service connections in another Manager
- Promote service connections to another environment

To export a service connection:

1. Select Services > Service Connections.
2. Locate a service connection and click Export Archive to export a service connection as an archive ZIP file.
3. Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

NOTE

The sensitive information, such as user names, passwords and access keys, are not included in an archive file.

Next, learn how to import service connections.

Import Service Connections

Manager allows you to import service connections. Use this functionality to:

- Roll-back service connections
- Re-create service connections in another instance of Manager
- Promote service connections to another environment

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

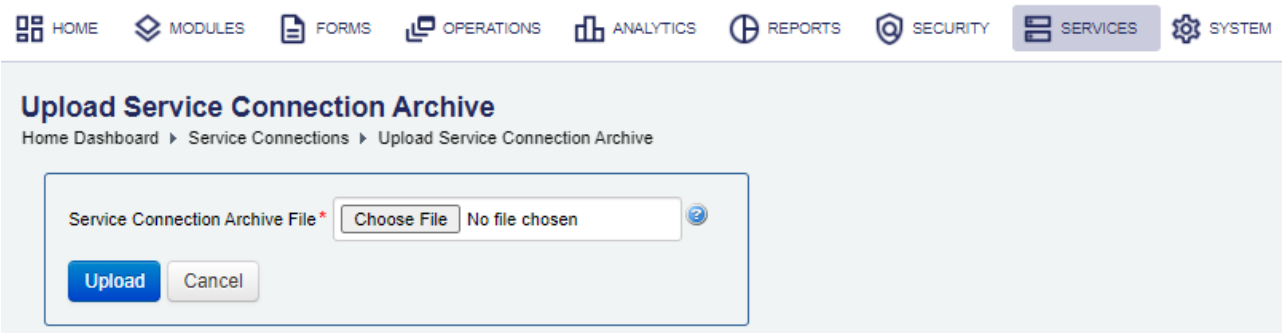
You can import a service connectionusing one of following options:

- UI
- SDK
- API

Import Using UI

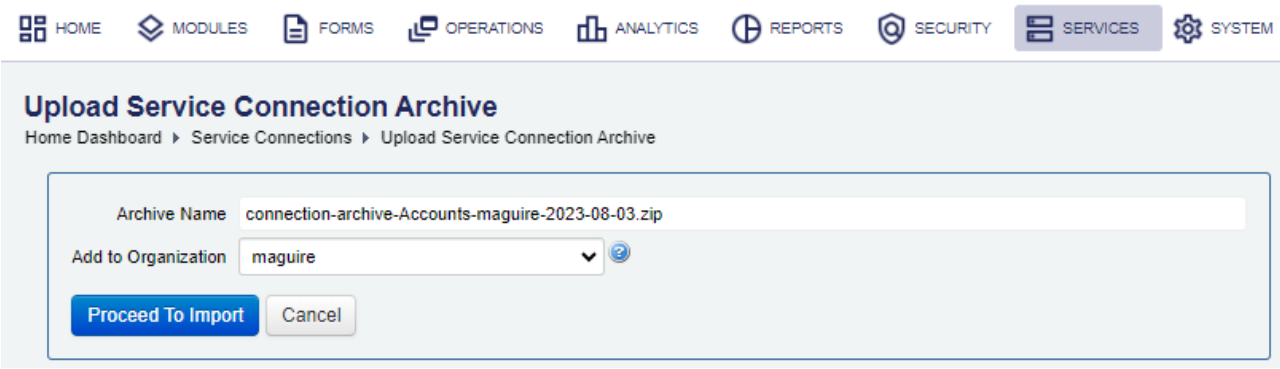
To import a service connection:

1. Select Services > Service Connections.
2. Click Import.

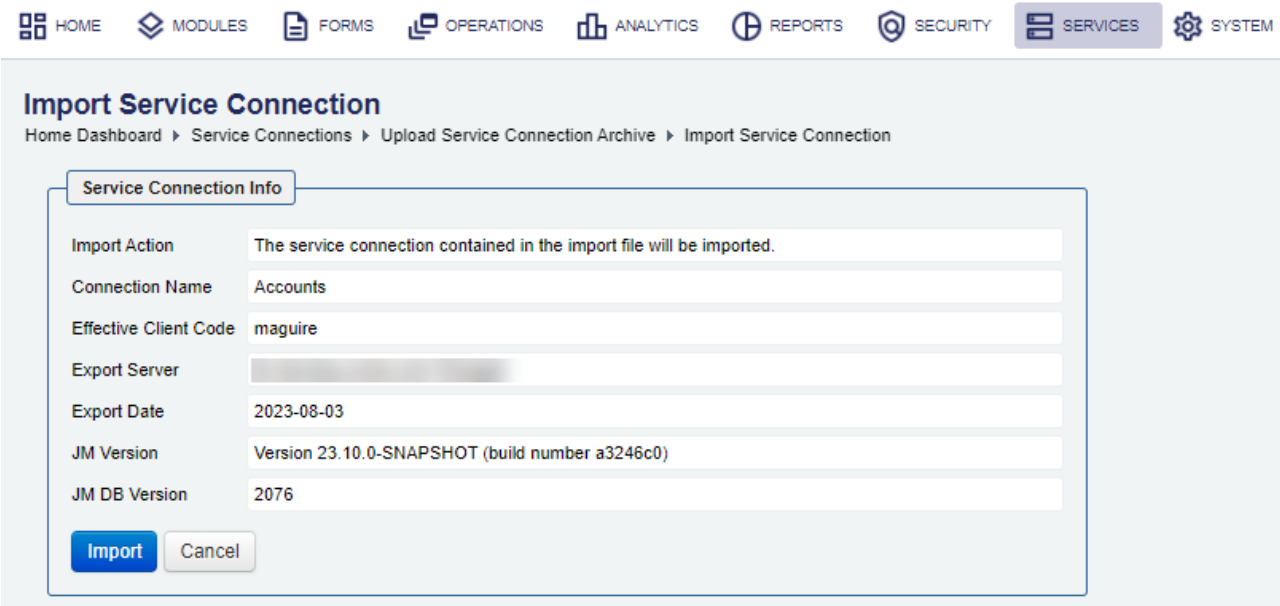


3. Click Choose File and choose a service connection archive ZIP file, which has been previously exported.

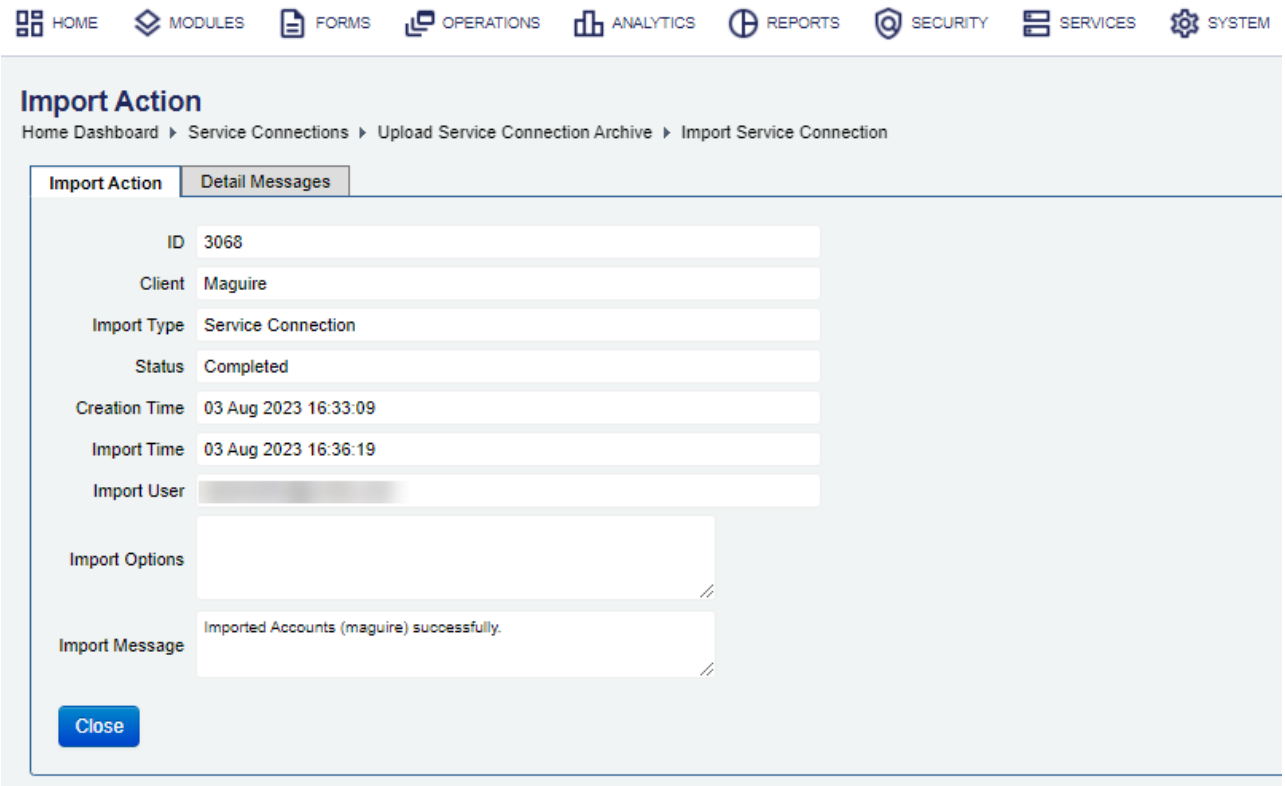
4. Click Upload.



5. Select an organization, to where the service connection applies, from the Add to Organization dropdown list. Otherwise, Manager uses the organization that the service connection belongs to from the export archive file.
6. Click Proceed To import.
7. Review the Import Options as shown below.



8. Click Import to complete the process. The expected result page is shown below.



9. Select the Detail Messages tab to see more information about the import. This tab pops up open if the import operation has any issues you may need to check and resolve.
10. Click Close and check the list of service connections to verify the imported service connection have been added.

NOTE

You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

Import Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:archive-deploy goal to import a service connection.

Import Using API

The Journey Manager REST API provides the endpoint URL to import a service connection.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s):///<JM_SERVER>/manager/secure/rest/archive/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as Base64[1].

`--form ‘archiveFile=@“/Users/my-service-connection.zip”’` - a path to an archive file to be imported.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

Example

```
curl --location --request PUT ‘https://<JM_SERVER>/manager/secure/rest/archive/v1/’
\
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
--form ‘archiveFile=@“/Users/my-service-connection.zip”’
```

Next, learn about service connection types.

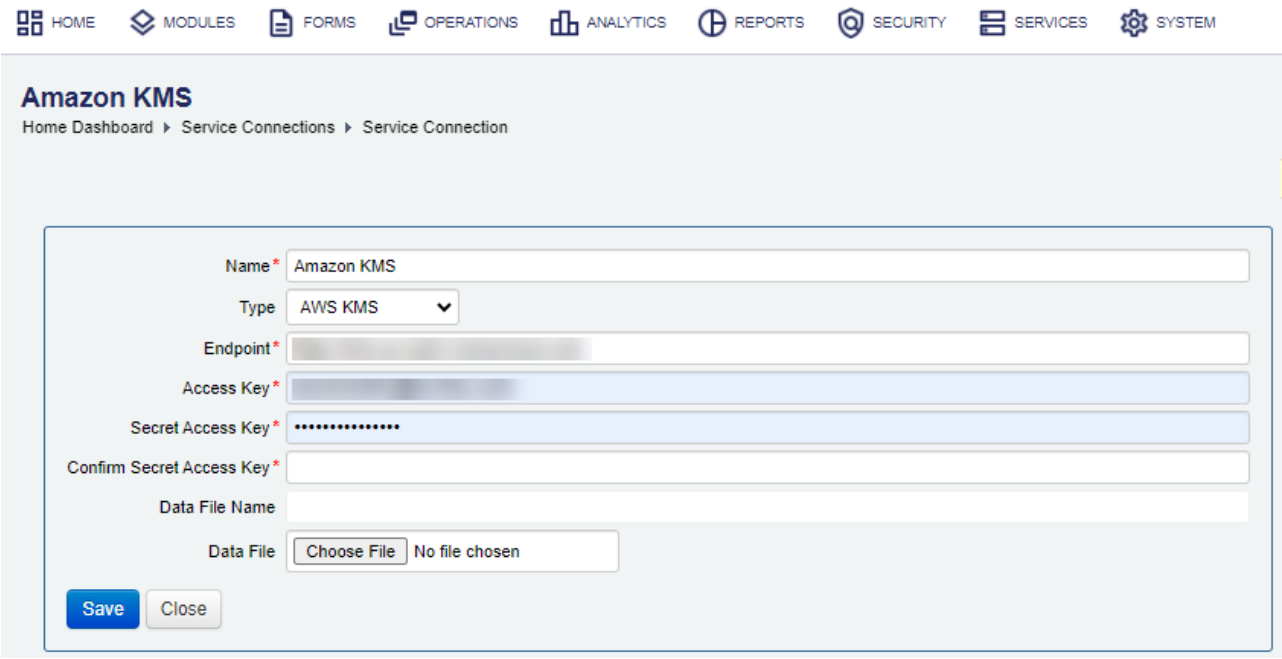
[1]Base64 is a binary to a text encoding scheme that represents binary data in an American Standard Code for Information Interchange (ASCII) string format.

AWS KMS Service Connection

Manager allows you to create and configure a AWS KMS[1] service connection to use it in a service that requires data storage.

To configure a service connection:

1. Select Services > Service Connections.
2. Locate an Amazon KMS service connection or click New to create a new service connection.



3. Provide a unique service connection name in the Name field.
4. Select the AWS KMS type from the Type dropdown list.
5. Enter the Endpoint as a URL for the AWS KMS queue service.
6. Provide the Access Key, which is an AWS access key ID. You use access keys to sign programmatic requests that you make to AWS if you use AWS CLI commands or using AWS

---

[1]AWS Key Management Service (KMS) is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data, and uses FIPS 140-2 validated hardware security modules to protect the security of your keys. AWS Key Management Service is integrated with most other AWS services to help you protect the data you store with these services. AWS Key Management Service is also integrated with AWS CloudTrail to provide you with logs of all key usage to help meet your regulatory and compliance needs.

API operations.You must use both the access key and secret access key together to authenticate your requests. For more information, see AWS documentation.

7. Provide the Secret Access Key.
8. Confirm the secret access key in the Confirm Secret Access Key field.
9. Enter the Data File Name.
10. Click Data File to upload a data file, if it's required for this service connection.
11. Click Save to update the changes.

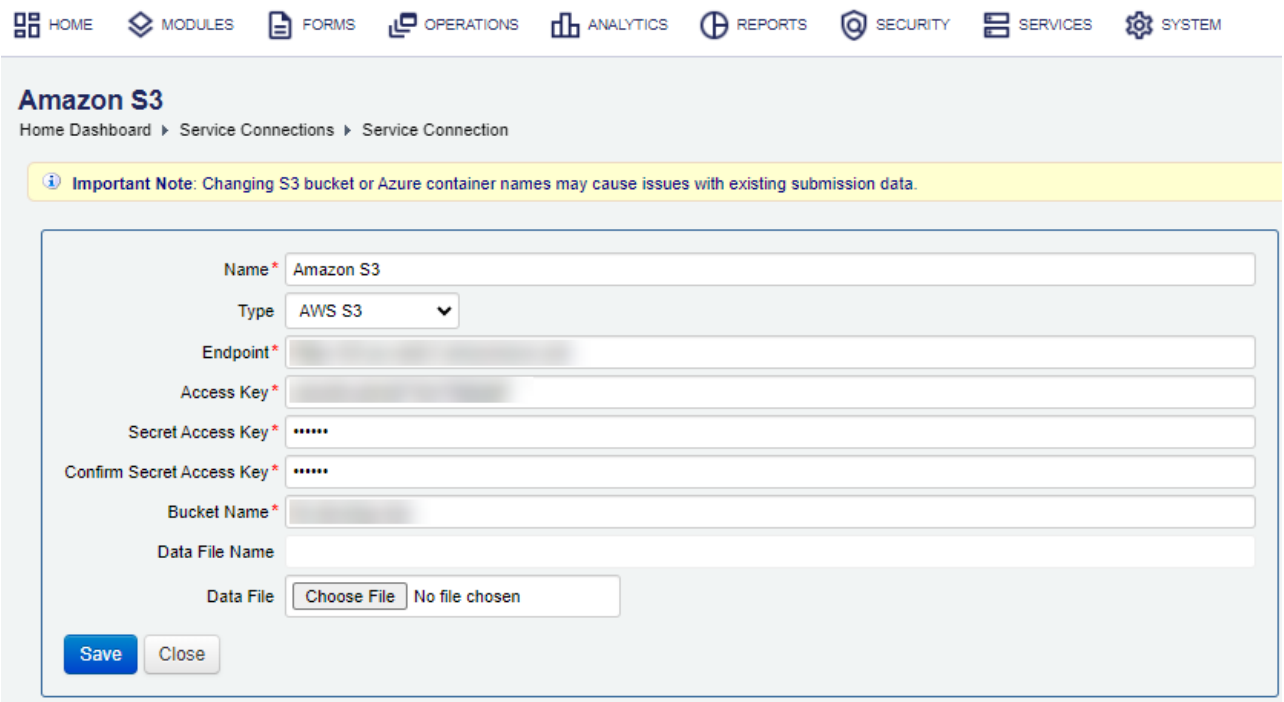
Next, learn about other service connection types.

AWS S3 Service Connection

Manager allows you to create and configure a AWS S3 service connection to be used it in services that require data storage.

To configure a service connection:

1. Select Services > Service Connections.
2. Locate an Amazon S3 service connection or click New to create a new service connection.



3. Provide a unique service connection name in the Name field.

   NOTE

   Changing S3 bucket names may cause issues with existing submission data.

4. Select the AWS S3 type from the Type dropdown list.
5. Enter the Endpoint as a URL for the AWS S3 service, for example, `https://my-domain.amazon-aws.com`.
6. Provide the Access Key, which is an AWS access key ID. You use access keys to sign programmatic requests that you make to AWS if you use AWS CLI commands or using AWS API operations. You must use both the access key and secret access key together to authenticate your requests. For more information, see AWS documentation.

7. Provide a secret key in the Secret Access Key field.
8. Confirm the secret key in the Confirm Secret Key field.
9. Enter a name of the S3 bucket in the Bucket Name field, for example, `my-domain-dev-env-1`.
10. Enter a name of a file in the Data File Name file.
11. Click Data File to upload a data file, if it's required for this service connection.
12. Click Save to update the changes.

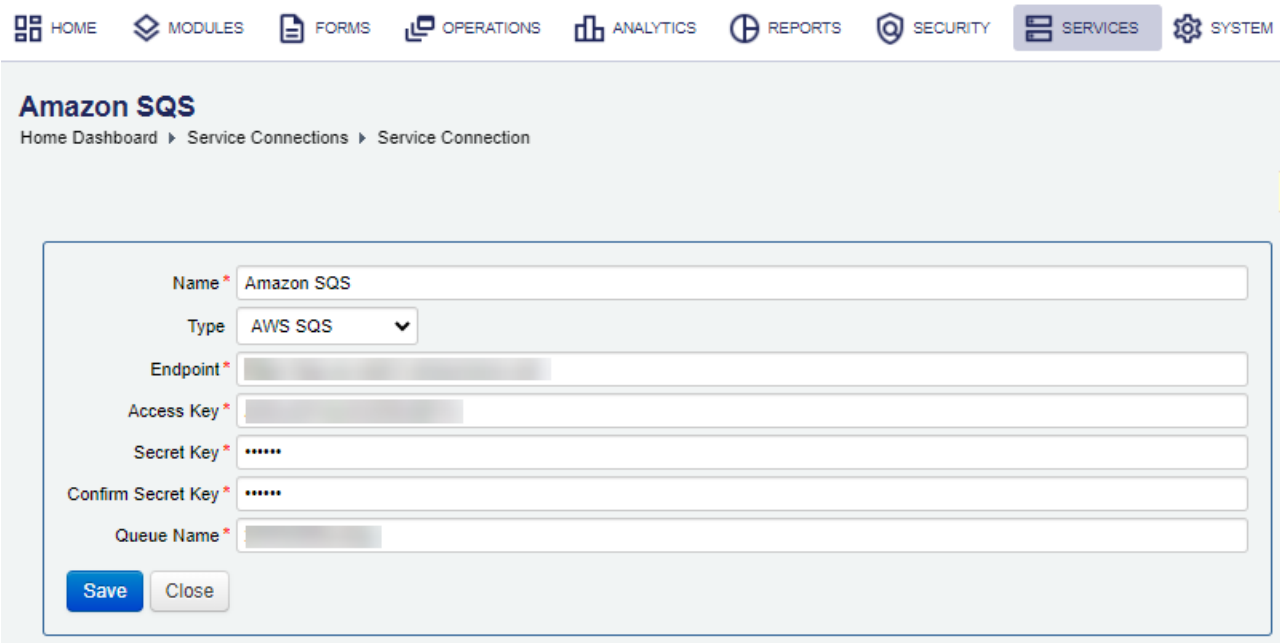
Next, learn about other service connection types.

AWS SQS Service Connection

Manager allows you to create and configure a service connection to an AWS SQS[1] queue, which is used in the AWS SQS System Event Publisher service to publish events to that queue.

To configure a service connection:

1. Select Services > Service Connections.
2. Locate an Amazon SQS service connection or click New to create a new service connection.



3. Provide a unique service connection name in the Name field.
4. Select the AWS SQS type from the Type dropdown list.
5. Enter the Endpoint as a URL for the AWS SQS queue service, for example, `https://sqs.my-domain.com`.
6. Provide the Access Key, which is an AWS access key ID. You use access keys to sign programmatic requests that you make to AWS if you use AWS CLI commands or using AWS API operations. You must use both the access key and secret access key together to authenticate your requests. For more information, see AWS documentation.

---

[1]Amazon Simple Queue Service (Amazon SQS) is a pay-per-use web service for storing messages in transit between computers. Developers use SQS to build distributed applications with decoupled components without having to deal with the overhead of creating and maintaining message queues.

7. Provide the Secret Key.
8. Confirm the Secret Key in the Confirm Secret Key field.
9. Enter the Queue Name configured at the endpoint URL, for example, `my-sqs-queue`.
10. Click Save to update the changes.

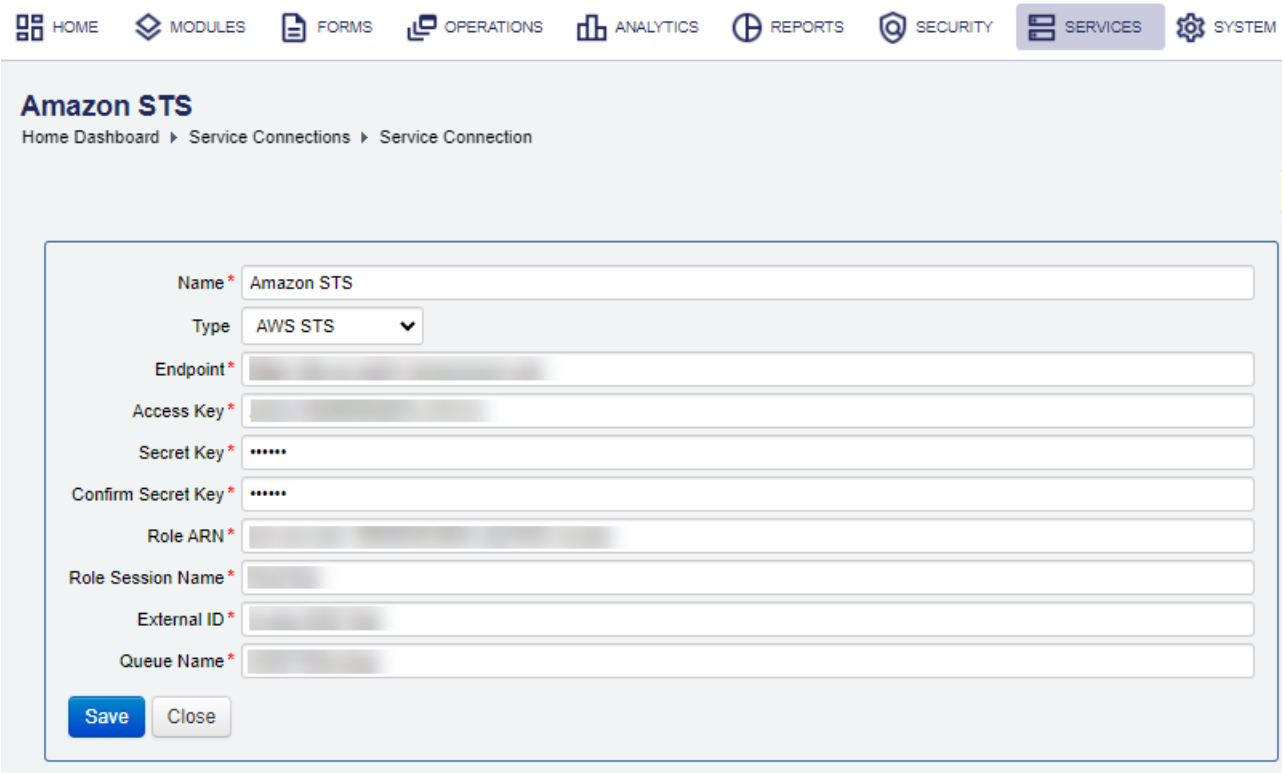
Next, learn about other service connection types.

AWS STS Service Connection

Manager allows you to create and configure a AWS STS[1] service connection to be used in the AWS SQS System Event Publisher services, which publish events to the various queues using the AWS STS protocol.

To configure a service connection:

1. Select Services > Service Connections.
2. Locate an Amazon STS service connection or click New to create a new service connection.



3. Provide a unique service connection name in the Name field.
4. Select the AWS STS type from the Type dropdown list.
5. Enter the Endpoint as a URL for the AWS STS queue service, for example, `https://sts.my-domain.com`.
6. Provide the Access Key, which is an AWS access key ID. You use access keys to sign programmatic requests that you make to AWS if you use AWS CLI commands or using AWS

[1]The AWS Security Token Service (STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users).

API operations. You must use both the access key and secret access key together to authenticate your requests. For more information, see AWS documentation.

7. Provide the Secret Key.
8. Confirm the Secret Key in the Confirm Secret Key field.
9. Provide the Role ARN[1] , which is the Amazon Resource Name (ARN) of the role to assume.
10. Enter the Role Session Name, which is an identifier for the assumed role session. Use the role session name to uniquely identify a session when the same role is assumed by different principals or for different reasons. In cross-account scenarios, the role session name is visible to, and can be logged by the account that owns the role. For example, `MyRoleTest`.
11. Provide the External ID. A unique identifier that might be required when you assume a role in another account. If the administrator of the account to which the role belongs provided you with an external ID, then provide that value in the ExternalId parameter. This value can be any string, such as a passphrase or account number. Because a cross-account role is usually set up to trust everyone in an account, the administrator of the trusting account might send an external ID to the administrator of the trusted account. For more information, see AWS documentation.
12. Enter the Queue Name configured at the endpoint URL, for example, `my-sts-queue`.
13. Click Save to update the changes.

Next, learn about other service connection types.

[1]An Amazon Resource Name (ARN) is a file naming convention used to identify a particular resource in the Amazon Web Services (AWS) public cloud. ARNs, which are specific to AWS, help an administrator track and use AWS items and policies across AWS products and API calls.

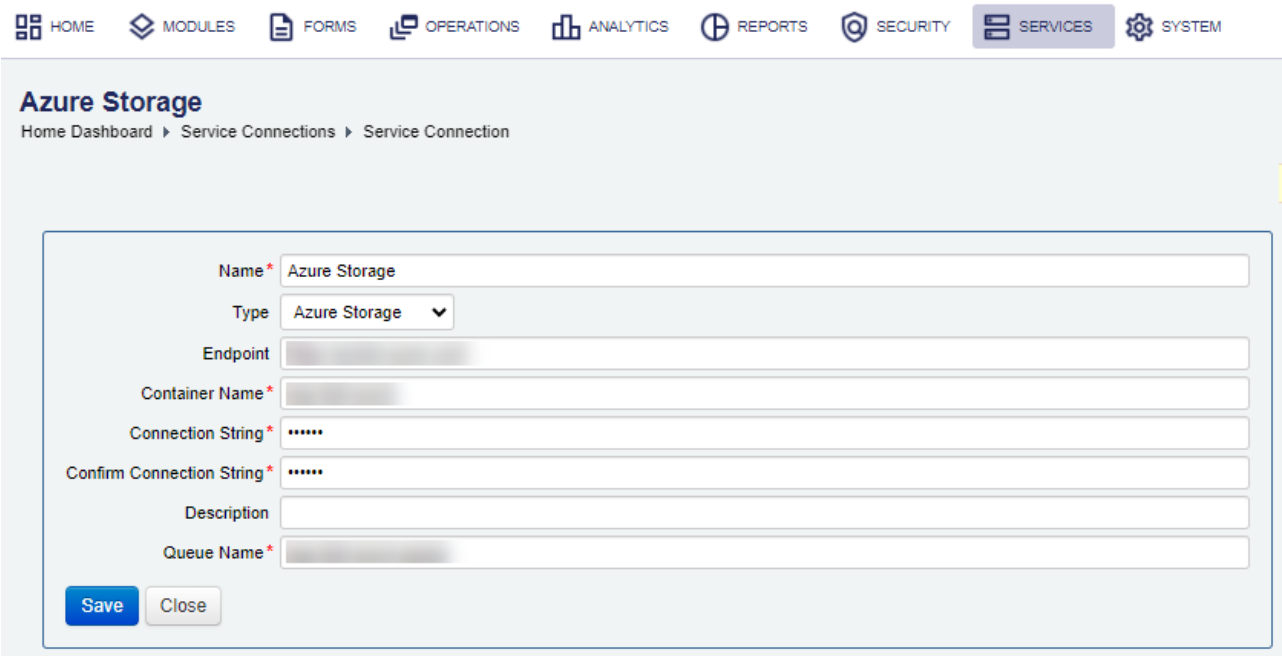
Azure Service Connection

Manager allows you to create and configure a service connection to an Azure[1] Storage, which is used by various services, such as the Azure service connection service and Azure submission data storage service.

> NOTE
>
> Changing Azure container names may cause issues with existing submission data.

To configure a service connection:

1. Select Services > Service Connections.
2. Locate an Azure Storage service connection or click New to create a new service connection.



3. Provide a unique service connection name in the Name field.
4. Select the Azure Storage type from the Type dropdown list.
5. Enter a URL to the Azure storage in the Endpoint field, for example, `https://storage.my-domain.com/`.

[1]Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers.

6. Provide a container name in the Container Name field, which is used for Azure Blob. Both the Azure Blob and Azure Queue are the part of Azure Storage so they use the same connection. For example, `my-test-azure-storage`.
7. Provide a connection string for Azure storage in the Connection String field. For more information, see Microsoft Azure documentation.
8. Enter a description of this service connection in the Description field.
9. Enter a default queue name in the Queue Name field, which is used as a fallback if an Azure service doesn't explicitly specify queue names in the Audit Log Queue, Job Queue, Submission Queue, System Health Queue, User Auth Queue. For example, `my-test-azure-storage-queue`.
10. Click Save to update the changes.

Next, learn about other service connection types.

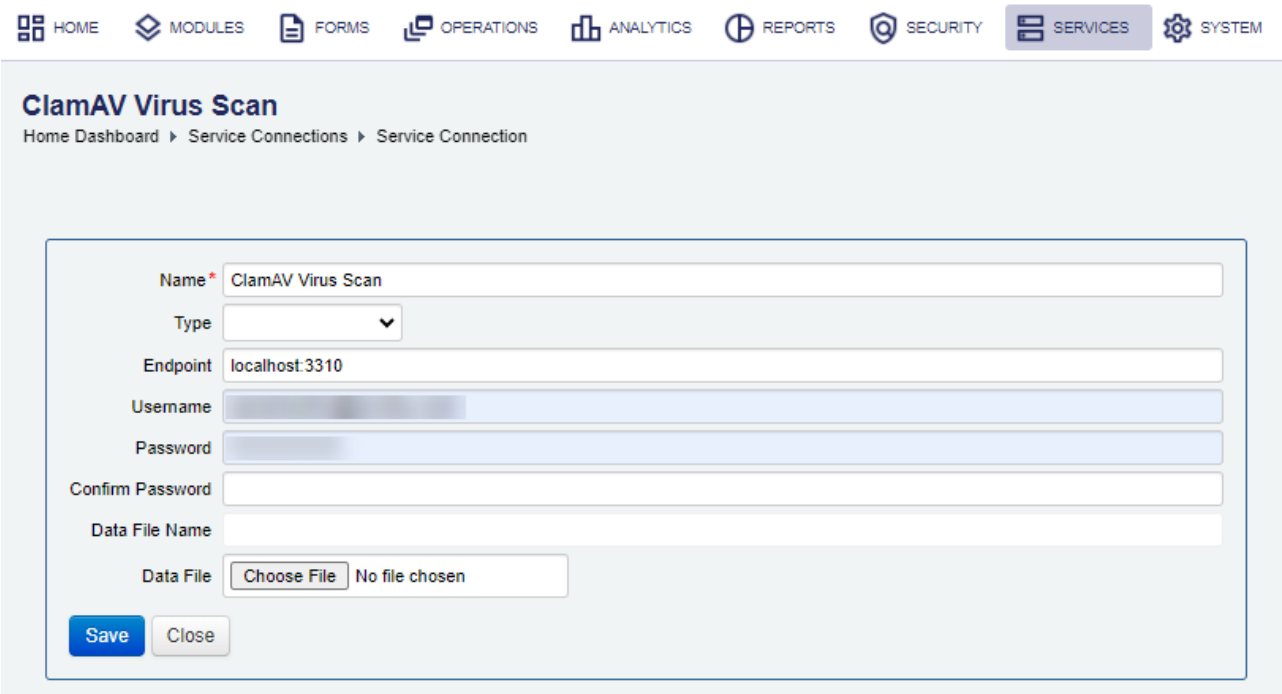
ClamAV Virus Scan Service Connection

Manager allows you to create and configure a ClamAV service connection to connect to the ClamAV engine.

> NOTE
>
> Make sure the ClamAV engine is running and is correctly configured before configuring a service connection.

To configure a service connection:

1. Select Services > Service Connections.
2. Locate a ClamAV Virus Scan service connection to edit it or click New to create a ClamAV Virus Scan service connection.



3. Provide a unique service connection name in the Name field, for example, ClamAV Virus Scan.
4. Leave the Type blank.
5. Update the Endpoint with a server name and port number, where the ClamAV Engine is installed, in the following format `hostName:portNumber`. If the ClamAV Engine runs on the same server as Manager and uses the default port number, you can leave the default configuration as `localhost:3310`.

6. Provide a user name in the Username field, which is used to authenticate the service connection. For more information, see ClamAV documentation.
7. Provide a password in the Password field, which is used to authenticate the service connection.
8. Click Save to update the changes.

Next, learn how to configure the attachment virus scan service.

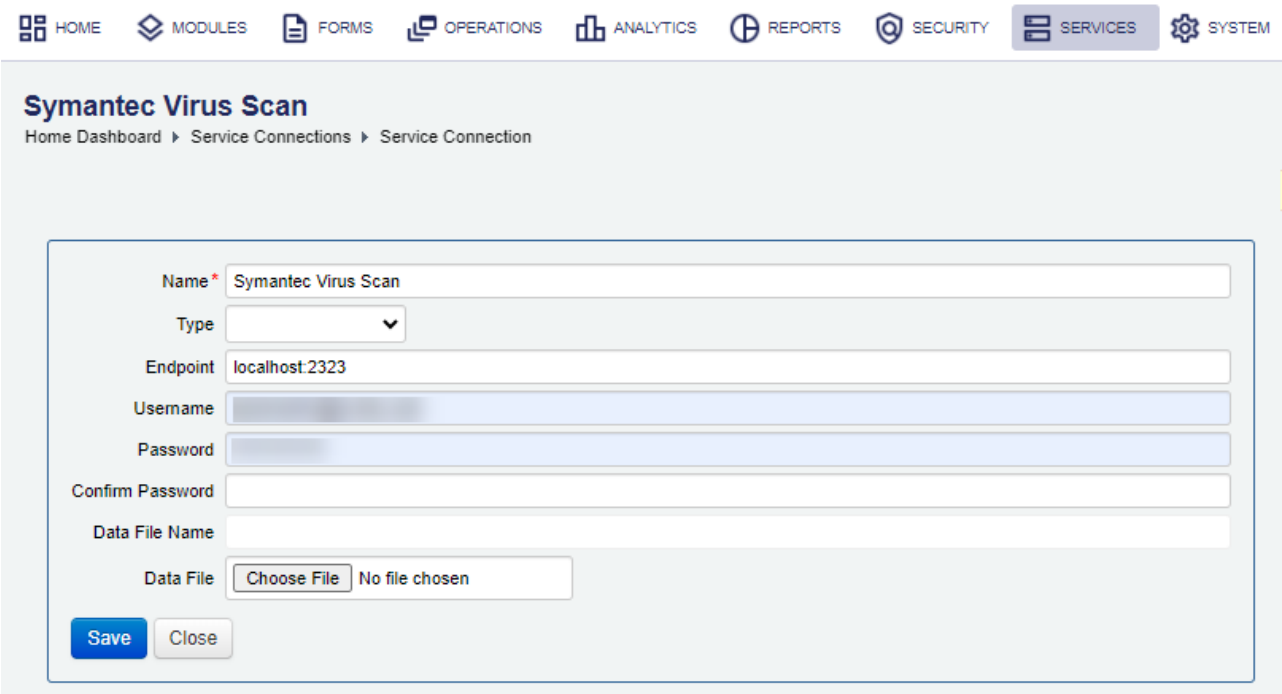
Symantec Virus Scan Service Connection

Manager allows you to create and configure a Symantec Virus service connection to connect to the Symantec Virus Scan engine.

> NOTE
>
> Make sure the Symantec Virus Scan engine is running and is correctly configured before configuring a service connection.

To configure a service connection:

1. Select Services > Service Connections.
2. Locate a Symantec Virus Scan service connection to edit it or click New to create a Symantec Virus Scan service connection.



3. Provide a unique service connection name in the Name field, for example, Symantec Virus Scan.
4. Leave the Type blank.
5. Update the Endpoint with a server name and port number, where the Symantec Protection Engine is installed, in the following format `hostName:portNumber`. If the Symantec Protection Engine runs on the same server as Manager and uses the default port number, you can leave the default configuration as `localhost:1344`.

6. Provide a user name in the Username field, which is used to authenticate the service connection. For more information, see Symantec documentation.
7. Provide a password in the Password field, which is used to authenticate the service connection.
8. Click Save to update the changes.

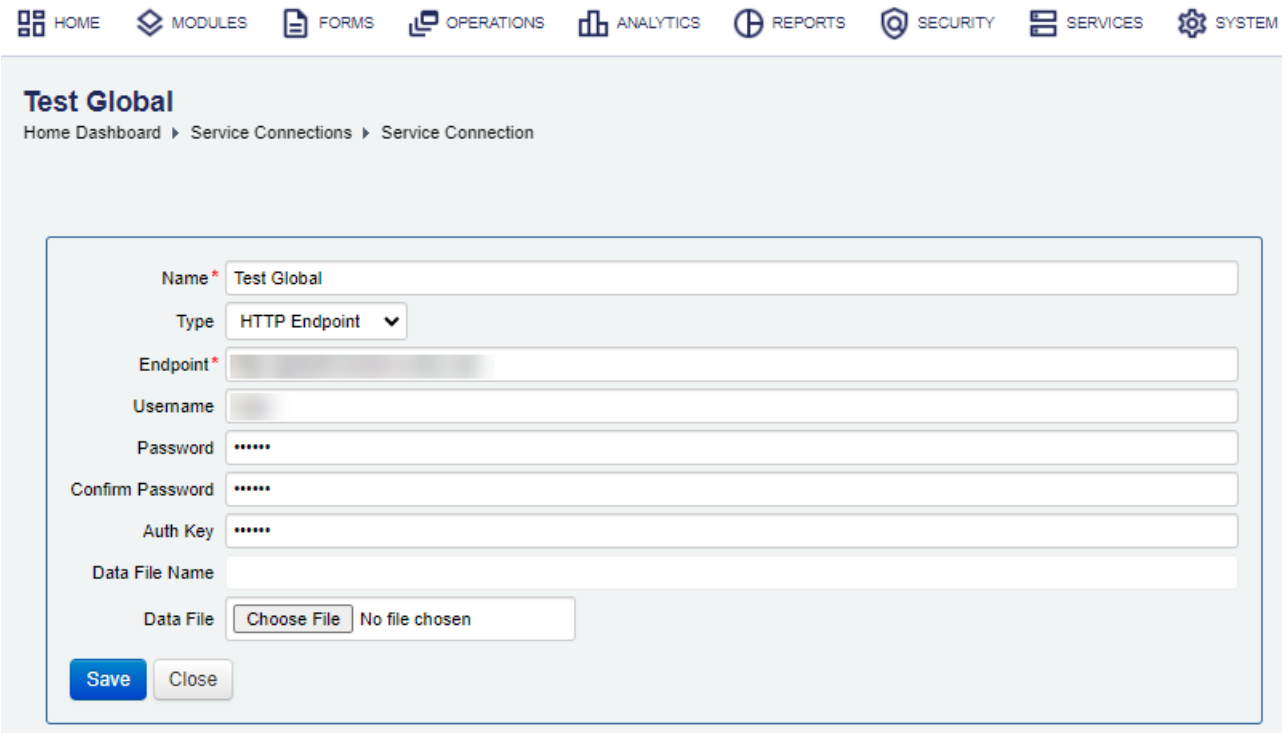
Next, learn how to configure attachment virus scan service.

HTTP Endpoint Service Connection

Manager allows you to create and configure service connections to communicate with external services over the HTTP[1] protocol by sending requests to specified endpoints[2].

To configure an HTTP Endpoint service connection:

1. Select Services > Service Connections.
2. Locate an HTTP Endpoint service connection or click New to create a new service connection.



3. Provide a unique service connection name in the Name field.
4. Select the HTTP Endpoint as a service connection type from the Type dropdown list.
5. Provide a URL to an external service in the Endpoint field, for example, `https://my-domain/my-servicer/secure/my-api`.
6. Enter a user name to authenticate a service connection in the Username field.

---

[1]HTTP (HyperText Transfer Protocol) is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.
[2]Endpoint is a web address (URL) at which clients of a specific service can gain access to it. By referencing that URL, clients can get to operations provided by that service.

7. Enter a password to authenticate a service connection in the Password field.
8. Confirm the password in the Confirm Password field.
9. Provide a secret key to authenticate a service connection in the Auth Key field.
10. Choose a certificate file from the Data File Name file picker.
11. Click Data File to upload a data file, if it's required for this service connection.
12. Enter request parameters in the Custom Parameters field.
13. Click Save to update the changes.

Next, learn about other service connection types.

DocuSign Service Connection

Manager allows you to create and configure a service connection to connect to DocuSign[1] using basic username and password authentication mechanism. Various services rely on this service connection to exchange contracts and sign documents electronically.

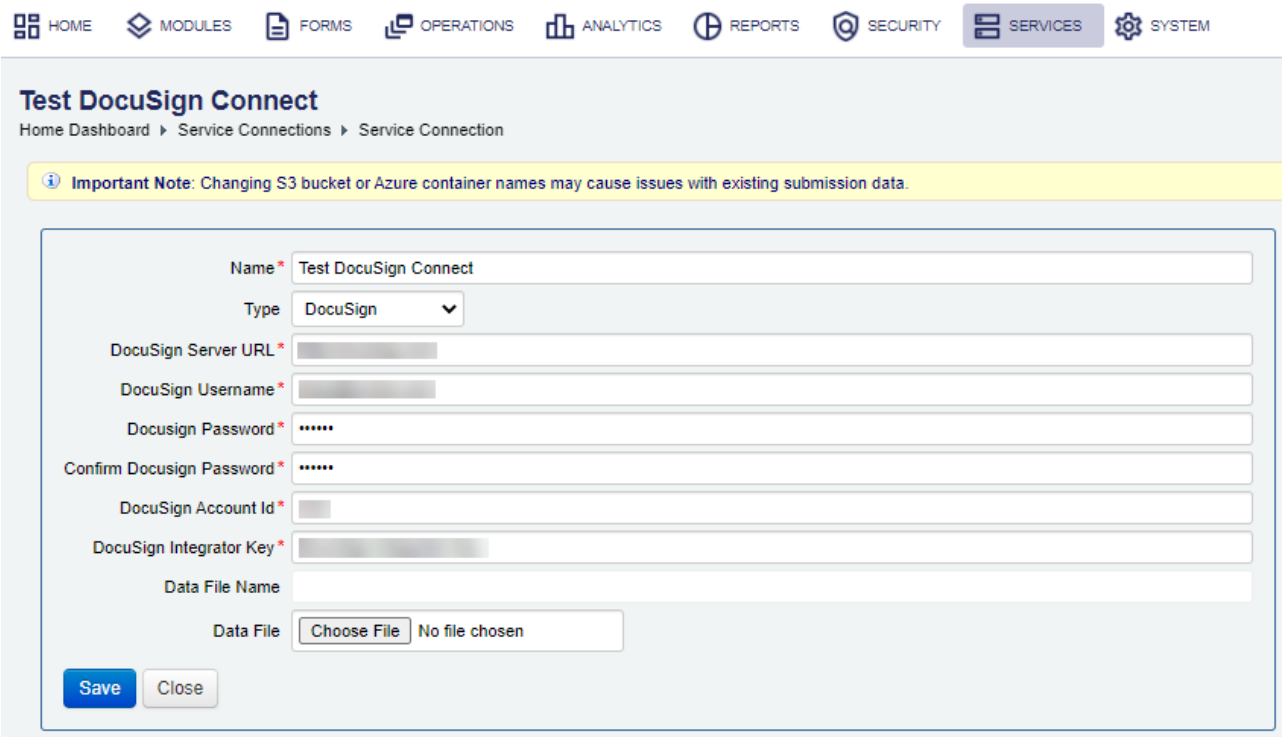
NOTE

DocuSign is in the process of retiring basic user password authentication which will no longer be supported after March 2023. For more information, see DocuSign API Basic User Password Authentication Retirement.
The DocuSign OAuth service connection replaces the DocuSign service connection, which uses basic username and password authentication.
However, you still can use the DocuSign service connection after March 2023 provided you will have configured it according to the new guidelines described below.

To configure a service connection:

[1]DocuSign is the leader in eSignature transaction management. The Temenos Journey Manager platform uses this solution to speed the rate at which all digital documents are completed, signed, and returned.

1. Select Services > Service Connections.
2. Locate a DocuSign service connection or click New to create a new service connection.



3. Provide a unique service connection name in the Name field.
4. Select DocuSign type from the Type dropdown list.
5. Enter one of the following DocuSign service's host names in the DocuSign Server URL field: | 22.10This feature was updated in 22.10.
   - Development environment - `account-d.docusign.com`
   - Production environment - `account.docusign.com`

   > NOTE
   >
   > Do not include the `https://` prefix and trailing `/`

6. Provide a username in the DocuSign Username field to log in on to the service.
7. Provide a user password in the DocuSign Password field.

   > NOTE
   >
   > The username and password must contain non-empty and valid values to enable the form to submit, but the values are ignored during user authentication.

8. Confirm the password in the Confirm DocuSign Password field.

9. Provide a User ID of your DocuSign account in the DocuSign Account Id. | 22.10This feature was updated in 22.10.

   NOTE

   Do not use the user's Account ID anymore.

10. Provide an integration key in the DocuSign Integration Key field. This key, also known as an API key, is a unique GUID[1] used to identify your API integration with DocuSign.
11. Click Choose File to upload a RSA[2] private key file, which you obtain from the DocuSign App page. | 22.10This feature was updated in 22.10.
12. Click Save to update the changes.

After the service connection is configured, you must manually grant one-time consent to your DocuSign API user.

To grant one-time consent:

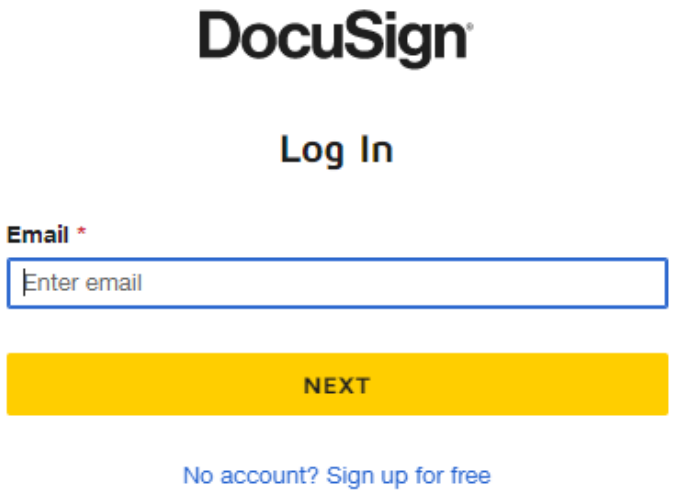
1. Populate your DocuSign Integration Key value into one of the following URLs:
   - Development environment - `https://account-d.docusign.com/oauth/auth?response_type=code&scope=signature+impersonation&client_id=<Integration Key>&redirect_uri=https://httpbin.org/get`
   - Production environment - `https://account.docusign.com/oauth/auth?response_type=code&scope=signature+impersonation&client_id=<Integration Key>&redirect_uri=https://httpbin.org/get`
2. Copy the URL and paste it in a browser to grant OAuth authentication access to a configured DocuSign API user permanently. You will be redirected to the DocuSign log in page to enter the user's email address and password to authenticate. You have to do this only once after the DocuSign API user is created or changed.

   NOTE

   You must configure the valid DocuSign Integration Key for the link to work correctly.

---

[1]A GUID (global unique identifier) is a term for a number that its programming generates to create a unique identity for an entity such as a Word document. GUIDs are widely used in Microsoft products to identify interfaces, replica sets, records, and other objects.
[2]RSA is a public-key cryptosystem that is widely used for secure data transmission.



After you log in successfully, the user is granted OAuth access. The next page shows some information regarding the authentication access, so you can check it was successful, but no further actions are required from you.

> NOTE
>
> Success is indicated by redirecting to a page at `httpbin.org`, which shows HTTP request details. You can ignore it and close the tab.

Next, learn more about DocuSign integration.

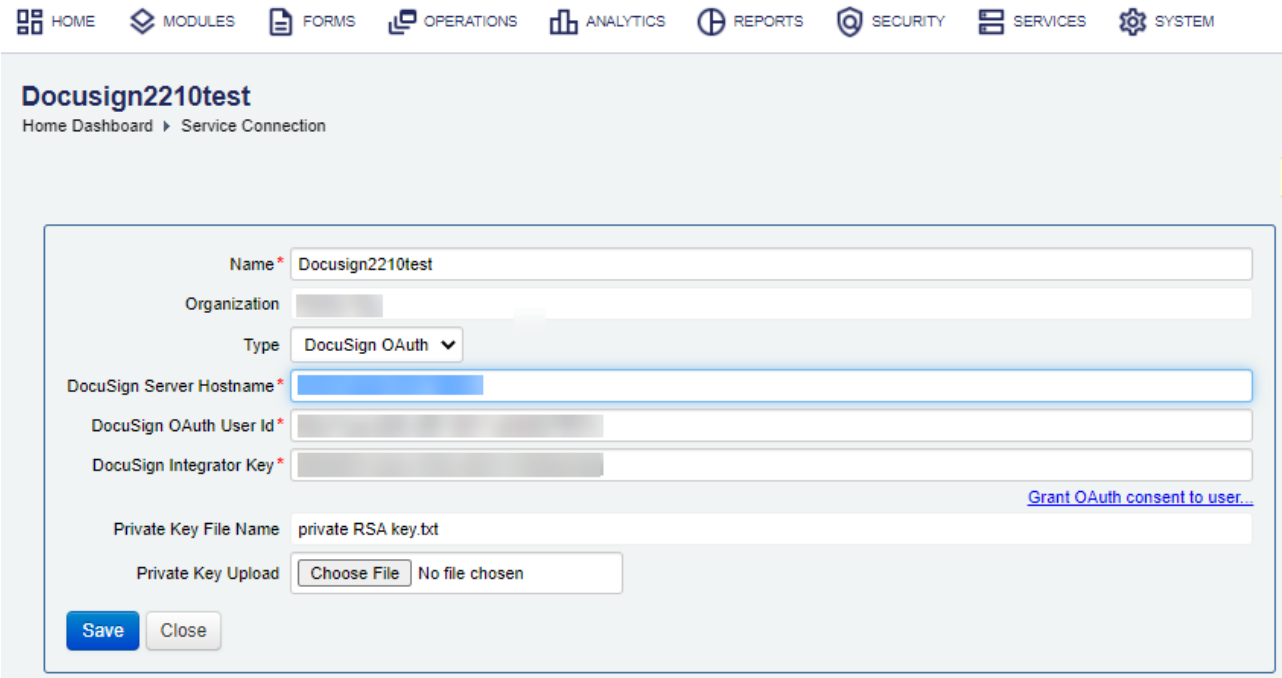
DocuSign OAuth Service Connection

Manager allows you to create and configure a service connection to connect to DocuSign[1] using OAuth user authentication mechanism. Various services rely on this service connection to exchange contracts and sign documents electronically.

> NOTE
>
> DocuSign is in the process of retiring basic user password authentication which will no longer be supported after March 2023. For more information, see DocuSign API Basic User Password Authentication Retirement.
> The DocuSign OAuth service connection replaces the DocuSign service connection, which uses basic username and password authentication.

To configure a service connection:

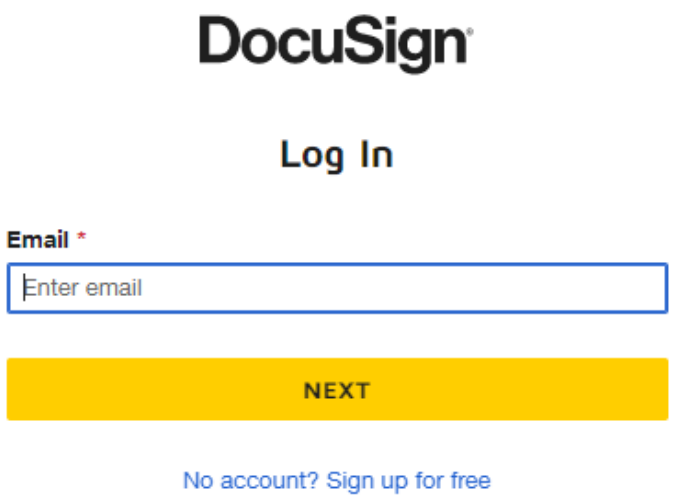
1. Select Services > Service Connections.
2. Locate a DocuSign OAuth service connection or click New to create a new service connection.



[1]DocuSign is the leader in eSignature transaction management. The Temenos Journey Manager platform uses this solution to speed the rate at which all digital documents are completed, signed, and returned.

3. Provide a unique service connection name in the Name field.
4. Select DocuSign OAuth as a service connection type from the Type dropdown list.
5. Enter a host name to the DocuSign service in the DocuSign Server Host Name field, for example, `my-test.com`.
6. Provide a User ID of your DocuSign account in the DocuSignOAuth User Id field.
7. Provide an integration key in the DocuSign Integration Key field. This key, also known as an API key, is a unique GUID[1] used to identify your API integration with DocuSign.
8. Click Choose File to upload a RSA[2] private key file, which you obtain from the DocuSign App page.
9. Click the Grant OAuth consent to user link to grant OAuth authentication access to a configured DocuSign API user permanently. You will be redirected to the DocuSign log in page (defined in the DocuSign Server Host Name) to enter the user's email address and password to authenticate. You have to do this only once after the DocuSign API user is created or changed.

> NOTE
>
> You must configure the valid DocuSign Integration Key for the link to work correctly.



After you log in successfully, the user is granted OAuth access. The next page shows

[1]A GUID (global unique identifier) is a term for a number that its programming generates to create a unique identity for an entity such as a Word document. GUIDs are widely used in Microsoft products to identify interfaces, replica sets, records, and other objects.
[2]RSA is a public-key cryptosystem that is widely used for secure data transmission.

some information regarding the authentication access, so you can check it was successful, but no further actions are required from you.

10. Click Save to update the changes.

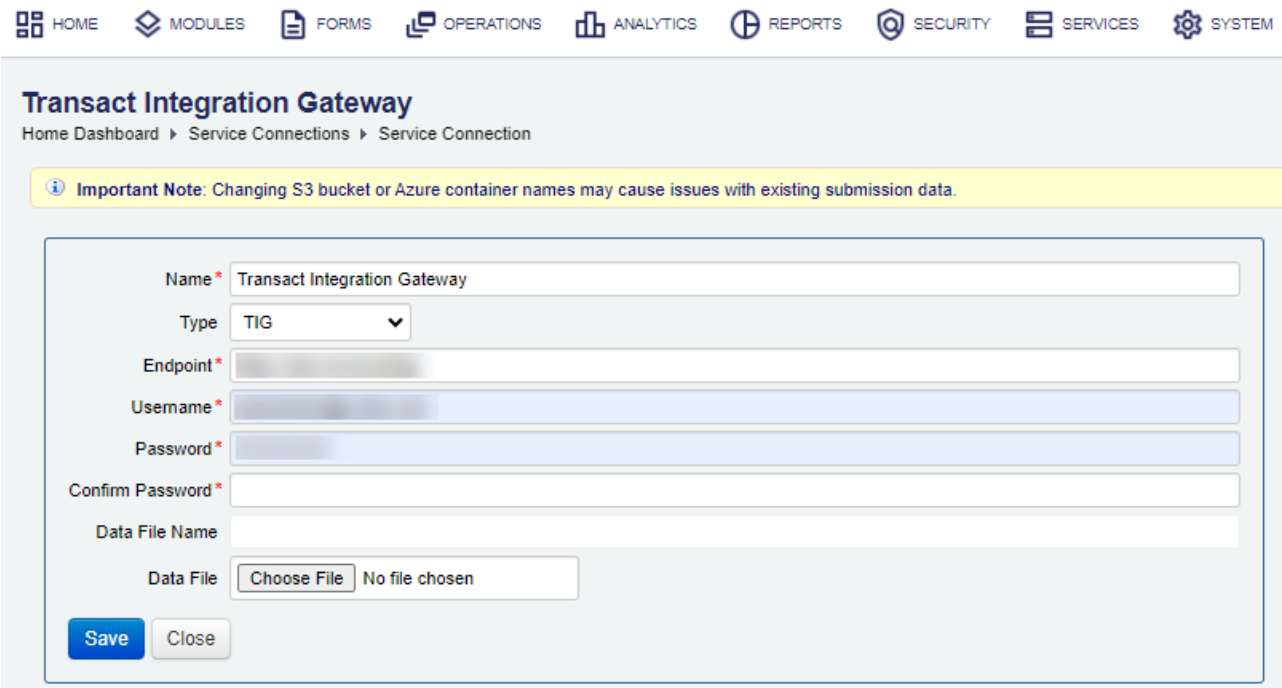
Next, learn more about DocuSign integration.

Integration Gateway Service Connection

Journey Manager allows you to create and configure a Transact Integration Gateway (TIG) service connection to Transact Integration Gateway.

To configure a service connection:

1. Select Services > Service Connections.
2. Locate a Transact Integration Gateway service connection or click New to create a new service connection.



3. Provide a unique service connection name in the Name field.
4. Select the TIG type from the Type dropdown list.
5. Enter the Endpoint as a URL of a TIG server you want to connect to, for example, `https://my-domain.com/tig/`.
6. Enter the Username.
7. Enter the Password.
8. Confirm the password in the Confirm Password field.
9. Provide the Data File Name.
10. Click Data File to upload a data file, if it's required for this service connection.
11. Click Save to update the changes.

Next, learn about other service connection types.

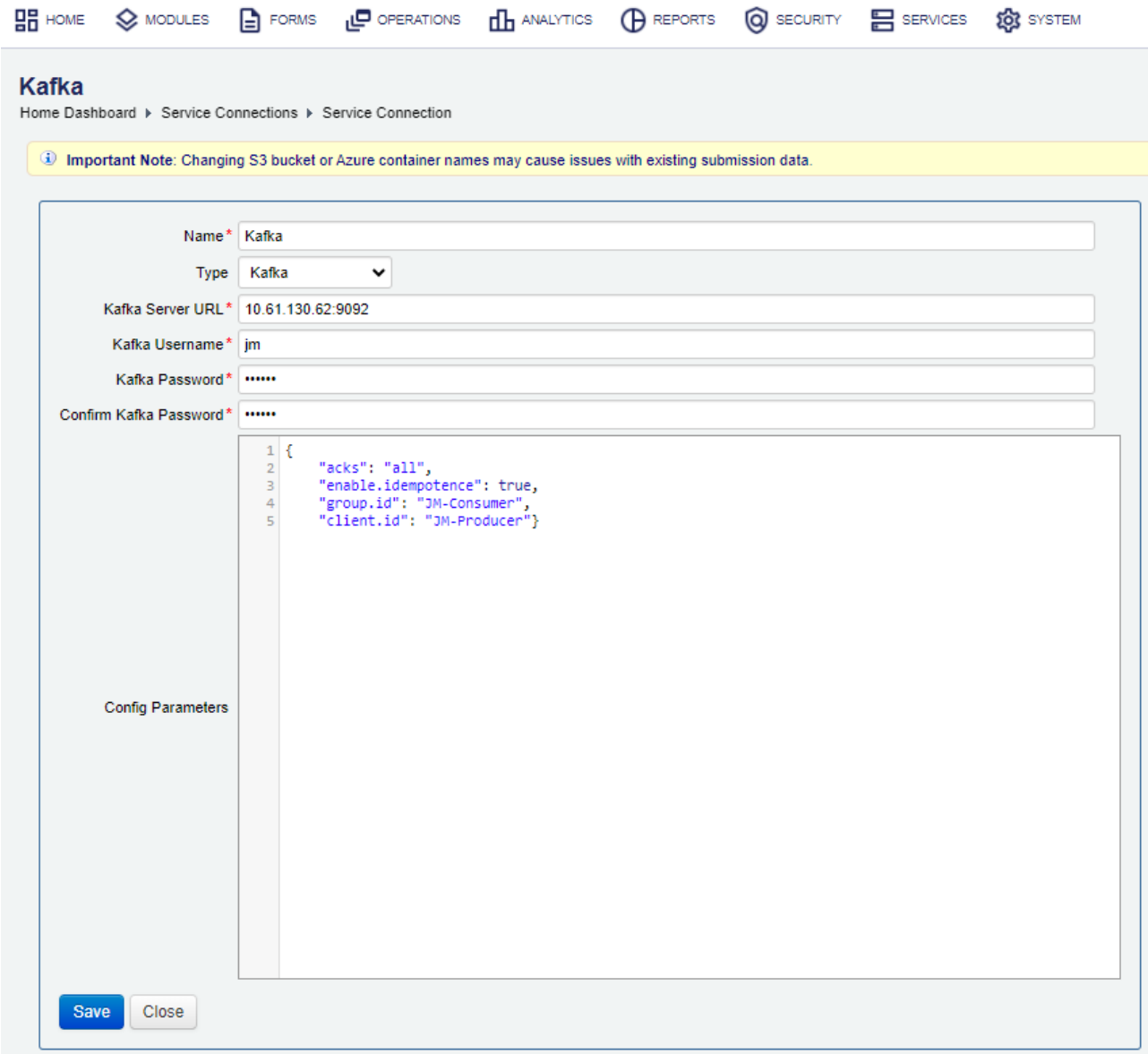
Kafka Service Connection

Manager allows you to create and configure a Kafka service connection to connect to an Apache Kafka[1] producer. Various services can use this service connection to publish events. Typically, you need this service connection for building services based on the Event-Based Architecture.

To configure a service connection:

[1]Apache Kafka is a distributed event store and stream-processing platform, which provide a unified, high-throughput, low-latency platform for handling real-time data feeds. For more information, see https://kafka.apache.org/intro

1. Select Services > Service Connections.
2. Locate a Kafka service connection or click New to create a new service connection.



3. Provide a unique service connection name in the Name field.
4. Enter an organization name in the Organization field or leave it blank.
5. Select Kafka as a service connection type from the Type dropdown list.
6. Enter a server URL, where a Kafka producer is installed, in the Kafka Server URL field, for example, `my-kafka-host.com`
7. Provide a user name to connect to a Kafka producer in the Kafka Username field.
8. Provide a user password to connect to a Kafka producer in the Kafka Password.
9. Confirm the user password in the Confirm Kafka Password field.

10. Enter Kafka producer parameters to establish a connection in the Config Parameters field. For more information, see producer configuration.
11. Click Save to update the changes.

Kafka Producer Configuration

The Kafka service connection supports the limited set of parameters that you can use to configure a connection to the Apache Kafka producer. Some of the parameters are set to default values in Manager and some are in a producer. We recommend checking Kafka documentation to learn about these parameters and their default values.

> NOTE
>
> Parameters are version specific, so check the Kafka version and Kafka client library version to ensure you use the correct parameters.

Supported producer configuration parameters:

`acks`
: default is `all`

`enable.idempotence`
: default is `true`

`client.id`
: default is `JM-Producer`

`compression.type`

`client.dns.lookup`

`connections.max.idle.ms`

`delivery.timeout.ms`

`group.id`
: To join a group, so listeners don't read an event multiple times. | 24.04 This feature was introduced in 24.04

`max.block.ms`

`max.request.size`

`metadata.max.age.ms`

`metadata.max.idle.ms`

`partitioner.class`

`reconnect.backoff.max.ms`

`reconnect.backoff.ms`

`request.timeout.ms`

`retries`

`retry.backoff.ms`

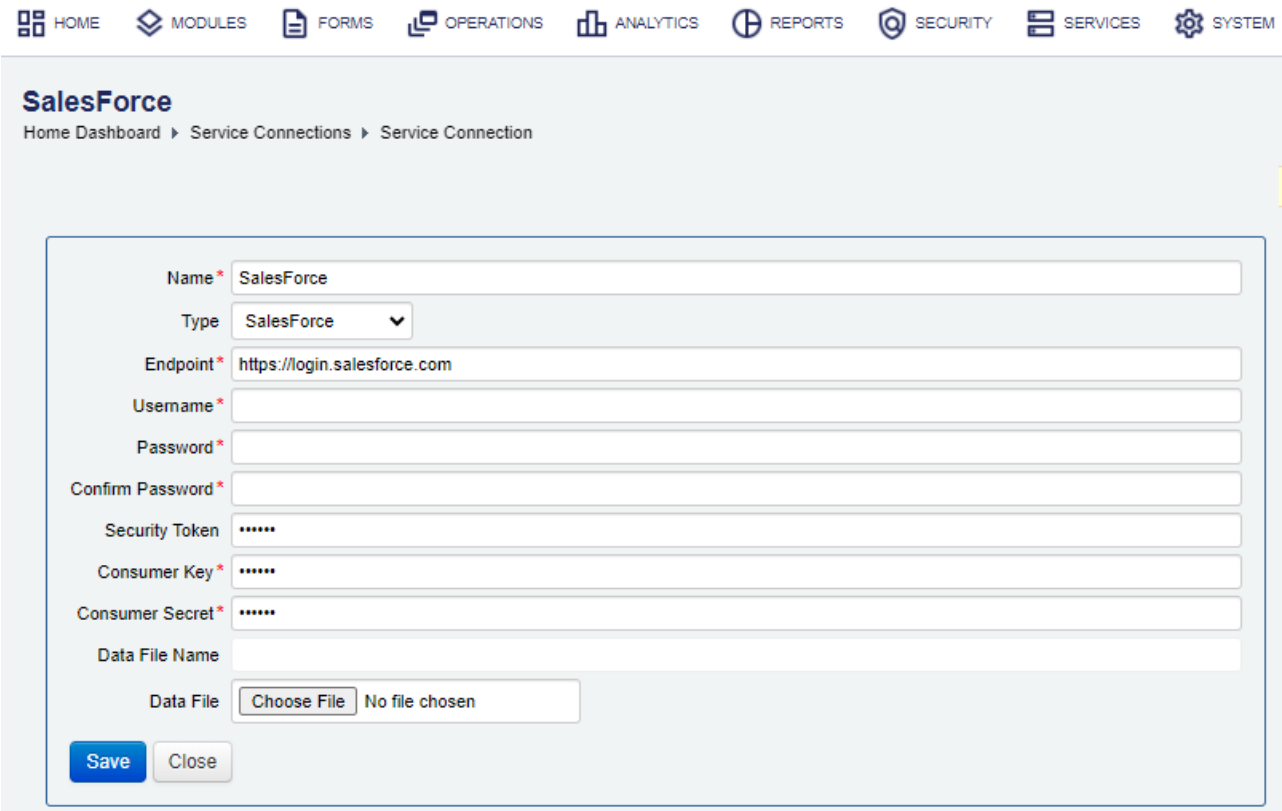
Next, learn about other service connection types.

Salesforce Service Connection

Manager allows you to create and configure a SalesForce[1] service connection to an external Salesforce service.

To configure a service connection:

1. Select Services > Service Connections.
2. Locate a SalesForce service connection or click New to create a new service connection.



3. Provide a unique service connection name in the Name field.
4. Select the HTTP[2] Endpoint type from the Type dropdown list.
5. Enter the Endpoint as a URL for the external service.
6. Enter the Username.
7. Enter the Password.
8. Confirm the password in the Confirm Password field.

---

[1]Salesforce is the customer relationship management (CRM) platform.

[2]HTTP (HyperText Transfer Protocol) is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

9. Enter the Security Token.
10. Provide the Consumer Key.
11. Enter the Consumer Secret.
12. Specify the Data File Name.
13. Click Data File to upload a data file, if it's required for this service connection.
14. Click Save to update the changes.

Next, learn about other service connection types.

Services Overview

Journey Manager comes with services that are software modules configured to perform specific business-related functionality, usually by executing Groovy[1] scripts. Services are often used to call external systems and other services hosted on premise or on cloud, such as the Amazon[2] or Azure[3] clouds. You can invoke services in various points of an onboarding journey as well as at different Transact Function triggers . An invoked service can call another service in turn making a chain of calls.

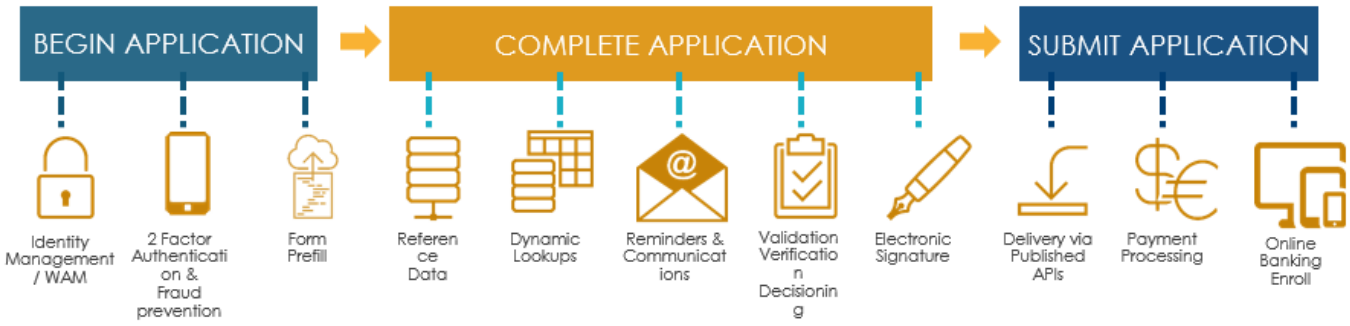
A customer onboarding journey may use the following Manager's services:

- Data prefill for existing customers or using photo ID
- Address look-up and validation
- Identity verification, KYC[4], OFAC[5]
- Deposit and credit decision-making
- Save and resume of a form application
- Follow-up leads
- eSign documents
- Channel cross-over
- Core system integration
- Cross-sell next product

A typical customer acquisition application may use the following services:

---

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.
[2]Amazon Web Services (AWS) is a comprehensive, evolving cloud computing platform provided by Amazon. It provides a mix of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings.
[3]Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers.
[4]Know your customer (KYC) requires that banks have a reasonable belief that they know the true identity of customers and establish reasonable and practical risk-based procedures for verifying the identity of each customer. The procedures must include account opening procedures that specify the identifying info that will be obtained from each customer, such as name, date of birth, address, ID number (e.g. SSN).
[5]Office of Foreign Asset Control (OFAC) enforces economic and trade sanctions. Banks must perform an OFAC screen on customers opening an account.



You can potentially have hundreds of distinct services plus several versions of each service, so Manager defines the following service categories to group the services:

- Core Global Services (Core): Generic Manager services that are part of the installation.
- Form Services: Usually customer specific Groovy[1] services.
- Job Services: Usually customer specific services.
- Transact Functions: Customer specific services using Transact Fluent API.

Each service has a service type, which defines what functionality it implements, for example, virus scanning. There can be multiple services per every service type. However, only one of them can be set as the default service for each type. You can define your own service types and create corresponding services to implement specific requirements, for example, to notify customers via SMS, using Fluent Groovy API.

A service can be:

- Organization-specific
- Global

If a service is defined for one specific organization, it is not available to all other organizations and their forms, as well as on the global level including administrators of other organizations, so it can't be used outside that organization. This is a most common case of defining a service.

A service can be assigned to multiple organizations so only these organizations and their forms can make use of it. | 24.10 This feature was introduced in 24.10

The global services don't have any organization assigned to them, so they can be used in global forms across multiple organizations. This is especially important for Manager installations in

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

multi-tenanted environments. You can also use global services in standard organization-specific forms

We only allow the following 3 types of Fluent services to be global | 23.10 This feature was introduced in 23.10:

- Fluent Function
- Delivery Function
- Fluent Job Action

When you create a global Fluent function of the above types, you don't need to select an organization it belongs to - the function resolves it from a submission. This is important to understand to design global functions correctly. Also, you must have a global access to create global services. For more information, see Create a Service, Create a Transact Function and Create a Job Service.

Some services must be strictly organization-specific, for example, the Submission Data Storage service, so there is absolutely no chance for multiple tenants to write to the same data storage.

For a collaboration job, a delivery channel belonging to an organization is used.

Services can have various parameters, which you can customize to control their run-time behavior.

Some services require a service connection, such as the AWS SQS service connection, to communicate with external systems and services.

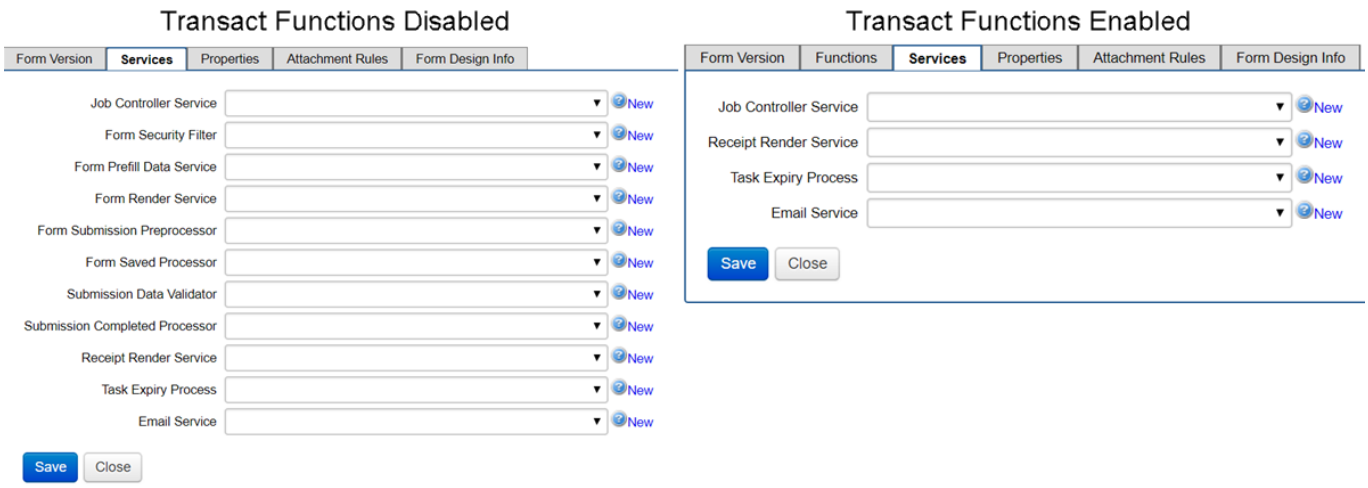
You can migrate existing services from one environment to another to improve your development process.

As services may be complex in their implementations, it can be difficult and time consuming for you to troubleshoot any errors associated with forms that use these services. Manager provides you with a quick way to access service logging information, using Groovy Service Log, where you can find some key service stats generated during its execution.

Forms use services extensively to process user input and fetch all required data, which is a big part of complex customer onboarding journeys. More often than not, you may want to use Transact functions to call services from Maestro forms.

Manager allows you to configure services that are used by a form. Additionally, you have the option of enabling or disabling Transact Functions when building a TM form version in Maestro

editor. However, you must be aware that enabling Transact Functions limits the range of services available to the form, which is illustrated below.



Services available to a form with Transact Functions disabled and enabled

Next, learn about service types.

Service Types

Manager uses service types to define services for various tasks, such as rendering forms or dynamic data calls.

A service type:

- Acts as a template by providing the applicable set of parameters to define a service.
- Determines how a service can be used.
- Can have a default service, if one is selected as the default.

Each service type belongs to one or more categories and has a defined scope.

- Core Global Services (Core): Generic Manager services that are part of the installation.
- Form Services: Usually customer specific Groovy[1] services.
- Job Services: Usually customer specific services.
- Transact Functions: Customer specific services using Transact Fluent API.

NOTE

To edit or remove services in the Core Global Services category, you must have the Core Service Edit permission.

A service scope can be:

- Global: a service has a global access within Manager.
- Organization: a service has access within a selected organization only.
- Both: Global or Organization.

Manager has the following predefined service types, by which services are grouped:

Service Type	Category	Scope	Description
Database Integrity Checker Service	Core Global Service	Global	Check integrity of a database used by a Manager server. | 23.04 This feature was introduced in 23.04

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

Data Retention Management	Core Global Service	Global	Performs data purging including PII[1] data.
Delivery Function	Transact Function	Both	Performs the submission data delivery using Fluent functions.
Delivery Process	Form Service		Passes the submission data and attachments, as well a customizable Groovy delivery service across, to perform a delivery.
Dynamic Data	Form Service		Obtains data dynamically from external systems or services. Forms can use this service for data pre-fill, lookup or validation.
Email Queue	Core Global Service		Processes emails stored in the email queue.
Email Service	Core Global Service Form Service		Provides a default SMTP[2] email service. You can also create a Groovy email service.

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

[2]SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with one of two other protocols, POP3 or IMAP, that let the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving e-mail. On Unix-based systems, sendmail is the most widely-used SMTP server for e-mail. A commercial package, Sendmail, includes a POP3 server. Microsoft Exchange includes an SMTP server and can also be set up to include POP3 support.

Event Configuration	Core Global Service		Provides a default Event Configuration Storage service. | 22.10 This feature was introduced in 22.10.
Event Listener Configuration	Core Global Service		Defines the Event Listener Configuration servicethat consumes Kafka topics and process them, for example, as a REST API request. | 24.04 This feature was introduced in 24.04
Find Delivery Process ID	Core Global Service		Updates the delivery process ID assigned by Adobe LiveCycle[1] for submissions delivered to an Adobe LiveCycle process.
Fluent Function	Transact Function	Both	Implements any service adhering to Transact Fluent API
Form Meta Data	Core Global Service	Global	Renders the metadata[2] related to the form as HTML
Form PDF Pre Render	Core Global Service		This service is called when a new Composer SmartForm with an XDP form template is created. It uses an Adobe LiveCycle orchestration to pre-render a PDF from the

[1]Adobe LiveCycle Enterprise Suite (ES4) is an service-oriented architecture Java EE server software product from Adobe Systems used to build applications that automate a broad range of business processes for enterprises and government agencies. LiveCycle ES4 is an enterprise document and form platform that allows capturing and processing information, delivering personalized communications, and protecting and tracking sensitive information. It is used for purposes such as account opening, services and benefits enrollment, correspondence management, request for proposal processes, and other manual based workflows.
[2]Metadata is data that describes other data. Meta is a prefix that in most information technology usages means "an underlying definition or description." Metadata summarizes basic information about data, which can make finding and working with particular instances of data easier. For example, author, date created and date modified and file size are examples of very basic document metadata. Having the abilty to filter through that metadata makes it much easier for someone to locate a specific document.

			XDP template. This PDF is used for form rendering, which speeds up rendering and reduces the load on LiveCycle's processes.
Form Prefill	Form Service		Retrieve prefill XML data for a form. The service can be defined at a form version level, otherwise a global default service is used. The service is called at form render time.
Form Saved Processor	Form Service		This service is called when a form is saved to Manager, and the form version in question has been associated with a Form Saved Processor service. The default service allows you to specify custom logic in a Groovy script.
Form Security Filter	Form Service		Provides a security filter for a form.
Form Server Load Balancer	Core Global Service		This service is called by various operations involving LiveCycle -for example, rendering forms and receipts - and is used to ensure capacity to perform the operation is available. It also performs load balancing if more than one LiveCycle server is available.
Form Submission Access Controller	Core Global Service		Provides security features based around the completion of a form.
Form Version Selector	Core Global Service Form Service		Selects the appropriate form version to render a form
Groovy Service	Form Service	Both	Provides Groovy[1] code that can be reused by other Groovy services in Manager.

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

Inbox Processor	Core Global Service	Defines services that consume events from Kafka topics. The default service is Event Inbox Processor Service. | 24.04 This feature was introduced in 24.04
Job Action	Job Service	Provides a Job Action service related to collaboration jobs. You can implement it using the `GrooveyJobActionService` or the `FluentJobActionService`. See `com.avoka.fc.core.service.job.impl.JobActionService` API.
Job Action Wait	Job Service	Provides a Job Action Wait service related to collaboration jobs. This service waits until all the steps of the previous Invoked actions have been completed. You can implement it using the `GrooveyJobActionService` or the `FluentJobActionService`. See `com.avoka.fc.core.service.job.impl.JobActionWaitService` API.
Job Controller	Job Service Form Service	Creates a new collaboration job for a form submission and then processes it. This service needs to be associated with the form version.
Job Delivery	Job Service	Provides a Job Delivery service related to collaboration jobs. This service performs actual delivery within a collaboration job. This service is used in conjunction with the Job Delivery Wait service to kick off the standard Delivery process. The delivery process is completed by the Transaction Processing Scheduled Job, not the job controller service. See `com.avoka.fc.core.service.job.impl.JobDeliveryService` API.
Job Delivery Wait	Job Service	Provides a Job Delivery Wait service related to collaboration jobs. This service waits for a delivery to be complete as a part of a collaboration job. The Job Delivery Wait service is an asynchronous mechanism. When the delivery is completed by the Transaction Processing job it calls a callback method that allows the Job Delivery Wait to com-

		plete. See `com.avoka.fc.core.service.job.impl.JobDeliveryWaitService` API.
Job Expiry	Job Service	Provides a Job Expiry service related to collaboration jobs. You can implement it using the `GroovyJobActionService` or the `FluentJobActionService`. .See `com.avoka.fc.core.service.job.impl.JobExpiryService` API.
Job Form Start	Job Service	Provides a Job Form Start service related to collaboration jobs. This service associates the initial form submission with the initial step. It can add a submission processing status message to the initial submission. These appear in the user's submission history. It can also send a status update via email. See `com.avoka.fc.core.service.job.impl.JobFormStartService` API.
Job Process Message	Job Service	Provides a Job Process Message service related to collaboration jobs. This service adds a submission processing status messages to the specified Submission. These appear in the user's submission history. This action service can be used anywhere in the job but is especially useful in endpoint steps. It can also send a status update via email. See `com.avoka.fc.core.service.job.impl.JobProcessMessageService` API.
Job Receipt Wait	Job Service	Provides a Job Receipt Wait service related to collaboration jobs. This action polls the submissions until the receipts have completed. The generation of the receipts is completed by a separate Scheduled Job 'Transaction Processing' which by default runs every 5 minute. This action will complete when all receipts have been generated for this job's submissions. If any receipt has not completed the action returns a result of "In Progress". This will mean the Job Action / Job processing will stop and retry again in 1 minute. This is required as the existing Job Task Wait service does not wait for receipts to be created. See `com.avoka.fc.core.service.job.impl.JobReceiptWaitService` API.

Job Task Assign	Job Service	Provides a Job Task Assign service related to collaboration jobs. The Job Task Assign Service is used for creating and assigning tasks in Journey Manager. It has the configuration properties associated with: • Status Updates and Email: This service can add a submission processing status messages to the specified Submission. These appear in the user's submission history. It can also send a status update via email • Task Assignment: These properties are used in the creation and assignment of user and group tasks. Tasks can be categorized as: • Standard Tasks - to an authenticated user or a group. • Review Tasks - these are the similar to the standard tasks but form xml and attachments are copied from a previous submission. • Anonymous Save - A submission record is saved but not assigned to a user in the database. They are usually assigned to a email address, the user is sent a notification email. They click on a link. The user has to enter a challenge response before they have access to the task form. See `com.avoka.fc.core.service.job.impl.JobTaskAssignService` API.
Job Task Wait	Job Service	Provides a Job Task Wait service related to collaboration jobs. A Job Task Wait Service is paired in the 1 step with 1 or more Job Task Assign Action that creates a task. It provided an asynchronous wait mechanism described below in State Transition section. that allows the user time to complete the task via submission. When a submission

			comes in the Job Task Wait will execute. The code will check that JobTaskAssign actions have completed. • Not All Complete: it will go back and wait for more submissions. • All Complete: It will allow execution to the next step. See `com.avoka.fc.core.service.job.impl.JobTaskWaitService` API.
Media Type Scan	Core Global Service		Scans and checks the contents of an attachment matches the specified file extension. | 20.11 This feature was introduced in 20.11.
Outbox Processor	Core Global Service		Defines services that publish events into Kafka topics. The default service is Event Outbox Processor Service. | 22.04 This feature was introduced in 22.04.
Payment Gateway	Core Global Service	Both	Processes payments by connecting to a payment gateway and making a payment transaction.
Payment Reminder	Core Global Service		Sends out reminder emails about submissions that have outstanding payments older than a certain age; that is, the Reminder Age Hours service parameter. Emails are sent to administrators belonging to the Receive Outstanding Payment Alerts group. This service is periodically called by the payment reminder job.
Receipt Number	Core Global Service Form Service		Calculates receipt numbers for new submissions.
Render Composer	Core Global	Global	Renders HTML forms created in Maestro or Composer.

HTML Form	Service	
Render Form	Core Global Service Form Service	Uses an orchestration in LiveCycle[1] to render a form when it is opened to be filled in. | 17.10 This feature was removed in 17.10
Render Off-line Sub-mission Form	Core Global Service	Uses an orchestration in LiveCycle to render a form when it is submitted after being saved offline.
Render Receipt	Core Global Service	Renders a form's receipt using one of the following orches-trations: LiveCycle[2], PhantomJS[3], Static PDF, IText.

[1]Adobe LiveCycle Enterprise Suite (ES4) is an service-oriented architecture Java EE server software product from Adobe Systems used to build applications that automate a broad range of business processes for enterprises and government agencies. LiveCycle ES4 is an enterprise document and form platform that allows capturing and processing information, delivering personalized communications, and protecting and tracking sensitive information. It is used for purposes such as account opening, services and benefits enrollment, correspondence management, request for proposal processes, and other manual based workflows.
[2]Adobe LiveCycle Enterprise Suite (ES4) is an service-oriented architecture Java EE server software product from Adobe Systems used to build applications that automate a broad range of business processes for enterprises and government agencies. LiveCycle ES4 is an enterprise document and form platform that allows capturing and processing information, delivering personalized communications, and protecting and tracking sensitive information. It is used for purposes such as account opening, services and benefits enrollment, correspondence management, request for proposal processes, and other manual based workflows.
[3]PhantomJS is a scripted, headless browser used for automating web page interaction. PhantomJS provides a JavaScript API enabling automated navigation, screenshots, user behavior and assertions making it a common tool used to run browser-based unit tests in a headless system like a continuous integration environment. PhantomJS is based on WebKit making it a similar browsing environment to Safari and Google Chrome (before Chrome's fork of WebKit evolved into Blink). It is open-source software released under the BSD License.

	Form Service	
Repository Publish	Core Global Service	Publishes a single form to the LiveCycle[1] repositories
Repository Synchronize	Core Global Service	Brings the LiveCycle repositories up to date after the set of form versions has changed.
Scheduled Service	Core Global Service Form Service	Executes custom logic defined as a Groovy script.
Submission Completed Processor	Form Service	Processes completed submissions.
Submission Data Extraction	Core Global Service	Provides a service to extract the XML data document from a HTML form submission.
Submission Data Storage	Core Global Service	Stores and retrieves submission, receipt and attachment data.
Submission Data Validator	Form Service	Validates submitted data.

[1]Adobe LiveCycle Enterprise Suite (ES4) is an service-oriented architecture Java EE server software product from Adobe Systems used to build applications that automate a broad range of business processes for enterprises and government agencies. LiveCycle ES4 is an enterprise document and form platform that allows capturing and processing information, delivering personalized communications, and protecting and tracking sensitive information. It is used for purposes such as account opening, services and benefits enrollment, correspondence management, request for proposal processes, and other manual based workflows.

Submission Delivery Controller	Core Global Service	Performs submission delivery, marks submissions as undeliverable, and sends reminders.
Submission Email Verification	Core Global Service	Sends email verification requests to users who have submitted a form.
Submission Preprocessor	Form Service	Process submission data before it's submitted.
Submission Receipt	Core Global Service	Creates and stores receipts for new submissions, and reattempts to render receipts for submissions where previous attempts failed.
Submission Service	Core Global Service	Performs submissions.
System Event Publisher	Core Global Service	Sends various server events to external systems. You can use both the Azure and AWS system event publishers.
Task Expiry Process	Core Global Service Form Service	Invokes an external process to notify interested parties that the task has expired.
Test Suite	Form Service	Provides a test suite.
Tracking Number	Core Global Service Form Service	Calculates tracking codes for new submissions.
Trans-	Core	Performs processing of all submissions.

action Processor	Global Service	
Two Factor Authentication	Core Global Service	Validates the second factor in two-factor authentication[1].
Virus Scan	Core Global Service Form Service	Scans attachments stored in Manager for viruses using various antivirus software.

You can create new services based on the existing types.

Next, learn how to view all services.

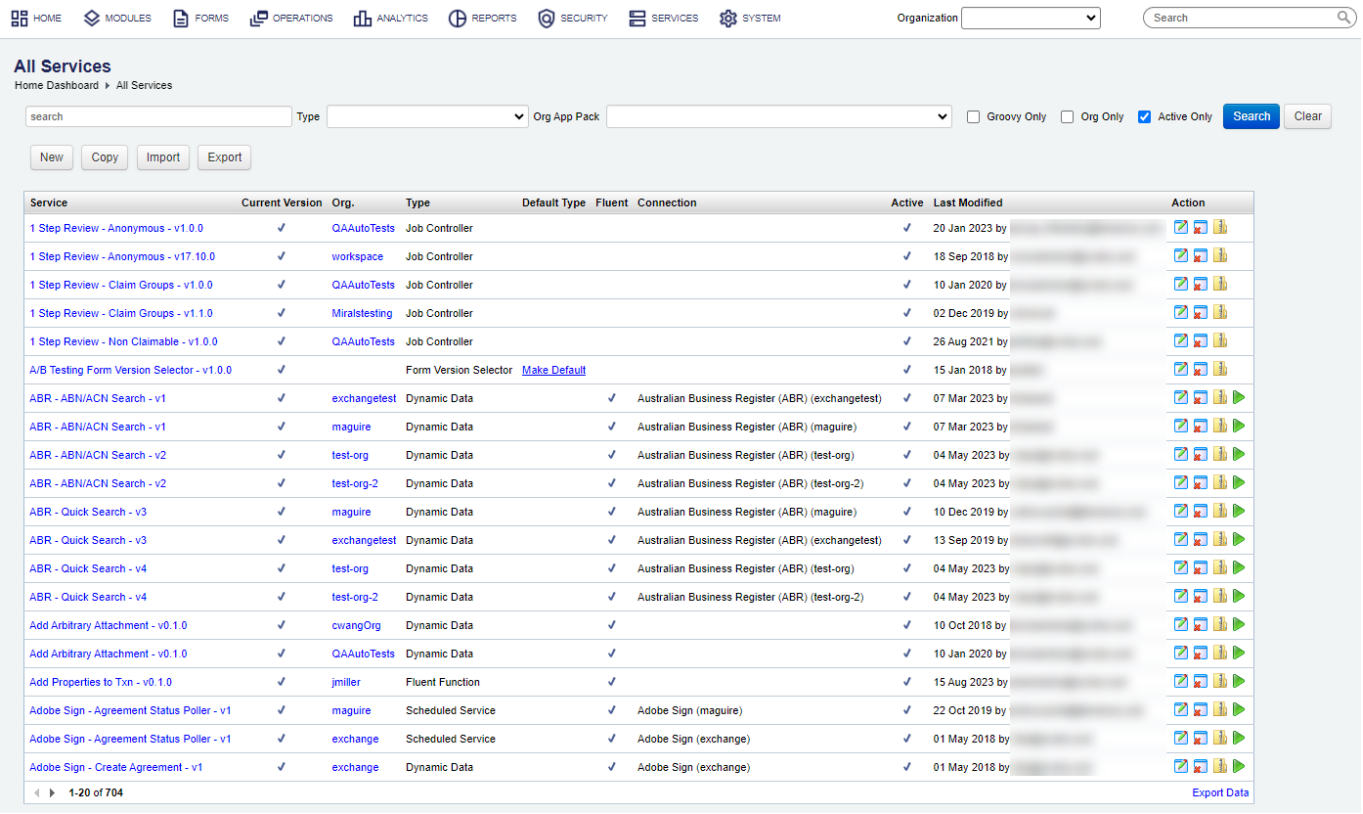
[1]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.

View Services

Manager allows you to view all services, so you can easily find and edit an existing service.

To view all services, select Services > All Services.

Services are shown page by page, with the maximum number of records per page configured in the users preferences.



The list displays the following details:

- Service: an organization-wide unique name of a service.
- Current Version: a version of the service currently used.
- Org: an organization the service belongs to or blank for a global service.
- Type: a type of a service.
- Default Type: whether this service is the default service for this service type or not. Only a global services can be marked as the default one.

- Fluent: marked with if this is a Fluent[1] service.
- Connection: a service connection used by this service, or blank if the service doesn't use or need a service connection.
- Active: marked with if this service is active. A service must be active so you could use it in your forms and applications.

  NOTE

  It is recommended to make a service not active rather then remove it when you don't need it any more.

- Last Modified: shows the date when this service was last modified and by whom.

To filter or search the services by one or more criteria, specify the following settings and click Search:

- Service name or part of a name.
- Service type.
- Org App Pack to see services belonging to a selected organization application package.
- Groovy Only to see Groovy[2] services only.
- Org Only to show organization service definitions only.
- Active Only.
- Organization to show services belonging to a selected organization.

Click Clear to reset the search criteria.

You can also perform the following actions:

- Create a service
- Configure a service
- Copy a service
- Remove a service

---

[1]Fluent is a programming style that can be applied to a wide range of programming languages. Across the Temenos Journey Manager platform, fluent is commonly associated with the Fluent SDK. The Fluent SDK is intended for use in developer integrated developer environment's (IDE) and contains libraries and project files for all the different services it supports.
[2]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

- Export services
- Import services
- Run a unit test to validate a selected service. This option is only available for Groovy, scheduled and virus scan services.
- Click Make Default to mark the selected service as a default one for all services of this type.

> NOTE
>
> To clear a service default status, open the service configuration and clear the Service Type Default checkbox.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to view form services.

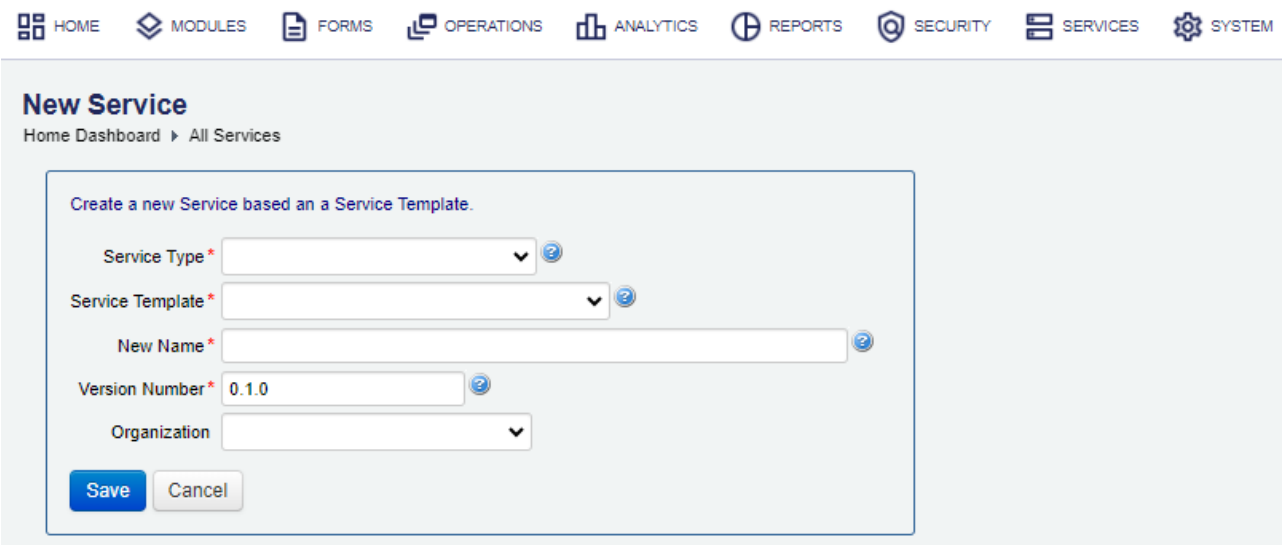
Create a Service

Manager allows you to create services so you can implement various onboarding application forms, for example, you can develop a service to process some specific data entered by a form applicant.

> NOTE
>
> You must be a Journey Manager administrator to create or configure this functionality.

To create a new service:

1. Select Services > All Services and click New.



2. Select a type of a service from the Service Type dropdown list. The service type dictates what service templates are available to create a new service from. Each service type may have multiple templates. A service type is also used to narrow down (filter) a list of services when viewing them.
3. Select a service template from the Service Template dropdown list to create your service from.
4. Give the new service a globally unique name in the New Name field. The service name needs to be globally unique, even if the service belongs to a specific organization. The name is automatically generated based on the selected template, so you may need to update it as required.
5. Edit a new version number in the Version Number field.
6. Select an organization from the Organization dropdown list so the service is associated only with forms and other entities belonging to that organization. If you do not select an

organization, the service will be globally available.

> NOTE
>
> You must have global access to create global Transact functions, both Fluent Functions and Delivery Functions, as well as global Job Actions. | 23.10 This feature was introduced in 23.10

> NOTE
>
> If the service is assigned to a specific organization, it is invisible to administrators of other organizations.

7. Click Save to update the changes.

While making a decision whether a service will be global or organization-specific, you should consider the following:

- A service is used by global forms in a multi-tenanted environment.
- System related service types such as Email Queue should be global, while other services such as Dynamic Data or Delivery Process might be candidates for organization-specific services.
- If you change the organization a service is assigned to, be aware that all entities belonging to other organizations, such as form versions, that are currently associated with this service will be disassociated and may have to be assigned new services.
- In most cases, it is better to create a new service for a particular organization rather than change a global service to be organization-specific.

> NOTE
>
> If you made a mistake configuring a service or don't need it any more, you can remove it.

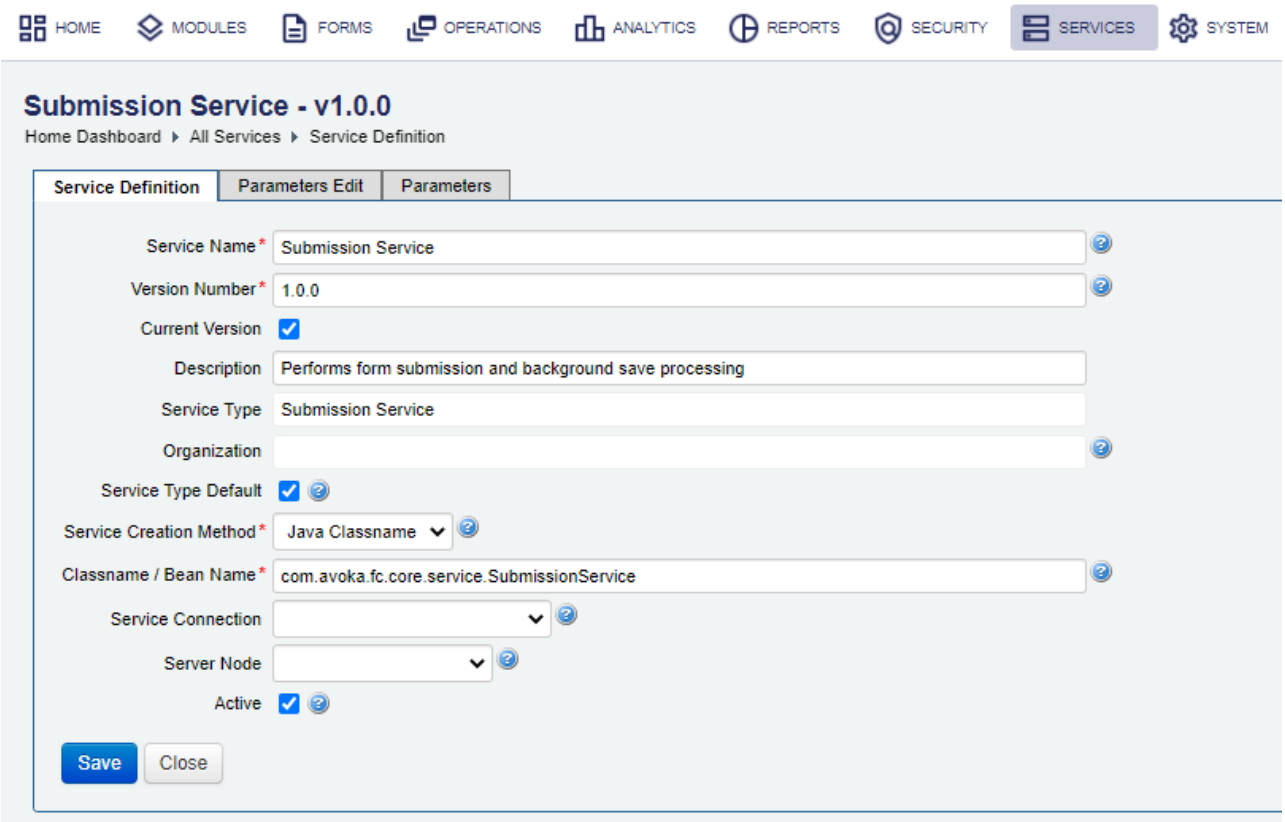
Next, learn how to configure a service.

Configure a Service

Manager allows you to configure services for your specific requirements. You usually need to do this after the service is created.

To configure a service:

1. Select Services > All Services.
2. Locate a service and click Edit.
3. Select the Service Definition tab to view or edit the standard service configuration.



4. Select the Current Version checkbox to make this service version current. This applies when there are multiple versions of this service, because a new version of this service is not automatically set as the current version.
5. Enter a meaningful service description in the Description field.
6. Select the Service Type Default checkbox to make this service as the default one, so it is executed for any service of this service type.

> NOTE
>
> You can also make a service as default from the list of all services.

7. Select the service creation method from the Service Creation Method dropdown list:
   - Java Classname
   - Spring Bean
8. Update the class name or bean name of the service in the Classname / Bean Name field. The class must be one of the classes pre-defined in Manager.

   NOTE

   The Service Creation Method and Classname / Bean Name fields are only available to users with appropriate permissions.

9. Select the service connection from the Service Connection dropdown list.
10. Select a server node from the Server Node dropdown list to associate the service with a specific server node defined in Manager. You may want to do it for the performance reasons, for example, to run a delivery service on a dedicated server only.

    NOTE

    The Server Node option is not available for a single-node installation and for Fluent services.

11. Select the Active checkbox to make this service active. It is selected by default.

    NOTE

    Clear the Active checkbox to make the service unavailable in various Manager's configurations by not showing it in the dropdown list of available services. This doesn't disable or deactivate the service itself, therefore existing forms will continue to use this service.

12. Check the minimum version of Manager required to run the service in the JM Min Version field.
13. Click Save to update the changes.

All services have parameters, which you need to edit before start using the service. However, depending on a service type and service template used to create the service, additional set of configuration tabs is also available. You may need to configure some or all of these tabs. For more information, see Configure a Transact Function Service Definition.

Next, learn how to configure service's parameters.

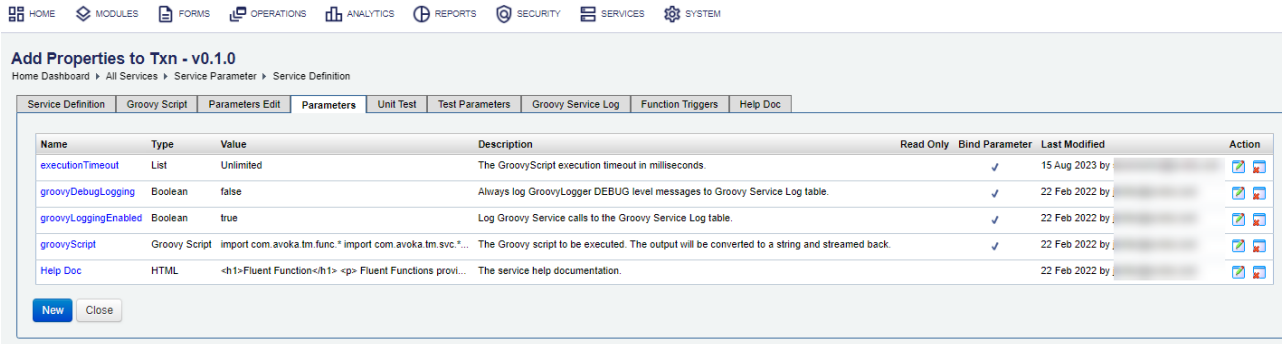
Configure Service Parameters

Manager allows you to view, configure, create, and remove service's parameters in the Parameters tab.

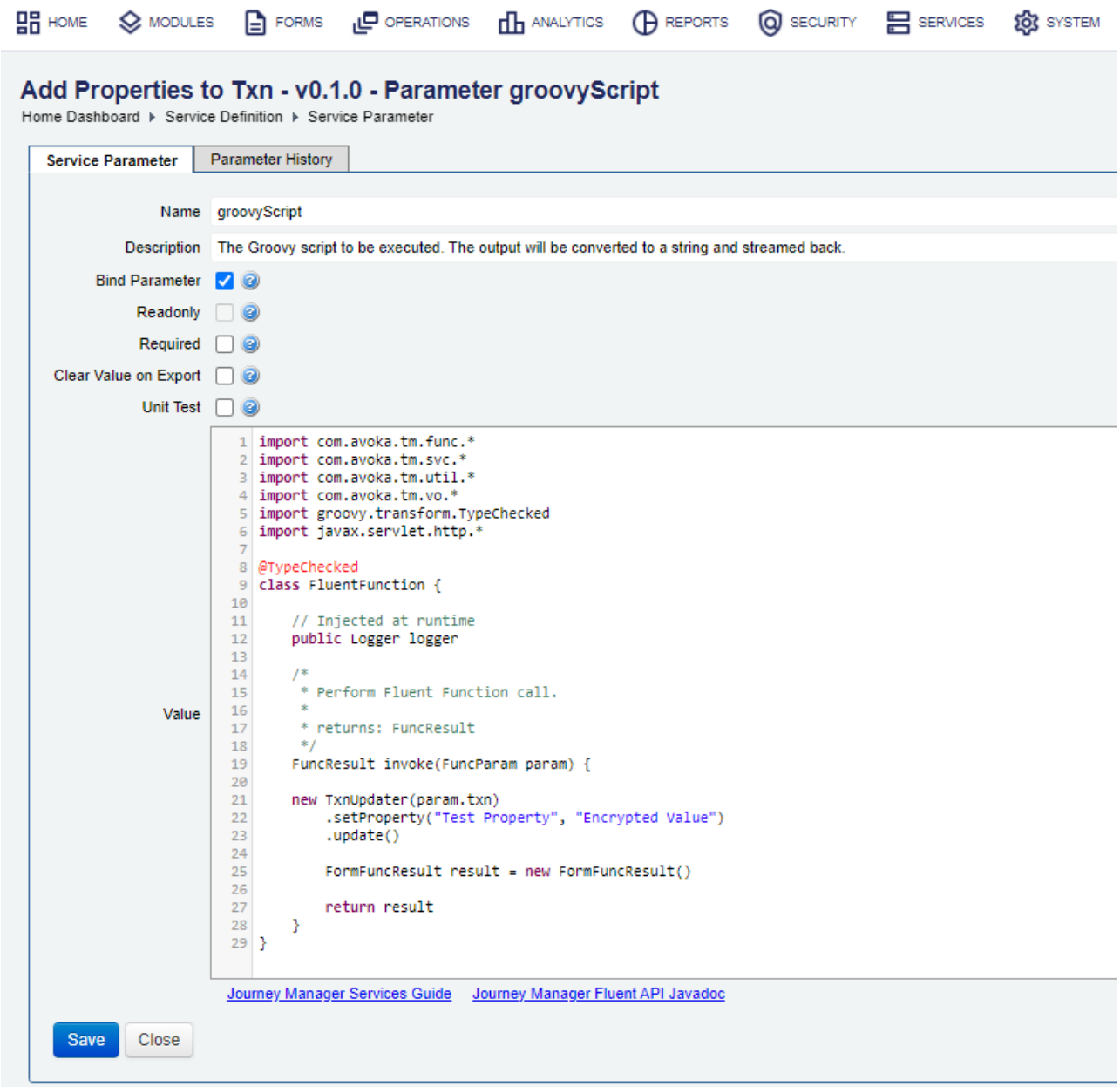
> NOTE
>
> You can update the same parameters in the Parameters Edit tab.

To configure service parameters:

1. Select Services > All Services.
2. Locate a service and click Edit.
3. Select the Parameters tab to view, edit and add service parameters.



4. Locate a parameter to modify and click Edit. For example, click `groovyScript` to view its configuration, as shown below.



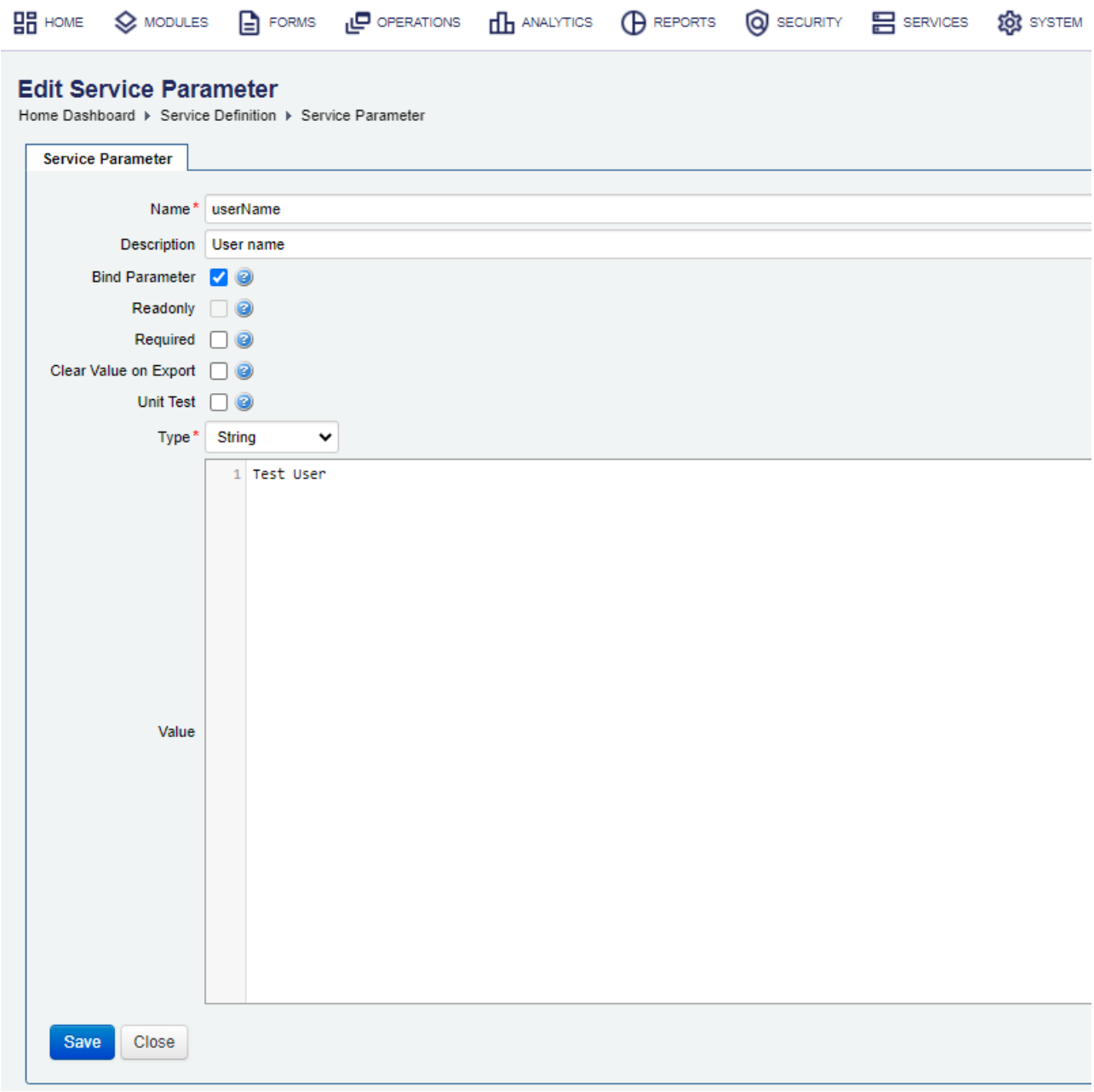
> NOTE
>
> For explanation of common parameters, see Configure Transact Function Parameters Edit. Otherwise, check the specific documentation for each service.

5. Click Save to update the changes.

To create a service parameter:

1. Select Services > All Services.
2. Locate a service and click Edit.
3. Select the Parameters tab to view, edit and add service parameters.

4. Click New to create a new parameter.



5. Select the Bind Parameter checkbox to inject the parameter value into the created service.
6. Select the type of the parameter, which is a String in this example, and provide a parameter's value in the Value field.
7. Click Save to update the changes.

Next, learn how to check where this service is used.

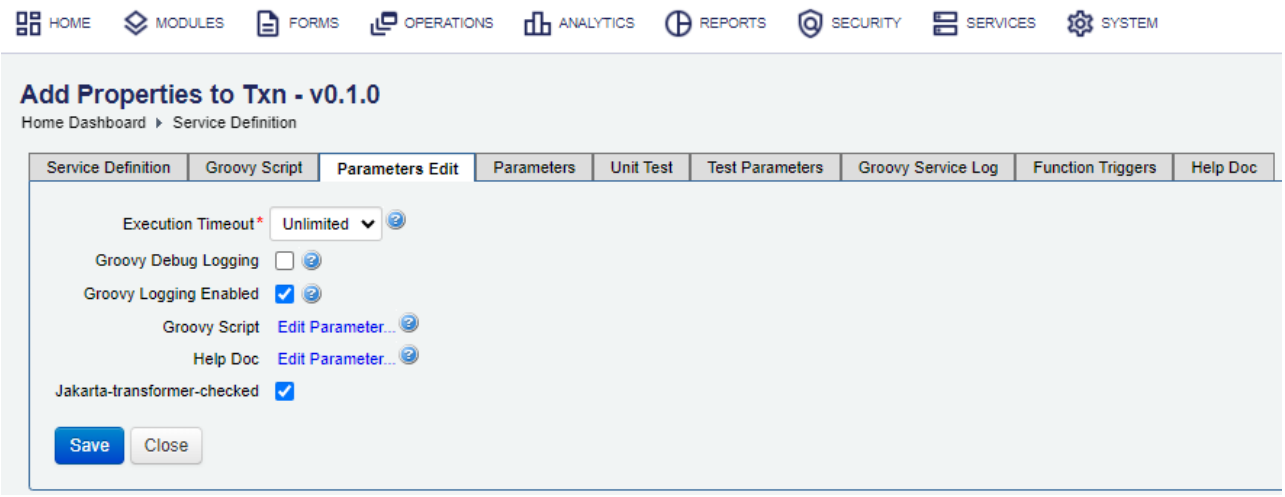
Configure Service Parameters Edit

Manager allows you to view and edit parameters used by a service. This adds flexibility of using services as well as make them more reusable.

To configure service parameters:

1. Select Services > All Services.
2. Locate a service and click Edit.
3. Select the Parameters Edit tab to view and edit parameters available to the service.

> NOTE
>
> The number of parameters and their types depend on the service implementation so the layout of this tab is different for each service.



4. Edit parameters as needed. For explanation of common parameters, see Configure Transact Function Parameters Edit. Otherwise, check the specific documentation of each service type.
5. Click the Parameter History tab to see a change history of the Groovy script.
6. Click View Content Difference to check the content that was changed. Select the View Type to show the difference side by side or inline.
7. Click Restore History Content to restore this history content.
8. Click Remove Service Parameter History to remove this history content.
9. Click Save to update the changes.

TIP

You can add new and edit existing parameters in the Parameters tab. For more information, see Configure Service Parameters.

Next, learn how to check where this service is used.

Assign Organizations to a Service

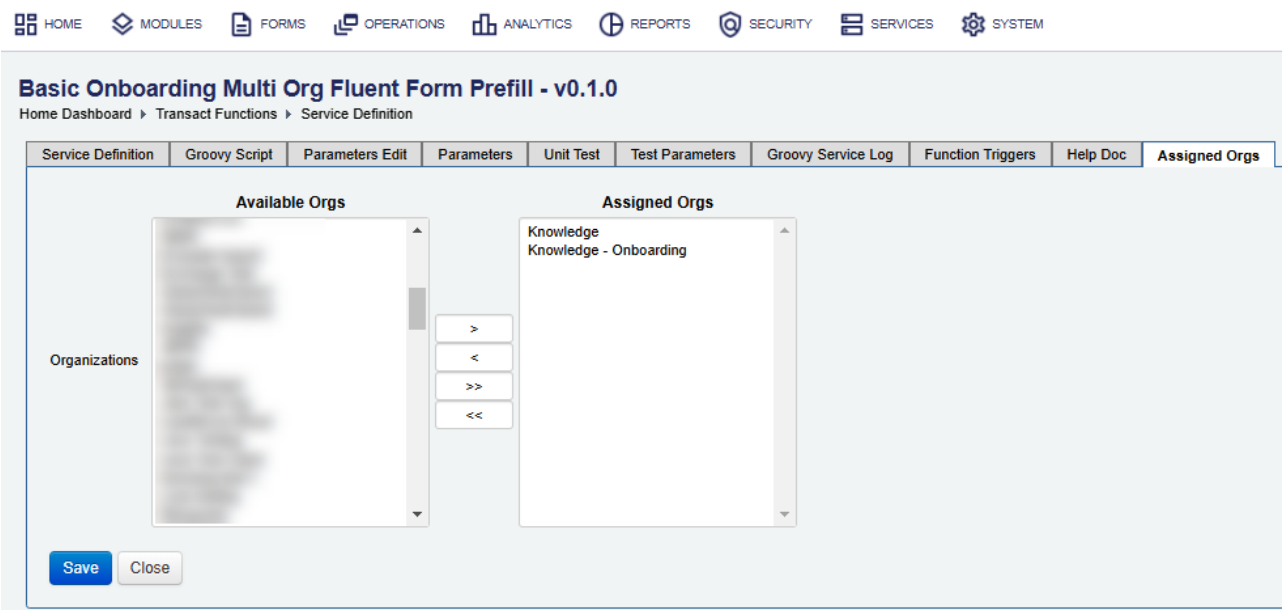
Manager allows you to assign multiple organizations to a service, so you can reuse it rather than replicate (copy) the service to all these organizations.

> NOTE
>
> You must have the Global Access to assign organizations to services.

To assign organizations to a service:

1. Select Services > All Services.
2. Locate a service and click Edit.
3. Select the Assigned Orgs tab to see all assigned organizations in the Assigned Orgs field.

   > NOTE
   >
   > This tab is only available to services of the following Service Template types:
   >
   > - Assignable Fluent Form Prefill. This is implemented in the Assignable Form Prefill service. | 24.10 This feature was introduced in 24.10



4. Select one or more organizations, which you want to assign to the service, from Available Orgs.

> NOTE
>
> You can use the `Shift` and `Ctrl` keys to select multiple organizations on Windows PC.

5. Click > to move the selected organizations from Available Orgs to Assigned Orgs.
6. Click < to revoke the selected organizations from Assigned Orgs.
7. Click >> to assign all organizations.
8. Click << to revoke all organizations.
9. Click Save to update the changes.

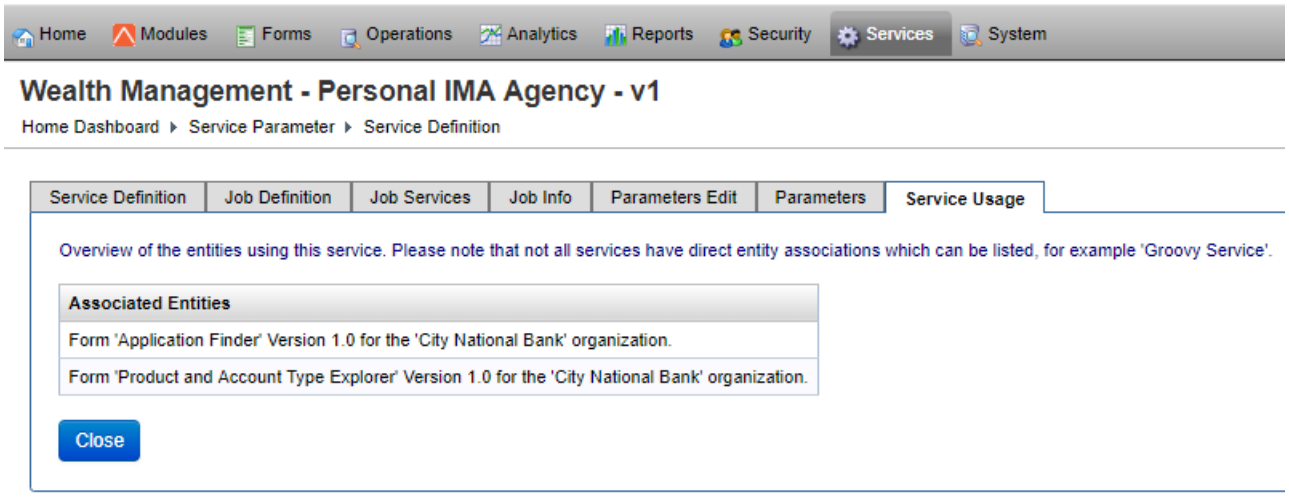
Next, learn how to check where this service is used.

Service Usage

Manager shows the forms linked to a service that allows you to determine where this service is used, so you can avoid unwanted side effects when modifying or removing the service.

To check service usage:

1. Select Services > All Services.
2. Locate a service and click the Service Usage tab.



> NOTE
>
> This tab is shown for service types that can be explicitly associated with entities such as form versions, organizations and delivery channels. However, not all services have direct entity associations which can be listed, for example the Groovy Service.

3. Click Close.

Next, learn how to view all services.

Copy a Service

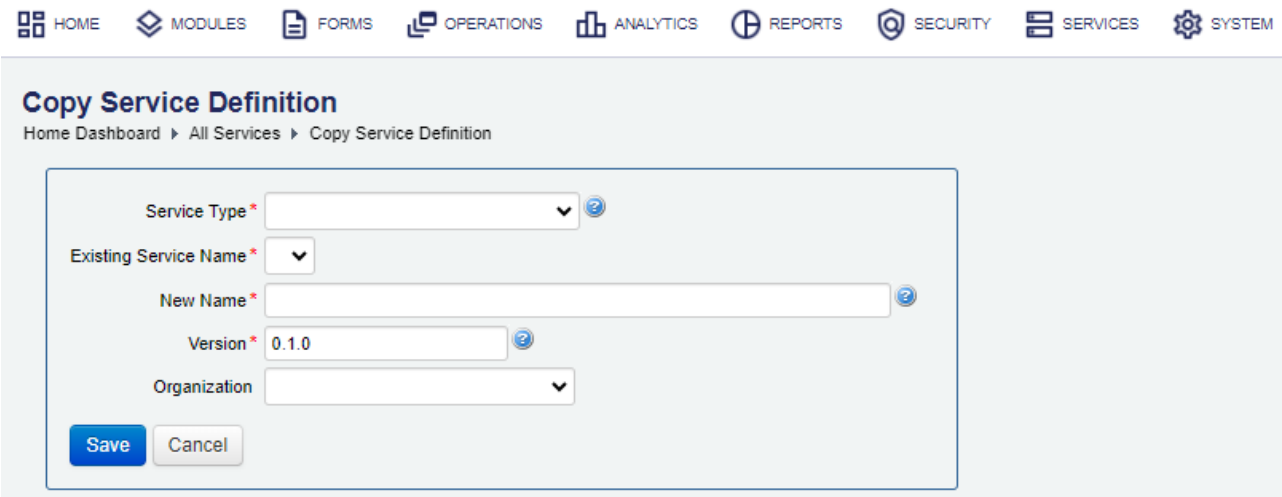
Manager allows you to copy an existing service. You can use this functionality to:

- Test services.
- Create multiple copies of the same service.
- Configure a new service from an existing service.

> NOTE
>
> The copied service inherits its base configuration and all service parameters from the original service rather than a service template.

To copy an existing service:

1. Select Services > All Services.
2. Click Copy.



3. Select a service type for which you want to create a new service definition from the Service Type dropdown list.
4. Select a service you want to create a copy of from the Existing Service Name dropdown list.
5. Enter a new globally unique service in the New Name field.
6. Select an organization from the Organization dropdown list to restrict the new service to a specific organization. If left blank, this service is a global service and is available to all organizations.
7. Click Save to update the changes. Manager creates the new service, which is identical to

the original apart from the name.

8. Configure the service according to your requirements.

Next, learn how to view all services.

Remove a Service

As you work on a service by improving its functionality to meet new requirements, you often create a new version of the service to keep tracks of changes. At some stage, a service or its old version is not required any more, so you want to disable it. Manager provides you with the following options:

- Deactivate or disable a service
- Remove a service

> NOTE
>
> We recommend deactivating a service, rather than removing it.

Deactivate a Service

To deactivate a service:

1. Select Services > All Services.
2. Locate the service and click Edit.
3. Select the Service Definition tab.
4. Clear the Active checkbox.

   > NOTE
   >
   > Clear the Active checkbox to make the service unavailable in various Manager's configurations by not showing it in the dropdown list of available services. This doesn't disable or deactivate the service itself, therefore existing forms will continue to use this service.

5. Click Save to update the changes.

> NOTE
>
> To reactivate the service, select its Active checkbox.

Remove a Service

When you remove a service, Manager irrevocably deletes all data associated with the service, so it can not be restored. Therefore, we recommend backing up the service before deleting it.

NOTE

To remove services in Core Global services, you must have the Core Service Edit permission.

To remove a service:

1. Select Services > All Services.
2. Locate the service.
3. Check the service is not used anywhere, for example form versions, to avoid errors.

   NOTE

   It is not recommended to delete a service that is the default for a service type.

4. Click Remove and then click OK to confirm.

Next, learn how to create a service.

Migrate Services

Manager provides a mechanism to migrate services from one environment to another. This allows you to develop and test a service in different environments before migrating it over to the production environment.

To migrate one or several services:

1. Export services
2. Import services

> NOTE
>
> You can use the same process to perform a service rollback.

Next, learn how to create a service.

Export Services

Manager allows you to export services. You can use this functionality to:

- Back up or archive services
- Re-create or migrate services in another Manager
- Promote services to another environment

> NOTE
>
> If you export a service with multiple endpoints, export the correct service connection before exporting the service.

To export a service:

1. Select Services > All Services.
2. Locate a service and click Export Archive to export a selected service as an archive ZIP file. The ZIP file contains a `service-config.xml` file with a service's definition.
3. Click Export Data to export the displayed data as a Microsoft Excel (XLS) file. The XLS file has the same column names and described in view services.

Next, learn how to import services.

Import Services

Manager allows you to import services, which were previously exported. Use this functionality to:

- Roll back one or several services.
- Re-create one or several services in another Manager.
- Promote services to another environment.

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

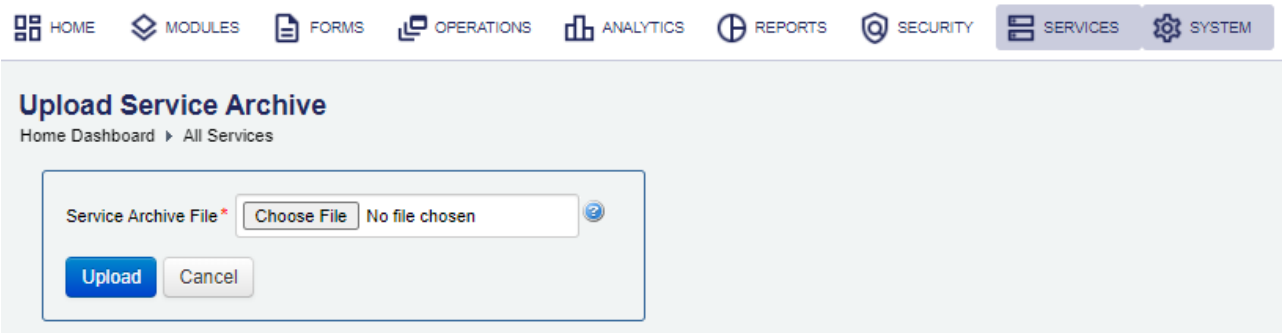
You can import a service using one of following options:

- UI
- SDK
- API

Import Using UI

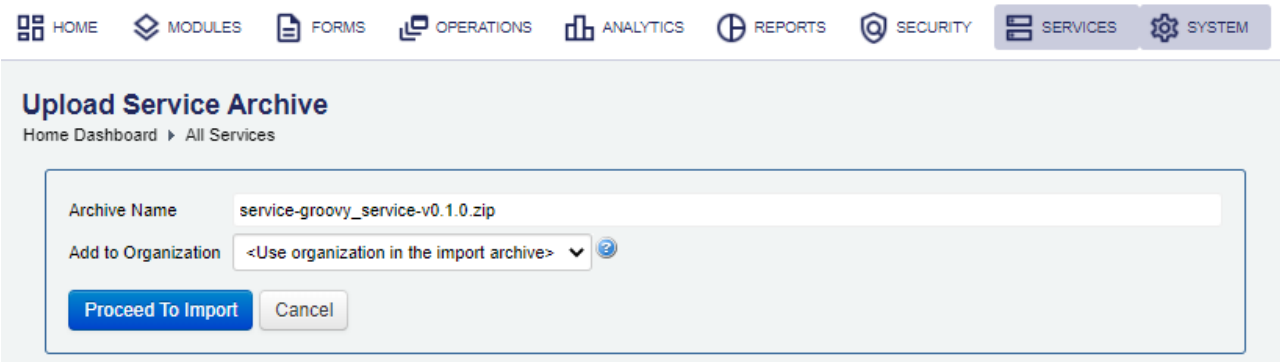
To import a service:

1. Select Services > All Services.
2. Click Import.

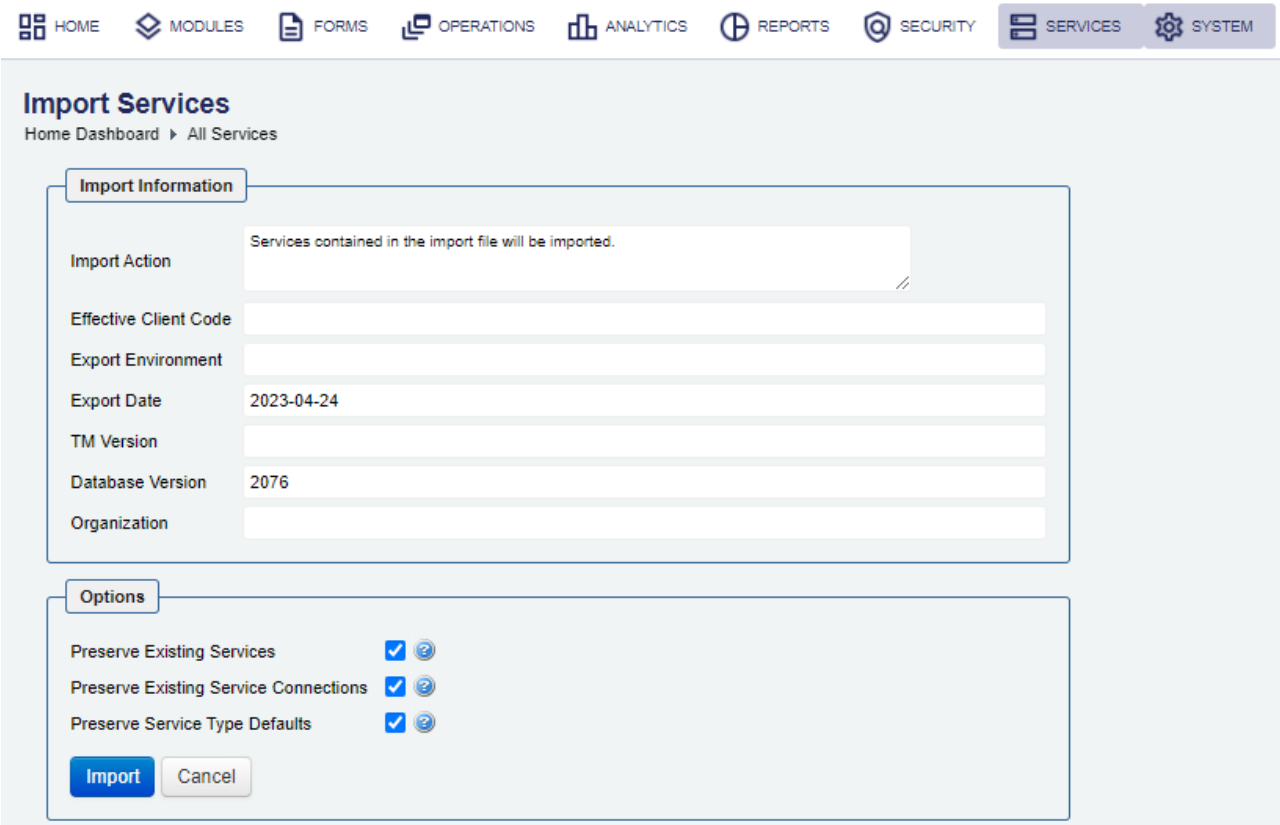


3. Click Choose File and choose a service archive ZIP file, which has been previously exported.

4. Click Upload.



5. Select an organization from the Add to Organization dropdown list where you want to add the services to or use the organization already configured in the import archive file.
6. Click Proceed To import.
7. Review the Import Options as shown below:

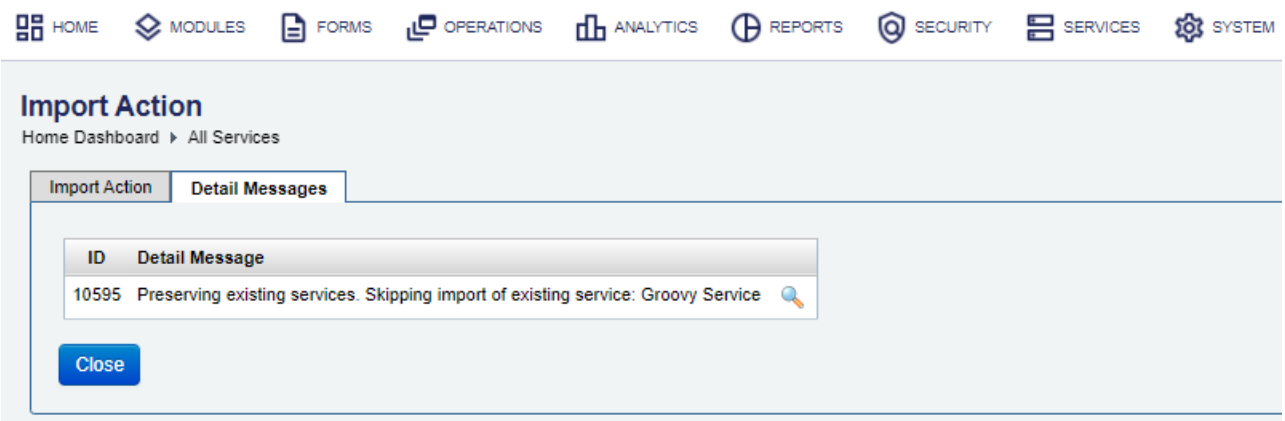


8. Select the Preserve Existing Services, so any services that already exist in the target environment will not be modified in any way. Only new to the target environment services will be imported.

NOTE

If you import a new version of a service, clear the Preserve Existing Services, then manually activate this service version and update the configurations that refer to the old version of the service to point to the new version.

9. Select the Preserve Existing Service Connections, so any service connections that already exist in the target environment will not be modified in any way. Only new to the target environment service connections will be created.
10. Select the Preserve Service Type Defaults, so none of the imported services will be promoted to be the default for their service types, regardless of their default status in the source environment. Otherwise, the service type default flag will be imported, and existing services may be denoted.
11. Click Import to complete the process. The expected result page looks like this.



12. Select the Detail Messages tab to see more information about the import. This tab pops up open if the import operation has any issues you may need to check and resolve.
13. Click Close and check the list of services to verify the imported services have been added.

NOTE

You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

To help you avoid enabling the Groovy Debug Logging option for a Groovy service or a function by mistake in a production environment, Manager checks if this option should be disabled on import. For more information, see the Production Mode option. | 23.04 This feature was introduced in 23.04

Import Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:archive-deploy goal to import a service.

Import Using API

The Journey Manager REST API provides the endpoint URL to import a service.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s)://<JM_SERVER>/manager/secure/rest/archive/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as Base64[1].

`--form ‘archiveFile=@“/Users/my-service.zip”’` - a path to an archive file to be imported.

`--form ‘importOptions=“PRESERVE_DEFAULT_SERVICES,PRESERVE_EXISTING_SERVICES”’` - a comma-separated list of import options to perform. The option names are the same as described in the Import Using UI section but they use an underscore, for example, `PRESERVE_DEFAULT_SERVICES` for Preserve Existing Services.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

Example

```
curl --location --request PUT ‘https://<JM_SERVER>/manager/secure/rest/archive/v1/’
\
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
--form ‘archiveFile=@“/Users/my-service.zip”’ \
--form ‘importOptions=“PRESERVE_DEFAULT_SERVICES,PRESERVE_EXISTING_
SERVICES,PRESERVE_SERVICE_CONNECTIONS”’
```

Next, learn how to migrate services.

---

[1]Base64 is a binary to a text encoding scheme that represents binary data in an American Standard Code for Information Interchange (ASCII) string format.

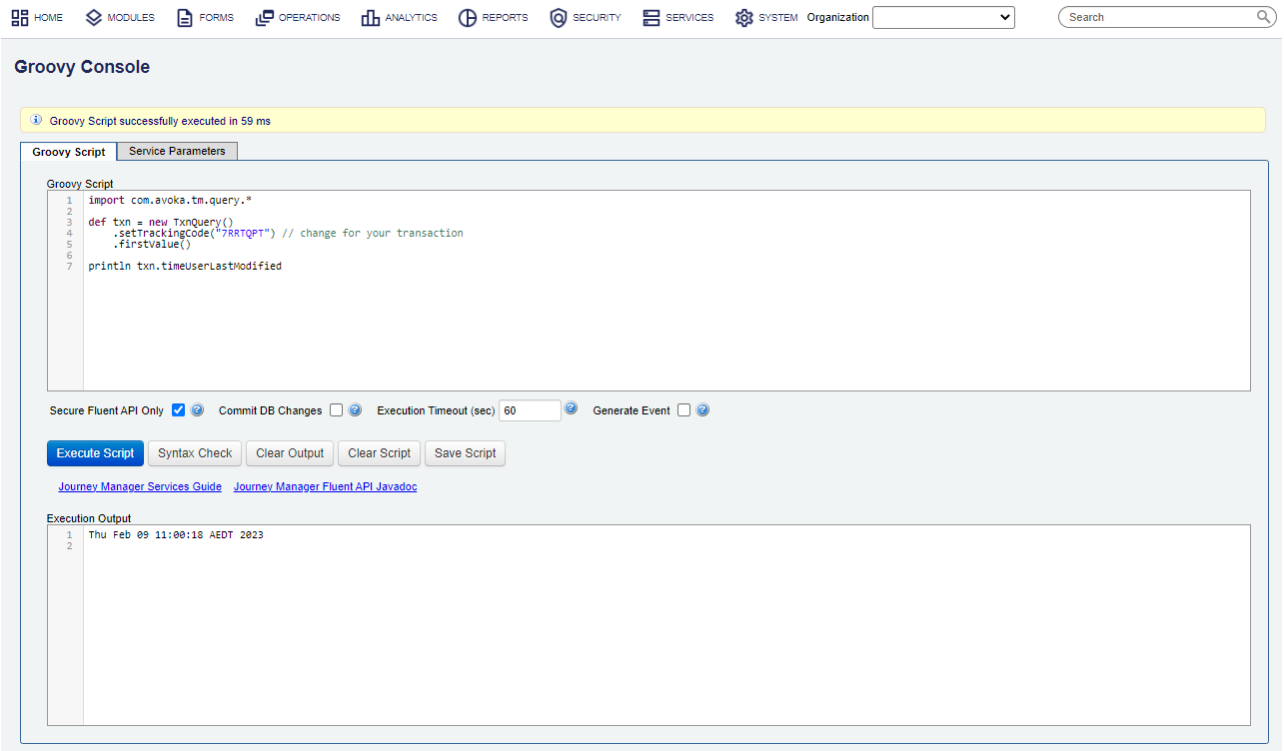
Groovy Console

Manager comes with the Groovy[1] console enabling developers to rapidly develop and test Groovy scripts. Groovy console has access to all services and libraries, which allows you to simulate the behaviors of any Groovy service.

> NOTE
>
> You must have the Groovy Console permission set to your user account to run Groovy scripts in the Groovy console.

To run a script in the Groovy console:

1. Select Services > Groovy Console.



2. Select the Groovy Script tab to write or edit a Groovy script. You can write a part of a script or a simple function to be executed and tested.

---

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

3. Select the Secure Fluent API Only checkbox to enforces the use of Transact Fluent API in this Groovy script. It is selected by default. Clear this checkbox if you want to test the Groovy script that uses Transact Core API.

   NOTE

   At installation time, the system wide build property `groovy.secure.api` defines whether the Secure Fluent API Only service parameter is editable or available.

4. Select the Commit DB Changes checkbox to commit any changes, made by the Groovy script, to the database. Otherwise, the changes will be rolled back. This is handy when you want to test a Groovy script that writes to the database, but you don't want to delete these test records from the database manually. It is not selected by default.
5. Enter a timeout in the Execution Timeout (sec) field to specify the Groovy script execution timeout in seconds, after which a timeout exception will be thrown. The default is 60 sec.
6. Select the Generate Event checkbox to generate events for the changes, made by the Groovy script. It is not selected by default. You can see generated events in the Event Log as well as in the transaction time line of the respective submission.

   NOTE

   The events are created only when both the Commit DB Changes and the Generate Event checkboxes are selected and the changes, applied to the Manager's entities, satisfy criteria defined in the Event Configuration Storage Definition.

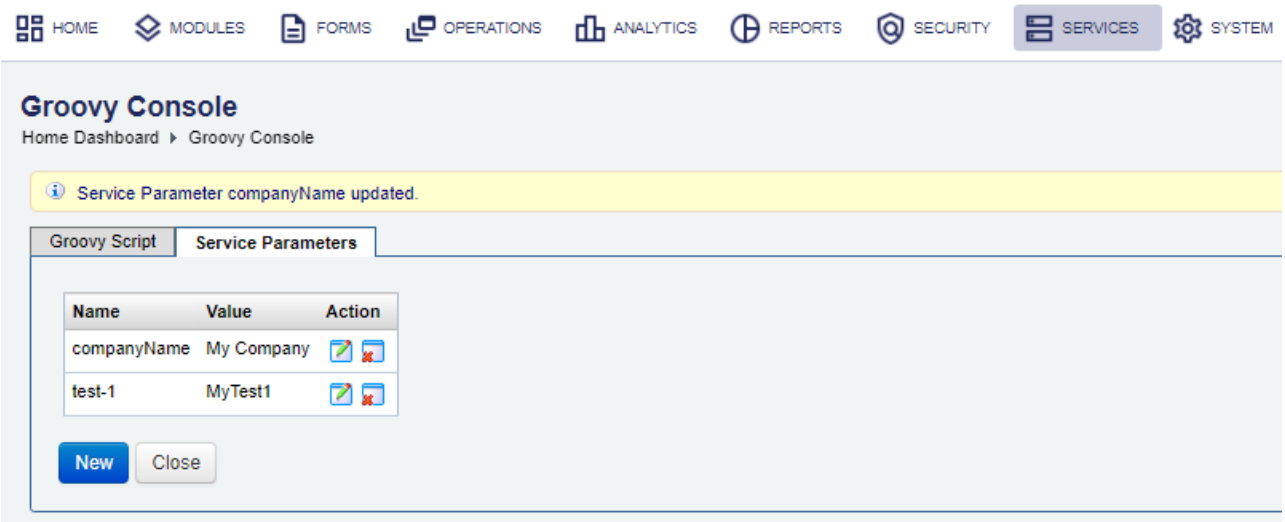
   NOTE

   The Generate Event checkbox is not shown if the eventing feature is disabled in deployment properties.

7. Click Execute Script to run it. The script output or an error message is shown in the Execution Output window.
8. Click Syntax Check to validate the Groovy script, which checks the script spelling, as shown below.

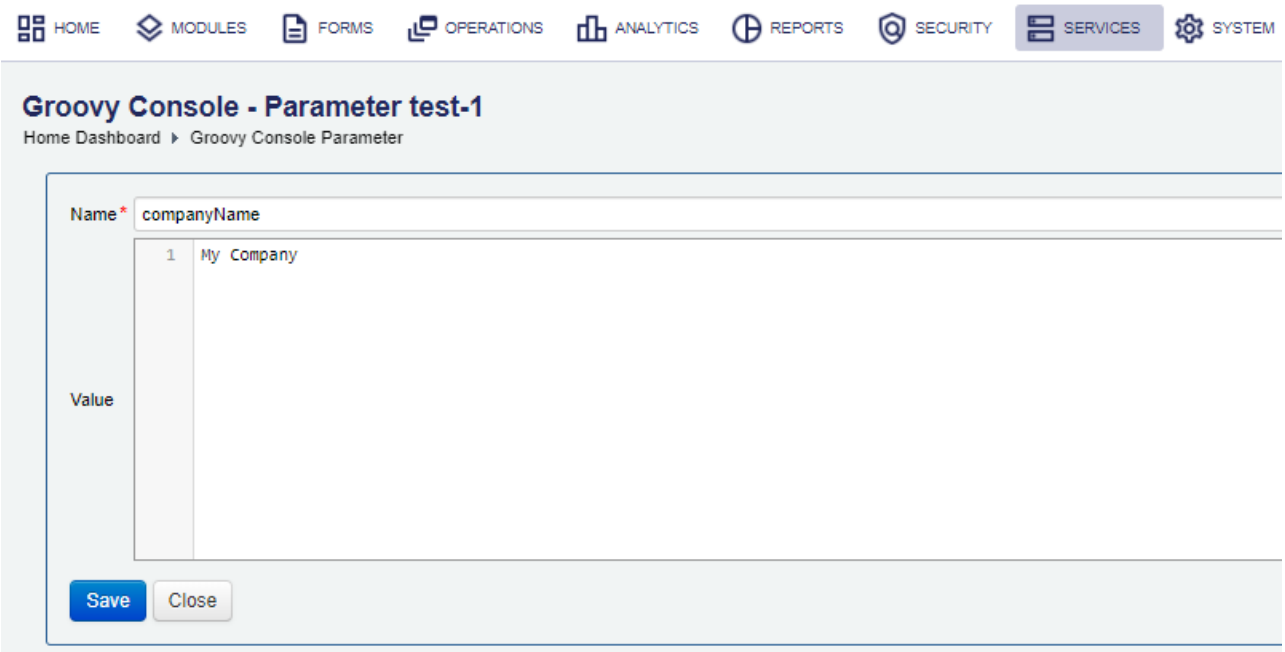
```
[Static type checking] - Cannot find matching method Script1#printl
(groovy.lang.GString).
       Please check if the declared type is correct and if the method exists.
       @ line 2, column 1.
       printl "Hello world, $it"
       ^

       1 error
```

9. Click Clear Output to clear the Execution Output text field.
10. Click Clear Script to clear the Groovy Script text field.
11. Click Save Script to save the script.
12. Select the Service Parameters tab to create or edit existing Groovy script parameters.



13. Click New to create a new parameter or click Edit to update an existing parameter.

14. Edit a name and a value.



15. Click Save to update the changes.

Now, you can use `companyName` parameter in the Groovy script, for example:

```
def name = serviceParameters.companyName
        (0..9).each {
                println "Hello $name, $it"
        }
```

You can use more complex Groovy code snippets including imports, as shown below:

```
import com.avoka.tm.query.*
import com.avoka.tm.vo.*

String trackingCode = "LHC5DD"
String txnPropertyName = "Monty"

Txn txn = new TxnQuery()
       .setTrackingCode(trackingCode)
       .firstValue()

// PropertyQuery() to access the Monty property value
String txnPropertyValue = new PropertyQuery()
       .setName(txnPropertyName)
       .setTxn(txn)
       .getValue()

// expected result: txnPropertyValue = Python
println "txnPropertyValue = ${txnPropertyValue}"
```

NOTE

A script saved in the Groovy console is specific to each user, so when you open the console next time, you will see your recently accessed script in there.

Next, learn about Groovy service log.

Services Troubleshooting

Manager allows you to create and use various services. This provides great flexibility to your application design, but it can also be daunting at times when you need to troubleshoot services for errors or erroneous (buggy) behavior.

Let's look at several common recommendations that may help you resolve problems with your services.

Service Configuration

A service configuration and definition should be identical across all environments with the exception of environment specific parameters, such as a host name, and so on, which must be parametrized, if possible.

Make sure you test the service well in the test environments before promoting it into a production environment. If you changed some service or form configuration in the test environment to perform some debugging activities, record what was changed, and restore to the original one after the testing is done. Even slight deviation in configuration can cause a significant change in an application behavior. For example, if you disable the Form Data Encryption option in a form in order to see and analyze form data passed between a client and a server, make sure to enable this option again after you are done with testing.

You must ensure that services don't have the Groovy Debug Logging option enabled when imported to or configured in production. Currently, if enabled, this adds significant amount of additional entries into the Groovy Service Log table that, over a short period of time, will place a strain on the Transaction Processor when purging the entries. This also affects other areas of the system. The debug logs are not intended for production; however, clients will often deploy forms with the setting enabled in older versions of Manager by accident and won't notice it until the entries are well into their millions and affecting performance.

To help you avoid enabling the Groovy Debug Logging option for a Groovy service or a function by mistake in a production environment, Manager checks if this option should be disabled in a production environment. For more information, see the Production Mode option. | 23.04 This feature was introduced in 23.04

You may also need to check whether the Debug Mode deployment property is enabled or not.

When you deploy a function or a service using the `svc-packag` and `svc-deploy` commands and move it through development, test and then production environments, ensure the Service

Creation Method or Classname / Bean Name names stay the same. Otherwise, the mismatched parameter can cause `java.lang.ClassCastException`.

Service Errors

A well-designed service is expected to log all relevant error messages into the Error Log, so you should always start from checking errors pertinent to the service. When you find the error of your interest, check the detailed description and the error stack trace. If you need to collaborate with other teams or departments to investigate the error, we recommend downloading the error as an HTML file and sending it to all parties involved.

Service Timeout

Event a properly functional and well-tested service might throw a timeout exception, for example, `TimeoutException occurred after 60,000 ms`, in some situations. There are could be several reasons for this. The most common is that it has taken too long for a service to process a task, so it has exceeded the service's timeout. You can easily resolve the issue by increasing the timeout. For example, the Jakarta Transformer service has the Execution Timeout property that you can update.

Next, learn how to view all services.

Core Global Services Overview

Journey Manager comes with a set of standard services, which are designed to perform specific tasks pertinent to server's core functionality, for example, data retention and email queues. These services are internal to Manager ensuring it runs as expected. They are not part of form submission flows, so Form Builders and Customer Services don't need to know about them or modify them as long as the system performs as designed. If you are a Form Builder, you should check form services.

Most of global core services don't have an organization assigned to them, especially for standard Journey Manager deployments. However, you can assign a global core service to a specific organization.

Services are grouped by service types so you can filter only services you need.

Manager allows you to view, create new and manage the existing core global services.

> NOTE
>
> You must have the Core Service Edit permission to be able to edit and remove Core Global services.

Next, learn about other services.

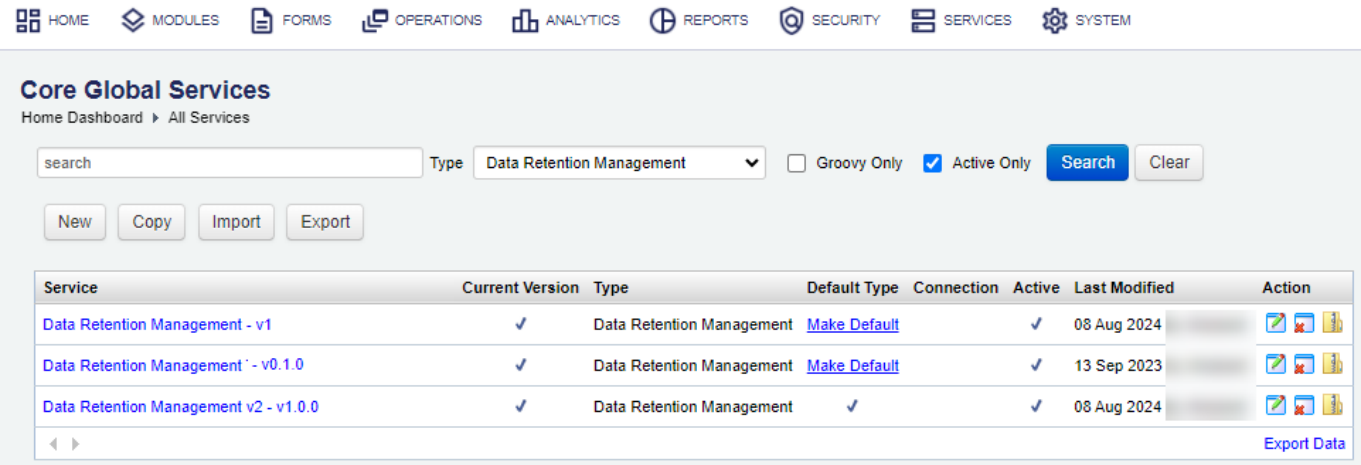
View Core Global Services

Manager allows you to view core global services, which belong to the Core Global Service category, so you can easily find and edit a service you need.

> TIP
>
> You can also check core global services using View Services UI.

To view core global services, select Services > Core Global Services.

Core global services are shown page by page, with the maximum number of records per page configured in the users preferences.



The list displays the following details:

- Service name:
- Current Version:
- Type:
- Default Type:
- Connection
- Active:
- Last Modified:

For more information, see View Services.

To filter or search the services by one or more criteria, specify the following settings and click Search:

- Service name or part of a name
- Service type
- Groovy Only to see only Groovy[1] services
- Active Only

Click Clear to reset the search criteria.

You can also perform the following actions:

- Create a service
- Configure a service
- Copy a service
- Remove a service

> NOTE
>
> This option is available only if you have the Core Service Edit permission.

- Export services
- Import services
- Run a unit test. This option is only available for groovy, scheduled and virus scan services.
- Click Make Default to mark the selected service as a default one for all services of this type.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

A/B Testing Form Version Selector Service

Manager comes with the A/B Testing Form Version Selector core global service, which you can link to a form that needs to support A/B testing[1]. This service extends the current version selector service.

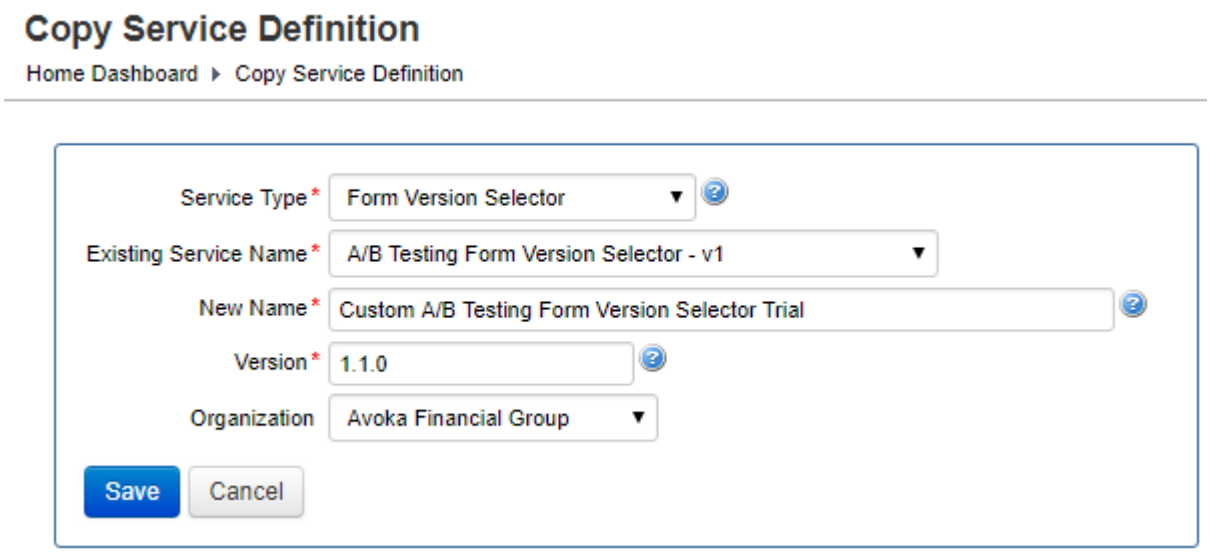
By default, when a user selects a form, the A/B Testing Form Version Selector service chooses a form to render from all its form versions. Usually, you may need to test only the current version against the previous version.

> NOTE
>
> It's recommended to copy the default A/B Testing Form Version Selector service before using it for A/B testing.

To copy the A/B Testing Form Version Selector service:

1. Select Services > Core Global Services.
2. Copy the service.
3. Select a service of the Form Version Selector service type from the Service Type drop-down.
4. Select A/B Testing Form Version Selector from the Existing Service Name dropdown.

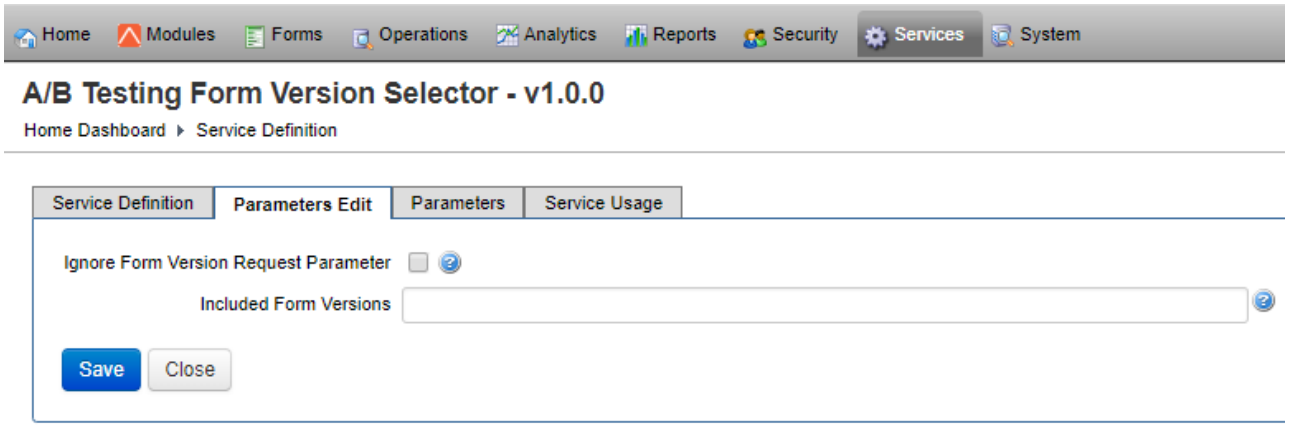
[1] A/B testing (bucket tests or split-run testing) is a randomized experiment with two variants, A and B. It includes application of statistical hypothesis testing or "two-sample hypothesis testing" as used in the field of statistics. A/B testing is a way to compare two versions of a single variable, typically by testing a subject's response to variant A against variant B, and determining which of the two variants is more effective.



5. Click Save to update the changes.

To customize the A/B Testing Form Version Selector service:

1. Select Services > Core Global Services. If the option isn't available, follow the above steps for copying the service.
2. Locate the service and click Edit.
3. Configure the standard service settings.
4. Click the Parameters Edit tab to view and edit the configuration.



5. Select the Ignore Form Version Request Parameter checkbox to always use the current form version regardless of the value of the `tmFormVersion` request parameter.
6. In the Included Form Versions text field, enter a comma-separated list of form versions to include in the selection. Leave it blank to include all versions.
7. Click Save to update the changes.

NOTE

The global A/B Testing Form Version Selector service can also be customized from your form.

Database Submission Data Storage Service

Manager provides the database submission data storage service, which is a service for storing form submissions in a database. This service is the default submission data storage service for on premise installations, so it's created and configured for you and you don't need to do anything. If Manager is installed on AWS cloud, this service is set as the default, so you need to configure the Amazon submission data storage service and set it as the default. For Azure cloud installations, the default service is already set to the Azure submission data storage service.

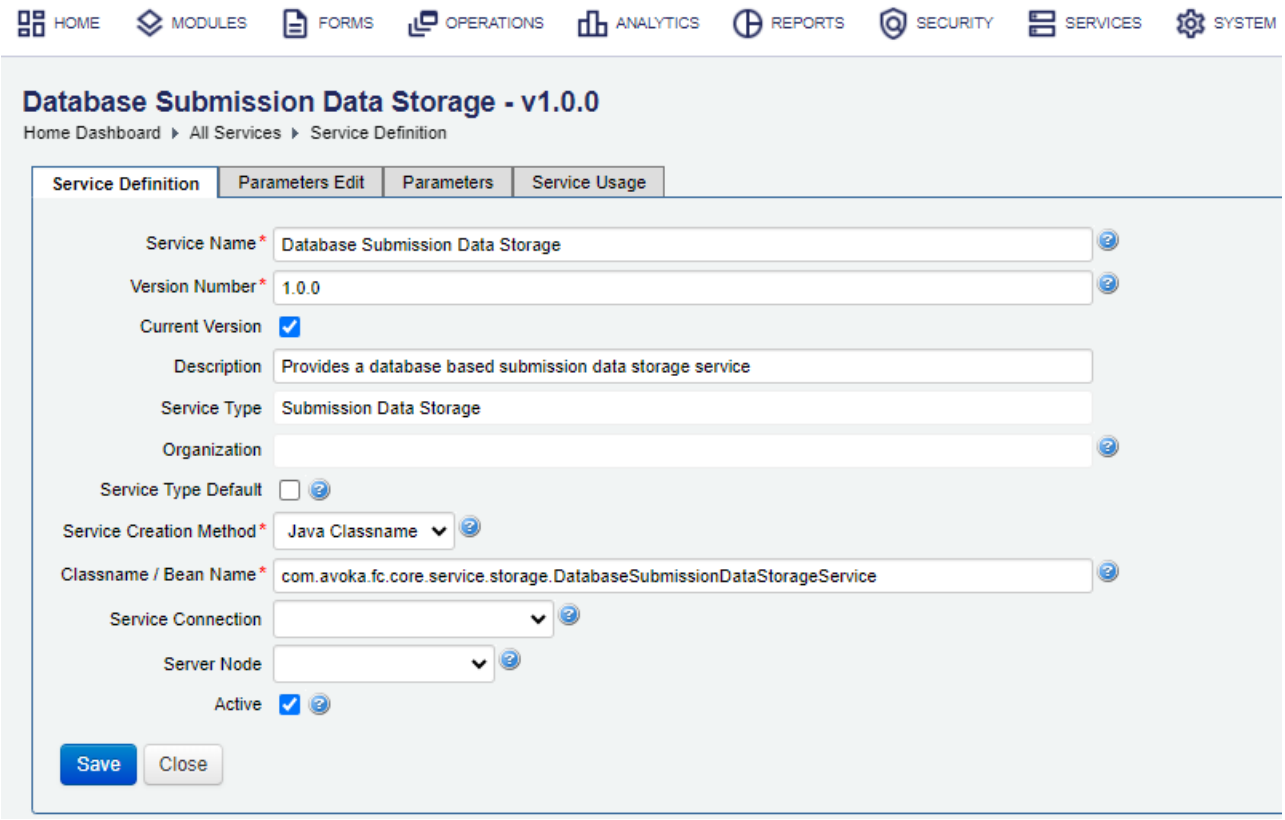
To create the database submission data storage service:

1. Select Services > Core Global Services and click New.
2. Select Submission Data Storage from the Service Type dropdown list.
3. Select Database Submission Data Storage from the Service Template dropdown list.
4. Configure the rest of parameters as described in Create a Service.
5. Click Save to update the changes.

To configure the database submission data storage service:

1. Select Services > Core Global Services.
2. Locate the Database Submission Data Storage service and click Edit.

3. Click the Service Definition tab to configure the service.



4. Click Save to update the changes.
5. Click the Parameters Edit tab to configure the service parameters.

NOTE

The Parameters fields are only available to users with appropriate permissions.

6. Select a file attachments storage encoding from the Attachments Encoding dropdown list, which can be:
    - Compressed / Encrypted - the secure way of data submission
    - Compressed / Plain Text
    - Uncompressed / Encrypted
    - Uncompressed / Plain Text
7. Select a PDF receipt storage encoding from the Receipt PDF Encoding dropdown list.
8. Select a Submission history XML storage encoding from the Submission History XML Encoding dropdown list.
9. Select a Submission XML storage encoding from the Submission XML Encoding drop-down list.
10. Click Save to update the changes.

Next, learn about the Amazon submission data storage service.

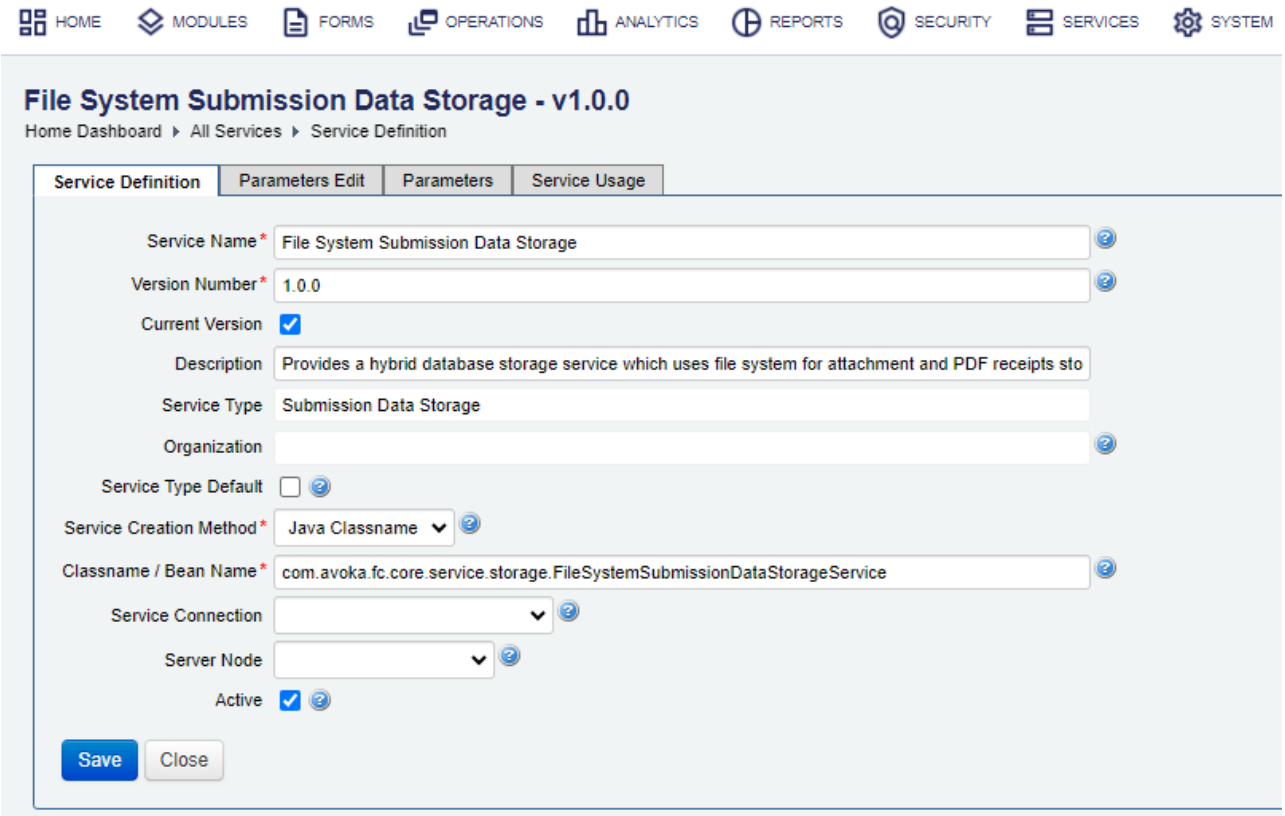
File System Submission Data Storage Service

Manager provides the file system submission data storage service, which is a hybrid database storage service that uses file system for attachment and PDF receipts storage. This service is created when your Manager is installed on premise.

> NOTE
>
> Cloud specific services and some service connections are not available and can't be manually created, unless the deployment environment matches your installation mode. To check the installation mode of your Manager server, select System > System Info, find the Build Properties section, and check the `deployment.env` value. For more information, see system info.

To configure the file system submission data storage service:

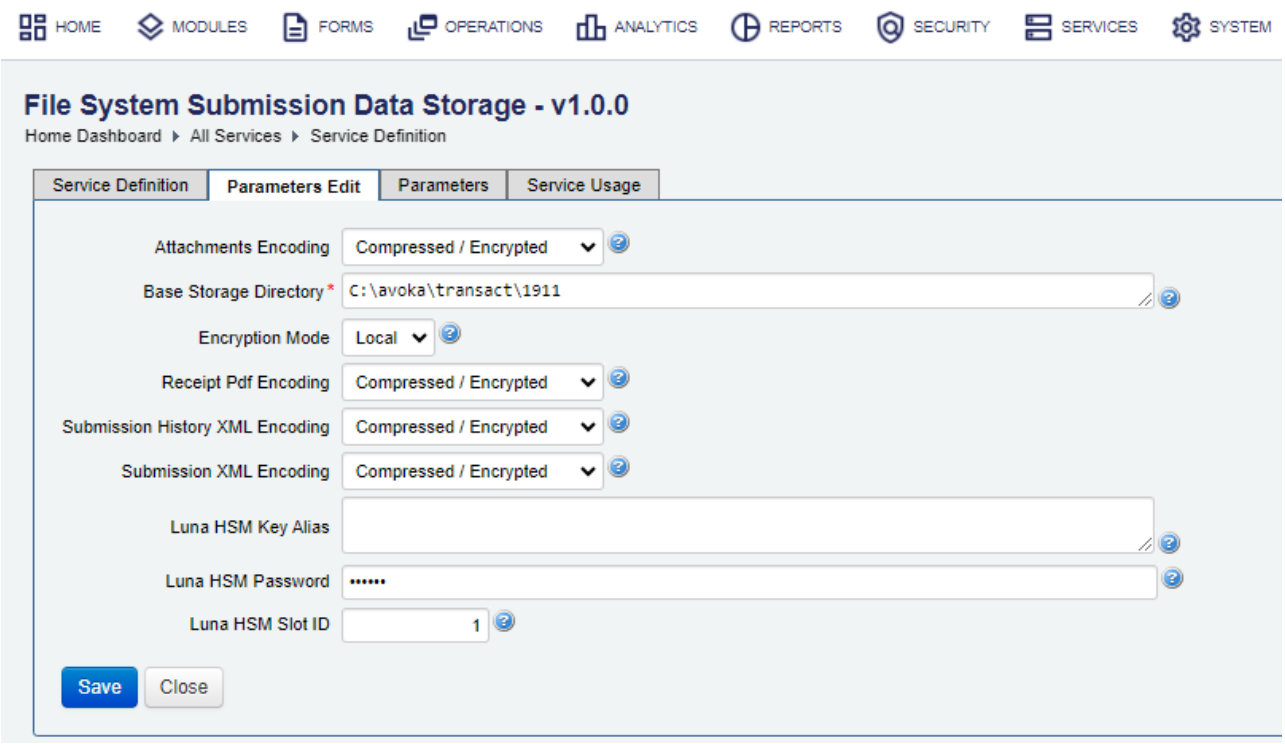
1. Select Services > Core Global Services.
2. Locate the File System Submission Data Storage service and click Edit.
3. Click the Service Definition tab to configure the service.



NOTE

The Service Creation Method and Classname / Bean Name fields are only available to users with appropriate permissions.

4. Click Save to update the changes.
5. Click the Parameters Edit tab to configure the service parameters.



NOTE

The Parameters fields are only available to users with appropriate permissions.

6. Select a file attachments storage encoding from the Attachments Encoding dropdown list, which can be:
   - Compressed / Encrypted
   - Compressed / Plain Text
   - Uncompressed / Encrypted
   - Uncompressed / Plain Text
7. Specify a base file storage directory for submission attachments in the Base Storage Directory field. This is a mandatory field and needs to be set up correctly.

8. Select a submission data encryption mode from the Encryption Mode dropdown list, which can be one of the following:
    - Local
    - HSM
9. Select a PDF receipt storage encoding from the Receipt PDF Encoding dropdown list.
10. Select a submission history XML storage encoding from the Submission History XML Encoding dropdown list.
11. Select a submission XML storage encoding from the Submission XML Encoding dropdown list.
12. Provide an Amazon CloudHSM SafeNet Luna HSM Key Alias in the Luna HSM Key Alias field.
13. Provide an Amazon CloudHSM SafeNet client password in the Luna HSM Password field.
14. Enter an Amazon CloudHSM SafeNet Luna HSM Slot ID in the Luna HSM Slot ID field.
15. Click Save to update the changes.

Next, learn about the database submission data storage service.

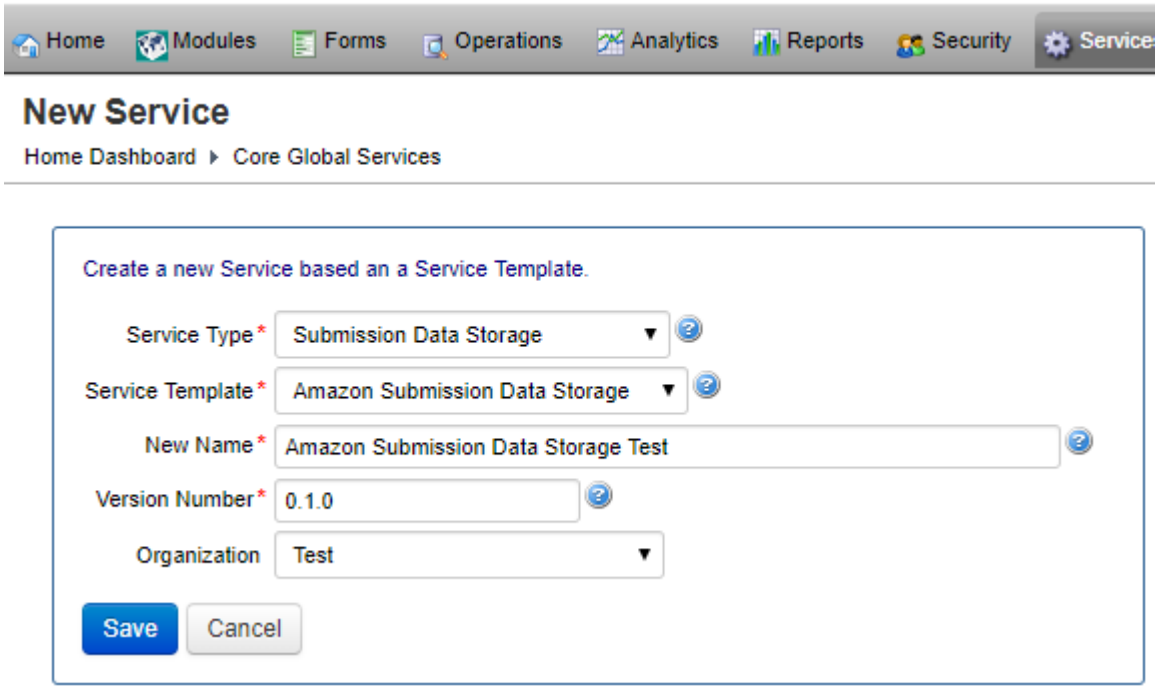
Amazon Submission Data Storage Service

Manager provides the Amazon submission data storage service to persist form submissions and related transactions in the AWS[1] cloud. The service requires an Amazon service connection to connect to an AWS queue. For AWS cloud installations, the database submission data storage service is initially set as the default, so you need to configure the Amazon submission data storage service and set it as the default.

> NOTE
>
> Cloud specific services and some service connections are not available and can't be manually created, unless the deployment environment matches your installation mode. To check the installation mode of your Manager server, select System > System Info, find the Build Properties section, and check the `deployment.env` value. For more information, see system info.

To create the Amazon submission data storage service:

1. Select Services > Core Global Services and click New.

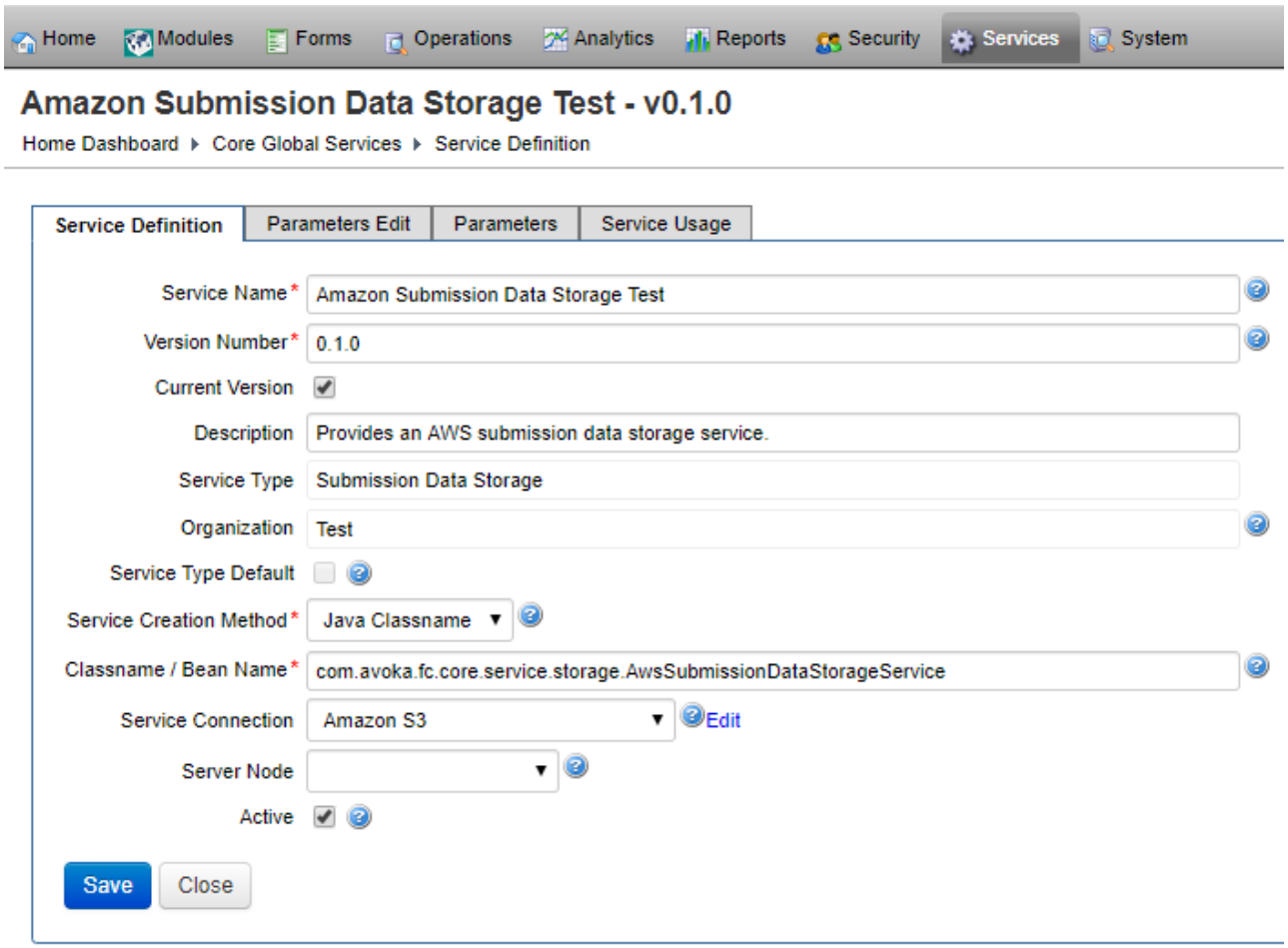


[1]Amazon Web Services (AWS) is a comprehensive, evolving cloud computing platform provided by Amazon. It provides a mix of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings.

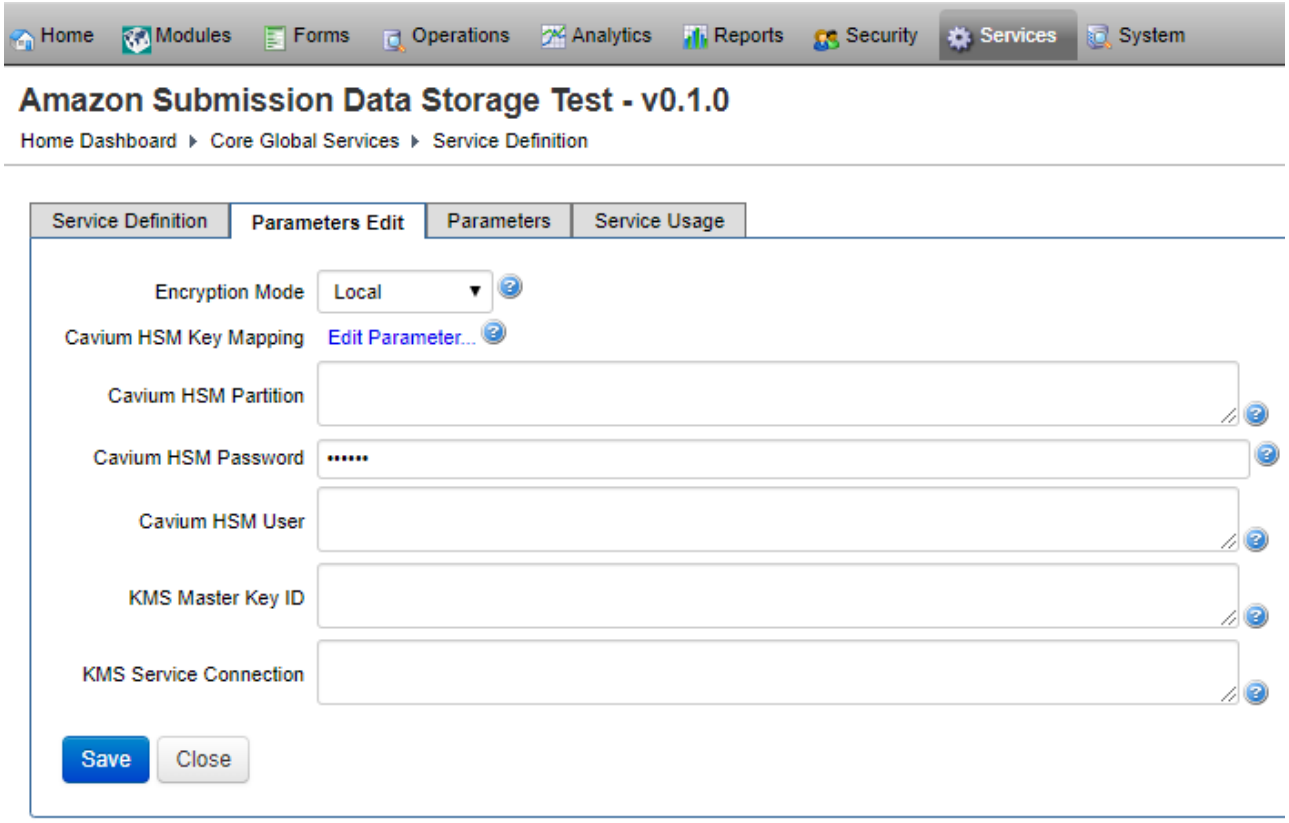
2. Select Submission Data Storage from the Service Type dropdown list.
3. Select Amazon Submission Data Storage from the Service Template dropdown list.
4. Provide a unique service name in the New Name field.
5. Specify a version in the Version Number field.
6. Select an organization from the Organization dropdown list.
7. Click Save to update the changes.

To configure the Amazon submission data storage service:

1. Select Services > Core Global Services.
2. Locate the Amazon Submission Data Storage service and click Edit.
3. Click the Service Definition tab to edit the standard service configuration.



4. Click the Parameters Edit tab to view or update the service configuration.



5. Select a submission data encryption mode from the Encryption Mode dropdown list, which can be Local, KMS[1], or Cavium HSM[2].
6. Provide a Cavium HSM Master Key Mappings as a JSON[3] document in the Cavium Hsm Key Mapping field.
7. Enter a Cavium HSM key alias in the Cavium Hsm Partition field.
8. Specify a Cavium HSM password in the Cavium HSM Password field.

---

[1]AWS Key Management Service (KMS) is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data, and uses FIPS 140-2 validated hardware security modules to protect the security of your keys. AWS Key Management Service is integrated with most other AWS services to help you protect the data you store with these services. AWS Key Management Service is also integrated with AWS CloudTrail to provide you with logs of all key usage to help meet your regulatory and compliance needs.
[2]AWS CloudHSM is a cloud-based hardware security module (HSM) that enables you to easily generate and use your own encryption keys on the AWS Cloud. With CloudHSM, you can manage your own encryption keys using FIPS 140-2 Level 3 validated HSMs.
[3]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

9. Enter a Cavium HSM user in the Cavium HSM User field.
10. Provide an AWS Key Management System (KMS) Master Key ID used to generate transaction data keys in the KMS Master Key ID field.
11. Enter an AWS KMS Service Connection name in the KMS Service Connection field.
12. Click Save to update the changes.

NOTE

For Azure installations, check the Azure Submission Data Storage service that persists form submissions and related transactions.

Next, learn about the Azure Submission Data Storage service.

Azure Submission Data Storage Service

Manager provides the Azure submission data storage service to persist form submissions and related transactions in the Azure[1] Cloud Storage. The service requires an Azure service connection to connect to the cloud using the Azure Key Vault service.

This service is the default submission data storage service for Azure cloud installations. For more information, see the installation guide.

> NOTE
>
> Cloud specific services and some service connections are not available and can't be manually created, unless the deployment environment matches your installation mode. To check the installation mode of your Manager server, select System > System Info, find the Build Properties section, and check the `deployment.env` value. For more information, see system info.

Manager's implementation of an Azure service caches a connection to the Azure cloud service for 5 minutes to improve server performance, reduce network bandwidth, and prevent unnecessary load on the systems. However, if an error occurs when configuring the storage service, there may be a delay up to 5 minutes once the problem is rectified for the changes to have effect.

To create the Azure submission data storage service:

1. Select Services > Core Global Services and click New.
2. Select Submission Data Storage from the Service Type dropdown list.
3. Select Azure Submission Data Storage from the Service Template dropdown list.
4. Provide a unique service name in the New Name field, for example, Azure Submission Data Storage.
5. Specify a version in the Version Number field, for example, 1.0.0.
6. Select an organization from the Organization dropdown list.
7. Click Save to update the changes.

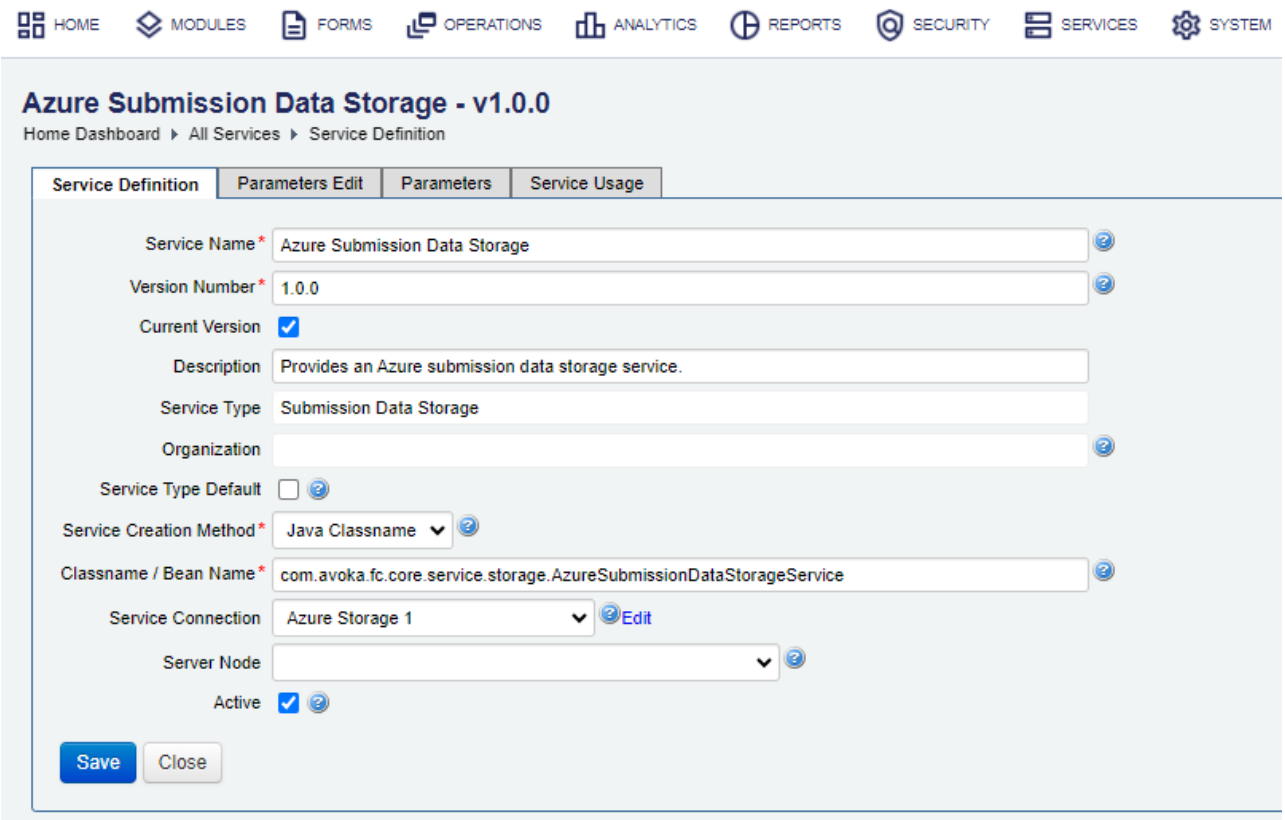
To configure the Azure submission data storage service:

1. Select Services > Core Global Services.
2. Locate the Azure Submission Data Storage service and click Edit.

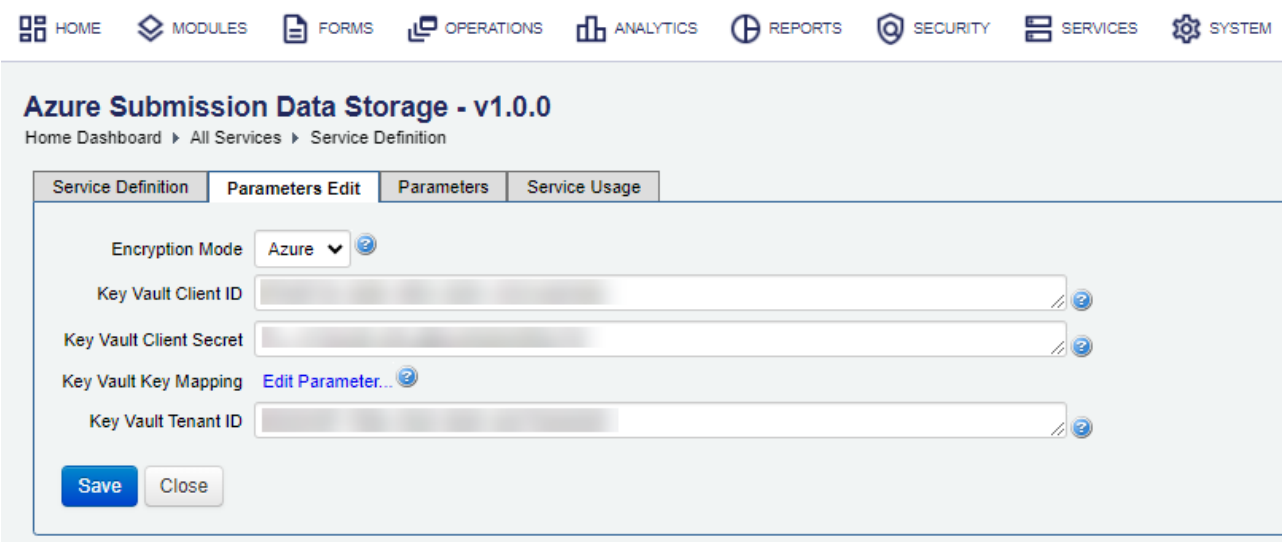
---

[1]Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers.

3. Click the Service Definition tab to configure the service.



4. Select an Azure Storage service connection from the Service Connection dropdown list, which the service will use to publish the events to. For example, Azure Storage 1. Alternatively, click the Edit link to configure the selected service connection.
5. Click Save to update the changes.
6. Click the Parameters Edit tab to configure the service parameters used for encryption key management. For more information, see Microsoft Azure documentation.



7. Select a submission data encryption mode from the Encryption Mode dropdown list, which can be one of the following:
   - Local
   - Azure
8. Provide a key vault client id in the Key Vault Client ID field.
9. Enter a key vault client secret in the Key Vault Client Secret field.
10. Specify a key vault master key mappings as JSON[1] document in the Key Vault Key Mapping field.
11. Enter a key vault tenant id in the Key Vault Tenant ID field.
12. Click Save to update the changes.

NOTE

For AWS installations, check the Amazon Submission Data Storage service that persists form submissions and related transactions.

Next, learn how to view all core global services.

[1] JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

System Event Publisher Service

Manager allows you to create and configure a System Event Publisher as a service to publish events to various queues, such as Amazon[1] or Azure[2]. Different System Event Publishers have the same configuration with the exception to a service connection.

> NOTE
>
> You need to create and configure a specific service connection before you can configure your System Event Publisher.

Manager comes with the following ready-to-use System Event Publishers:

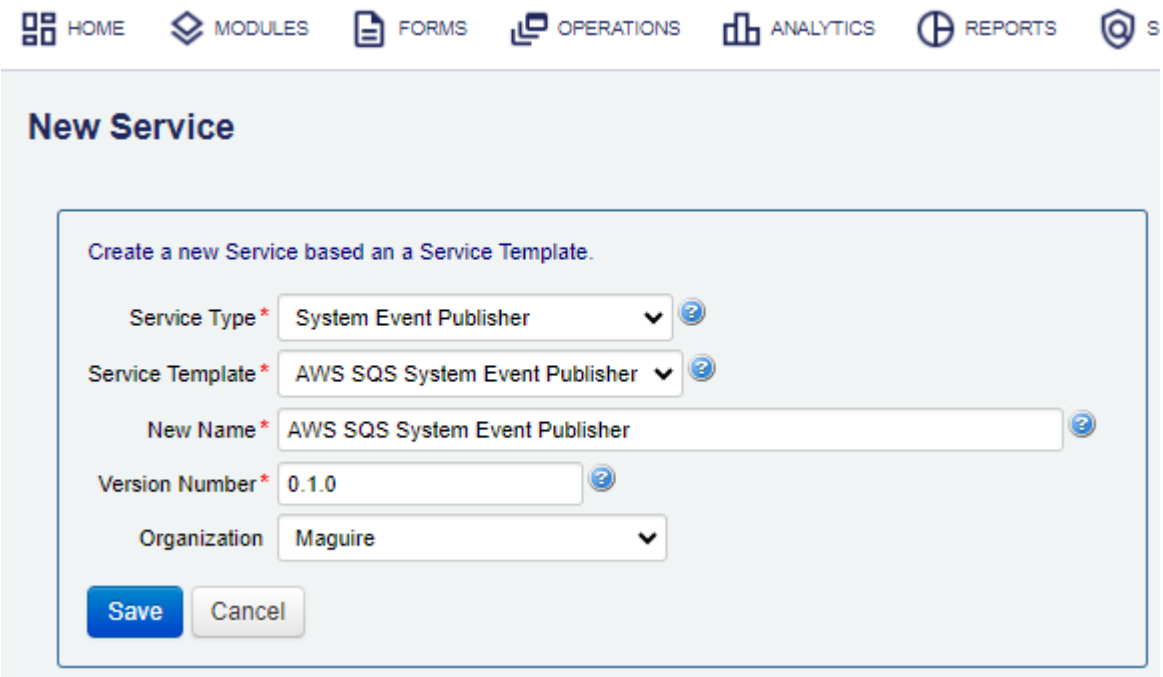
- AWS SQS System Event Publisher service
- Azure System Event Publisher service

> NOTE
>
> You must be a Journey Manager administrator to create or configure this functionality.

To create a System Event Publisher service:

[1]Amazon Web Services (AWS) is a comprehensive, evolving cloud computing platform provided by Amazon. It provides a mix of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings.
[2]Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers.

1. Select Services > Core Global Services.
2. Click New.

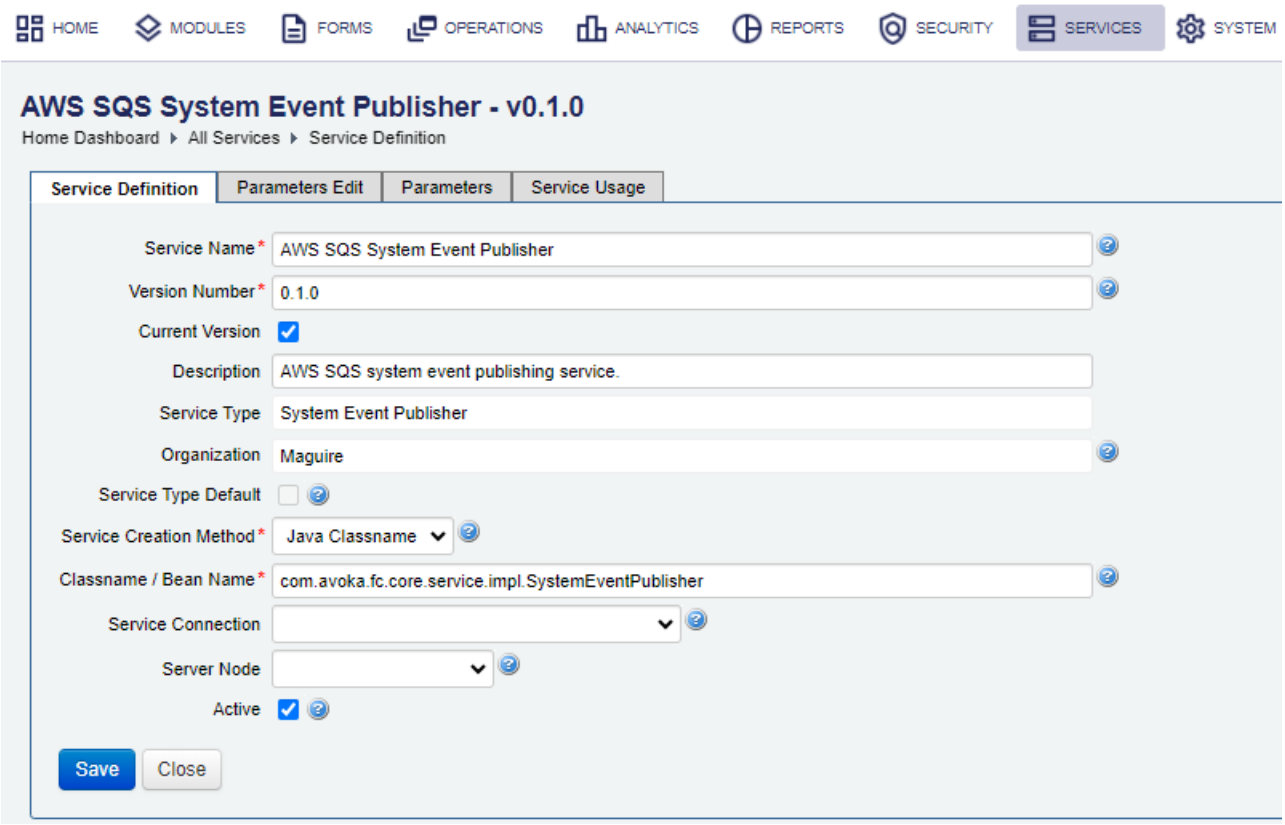


3. Select the System Event Publisher service type from the Service Type dropdown list.
4. Select a System Event Publisher template from the Service Template dropdown list to automatically prefills the new service with the default parameters.
5. Edit the service name in the New Name field, which is by default the name of a service template.
6. Enter a unique name in the Version Number field.
7. Select an organization for this service from the Organization dropdown list, or leave it blank for the global service.
8. Click Save to update the changes.

To configure a System Event Publisher service:

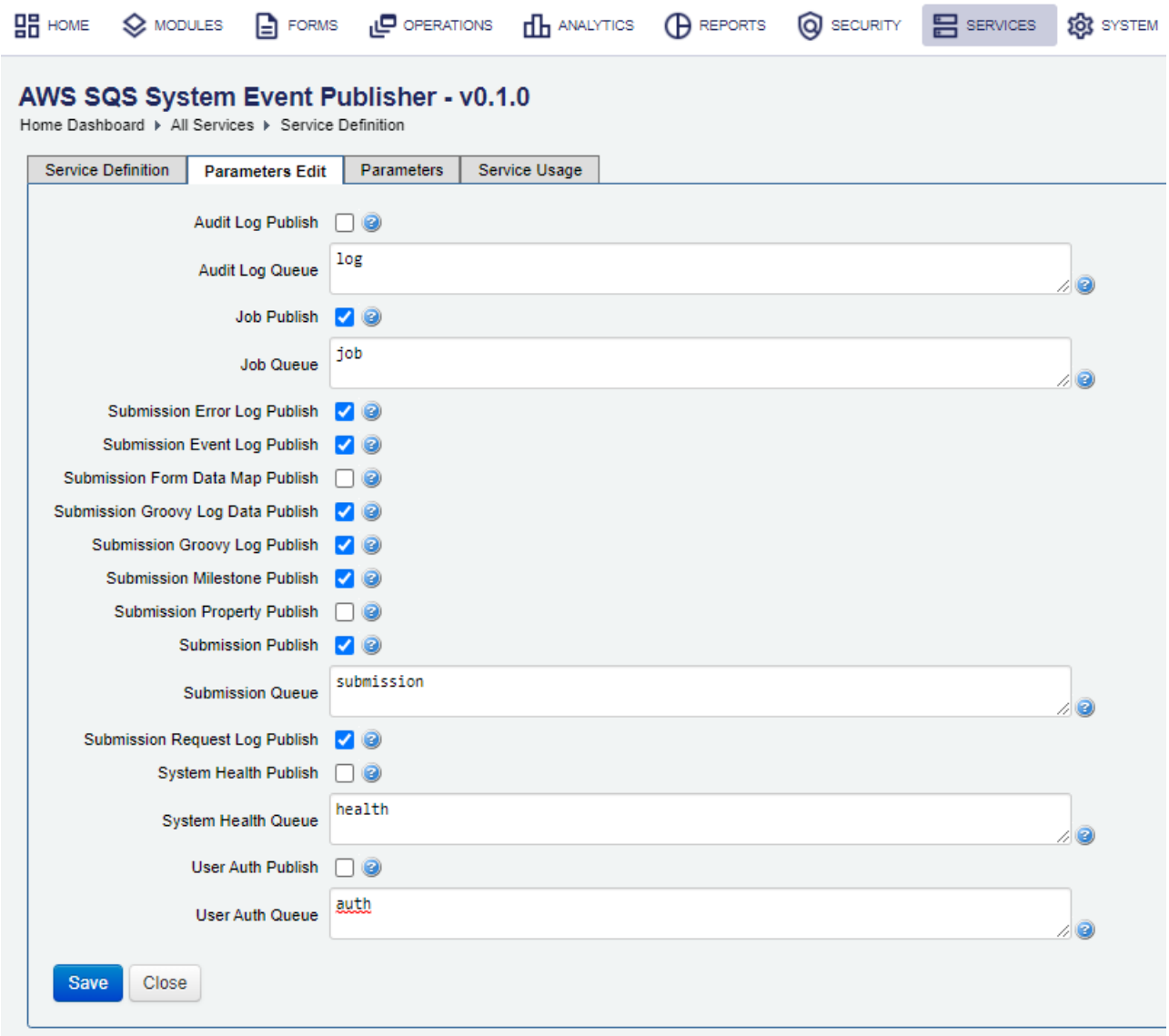
1. Select Services > Core Global Services.
2. Locate a System Event Publisher service and click Edit.

3. Select the Service Definition tab to configure the service.



4. Select a service connection from the Service Connection dropdown list, which the service will use to publish the events. Alternatively, click the Edit link to configure the selected service connection.
5. Click Save to update the changes.

6. Select the Parameters Edit tab to configure the service parameters.



> NOTE
>
> You can edit the parameters in the Parameters tab as well.

7. Select Audit Log Publish to publish audit log events. It is not selected by default.
8. Specify a queue name in the Audit Log Queue field to publish audit log events to, or leave it blank to use the default queue specified in the Queue Name of this service connection.
9. Select Job Publish to publish job events.
10. Specify a queue name in the Job Queue field to publish job events to, or leave it blank to use the default queue specified in the Queue Name of this service connection.

11. Select Submission Error Log Publish to publish the submission error log as part of a submission event. The Submission Publish checkbox below must also be selected. It is selected by default.
12. Select Submission Event Log Publish to publish the submission event log as part of a submission event. The Submission Publish checkbox below must also be selected. It is selected by default.
13. Select Submission Form Data Map Publish to publish the submission form data map as part of a submission event. The Submission Publish checkbox must also be selected. It is selected by default.
14. Select Submission Groovy Log Data Publish to publish added submission groovy service log data as part of a submission event. The Submission Publish checkbox must also be ticked. It is selected by default.
15. Select Submission Groovy Log Publish to publish added submission groovy service logs as part of a submission event. The Submission Publish checkbox below must also be selected. It is selected by default.
16. Select Submission Milestone Publish to publish added submission milestones as part of a submission event. The Submission Publish checkbox below must also be selected. It is not selected by default.
17. Select Submission Property Publish to publish submission properties as part of a submission event. It is not selected by default.

    > NOTE
    >
    > You also must select the Submission Publish checkbox below for the Submission Property Publish setting to take effect.

    Adding all properties to the event can significantly increase its size, which, in turn, may cause a system event publisher delivery error when using AWS. It depends on the underlying queue configuration. For AWS, if a message size exceeds the maximum allowed, an error will occur, for example, `One or more parameters are invalid. Reason: Message must be shorter than 262144 bytes`.

18. Select Submission Publish to publish submission events. It is selected by default.
19. Specify a queue name in the Submission Queue field to publish submission events to, or leave it blank to use the default queue specified in the Queue Name of this service connection.

20. Select Submission Request Log Publish to publish added submission request log as part of a submission event. The Submission Publish checkbox below must also be selected. It is selected by default.
21. Select System Health Publish to publish hourly system health events. It is not selected by default.
22. Specify a queue name in the System Health Queue field to publish system health events to, or leave it blank to use the default queue specified in the Queue Name of this service connection.
23. Select User Auth Publish to publish user authentication events. | 18.11 This feature was introduced in 18.11.
24. Specify a queue name in the User Auth Queue field to publish authentication events to, or leave it blank to use the default queue specified in the Queue Name of this service connection. | 18.11 This feature was introduced in 18.11.
25. Click Save to update the changes.

Next, learn how to configure an AWS SQS System Event Publisher service.

AWS SQS System Event Publisher Service

Manager comes with the AWS SQS System Event Publisher service that allows you to publish events and elements to AWS SQS[1] queues. The service uses the AWS Security Token Service (STS) protocol to authenticate the AWS STS Service Connection. However, you can use the standard AWS SQS Service Connection as well.

> NOTE
>
> You must be a Journey Manager administrator to create or configure this functionality.

To create and configure the AWS SQS System Event Publisher service:

1. Create and configure the AWS SQS service connection or the AWS STS service connection for the STS authentication protocol.
2. Configure the AWS SQS System Event Publisher service using this service connection.

> NOTE
>
> AWS SQS has a maximum message size, so if an event you're trying to publish is larger than that, you'll get an error. For more information, see https://aws.amazon.com/sqs/faqs/

Next, learn how to configure an Azure system event publisher service.

[1]Amazon Simple Queue Service (Amazon SQS) is a pay-per-use web service for storing messages in transit between computers. Developers use SQS to build distributed applications with decoupled components without having to deal with the overhead of creating and maintaining message queues.

Azure System Event Publisher Service

Manager comes with the Azure System Event Publisher service that allows you to publish events and elements to queues stored in Azure[1] Storage.

> NOTE
>
> You must be a Journey Manager administrator to create or configure this functionality.

To create and configure the Azure System Event Publisher service:

1. Create and configure the Azure Storage service connection.
2. Configure the Azure System Event Publisher service using this service connection.

Next, learn how to configure an AWS SQS system event publisher service.

[1]Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers.

Chromium Puppeteer Receipt Service

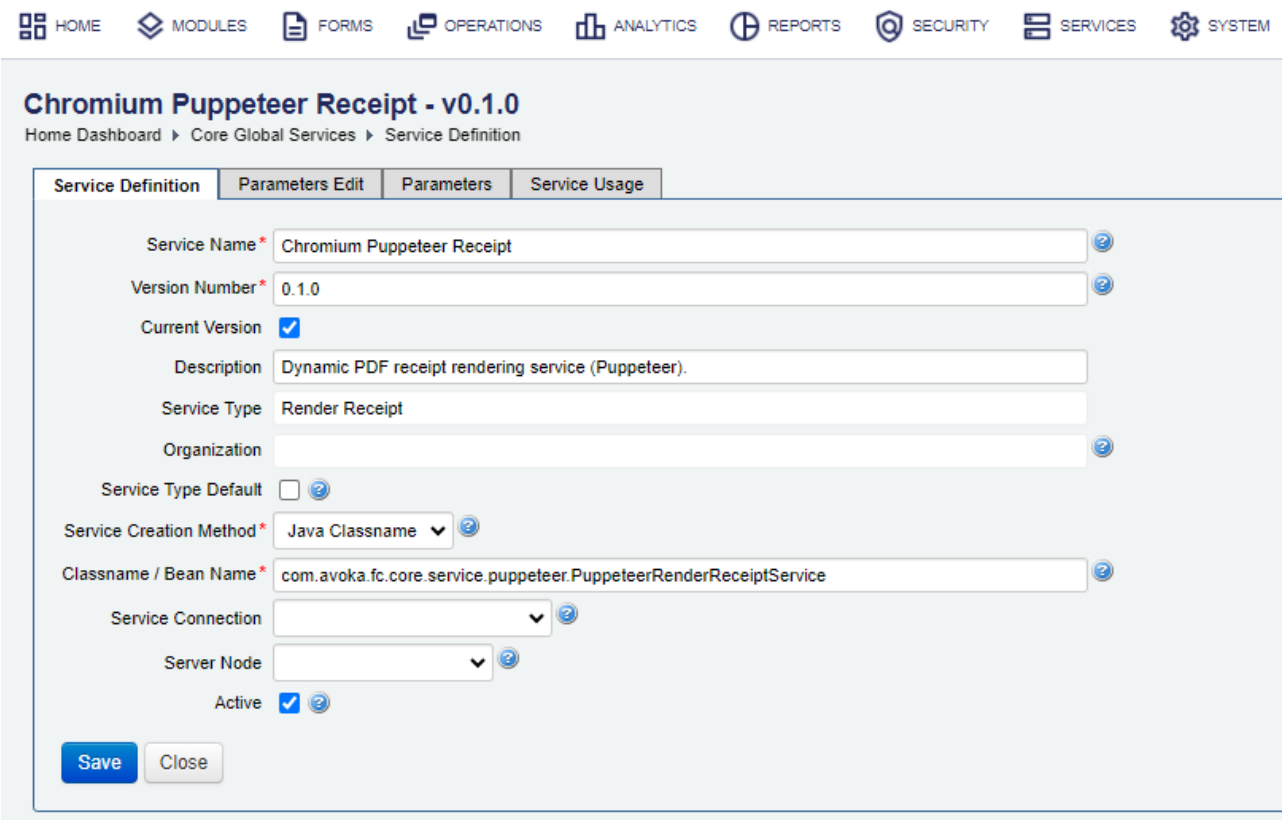
Manager provides the Chromium Puppeteer Receipt service, which you can use to generate dynamic PDF receipts of forms. The service utilizes the Puppeteer[1] library that supports a wide range of styling options.

To create the Chromium Puppeteer Receipt service:

1. Select Services > Core Global Services and click New.
2. Select Render Receipt from the Service Type dropdown list.
3. Select Chromium Puppeteer Receipt from the Service Template dropdown list.
4. Update the service name in the New Name field.
5. Update the service version in the Version Number field.
6. Select an optional organization from the Organization dropdown list.

> NOTE
>
> If organization is blank, the service will be global, otherwise it will be available to the selected organization only.

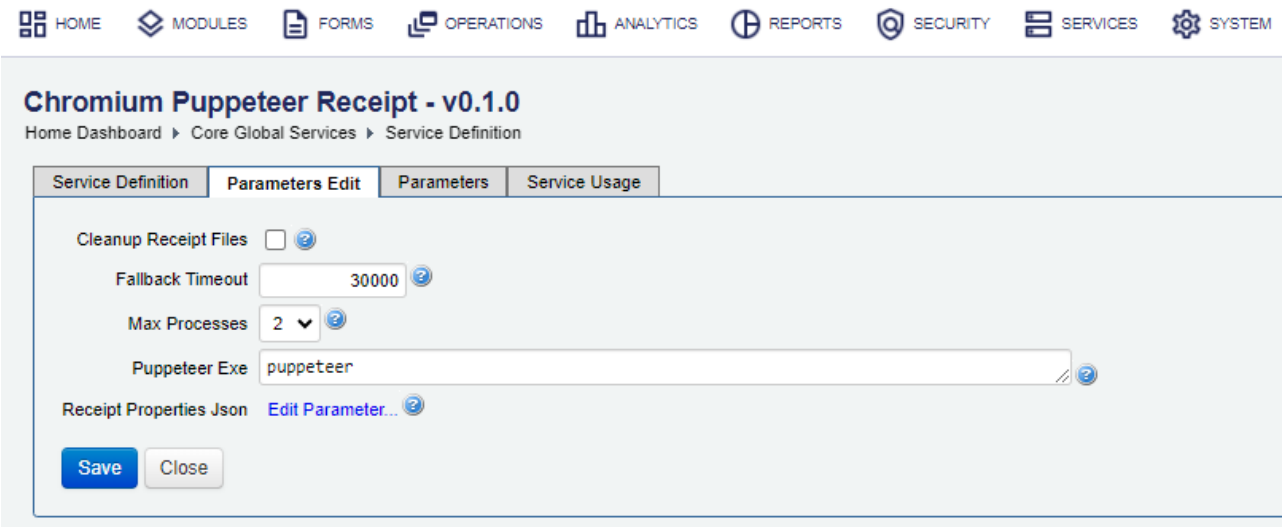
[1]Puppeteer is a Node library which provides a high-level API to control headless Chrome or Chromium over the DevTools Protocol. It can also be configured to use full (non-headless) Chrome or Chromium.



7. Click Save to update the changes.

To configure the Chromium Puppeteer Receipt service:

1. Select Services > Core Global Services.
2. Locate the service and click Edit.
3. Click the Service Definition tab to edit the standard service configuration.
4. Click the Parameters Edit tab to view or update the service configuration.



5. Select the Cleanup Receipt Files checkbox to clean up the receipt files after rendering has completed.
6. Edit the fallback process timeout (in milliseconds) in the Fallback Timeout field. The default is 3000.
7. Select the maximum number of concurrent processes from the Max Processes dropdown list. The default is 2.

   NOTE

   Changing Max Processes will not take effect until a Manager server is restarted.

8. The Puppeteer Exe field contains the file name of the puppeteer executable that will be run to generate a receipt. Modify the file name as required, without the Windows .exe file extension. The default value is puppeteer.
9. Click Receipt Properties Json to edit the properties that will be passed to the `puppeteer.js` script for rendering of customized headers and footers.

```
{
        "viewportSize": {
                "width": 800,
                "height": 600
        },

        "paperSize": {
                "format": "A4",
                "landscape": false,
                "margin": {
                        "top": "1.5cm",
                        "left": "1.5cm",
                        "bottom": "1.5cm",
                        "right": "1.5cm"
                }
        },
        "headerTemplate": "",
        "footerTemplate": "<div style='width:100%;text-align: center;'><span style='font-
size: 8px;'>Page <span class='pageNumber'></span> of <span class-
='totalPages'></span></span></div>",
        /* TJM configuration options */
        "config": {
                "loadTimeoutMS": 20000,
                "postLoadTimeoutMS": 500,
                "logOptions": true,
                "logHeadersFooters": true,
                "logResourceEvents": true,
                "logFormCallbacks": true,
                "disableFormOverrides": true,
                "supportOldStyleFormOverrides": "composer",
                "treatFormErrorsAsFatal": false,
                "forceError": false,
                "customCSS": ".phantom-header {} .phantom-footer {}"
        }
}
```

10. Click Save to update the changes.
11. Assign this service to your form version as the Receipt Render Service, so it can render a form submission as a PDF receipt. You can also make this service as default.

NOTE

Customizing the Chromium Puppeteer Receipt service will impact all receipts of forms that are linked to this service.

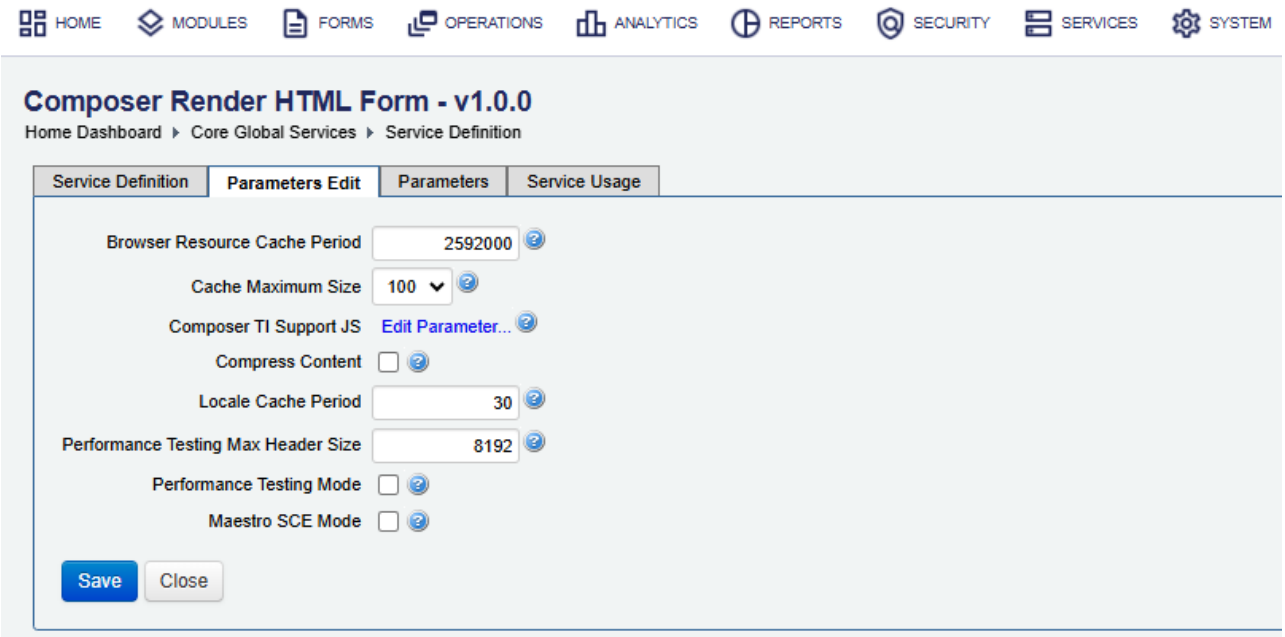
Next, learn how to view core global services.

Composer Render HTML Form Service

Manager comes with the Composer Render HTML Form core global service, which renders Composer SmartForms.

To configure the Composer Render HTML Form service:

1. Select Services > Core Global Services.
2. Locate the service and click Edit.
3. Configure the standard service settings.
4. Select the Parameters Edit tab to view and edit the configuration.



5. Edit the browser resource caching period in seconds in the Browser Resource Cache Period field. Default is 2592000.
6. Edit the server side form HTML and resource cache maximum size in the Cache Maximum Size field. Default is 100.
7. Click Composer TI Support JS to view and edit the Composer Transact Insights Support JavaScript file include. You can use the include file when Composer Form was not published with TI support.

8. Select the Compress Content checkbox to GZIP[1] compress the HTML content.

> NOTE
>
> It's recommended to use Apache `mode_deflate` for HTML compression.

9. Edit the locale JSON[2] resources caching period in seconds in the Locale Cache Period field. Default is 30.
10. Select the Performance Testing Mode checkbox to enable Performance Testing mode which renders HTTP response header `xml_data` for use by automation tools.
11. Select the Maestro SCE Mode checkbox to enable Maestro Static Content Externalization[3] rendering mode. | 18.11 This feature was introduced in 18.11.
12. Click Save to update the changes.

Next, learn how to view core global services.

---

[1]GZIP is a file format and a software application used for file compression and decompression.
[2]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.
[3]Static Content Externalization (SCE).

Current Version Selector Service

Manager comes with the Current Version Selector service, which is a Core Global service, allowing you to choose the most appropriate form version of to a form to be rendered. The default service returns the current form version unless a specific version was requested.

If you would like to customize the way versions are selected, for example, for A/B testing, you can create your own form version selector service and implement custom logic in a Groovy[1] script.

> NOTE
>
> This service can be configured on the form level, otherwise, the system default version selector service is used.

To configure the Current Version Selector service:

1. Select Services > Core Global Services.
2. Locate the service and click Edit.
3. Configure the standard service settings.
4. Select the Parameters Edit tab to view and edit the specific service configuration.
5. Select the Ignore Form Version Request Parameter checkbox to always use the current form version regardless of the value of the `tmFormVersion` request parameter. This can prevent users loading and creating transactions of an old form version by providing the `tmFormVersion` URL parameter. This option is not selected by default.
6. Click Save to update the changes.

Next, learn about A/B testing form version selector service.

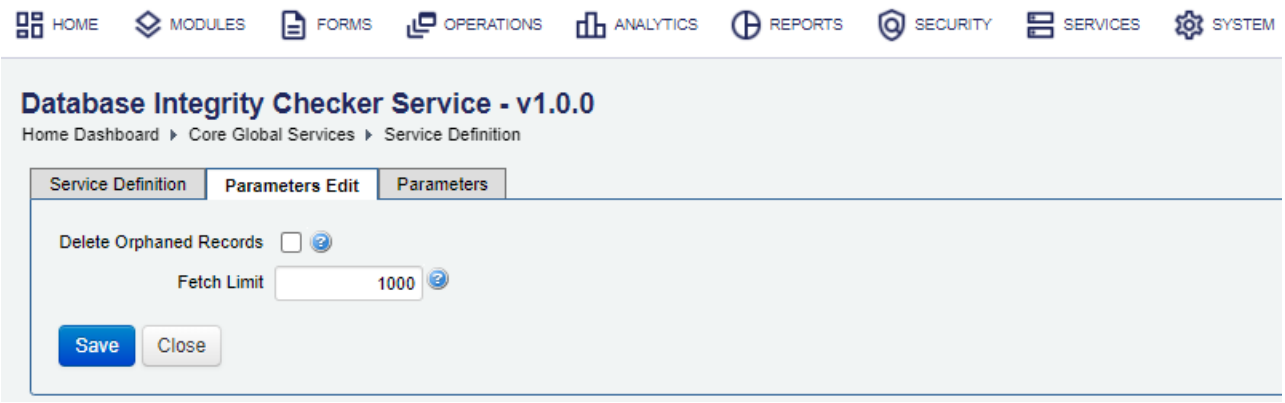
[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

Database Integrity Checker Service

Manager comes with the Database Integrity Checker service, which checks the integrity of the database and optionally performs remedial actions for database records that can potentially cause system instability. The service is invoked by the Database Integrity Checker scheduled job, which you need set up according to your requirements.

To configure the Database Integrity Checker service:

1. Select Services > Core Global Services.
2. Locate the Database Integrity Checker service and click Edit.
3. Configure the standard service settings.
4. Select the Parameters Edit tab to configure the service parameters.



5. Select the Delete Orphaned Records checkbox to automatically purge invalid orphaned database records, which are causing a problem to the stability of the system. This is not selected by default.

   WARNING

   Before enabling this option, you must discuss it with our Support Team. A comprehensive analysis of your system may be required to understand why orphaned records do appear and then apply some configuration changes to stop them from happening.

   Each time the Database Integrity Checker service runs, it checks whether this option is enabled, and, if so, performs the following actions:

   - Sets Database Integrity Checker back to false, so it isn't left permanently on.
   - Deletes all orphaned records within Fetch Limit.

- Adds a new entry to event log, as shown below:

```
Database Integrity Checker Job removed 2 invalid/orphaned groups:
Deleted 1 record(s) for Entity: submission_property_search Field: submission_
property_oid TargetEntity: submission_property
Deleted 2 record(s) for Entity: submission_property Field: submission_oid Tar-
getEntity: submission
```

  The existing event log entry for checking database integrity now includes the number of entries found, as illustrated below:

```
Database Integrity Checker Job found 2 invalid/orphaned groups:
1 record(s) for Entity: submission_property_search Field: submission_property_
oid TargetEntity: submission_property
2 record(s) for Entity: submission_property Field: submission_oid Tar-
getEntity: submission
```

6. Set the maximum number of records that needs to be processed in the Fetch Limit field. The default is 1000.
7. Click Save to update the changes.

Next, learn how to view core global services.

Data Retention Management Service

Manager comes with the Data Retention Management service that performs the following critical steps in the transaction life cycle:

- Task expiry
- Abandonment
- Data purging

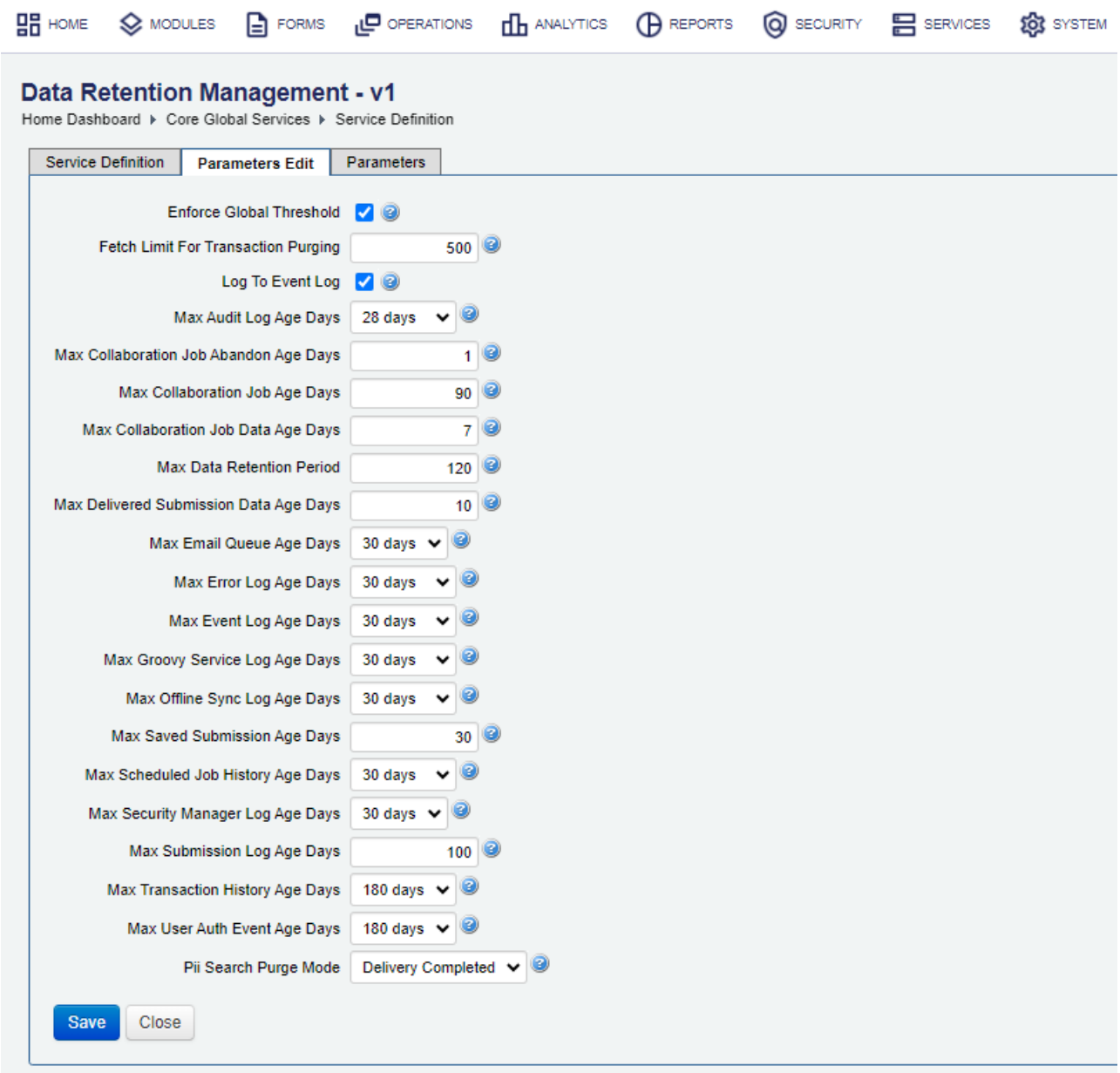
This service is periodically called by the Data Retention Job and the Transaction Processor service, if its configuration has the Include Data Retention checkbox selected.

> NOTE
>
> The Data Retention Management service is the default service for the Data Retention Management type. However, you can use the Data Retention Management v2 service, which provides performance improvements for the Transaction Processor service.
> | 24.04 This feature was introduced in 24.04

To configure the Data Retention Management service:

1. Select Services > Core Global Services.
2. Select the Data Retention Management from the Type dropdown list to show all services of this type.
3. Locate the Data Retention Management service and click Edit.
4. Check the standard service settings.

5. Select the Parameters Edit tab to view and customize the service configuration.



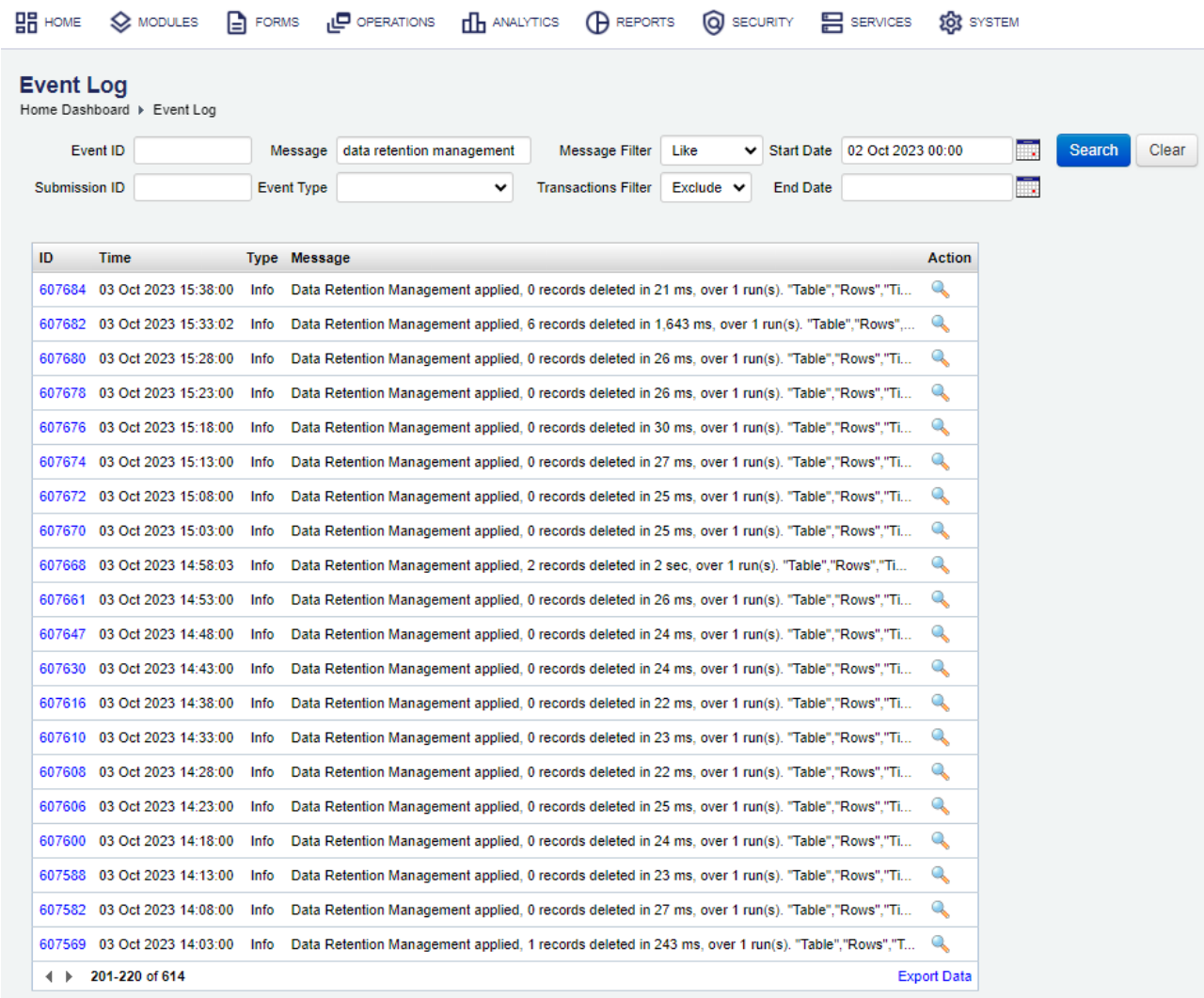
> NOTE
>
> For explanation of the parameters, see Global Data Retention Configuration.
> Some of the service's settings duplicate the global data retention configuration settings, so you can override them.

6. Select the Enforce Global Threshold checkbox to ensure that data retention policies can't be longer than that of global system or organization settings.

7. Edit the maximum number of items to be purged per each category: submissions, collaboration jobs, transaction history, error logs, and so on in the Fetch Limit For Transaction Purging field, which is. The default is 500. For more information, see Transaction Processor service.
8. Select the Log to Event Log checkbox to log the results of applying the data retention policy to the event log, so you can see what data was deleted from the system. The example of a message recorded in the even logs is shown bellow:

```
Data Retention Management applied, 0 records deleted in 3 sec.

"Table","Rows","Time (ms)","Percentage"
"Delete Submission Data",0,77,2.6
"Job",0,7,0.2
"Submission",0,10,0.3
"Transaction History",0,4,0.1
"Audit Log",0,4,0.1
"User Auth Event",0,3,0.1
"Error Log",0,13,0.3
"Event Log",0,3784,96.2
"Groovy Service Log",0,3,0.1
"Offline Client System Log",0,3,0.1
"Offline Sync Call",0,4,0.1
"Offline Sync Log",0,3,0.1
"Scheduled Job History",0,3,0.1
"Security Manager Log",0,3,0.1
"Email Queue",0,4,0.1
"Composer Package Queue",0,3,0.1
"Import Action",0,5,0.1
```

Then you can search for particular retention event in the event logs using the following criteria:



9. Select the age in days at which an audit log entry becomes eligible for purging from the Max Audit Log Age Days dropdown list.
10. Edit the number of days in the Max Collaboration Job Abandon Age Days field to specify when a collaboration job and all its transactions become eligible for abandonment.

> NOTE
>
> You can disable collaboration job abandonment by setting the Max Collaboration Job Abandon Age Days to `0`.

> NOTE
>
> Check collaboration job abandonment configuration in Global Data Retention Configuration and Configure Organization Data Retention.

11. Edit the age in days at which a collaboration job and its transactions become eligible for abandonment in the Max Collaboration Job Age Days field. The enforced range of this value is:
    - Min: 1 day
    - Recommended: 30 days
    - Max (Strict mode): 730 days
    - Max (Relaxed mode): 730 days

    NOTE

    Manager enforces the maximum limits of data retentions to provide required system stability.

12. Edit the age in days at which a collaboration job data become eligible for purging in the Max Collaboration Job Data Age Days field.
13. Edit the maximum period of time, in seconds, allocated for data retention purging in the Max Data Retention Period field. The default is 120. For more information, see Transaction Processor Service.
14. Edit the age in days at which delivered transaction data (XML, PDFs, file attachments) become eligible for purging in the Max Delivered Submission Data Age Days field. The default is 10.
15. Select the age in days at which an email queue entry becomes eligible for purging from the Max Email Queue Age Days dropdown list. The default is 30 days.
16. Select the age in days at which an error log entry becomes eligible for purging from the Max Error Log Age Days dropdown list. The default is 30 days.
17. Select the age in days at which an event log entry becomes eligible for purging from the Max Event Log Age Days dropdown list. The default is 30 days.
18. Select the age in days at which a Groovy service log entry becomes eligible for purging from the Max Groovy Service Log Age Days dropdown list. The default is 30 days.
19. Select the age in days at which a T.Field App client and sync log entry become eligible for purging from the Max Offline Sync Log Age Days dropdown list. The default is 30 days.
20. Edit the age in days after which saved transactions will be abandoned in the Max Saved Submission Age Days field. The default is 30 days.
21. Select the age in days at which a scheduled job history entry becomes eligible for purging from the Max Scheduled Job History Age Days dropdown list. The default is 30 days.
22. Select the age in days at which a security manager log entry becomes eligible for purging from the Max Security Manager Log Age Days dropdown list. The default is 30 days.

23. Edit the age in days after which a transaction record becomes eligible for purging in the Max Submission Log Age Days field. The enforced range of this value is:
    - Min: 1 day
    - Recommended: 100 days
    - Max (Strict mode): 730 days
    - Max (Relaxed mode): 1825 days
24. Select the age in days at which a transaction history entry becomes eligible for purging from the Max Transaction History Age Days dropdown list. The default is 180 days.
25. Select the age in days at which a user authentication event entry becomes eligible for purging from the Max User Auth Event Age Days dropdown list. The default is 180 days.
26. Select the required purge mode from the PII Search Purge Mode dropdown list to specify when PII searching metadata, such as data extracts and transactions properties, will be deleted:
    - Form Completed: the smallest set of data to be deleted.
    - Delivery Completed: a bigger set of data to be deleted.
    - PII Purge Time: the biggest set of data to be deleted and it takes the longest time to do it.
27. Click Save to update the changes.

Next, learn how to view all core global services.

Data Retention Management v2 Service

Manager comes with the Data Retention Management v2 service, which extends the Data Retention Management service and provides performance improvements, such as reduced memory footprint and more efficient database queries for the Transaction Processor service. Enabling and running the Data Retention Management v2 servicee will help clear any data backlog and gradually improve system performance.

IMPORTANT

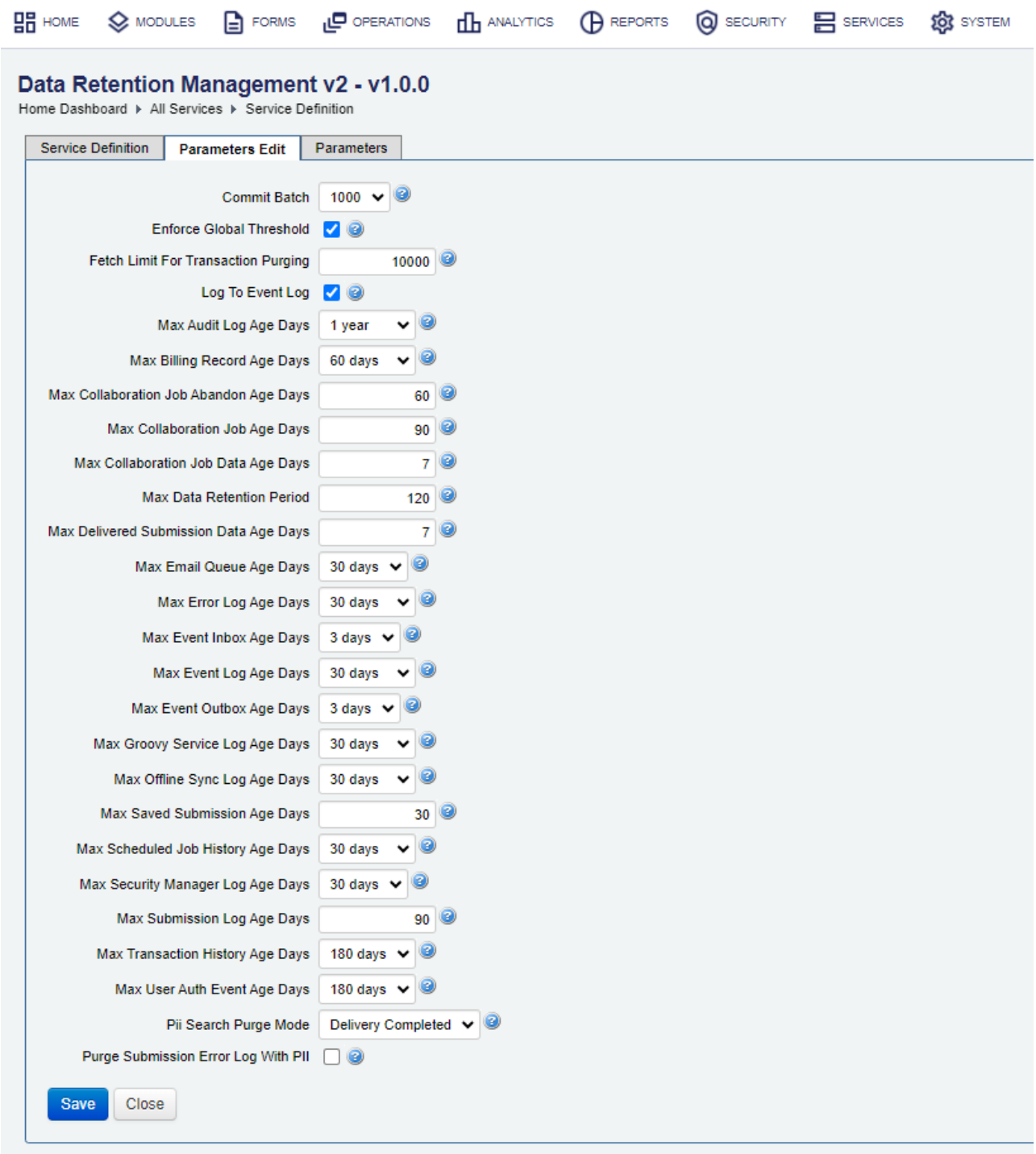
When you install or upgrade Manager, the Data Retention Management v2 service is not set as the default service for the Data Retention Management type - the Data Retention Management service is set by default. However, as with any other services, you can change it by following these steps:

1. Apply any existing Data Retention Management service settings to the Data Retention Management v2 service. This is particularly important as every client has their own customized settings which are aligned with particular business requirements and legal obligations to keep and purge PII data.
2. Set the Data Retention Management v2 service as default by clicking Make Default or selecting the Service Type Default checkbox.
3. Test the Data Retention Management v2 service in your lower environments, such as DEV and TEST.
4. Confirm successful results before moving to the next environment.
5. Enable it in your production environment when you are completely satisfied with test results.

To configure the Data Retention Management v2 service:

1. Select Services > Core Global Services.
2. Select the Data Retention Management from the Type dropdown list to show all services of this type.
3. Locate the Data Retention Management v2 service and click Edit.
4. Check and update the standard service settings.

5. Select the Parameters Edit tab to view and customize the service configuration.



NOTE

For explanation of the parameters, see Data Retention Management Service.
Some of the service's settings duplicate the Global Data Retention Configuration, so you can override them.

WARNING

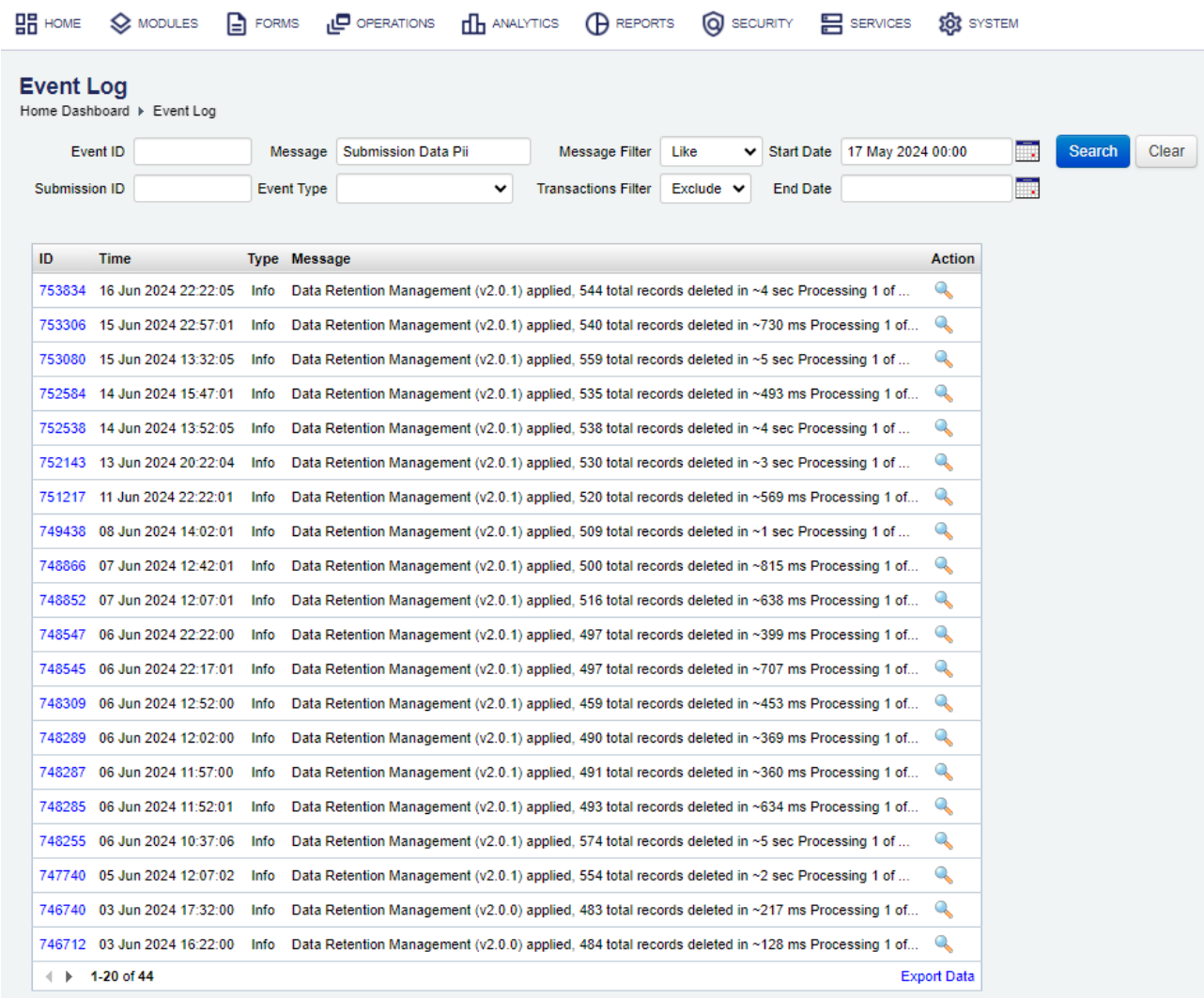
This service was introduced in Journey Manager 24.04 and then backported to all supported releases. However, there are some difference in the following versions:

- Prior to 24.04, not all of these settings are available.
- Prior to 23.10, some of these settings, such as Fetch Limit For Transaction Purging and Max Data Retention Period are only configured in the Transaction Processor service.

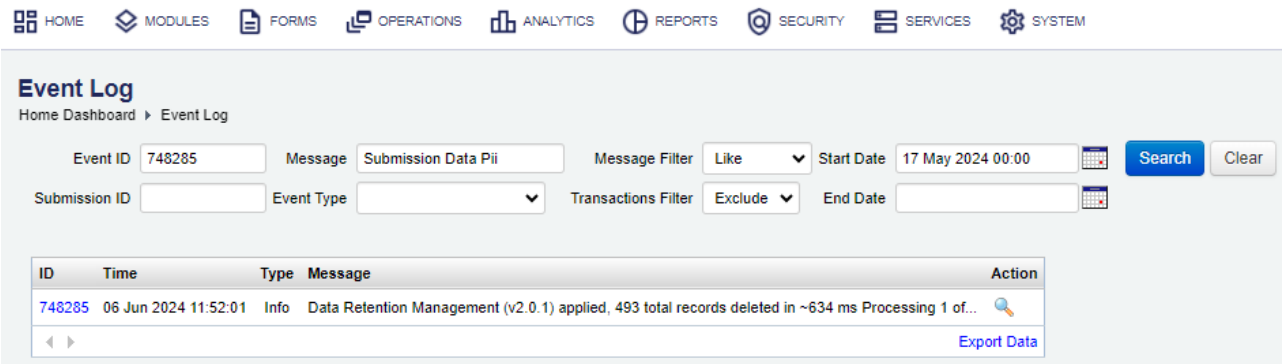
6. Select a size of a commit batch, which is used for transaction purging, from the Commit Batch dropdown list. The value must be within the fetch limit specified below. The default is 1000.
7. Select the Log to Event Log checkbox to log the results of applying the data retention policy to the event log, so you can see what data was deleted from the system. The example of a message recorded in the even logs is shown bellow:

```
Data Retention Management (v2.0.1) applied, 493 total records deleted in ~634 ms

Processing 1 of 2 - Transactions PII Data purged 1 records in ~18 ms:

"AWS Submission Data Storage" removed 1 records in 0 ms
"File Upload Batch Update" updated 0 records in 0 ms
"Submission History Batch Update" updated 0 records in 0 ms
"Async Msg and Body PII" removed 0 records in 0 ms
"Extracts and Search PII" removed 0 records in 3 ms
"File Uploads and Data PII" removed 0 records in 0 ms
"Groovy Service Logs and Data PII" removed 0 records in 1 ms
"Submission History Data PII" removed 0 records in 2 ms
"Job Properties and Search PII" removed 0 records in 2 ms
"Submission Error Logs" removed 0 records in 0 ms
"Submission Properties and Search PII" removed 0 records in 2 ms
"Submission Data PII" removed 0 records in 0 ms

Processing 2 of 2 - Transactions and Related Data purged 492 records in ~615 ms:

"Audit Log(s)" removed 0 records in 0 ms
"Email Queue(s)" removed 456 records in 42 ms
"Error Log(s)" removed 0 records in 18 ms
"Event Log(s)" removed 2 records in 14 ms
"Groovy Service Log(s)" removed 0 records in 13 ms
"Scheduled Job History(s)" removed 33 records in 15 ms
"Security Manager Log(s)" removed 0 records in 13 ms
"Submission(s)" removed 1 records in 453 ms
"Transaction History(s)" removed 0 records in 13 ms
"User Authorized Event(s)" removed 0 records in 14 ms
"Composer Package Queue(s)" removed 0 records in 15 ms
"Import Action(s)" removed 0 records in 2 ms
"Job(s)" removed 0 records in 1 ms
"Job Group(s)" removed 0 records in 1 ms
"Request Log(s)" removed 0 records in 0 ms
"Submission Group(s)" removed 0 records in 1 ms
```

Then you can search for particular retention event in the event logs using the following criteria: `Submission Data Pii` as a message. This is illustrated below.



You can further narrow the search down by using an event ID, for example, `748285`, as shown below:



8. Select the age in days at which billing records become eligible for purging from the Max Billing Record Age Days dropdown list. Default is 60 days. | 24.10 This feature was introduced in 24.10

9. Select the age in days at which Event Inbox records (both Delivered and Failed) become eligible for purging from the Max Event Inbox Age Days dropdown list. Default is 3 days.
   | 24.10 This feature was introduced in 24.10
10. Select the age in days at which Event Outbox records (both Delivered and Failed) become eligible for purging from the Max Event Outbox Age Days dropdown list. Default is 3 days.
   | 24.10 This feature was introduced in 24.10
11. Select the Purge Submission Error Log With PII checkbox to delete all PII[1] data of a submission from the error log.
12. Click Save to update the changes.

Next, learn how to view all core global services.

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

Dynamic PDF Receipt Service

Manager provides the following form-specific services for generating PDF receipts:

- The Dynamic PDF Receipt service, which is the legacy receipt render service.
- The Dynamic PDF Receipt 2 service, which provides more options for customizing the receipts.

The service is called when a form receipt is opened. The service uses an orchestration in Adobe LiveCycle ES[1] to render the receipt. Typically, receipts are rendered without buttons, text fields, and other interactive form feature.

This service uses the PhantomJS[2] headless internet browser, which has no user interface.

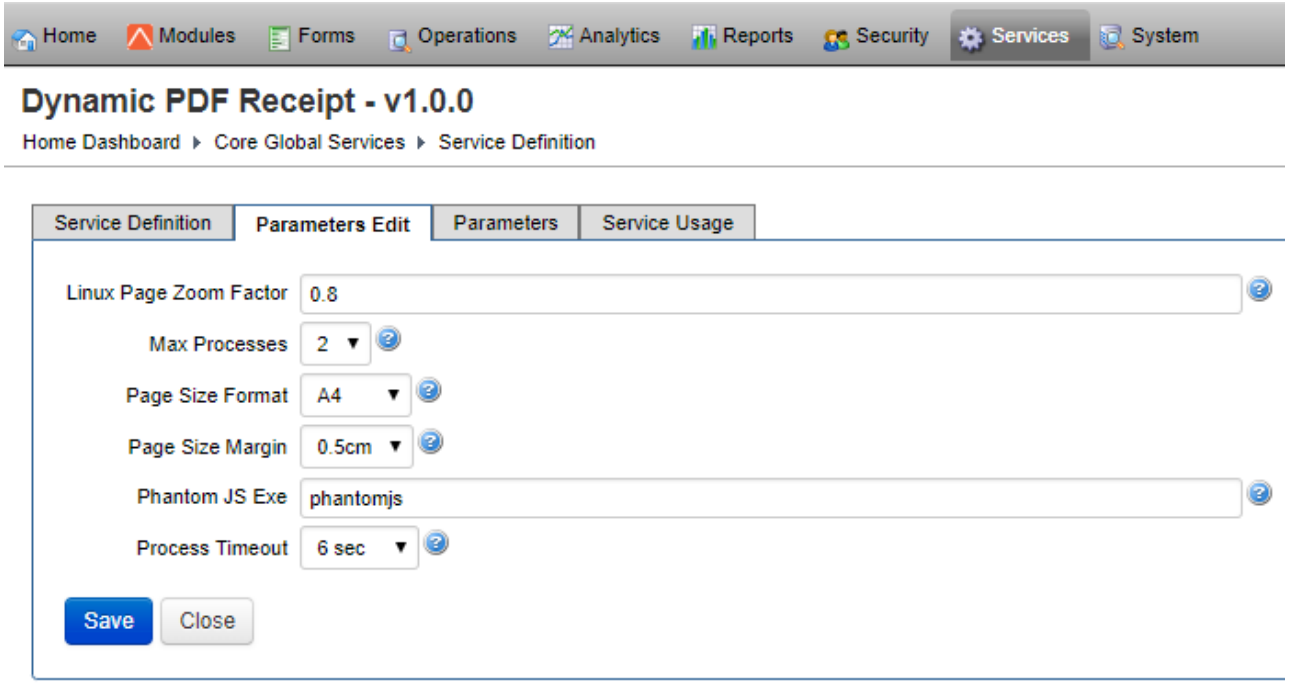
> NOTE
>
> Adobe LiveCycle support has been completely removed in 19.11, so you should use the Dynamic PDF Receipt 2 service.

To configure the Dynamic PDF Receipt service:

1. Select Services > Core Global Services.
2. Locate the service and click Edit.
3. Click the Service Definition tab to edit the standard service configuration.
4. Click the Parameters Edit tab to view or update the service configuration.

---

[1]Adobe LiveCycle Enterprise Suite (ES4) is an service-oriented architecture Java EE server software product from Adobe Systems used to build applications that automate a broad range of business processes for enterprises and government agencies. LiveCycle ES4 is an enterprise document and form platform that allows capturing and processing information, delivering personalized communications, and protecting and tracking sensitive information. It is used for purposes such as account opening, services and benefits enrollment, correspondence management, request for proposal processes, and other manual based workflows.
[2]PhantomJS is a scripted, headless browser used for automating web page interaction. PhantomJS provides a JavaScript API enabling automated navigation, screenshots, user behavior and assertions making it a common tool used to run browser-based unit tests in a headless system like a continuous integration environment. PhantomJS is based on WebKit making it a similar browsing environment to Safari and Google Chrome (before Chrome's fork of WebKit evolved into Blink). It is open-source software released under the BSD License.



5. Enter Linux Page Zoom Factor, which is the paper zoom factor for Linux servers. The default is 0.8.
6. Select the maximum number of concurrent PhantomJS[1] processes from the Max Processes dropdown list. The default is 2.

> NOTE
>
> Changing Max Processes will not take effect until Manager is restarted.

7. Select the paper size format from the Page Size Format dropdown list. The default is A4.
8. Select the paper size margin from the Page Size Margin dropdown list. The default is 0.5 cm.

[1]PhantomJS is a scripted, headless browser used for automating web page interaction. PhantomJS provides a JavaScript API enabling automated navigation, screenshots, user behavior and assertions making it a common tool used to run browser-based unit tests in a headless system like a continuous integration environment. PhantomJS is based on WebKit making it a similar browsing environment to Safari and Google Chrome (before Chrome's fork of WebKit evolved into Blink). It is open-source software released under the BSD License.

9. Provide the PhantomJS[1] executable file name (excluding the Windows .exe file extension) in the Phantom JS Exe field. The default is phantomjs.
10. Select the PhantomJS rendering process timeout in milliseconds from the Process Timeout dropdown list. The default is 6 seconds.
11. Click Save to update the changes.

NOTE

Customizing the Dynamic PDF Receipt service will impact all receipts of forms that are linked to this service.

Next, learn how to configure dynamic PDF receipt 2 service.

[1]PhantomJS is a scripted, headless browser used for automating web page interaction. PhantomJS provides a JavaScript API enabling automated navigation, screenshots, user behavior and assertions making it a common tool used to run browser-based unit tests in a headless system like a continuous integration environment. PhantomJS is based on WebKit making it a similar browsing environment to Safari and Google Chrome (before Chrome's fork of WebKit evolved into Blink). It is open-source software released under the BSD License.

Dynamic PDF Receipt 2 Service

Manager provides the Dynamic PDF Receipt 2 service for generating PDF receipts. The service is the extension of the legacy Dynamic PDF Receipt service, which belongs to Render Receipt core global services, and provides several extra customization options. The Dynamic PDF Receipt 2 service uses the PhantomJS[1] headless Internet browser to produce a receipt. Receipts can have headers and footers, which you can customize by adjusting their properties in a JSON file.

To create the Dynamic PDF Receipt 2 service:

1. Select Services > Core Global Services. and click New.
2. Select Render Receipt from the Service Type dropdown list.
3. Select Dynamic PDF Receipt 2 from the Service Template dropdown list.
4. Update the service name in the New Name field.
5. Update the service version in the Version Number field.
6. Click Save to update the changes.

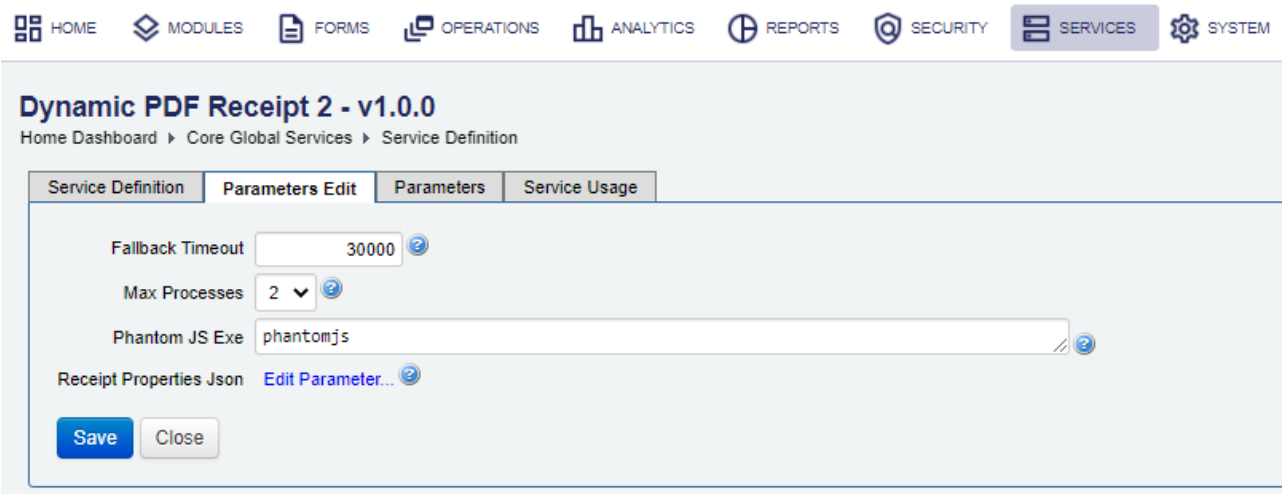
To configure the Dynamic PDF Receipt 2 service:

1. Select Services > Core Global Services.
2. Locate the service and click Edit.
3. Click the Service Definition tab to edit the standard service configuration.

---

[1]PhantomJS is a scripted, headless browser used for automating web page interaction. PhantomJS provides a JavaScript API enabling automated navigation, screenshots, user behavior and assertions making it a common tool used to run browser-based unit tests in a headless system like a continuous integration environment. PhantomJS is based on WebKit making it a similar browsing environment to Safari and Google Chrome (before Chrome's fork of WebKit evolved into Blink). It is open-source software released under the BSD License.

4. Click the Parameters Edit tab to view or update the service configuration.

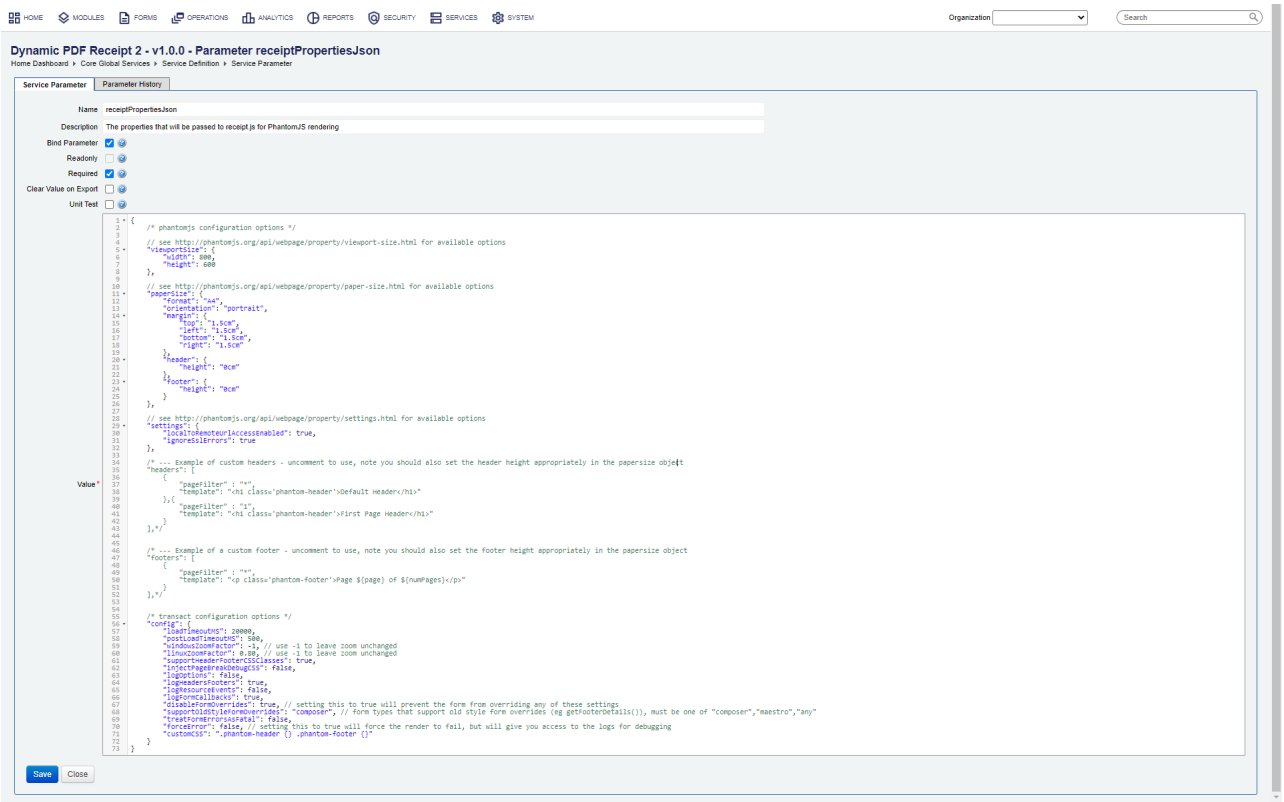


5. Edit Fallback Timeout, which is the fallback process timeout in milliseconds. The default is 3000.
6. Select Max Processes, which is the maximum number of concurrent PhantomJS processes. The default is 2.

> NOTE
>
> Changing Max Processes will not take effect until Manager is restarted.

7. Edit Phantom JS Exe, which is the PhantomJS[1] executable file name (excluding the Windows .exe file extension). The default is phantomjs.
8. Click Receipt Properties Json to edit the properties that will be passed to the receipt.js for customized headers and footers rendering.

[1]PhantomJS is a scripted, headless browser used for automating web page interaction. PhantomJS provides a JavaScript API enabling automated navigation, screenshots, user behavior and assertions making it a common tool used to run browser-based unit tests in a headless system like a continuous integration environment. PhantomJS is based on WebKit making it a similar browsing environment to Safari and Google Chrome (before Chrome's fork of WebKit evolved into Blink). It is open-source software released under the BSD License.



9. Click Save to update the changes.

Next, learn how to [view core global services](#).

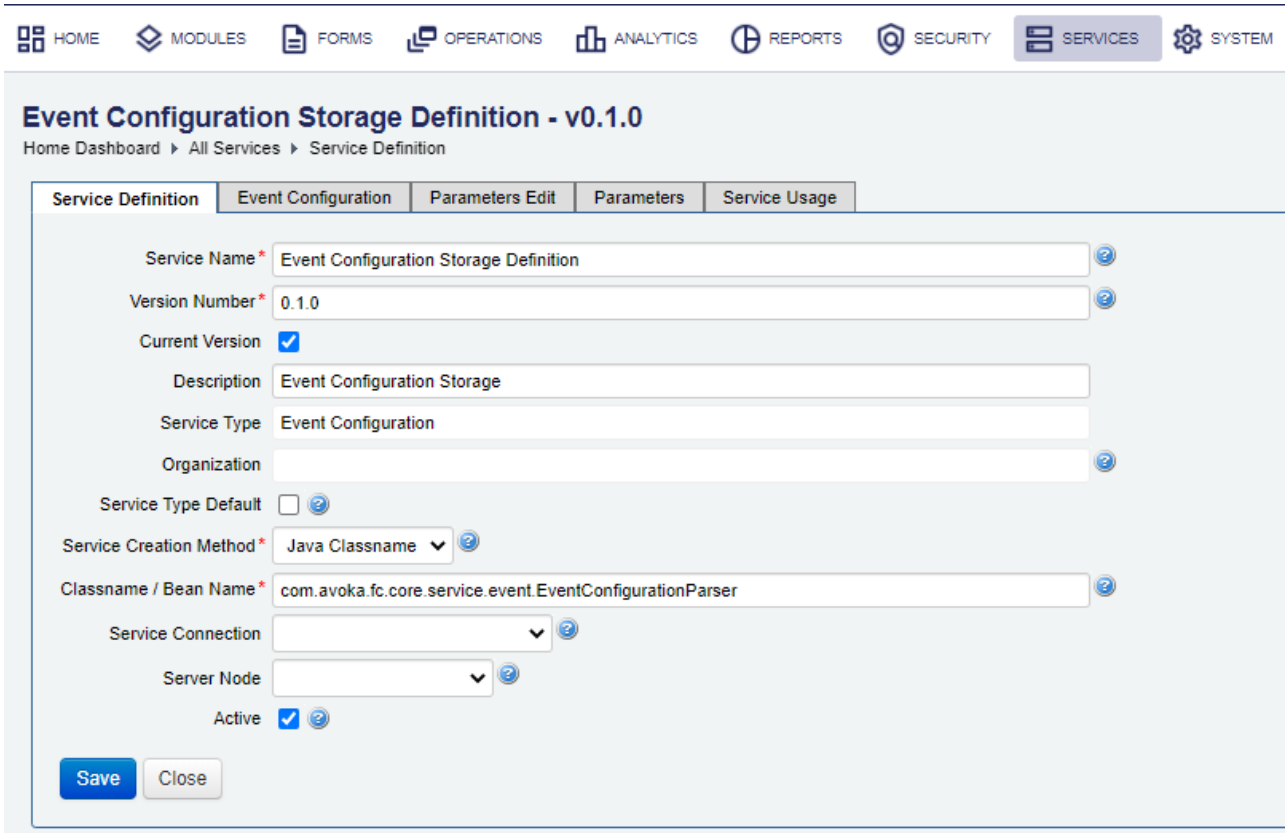
Event Configuration Storage Service

Manager comes with the Event Configuration Storage service, which allows you to publish various types of events to Kafka topics. This service is an integral part of the Event-Based Architecture.

> NOTE
>
> You may need to create and configure a Kafka service connection to use this service.

To configure the Event Configuration Storage service:

1. Select Services > Core Global Services.
2. Locate the Event Configuration Storage service and click Edit.
3. Configure the standard service settings using the Event Configuration service type as the Service Type and the Event Configuration Storage Definition template as the Service Template. Select a relevant service connection to use to connect to the external service.



4. Select the Event Configuration tab to specify filter criteria that are used to decide which Manager's entity to published to which Kafka topic.

5. Copy an paste an existing JSON configuration file or edit it using JSON syntax in the Event Configuration field. You must specify at least the following properties:
   - Topic name: a name of a topic configured in Kafka, for example, `KAFKA_TOPIC`.
   - Entity Name: a name of the Manager's entity that is published to the Kafka topic identified with the topic name, for example, `Submission`.
   - Entity Conditions: query to list entities to be published to the Kafka topic.

   For more information, see event definition.

6. Select the Auto Format checkbox to apply JSON file formatting, such as indentation.
7. Click Journey Manager Services Guide and Journey Manager Fluent API Javadoc links to access the relevant documentation.
8. Click Save to update the changes.

> NOTE
>
> You must enable the Eventing Feature Enabled deployment property to activate the eventing framework, so all triggered events are stored in the Event Outbox.

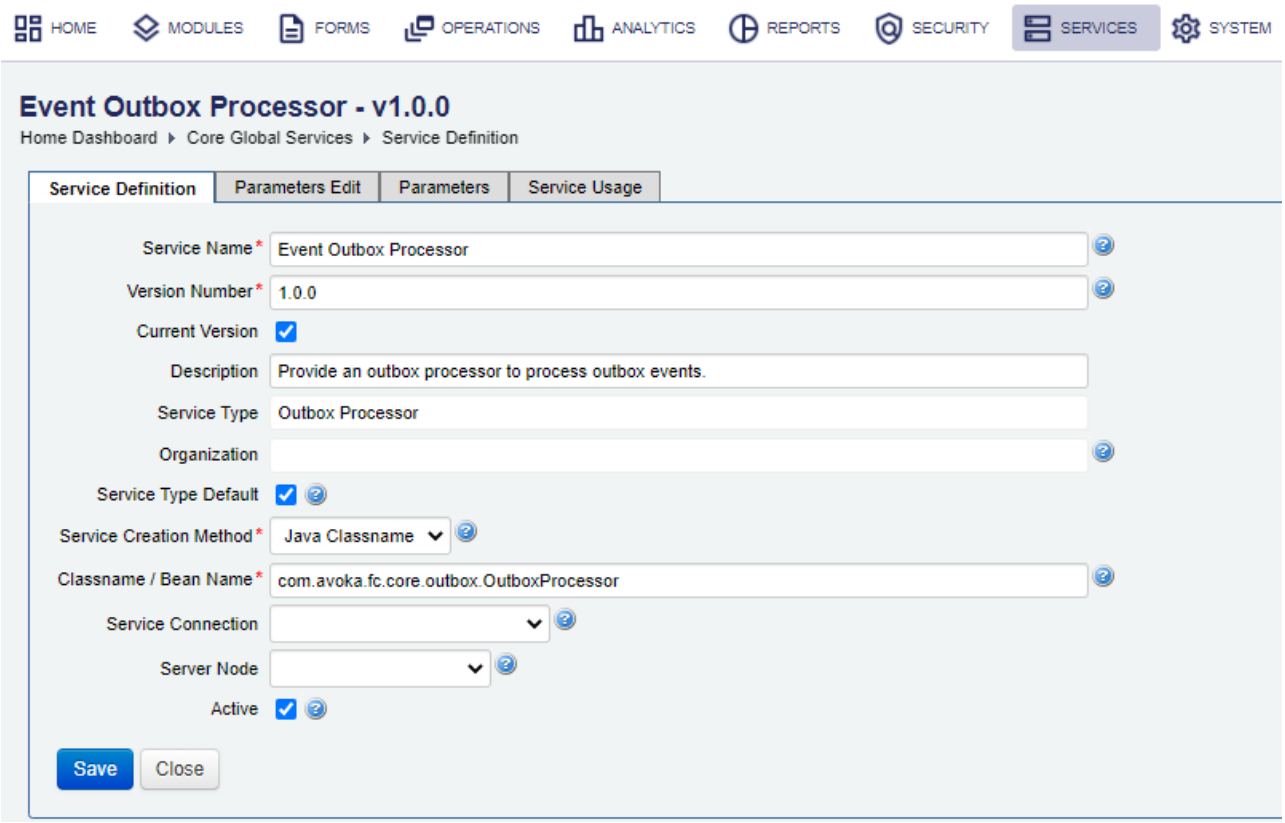
Next, learn how to view core global services.

Event Outbox Processor Service

Manager provides the Event Outbox Processor service to publish events into a Kafka topic. Normally, this service is called by the Event Outbox Processor job.

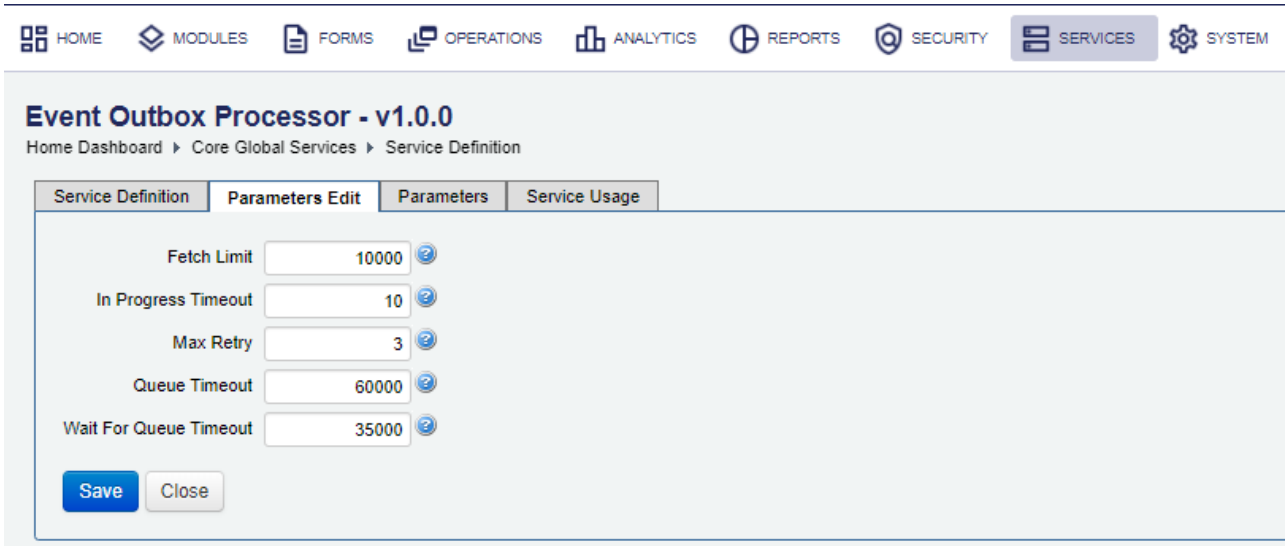
To configure the Event Outbox Processor service:

1. Select Services > Core Global Services.
2. Select the Outbox Processor from the Type dropdown list to shown only services of this type.
3. Locate the Event Outbox Processor service and click Edit.
4. Click the Service Definition tab to configure the service.



5. Click Save to update the changes.

6. Click the Parameters Edit tab to configure the service parameters.



7. Edit a maximum number of events to be processed in the Fetch Limit field. The default is 10000.
8. Specify how long to wait for an event to be processed (in milliseconds) in the In Progress Timeout field.

   NOTE

   The event's status is set to In Progress when Manager starts sending it to Kafka

9. Specify a maximum number of retries to process an event if it failed in the Max Retry field.
10. Specify how long to wait in the event queue (in milliseconds) in the Queue Timeout field.
11. Specify how long to wait to add an event to the event queue (in milliseconds) in the Wait For Queue Timeout field.
12. Click Save to update the changes.

The service adds an entry to the event log every time it runs. The entry can be one of the following types:

- Send Event/Delivered
- Send Event/Failed

The log entry example is shown below:

```
Event sent from Journey Manager, Event ID:10, Service Connection:KafkaConnection, Event
Topic:topic_1, Event Details:
{"entity":"Submission"}
```

Next, learn how to view all core global services.

Form Submission Access Controller Service

Manager has a variety of built-in security access rules. One of them is to allow a user to access form's submissions only if a user's session matches a session used to render that form. If the sessions don't match, the user is expected to re-enter the save challenge details.

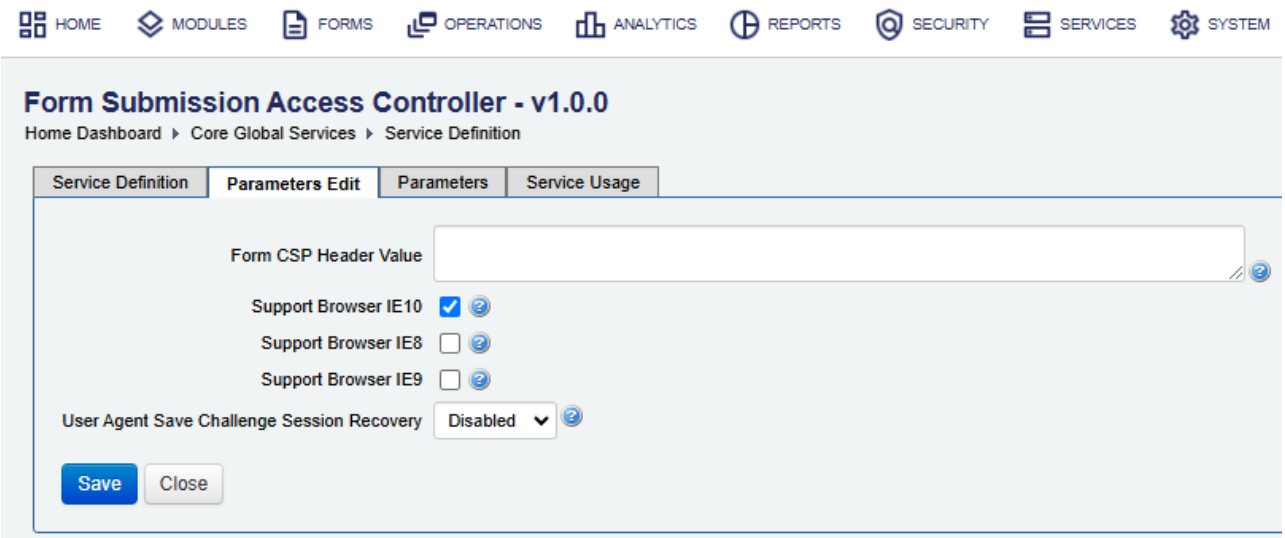
> NOTE
>
> Composer forms prior to version 4.3 SP3 don't have this functionality within the form flow.

Manager comes with the form submission access controller service, which you can use to change (override) the existing users access rules to form's submissions.

> NOTE
>
> If a session is re-established, access to the submissions will only be valid within the Anonymous Submission Access Timeout time configured for this form space.

To configure the form submission access controller service:

1. Select Services > Core Global Services.
2. Locate the service and click Edit.
3. Click the Service Definition tab to edit the standard service configuration.
4. Click the Parameters Edit tab to view or update the service configuration.



5. Specify the system wide Content Security Policy[1] (CSP) security HTTP header value in the Form CSP Header Value field.

   NOTE

   Check the CORS Headers setting of your form space and the System Default CSP Header setting of your organization to ensure the correct CSP is used.

6. Select the Support Browser IE10 checkbox to support the legacy Internet Explorer 10 (IE10) browser.
7. Select the Support Browser IE8 checkbox to support the legacy Internet Explorer 8 (IE8) browser.
8. Select the Support Browser IE9 checkbox to support the legacy Internet Explorer 9 (IE9) browser.

   NOTE

   IE8, 9 and 10 don't support TLS 1.1-1.2 or Content Security Protection (CSP).

9. Select a save challenge session recovery mode from the User Agent Save Challenge Session Recovery dropdown list to override the existing behavior. You can use one of the following options:
   - Disabled: sessions aren't re-establish, so users must re-enter the save challenge details to recover sessions. This is the default option.
   - Enabled: user agent[2] check is enabled to allow sessions to be re-established.

---

[1]Content Security Policy (CSP) is a computer security standard introduced to prevent cross-site scripting (XSS), clickjacking and other code injection attacks resulting from execution of malicious content in the trusted web page context. CSP provides a standard method for website owners to declare approved origins of content that browsers should be allowed to load on that website—covered types are JavaScript, CSS, HTML frames, web workers, fonts, images, embeddable objects such as Java applets, ActiveX, audio and video files, and other HTML5 features. For more information, see https://www.w3.org/TR/CSP/

[2]User agent is software (a software agent) that is acting on behalf of a user. One common use of the term refers to a web browser that retrieves, renders and facilitates end user interaction with Web content.

- 4.3.0SP2: user agent[1] check is enabled to allow sessions to be re-established for forms built using Composer version 4.3 SP2 and higher.

10. Click Save to update the changes.

Next, learn how to view core global services.

[1]User agent is software (a software agent) that is acting on behalf of a user. One common use of the term refers to a web browser that retrieves, renders and facilitates end user interaction with Web content.

Jakarta Transformer Service

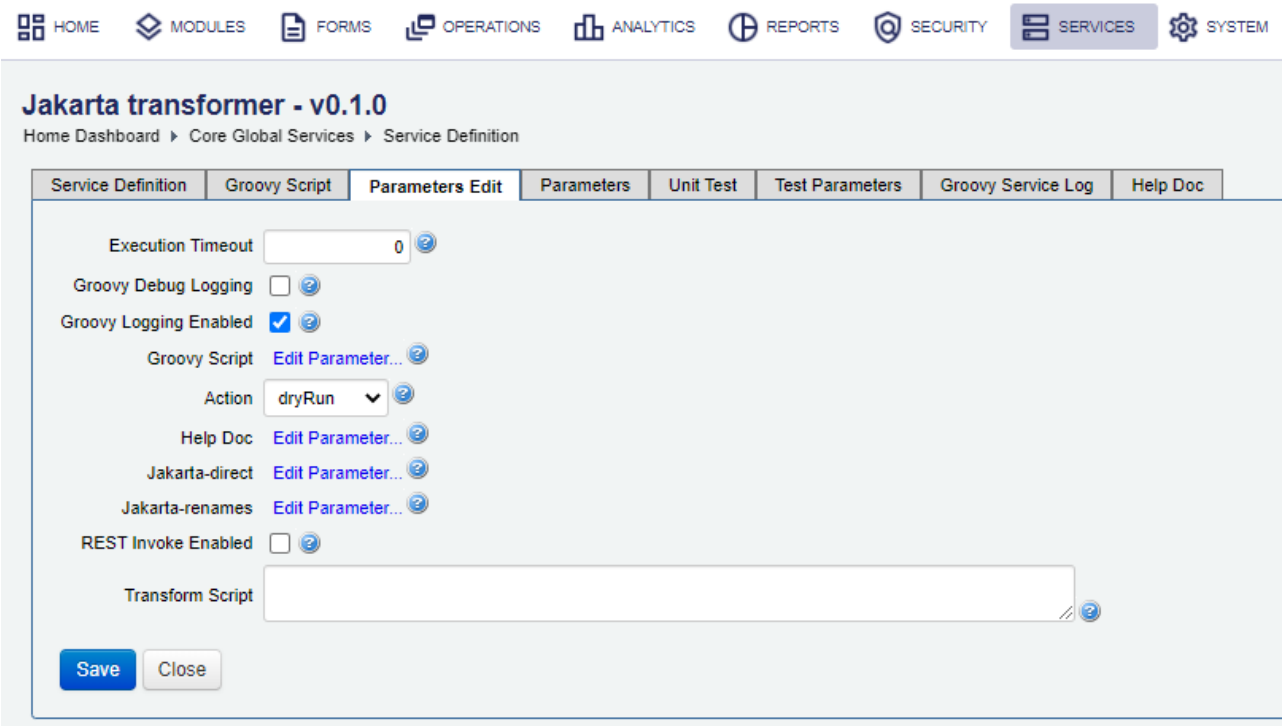
Manager 23.04 comes with a Jakarta Transformer service, which allows you to upgrade your existing Groovy services and Security Managers to be compatible with Jakarta EE. You can run this service from the Unit Test tab to verify it or from Groovy Console to update your system.

> IMPORTANT
>
> This service exists in Manager 23.04 only. For Manager 23.10 and later, use the Jakarta Transformer scheduled service.
> If you have upgraded Manager 23.04 to 23.10 and later, the Jakarta Transformer service must be deactivated.

> NOTE
>
> The Jakarta Transformer service is a global service.

To configure the Jakarta Transformer service:

1. Select Services > Core Global Services.
2. Locate the Jakarta Transformer service and click Edit.
3. Configure the standard service settings.
4. Configure the service parameters in the Parameters Edit tab.



5. Select an action from the Action dropdown list, which can be one of the following:
   - `list` - lists all services that contain a Groovy script or a unit test script that requires change. It also lists Security Managers and Authentication Providers that contain Groovy scripts and require changes.
   - `dryRun`- in addition to `list` functionality, this option displays the required changes for the Groovy script by line number and by type of scripts. This is the default option. For services, it shows if the change is done in the Groovy script or Unit test script. For security managers, it shows if the change is done in SSO Get Auth Token, SSO Auth Ok Response or SSO Revalidation scripts. Additionally, if the security manager has an authentication provider with a Groovy script that requires change, it also mentions this.
   - `transform` - in addition to `dryRun` functionality, this option applies and commits the changes in the database.

     > WARNING
     >
     > This action is irreversible.

6. Click Jakarta-direct to view and edit the `jakarta-direct.properties` configuration file.
7. Click Jakarta-renames to view and edit the `jakarta-renames.properties` configuration file.
8. Select the REST Invoke Enabled checkbox to enable this service to be called via REST Groovy Service Invoke.
9. Provide a script in the Transform Script field to test a transformation script.

   > NOTE
   >
   > This field is for testing purposes only. Otherwise, keep it empty to ensure the service runs as expected.

10. Click Save to update the changes.

Next, learn how to view core global services.

Jakarta Transformer Scheduled Service

Manager comes with a Jakarta Transformer scheduled service, which must be executed as a scheduled job several times so it upgrades your existing Groovy services and Security Managers to be compatible with Jakarta EE. This service is designed to process a large number of services with extensive Groovy scripts. It also supports multiple versions of Exchange components by using a new deployment property - Enable transforming javax to jakarta.

Every time the service is run, it scans and updates services and then marks the transformed services by selecting its Jakarta-transformer-checked checkbox to mark it so it won't scan and transform it again during the next run.

To make sure the service transforms all service and security managers scripts in large Journey Manager environments, you must create a custom scheduled job to run the Jakarta Transformer scheduled service. For more information, see Migrate to Jakarta EE.

IMPORTANT

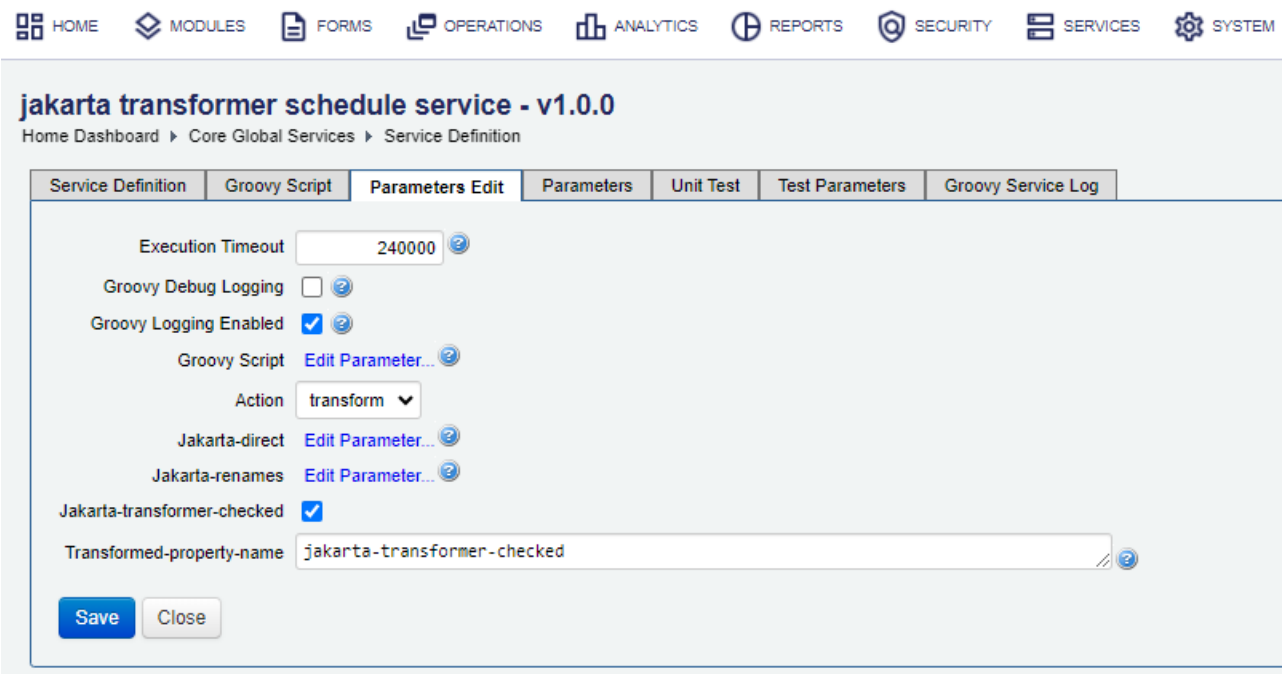
For Manager 23.04 only, use the Jakarta Transformer service.

NOTE

The Jakarta Transformer scheduled service is a global service.

To configure the Jakarta Transformer scheduled service:

1. Select Services > Core Global Services.
2. Locate the Jakarta Transformer scheduled service and click Edit.
3. Configure the standard service settings.
4. Configure the service parameters in the Parameters Edit tab. For explanation of the parameters, see Jakarta Transformer service.



5. Update a property name, which is added to services checked for Javax to Jakarta transformation, in the Transformed-property-name field. The default is jakarta-transformer-checked. For more information, see Configure Transact Function Parameters Edit.
6. Click Save to update the changes.

Next, learn how to view core global services.

Payment Reminder Service

Manager provides a service to send out reminder emails about submissions with outstanding payments older than a certain age.

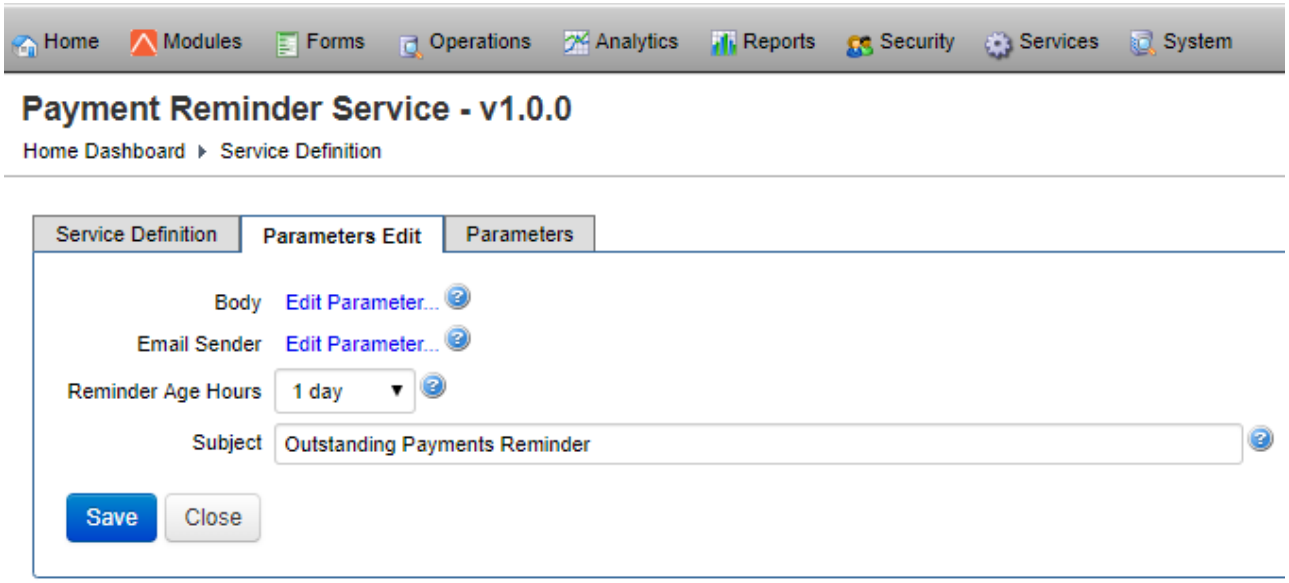
To configure a payment reminder service:

1. Locate an existing service or create a new one using the Service Type as Payment Reminder.

   NOTE

   You can also copy the existing payment reminder service to create a new one.

2. Click the Parameters Edit tab to view or change the service configuration.



3. Click Body to edit the email body template.
4. Click Email Sender to edit the email sender address for the payment alert emails.
5. Select the reminder age in hours for outstanding payment after which the alert email will be sent from the Reminder Age Hours dropdown list. Default is 1 day.
6. Edit the email subject template in the Subject field.
7. Click Save to update the changes.

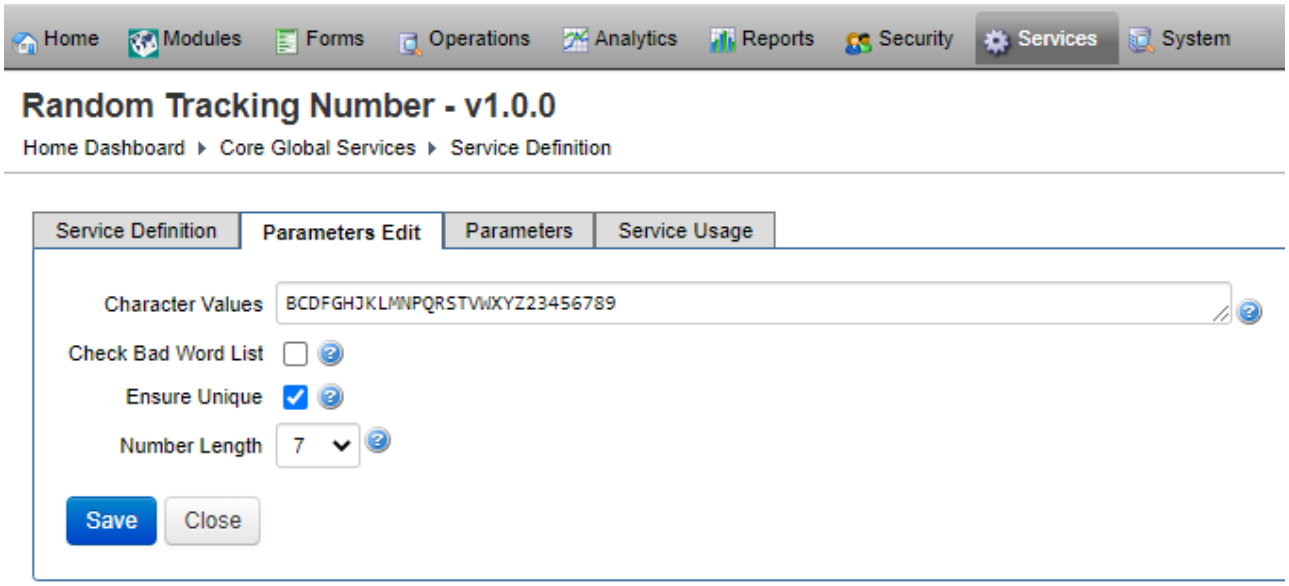
Next, learn how to view all services.

Random Tracking Number Service

Manager comes with the Random Tracking Number service, which is the core global service, that generates a unique tracking code for each transactions. For more information on the uniqueness of the generated codes and how to configure them, see configure receipt numbers and tracking codes.

To configure the Random Tracking Number service:

1. Select Services > Core Global Services.
2. Locate the service and click Edit.
3. Configure the standard service settings.
4. Click the Parameters Edit tab to view and edit the configuration.



5. Edit Character Values, which is the list of character values to generate the random tracking number from.
6. Select the Check Bad Word List to validate a random tracking number against badword list. This is not required as available letters exclude vowels, and does incur some performance cost.
7. Select Ensure Unique, which Ensures the number is unique by checking against tracking number table.
8. Edit Number Length, which is the length of the tracking number.
9. Click Save to update the changes.

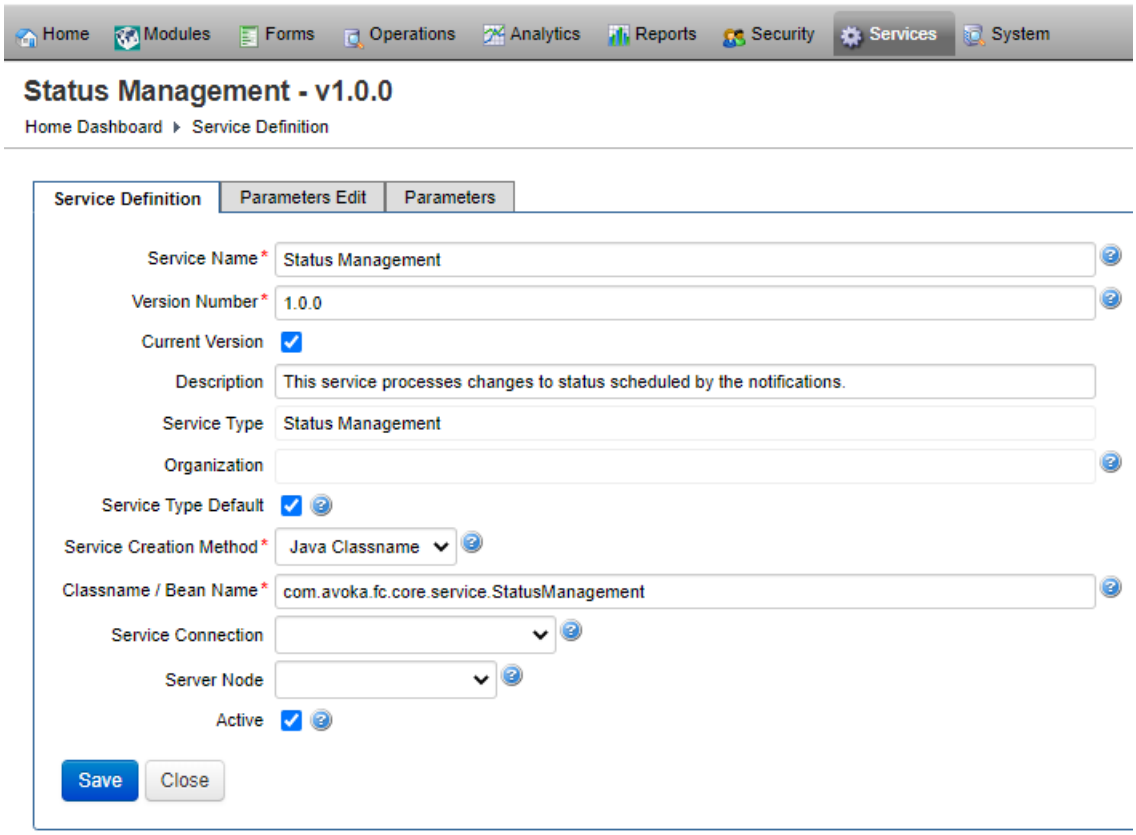
Next, learn how to view core global services.

Status Management Service

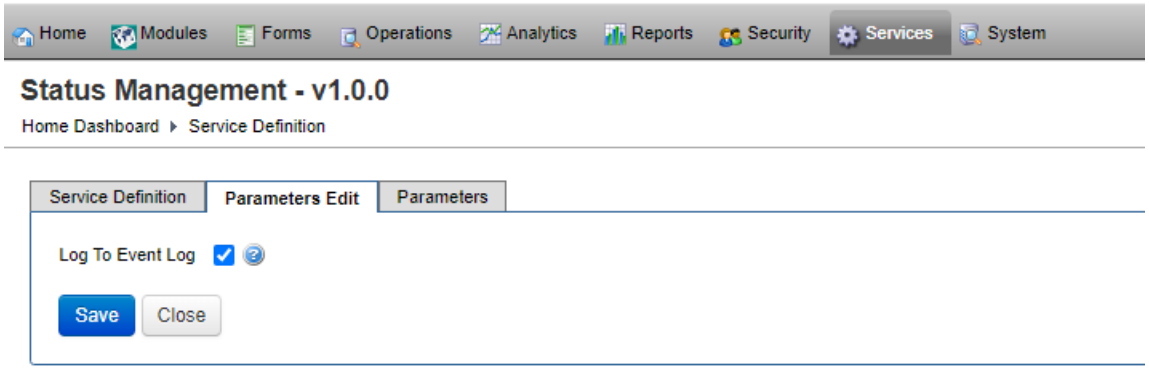
Manager provides the Status Management service to change statuses of form spaces, forms, and form versions, which have been scheduled using Notifications and Status Change. This service is a scheduled job that runs every 5 mins and processes status change events.

To configure the status management service:

1. Select Services > Core Global Services.
2. Locate the Status Management service and click Edit.
3. Click the Service Definition tab to configure the service.



4. Click Save to update the changes.
5. Click the Parameters Edit tab to configure the service parameters.



6. Select the Log To Event Log checkbox to add an entry to the event log every time the service runs.
7. Click Save to update the changes.

The service add an entry to the event log every time the service runs. The entry includes:

- Number of entities
- Names of specific entities changed

The log entry example is shown below:

```
Status Management completed in 0 sec. Duration 3 ms.
  Number of entities updated: 1 Detailed re
```

Next, learn how to view all core global services.

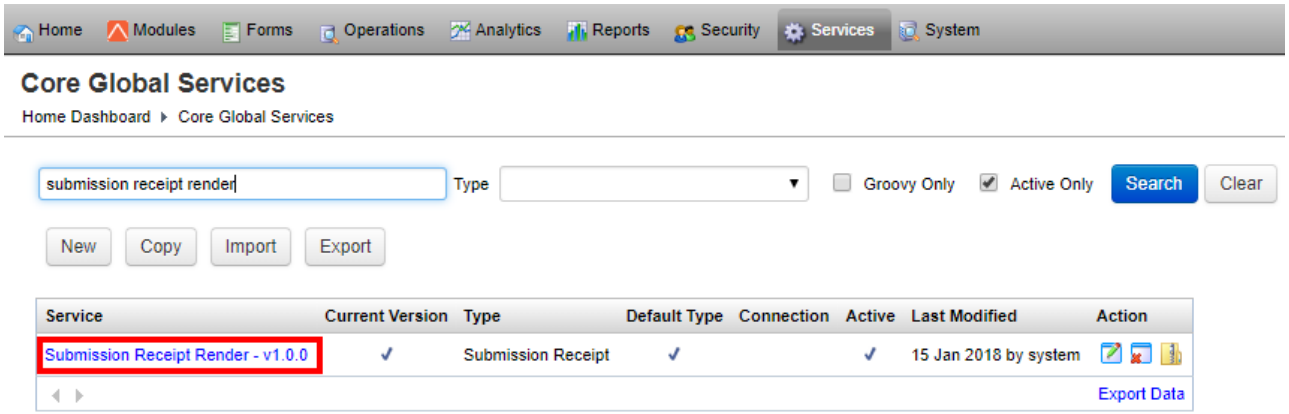
Submission Receipt Render Service

Manager provides the Submission Receipt Render core global service, which creates and stores receipts for new submissions, re-attempts to render receipts for submissions where previous attempts failed, and generates PDF receipts in the background.

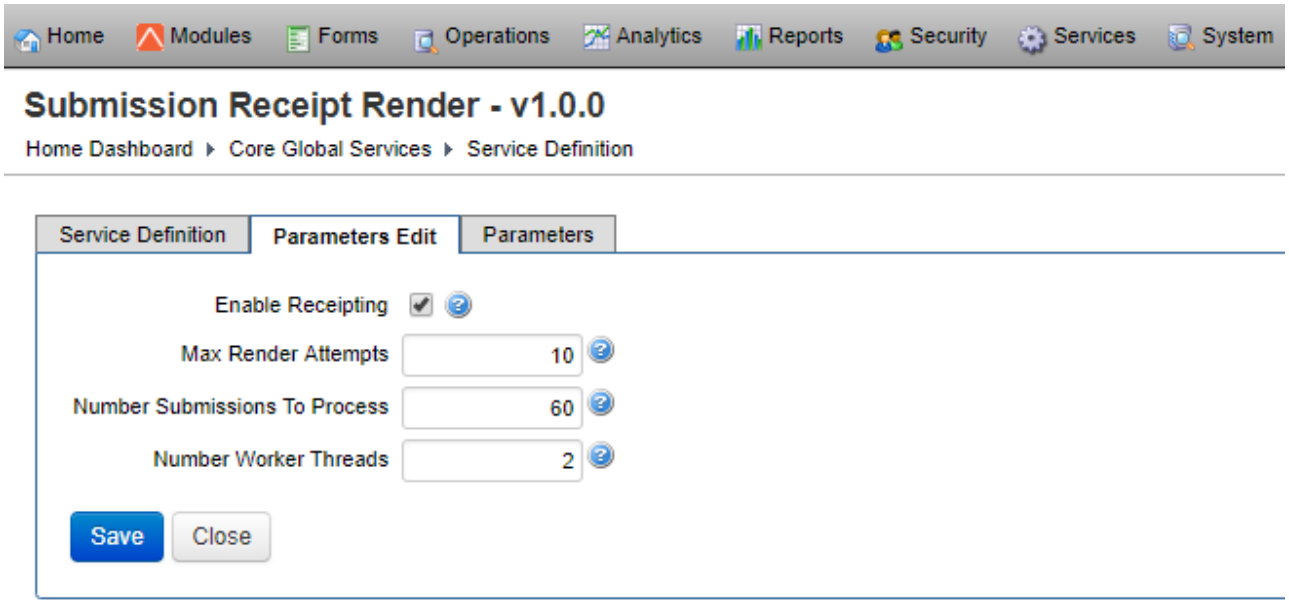
This service is called periodically by the Transaction Processor service.

To view the Submission Receipt Render service configuration:

1. Select Services > Core Global Services.
2. Locate the service and click Edit.



3. Configure the standard service settings.
4. Click the Parameters Edit tab to view and edit the configuration.



5. Select Enable Receipting to enable background PDF receipt generation. It is selected by default.
6. Edit Max Render Attempts, which is the maximum number of times a PDF receipt will attempt to render before failing and being placed into an error state. The default is 10.
7. Edit Number Submissions To Process, which is the maximum PDF receipt generation queue length.This is a balance, if the queue length is too long then during quiet times a single node may take all the receipts waiting for receipts and effectively serialise the process affecting responsiveness. The default is 60.
8. Edit Number Worker Threads, which is the number of concurrent PDF rendering processes to run on a single node. This number must not exceed the Max Processes parameter setting on the Dynamic PDF Receipt or Dynamic PDF Receipt 2 services.
9. Click Save to update the changes.

NOTE

You can use this service to improve Manager load balancing.

Next, learn how to view core global services.

Transaction Processor Service

Manager comes with the Transaction Processor service that plays an important role in the transaction life cycle by performing the following steps:

- Abandonment
- Receipt generation
- Delivery
- Task expiry
- Transaction history creation
- Data purging

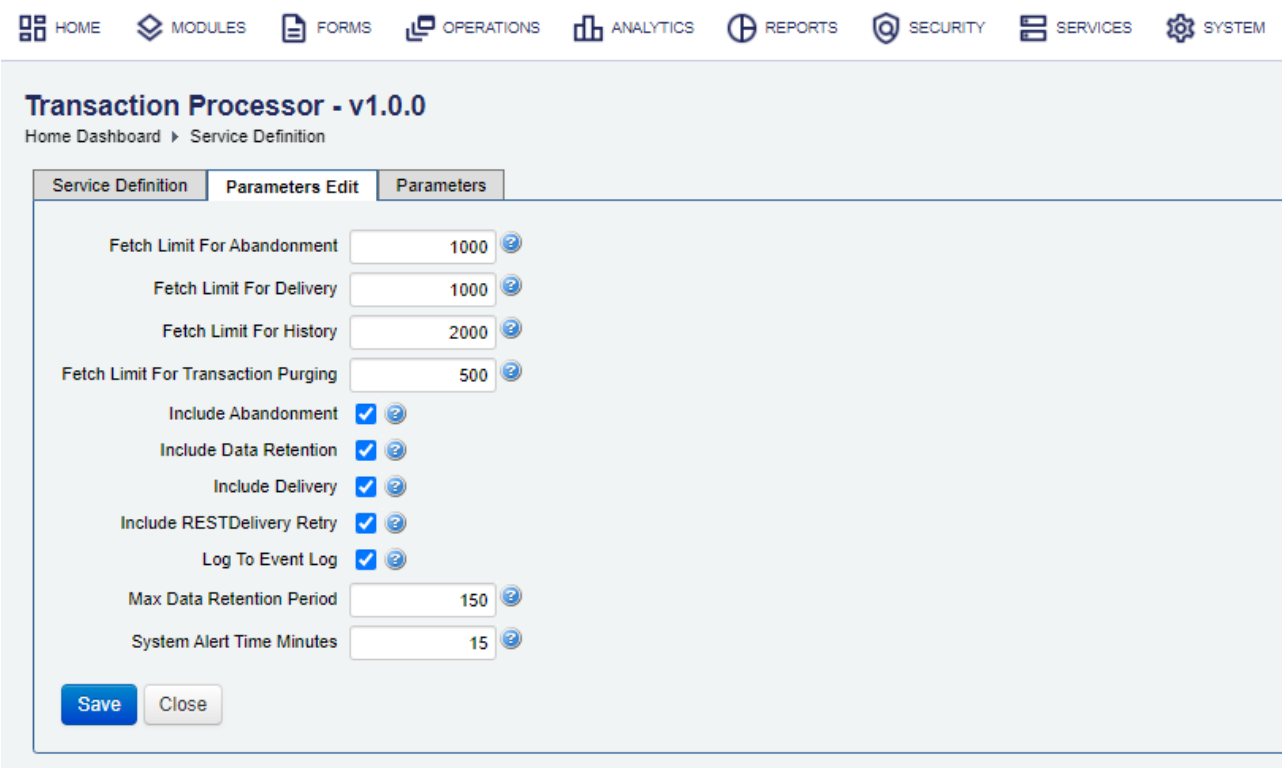
The Transaction Processor service is called by the Transaction Processing scheduled job.

> NOTE
>
> We recommend seeking advice from the Client Services team before changing this service configuration.

To configure the Transaction Processor service:

1. Select Services > Core Global Services.
2. Select the Transaction Processor from the Type dropdown list to show all services of this type.
3. Locate the service and click Edit.
4. Check the standard service settings.

5. Select the Parameters Edit tab to view and edit the configuration.



6. Edit the maximum number of records to consider for each abandonment type in the Fetch Limit For Abandonment field. The default is 1000.
7. Edit the the maximum number of records to consider for each delivery type (normal vs abandoned submission delivery) in the Fetch Limit For Delivery field. The default is 1000.
8. Edit the maximum number of records to process for transaction history creation in the Fetch Limit For History field. The default is 2000.
9. Edit the maximum number of items to be purged per each category: submissions, collaboration jobs, transaction history, error logs, and so on in the Fetch Limit For Transaction Purging field, which is. The default is 500.

   It is important to understand that, for each category, the full hierarchy is purged. This means that:

   - For a submission - all related error logs are purged.
   - For a collaboration job - all related submission are purged.

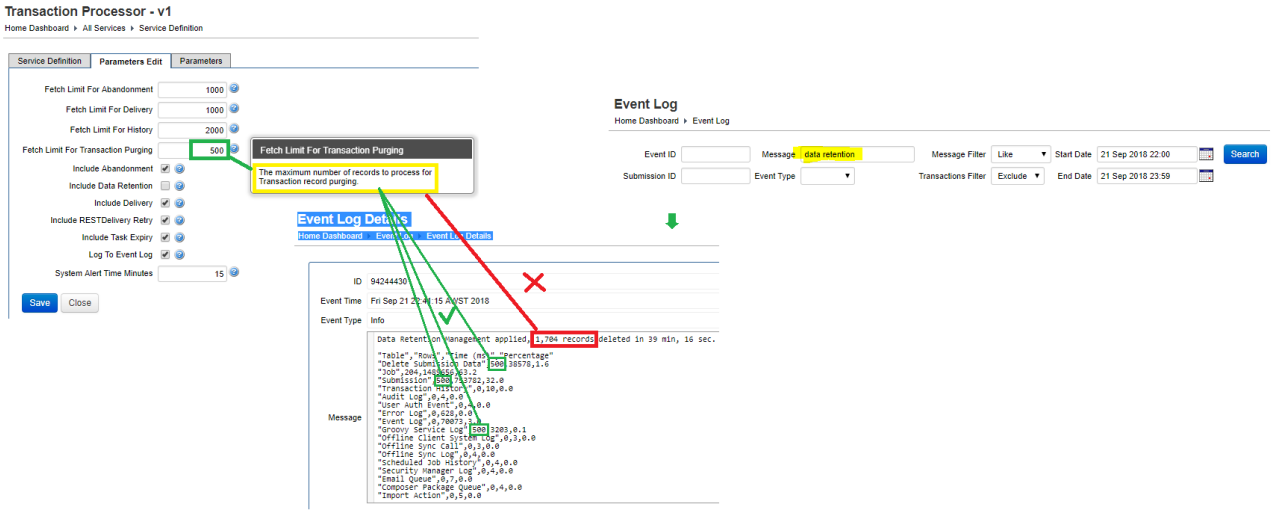
   The fetch limit only applies to the top level of the category. So in the example of submissions, 500 independent submission may be purged, but if collaboration jobs are purged, then the actual number of submission will be higher.

Open the event log and search for events using data retention key word to see the actual number of purged transactions of each category, as shown below.

ID	94244430
Event Time	Fri Sep 21 22:41:15 AWST 2018
Event Type	Info
Message	

```
Data Retention Management applied, 1,704 records deleted in 39 min, 16 sec.

"Table","Rows","Time (ms)","Percentage"
"Delete Submission Data",500,38578,1.6
"Job",204,1489656,63.2
"Submission",500,753782,32.0
"Transaction History",0,10,0.0
"Audit Log",0,4,0.0
"User Auth Event",0,4,0.0
"Error Log",0,628,0.0
"Event Log",0,70073,3.0
"Groovy Service Log",500,3203,0.1
"Offline Client System Log",0,3,0.0
"Offline Sync Call",0,3,0.0
"Offline Sync Log",0,4,0.0
"Scheduled Job History",0,4,0.0
"Security Manager Log",0,4,0.0
"Email Queue",0,7,0.0
"Composer Package Queue",0,4,0.0
"Import Action",0,5,0.0
```

This is also illustrated below.



10. Select the Include Abandonment checkbox to turn on the abandonment process. It is selected by default.

> NOTE
>
> If you clear this checkbox, transactions that should be abandoned will not be processed.

11. Select the Include Data Retention checkbox to turn on the data retention process, which executes the Data Retention Management service. It is selected by default.

    NOTE

    We don't recommended turning off data retention as no data will be purged. However, you can do it for a short period of time to troubleshoot transaction processing issues.

12. Select the Include Delivery checkbox to turn on the delivery process. It is selected by default.
13. Select the Include RESTDelivery Retry checkbox to turn on delivery. It is selected by default.
14. Select the Include Task Expiry checkbox to turn on task expiry. It is selected by default.

    NOTE

    If you clear this checkbox, transactions that should be expired will not be processed.

15. Select the Log To Event Log checkbox to log the results of the transaction processing to the event log.It adds two entries to the event log - one for the Transaction Processor and one for data retention, which includes timing information relating to the performance of the operations. It is not selected by default.

    NOTE

    Disable logging unless you need to troubleshoot transaction processing issues.

16. Edit the maximum period of time, in seconds, allocated for data retention purging in the Max Data Retention Period field. The default is 150 seconds. This allows the Transaction Processor to limit the amount of time it spends on data retention processing. That is, when purging takes longer than expected, the Transaction Processor will stop, move onto other tasks, and continue at a later time. This ensures that purging data does not affect other parts of the Manager system. | 18.11 This feature was introduced in 18.11.
17. Edit the number of minutes that the system monitoring job will use when determining whether system alerts need to be raised about the transaction processing service in the System Alert Time Minutes field. The default is 15 minutes.
18. Click Save to update the changes.

Next, learn how to view core global services.

Attachment Virus Scanning Service

Manager comes with several preconfigured antivirus services to scan any files that form users may attach to forms. In most of onboarding journeys, forms are often designed to include various supporting documentation, such as proof of ID, which are file attachments.

Attachment virus scan services ensure no malicious or infected files enter the system as attachments. For this to work, there must be at least one antivirus attachment scanning service configured and set as a default Virus Scan type service for each Manager instance. This is usually done after Manager software has been installed.

Manager runs a default antivirus service to scan files at upload time for all form-related flows with attachments. These flows include:

- Attachment widgets
- Form space attachment pages
- Fluent API
- Open UX commands

> NOTE
>
> Attachment data stored in the Manager database directly with custom core Groovy services using Core API is not scanned.

Manager delivers a form submission only if all attachments are scanned and marked as free of viruses or malicious software. If infected files are detected, they are either blocked from upload, or, if picked up by the Virus Scan scheduled job, deleted. In this case, a form user, who uploaded the attachment, is notified via email.

When a file is scanned, one of the following states is set as the Virus Status property, which you can check for each transaction with attachments:

- Clean - an attachment is free of viruses or malicious software
- Virus - an attachment is infected
- Scan Error - an error happened during the scan
- No Data - an attachment is missing actual data and doesn't need to be scanned anymore

Manager has the standard virus scan scheduled job, which queries the database for all file uploads with the Virus Status property set as Scan Error or Null and deletes the infected files. Null is different from No Data and it means there is no attachment specified.

Submissions that are pending virus scan results for their attachments remain in the Delivery Not Ready status until all virus scans are completed. After that, they can transition through delivery.

Manager also scans all files, such as configuration files, import archives or reference data, that are imported or uploaded via its UI, for viruses. For example, Manager will kick off a virus scan during a service import operation. This functionality is enabled by default, but you can turn it off using the Enable Virus Scan for JM Uploads deployment property. | 23.04 This feature was introduced in 23.04

Manager provides the following preconfigured virus scanning services, so you can choose one depending on your Virus Scanning Engine[1]:

- ClamAV
- Symantec
- Fluent

NOTE

You must configure at least one virus scanning service in Manager so it can perform regular virus scanning activities.

Next, learn how to configure the ClamAV attachment virus scanning service.

[1]Virus Scanning Engine is Antivirus software (AV software), also known as anti-malware, which is a computer program used to prevent, detect, and remove malware.

ClamAV Attachment Virus Scanning Service

Manager allows you to configure the ClamAV Virus Scanning service to check files for viruses using ClamAV antivirus software. You can also chose another antivirus software to use depending on your requirements.

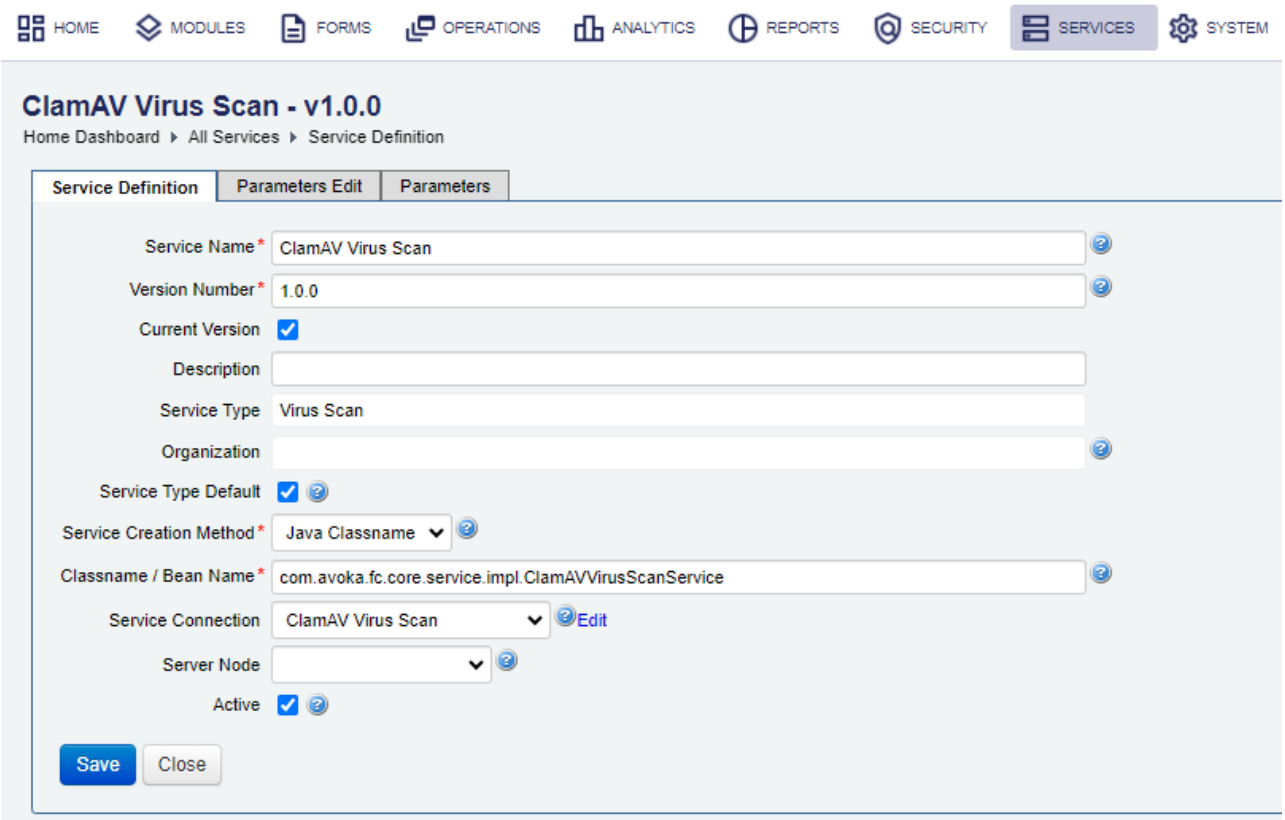
> NOTE
>
> Depending on your Journey Manager provisioning, you have certain antivirus software available to you, so check Manager installation guide for details.

To configure the ClamAV Virus Scanning service:

1. Configure ClamAV[1] engine. For more information, see ClamAV Virus Scanning documentation.
2. Configure the ClamAV Virus Scan service connection.
3. Select Services > Core Global Services.
4. Locate the ClamAV Virus Scan service and click Edit.

[1]Clam AntiVirus (ClamAV) is a free, cross-platform and open-source antivirus software toolkit able to detect many types of malicious software, including viruses. One of its main uses is on mail servers as a server-side email virus scanner. The application was developed for Unix and has third party versions available for AIX, BSD, HP-UX, Linux, macOS, OpenVMS, OSF (Tru64) and Solaris.

5. Select the Service Definition tab.



6. Select the Service Type Default checkbox to activate this service, which will be used as the default service for the virus scan type.

> NOTE
>
> There can be only one default service for each service type and only Global Services can be marked as the default ones.

> NOTE
>
> To deactivate a virus scanning service, clear the Service Type Default checkbox

7. Click Save to update the changes.

Next, learn how to configure Symantec virus scanning service.

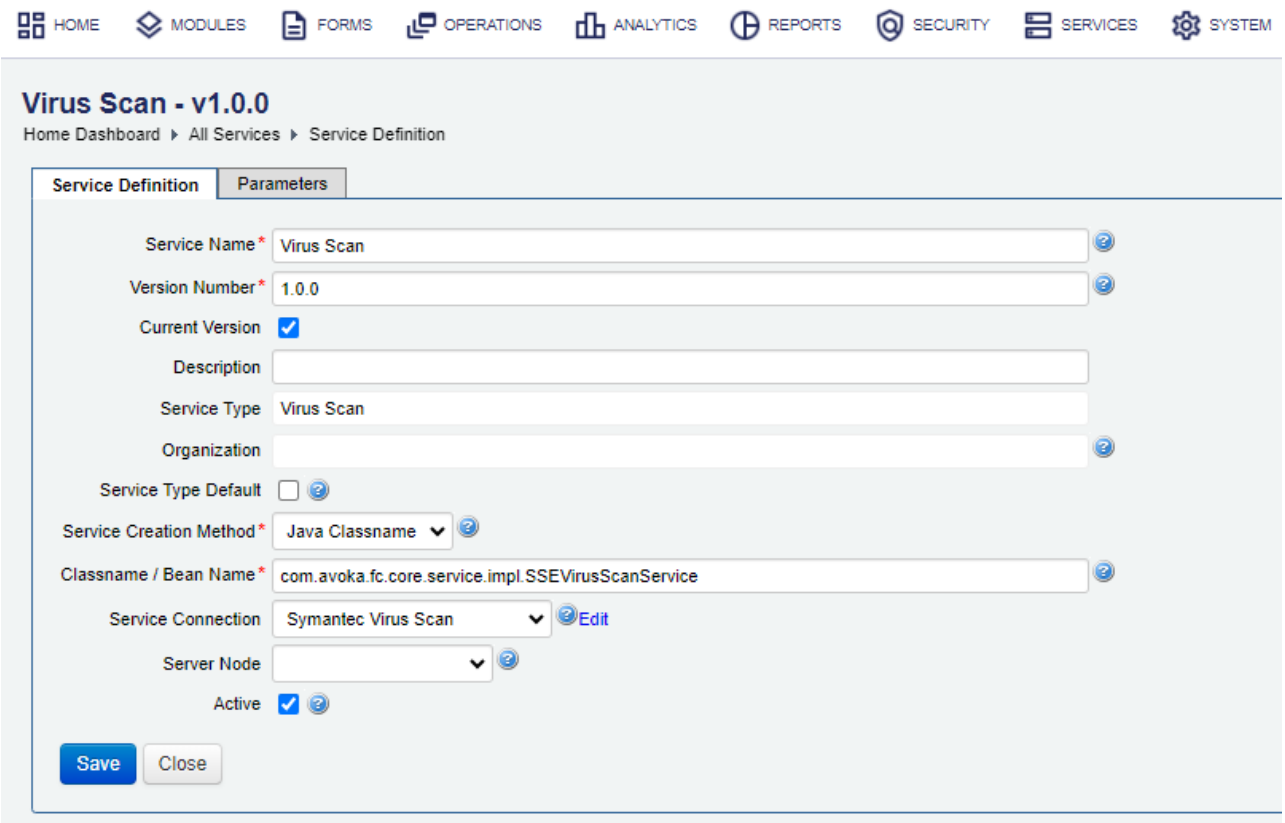
Symantec Attachment Virus Scanning Service

Manager allows you to configure the Symantec Virus Scanning service to check files for viruses using Symantec antivirus software. You can also chose another antivirus software to use depending on your requirements.

> NOTE
>
> Depending on your Journey Manager provisioning, you have certain antivirus software available to you, so check Manager installation guide for details.

To configure a service:

1. Configure the Symantec Protection Engine. For more information, see Symantec documentation.
2. Configure the Symantec Virus Scan service connection.
3. Select Services > Core Global Services.
4. Locate the Symantec Virus Scan service and click Edit.
5. Select the Service Definition tab.



6. Select the Service Type Default checkbox to activate this service, which will be used as the default service for the virus scan type.

   NOTE

   There can be only one default service for each service type and only Global Services can be marked as the default ones.

   NOTE

   To deactivate a virus scanning service, clear the Service Type Default checkbox

7. Select a Symantec Virus Scan from the Service Connection dropdown list.
8. Click Save to update the changes.

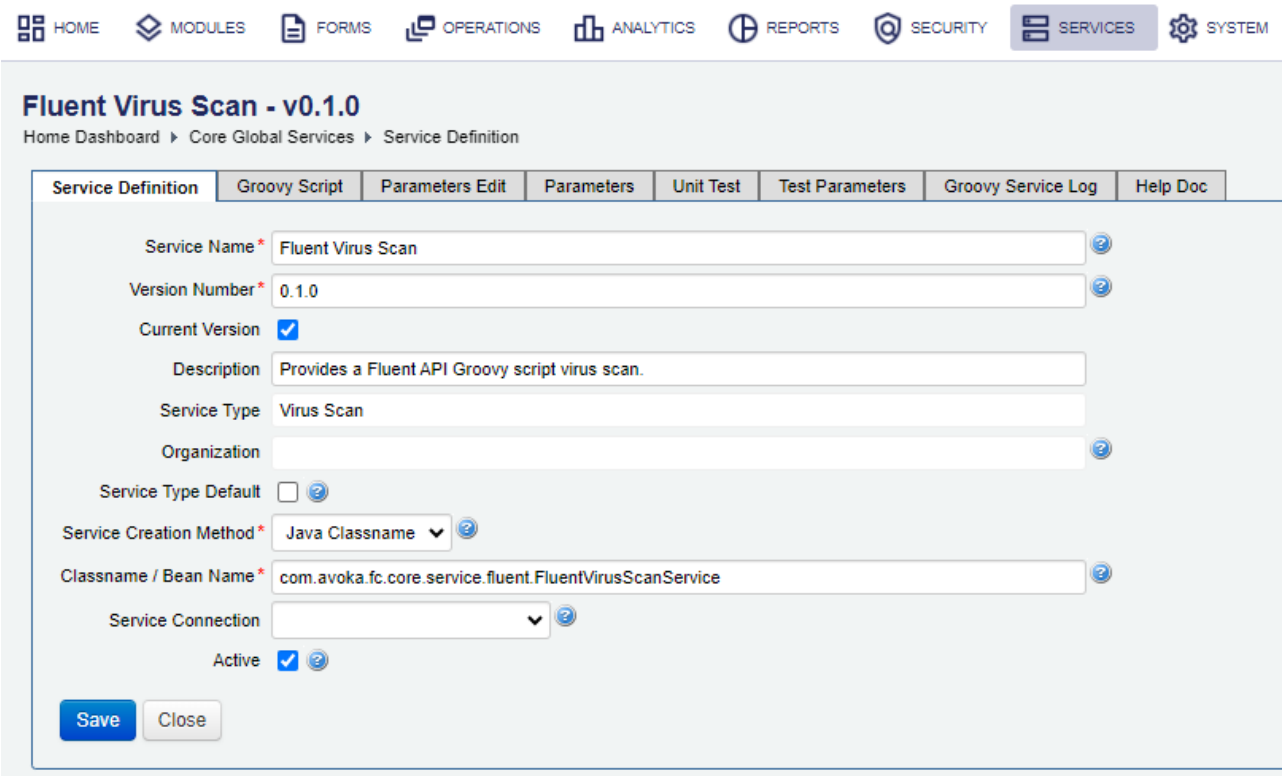
Next, learn how to configure ClamAV virus scanning service.

Fluent Attachment Virus Scanning Service

Manager allows you to create and configure a Fluent service, which checks files, such as, attachments, for viruses invoking antivirus software of your choice. You can also chose another antivirus software to use depending on your requirements.

To configure the Fluent service:

1. Configure a service connection, if required.
2. Select Services > Core Global Services.
3. Locate the Fluent virus scan service and click Edit.
4. Select the Service Definition tab.



5. Select the Service Type Default checkbox to activate this service, which will be used as the default service for the virus scan type.

> NOTE
>
> There can be only one default service for each service type and only Global Services can be marked as the default ones.

> NOTE
>
> To deactivate a virus scanning service, clear the Service Type Default checkbox

6. Select the Groovy Script tab to configure the Groovy code that performs a file scanning or invokes another service to do the job.

> NOTE
>
> Check the latest Fluent API to implement your changes.

7. Click Save to update the changes.

Next, learn how to configure ClamAV virus scanning service.

Dynamic Data Service

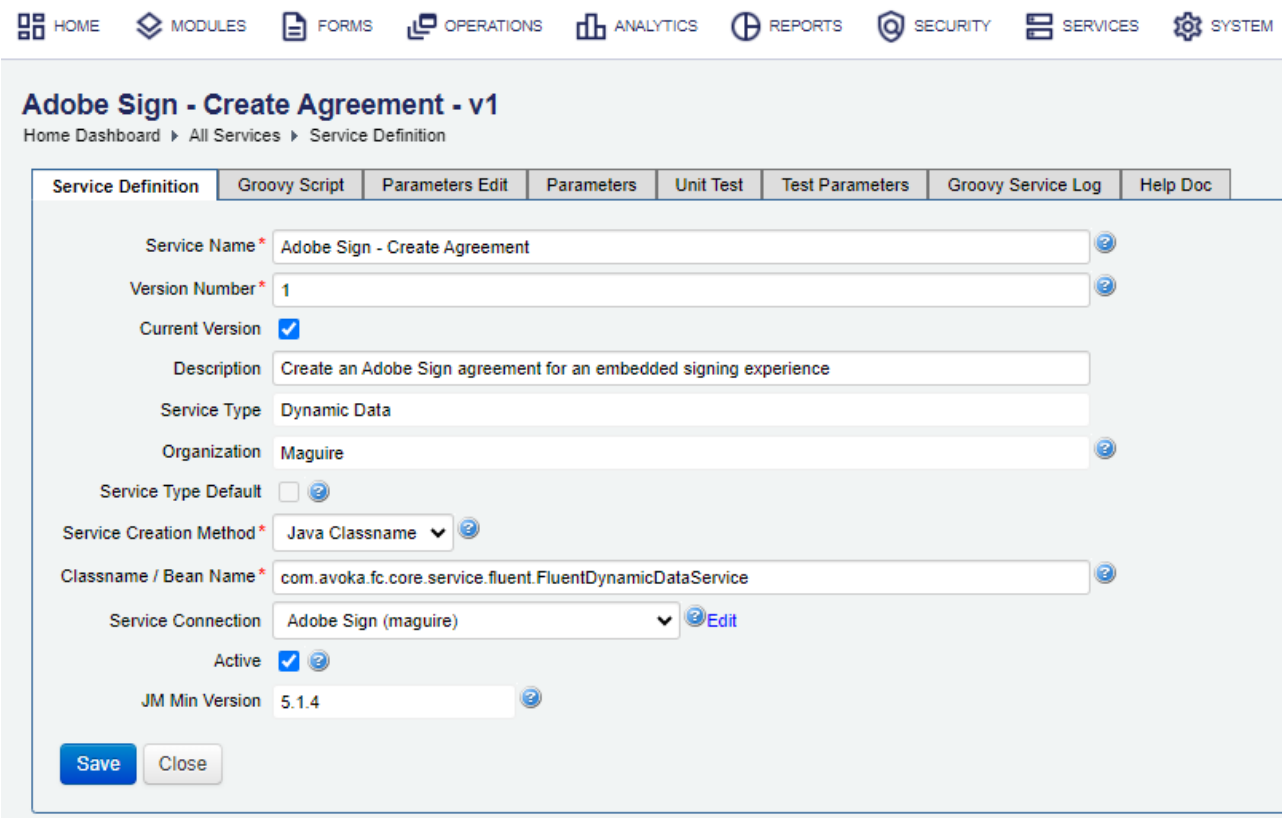
Manager comes with the dynamic data service (DDS) to retrieve required data dynamically, often from external systems. This service can be called by forms to obtain data dynamically while they are being filled in, for example, data lookup or validation using external services. The dynamic data service performs a lookup using a Groovy[1] script configured as one of the service's parameters. You can easily change the lookup behavior by modifying this service parameter, without a restart or rebuild of your Manager server.

Maestro forms use DDS as one of the methods to obtain form data dynamically. For more information, see dynamic data in forms.

To configure the dynamic data service:

1. Select Services > All Services.
2. Locate a service and click Edit.
3. Configure the standard service settings using the Dynamic Data service type as the Service Type and the Fluent Dynamic Data template as the Service Template. Select a relevant service connection to use to connect to the external service.

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.



4. Select the Groovy Script tab and update a Groovy script to make use of a service connection. You can use either Transact Fluent API or Core Transact API depending on the service template you have selected. The snippets of both APIs are shown below:

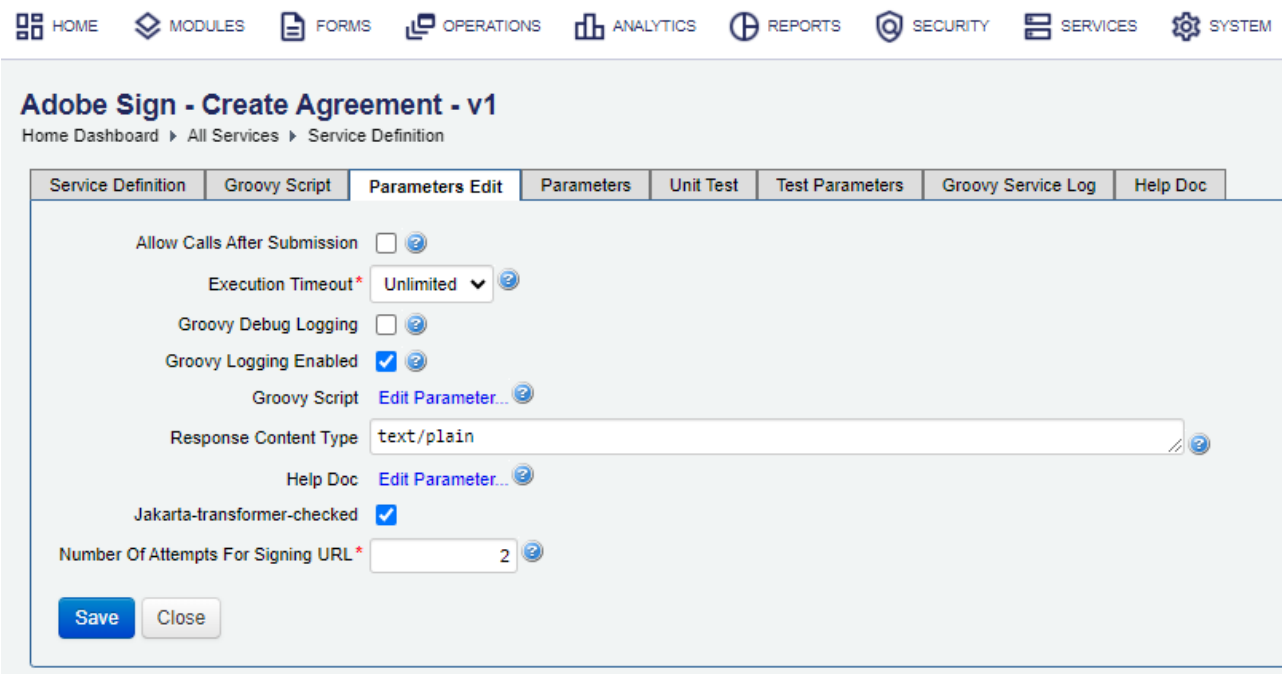
Transact Fluent API

```
String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {
        ...
        def endPoint = svcDef.svcConn.endpoint
        ...
}
```

Core Transact API

```
def serviceConnection = serviceDefinition.connection
def endPoint = serviceConnection.endpointValue
```

5. Select the Parameters Edit tab to edit the service configuration.
6. Click Groovy Script Edit Parameter to create and update parameters used withing the script.



You can retrieve a parameter using the following code:

Transact Fluent API

```
String invoke(SvcDef svcDef, Txn txn, HttpServletRequest request, User user) {
        ...
        def zipConnectionName = svcDef
                .paramsMap
                .get('GetInfoByZip Connection Name')
        ...
}
```

Core Transact API

```
def zipConnectionName = serviceParameters.get('GetInfoByZip Connection Name')
```

7. Configure the Unit Test and Test Parameters tabs.
8. Click Save to update the changes.

DDS services require a request log key, which is session specific, to be provided unless the services run in test mode. This provides some protection against CSRF[1]. DDS services are not

[1]Cross-site request forgery (CSRF), also known as one-click attack or session riding, is a type of malicious exploit of a website where unauthorized commands are transmitted from a user that the web application trusts.

directly related to forms, so you can call any within your organization unless the setting to enable calling after submission is set the submission related to the request log are not be in a finished state. Form Functions has a stronger security model when they are mapped to the form definition. For more information CSRF, see how to configure modules.

Next, learn how to configure a service with multiple endpoints to external services.

Submission Completed Processor Service

Manager allows you to create a service of the submission completed processor type, which is called immediately after a form submission has progressed to form status Completed.

To configure the submission completed processor service:

1. Select Services > All Services.
2. Locate a service and click Edit.
3. Configure the standard service settings.
4. Click Save to update the changes.

For more information, see groovy-services-api/submission-completed-processor.html.

Task Cancellation Processor Service

You can create a service of the same type to cancel or abandon a task submission or a task that is a part of a collaboration job. This service will use the following job controller API:

```
/**
* @see IJobController#cancelJob(Job)
* @param job the job instance to cancel (required)
* @return true if the job was cancelled, or false if the job was already finished
*/
public boolean cancelJob(final Job job)
```

To configure the task cancellation processor service:

1. Select Services > All Services.
2. Locate a service and click Edit.

3. Configure the standard service settings.

4. Select the Groovy Script tab and update a Groovy[1] script as shown below:

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

```
import com.avoka.component.xml.XmlDoc
import com.avoka.fc.core.dao.DaoFactory
import com.avoka.fc.core.service.ServiceFactory
import com.avoka.fc.core.entity.Job
import com.avoka.fc.core.service.ServiceLocator


def XPATH_TACKING_CODES = '//Get-
tingStarted/TrackingCode/rptTrackingCode/TrackingCode'
def PIPE_SEPARATOR = '||'
def xmlDoc = new XmlDoc(submissionXml)
def submissionDao = DaoFactory.getSubmissionDao()
def jobControllerService = new JobControllerService()
def abandonService = ServiceFactory.getAbandonmentService()
def cancelledTasks = []
def notCancelledTasks = []
def trackingCodeNodes = xmlDoc.getNodeList(XPATH_TACKING_CODES)
// Loop all given tracking codes

for (def node : trackingCodeNodes) {
       def isAbandoned = false
       def taskSubject = ''
       try {
               // Get the submission object
               def trackingCode = node.getTextContent()
               def oSubmission = submissionDao.getSubmissionByTrackingNumber(trackingCode)
               taskSubject = oSubmission.getTaskSubject()
               // Get the job that the submission is part of, if any
               def job = oSubmission.getJob()
               // Check if submission is part of collaboratino job
               if (job != null) {
                       // Submission is part of collaboration job
                       def jobController = ServiceLocator.getJobController(job)
                       isAbandoned = jobController.cancelJob(job)
               } else {
                       // Submission is not part of collaboration job
                       isAbandoned = abandonService.performExplicitAdminAbandonment(oSubmission)
               }
       } catch (Exception e) {
```

```
        } finally {

        if (isAbandoned) {
                cancelledTasks.add(taskSubject)
        } else {
                notCancelledTasks.add(taskSubject)
        }
}
```

5. Configure the service in the Unit Test and Test Parameters tabs.
6. Click Save to update the changes.

Next, learn how to view job services.

Job Services Overview

Job services are specialized services used in Journey Manager to create and process collaboration jobs.

Manager provides different job service types and comes with pre-configured job services and job service templates to help you create new job services.

Job services are defined per an organization. Each organization must have at least one job service configured. There is no limit on the number of job services an organization can have. Each form within the organization can use these job services. You can create new job controllers and job actions to implement your requirements.

Manager provides job service templates, which are enhanced job controller services. Templates allow a user to customize collaboration jobs without having to modify the job service definition. Templates are organization agnostic and you can't edit them.

It's important to understand a difference between a job service and a Transact function. A Transact function is usually used to fetch some data or make a payment. A job service, on the other hand, is used to create a workflow when a form application requires a workflow, for example, for back office employees to verify and review customers' applications. You can't replace a job service with a Transact function. You can create a form that triggers a job service and at the same time the form can trigger a number of fluent functions.

Next, learn how to view job services.

Job Definitions

Manager uses job services to process collaboration jobs. You can define and configure a job service as a single JSON[1] document, which is called a job definition. The job definition includes the list of steps that make up the collaboration job. Each of the individual steps has the list of actions to be performed when the step runs, and the routes property, which defines the next step transition by mapping the route name to the next step.

The job definition consists of the following properties:

- jobDetails
- jobGroups
- steps
- actions
- properties
- routes

jobDetails

The `jobDetails` property defines a collaboration job and how it is processed. The snippet of the `jobDetails` is shown below:

```
{
        "jobDetails": {
                "name": "1 Step Review",
                "processSubmitImmediate": "true",
                "allowEarlyCreation": true,
                "version": "0.1.0"
        },
        "steps":[
                { .. step definition ...},
                { .. step definition ...},
                ...
        ]
}
```

The `jobDetails` has the following attributes:

---

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

`name`

The name of the collaboration job, which is set by the job controller service when the collaboration job has been saved.

If defining multiple jobs, use distinct names for each job definition to ensure you can differentiate between them in the Journey Analytics scope selector.

`processSubmitImmediate`

The collaboration job processing mode. If this attribute is omitted or set to false, the job will use the standard mode. Otherwise, it will use the process immediate mode.

`allowEarlyCreation`

A boolean flag that allows you to create a collaboration job and associate it with a transaction that can only be in the Opened or Saved state.

This requires the Job Controller to be associated with a form version of a transaction and Job Controller service has the `allowEarlyCreation` flag set to true in its job definition. The `processJobAfterCreate` attribute will be ignored in this case.

| 24.10 This feature was introduced in 24.10

`version`

The version of Manager that the job was created in.

Let's look at a simple example of using the above configuration to create a job given the job controller service and version for a non submitted Txn as the initial Txn:

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
Job job3 = new JobBuilder()
   .setSvcName('1 Step Review')
   .setSvcVersion('0.1.0')
   .setReferenceNumber('ABC123')
   .setInitialTxnTrackingCode('ABC123')
   .build()
```

jobGroups

The `jobGroups` property lists groups, which job coordinators can use to manage a collaboration job. The `jobGroups` can be used to show collaboration jobs assigned to each group, for example, Journey Space Reviews. The snippet of the `jobGroups` is shown below:

```
"jobGroups" : [
      {
        "name" : "HL Melbourne",
        "description" : "Home Loan Customer Care Group for Vic"
      },
      {
        "name" : "HL Sydney",
        "description" : "Home Loan Customer Care Group for NSW"
      }
    ]
```

The `jobGroups` has the following attributes:

`name`
: The name of the job group, which will be displayed in the Job Groups dropdown from the Reviews tab of the form space. The name must be less than 100 characters.

`description`
: The description for the job group, which must be less than 200 characters.

The job groups can be created in one of the following ways:

1. If a job group defined in a job definition, as in above, has not been created, then the job group will be automatically created the first time a user submits a form associated to this job. This is the preferred way to create job groups.
2. Job groups can be manually created. By manually creating the job group, you must ensure that the job group names and descriptions match with the job definition. Any mismatch will not be subsequently updated to the job groups.

steps

The `steps` property is an JSON array that defines steps. The snippet of the `steps` property is shown below:

```
{
      "name": "Initial Review",
      "type": "",
      "expiryRule": "+5d"
      "expiryServiceName": "An Expiry Service"
      "actions": [
        { ... Action 1 Definition ... },
        { ... Action 2 Definition ... }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display":
"false" },
        { "name": "Terminate",  "nextStep": "Terminated Initial" }
      ]
},
```

The `steps` has the following attributes:

`name`

The name of the step, which is unique within a collaboration job.

`type`

The type of the step, that can be:

- `start`: the first step for the collaboration job. `"type":"start"`
- `endpoint`: the processing stops and the job completes. Endpoints do not specify actions, routes or `customProperties`. `"type":"endpoint"`
- `""`: empty string. `"type":""`

`expiryRule`

The step scheduled completion time, the step expiry, calculation rule. If specified, the `job_step.time_completion_scheduled` is set as the step creation time plus the expiry rule value, for example, `2014-03-05 + 5d = 2014-03-10`. The valid values are:

- `"expiryRule" : "+4h"` (hours)
- `"expiryRule" : "+5d"` (days)
- `"expiryRule" : "+4w` (weeks)
- `"expiryRule" : "+4m"`. (months)

This is an optional attribute.

> NOTE
>
> We recommend setting it up to some default value so an action doesn't stay in the queue forever, if an associated form never abandons and the task is never submitted, even as a transaction expires.

`expiryServiceName`

Specifies a job expiry service that is used to clean up some items, such as incomplete submissions. This service should specify a route.

> NOTE
>
> If the `expiryRule` is used, but the `expritysServiceName` is not specified, then `expiryRoute` must be provided.

actions

The `actions` property defines the list of actions for each step, which are executed sequentially. Actions are required for all steps except endpoints. The snippet of the `actions` property is shown below:

```
{
    "name": "Create Task",
    "type": "Job Task Assign",
    "properties": [
        { "name": "Process Message Send Email",  "value": "true" },
        { "name": "Process Message Submission Step",  "value": "Start" },
        { "name": "Process Message Text",  "value": "Your ${submission.formName} applic-
ation is at the Initial Review Step" },
}
```

The `actions` has the following attributes:

`name`

The name of the action, which is unique within a collaboration job.

`type`

The action service type, which can be one of the following:

- listed above in the Standard Actions Services.
- groovy action services can have a type of Job Action.

`properties`

The action-specific properties. For more information, see the `properties` object definition below. This is an optional attribute.

The standard Action Property classes (`com.avoka.fc.core.service.job.impl`) define a fixed set of Action Properties. The `JobTaskAssignService` defines a broad range of separate properties, so you can add them to your script when creating a new job action. This makes a job action re-useable across a collaboration job and the whole form application. For example, if the job definition has this property:

```
"properties": [
   { "name": "ABC Widget Name",  "value": "Aplle Blue Catapillar" },
   ...
 ]
```

Then, you can retrieve it using the Groovy code below:

```
def abcName = actionStepProperties.getProperty("ABC Widget Name")
```

However, you should consider the following:

- There is no submission associated with the job actions by default.
- `$formDataMap.name` doesn't resolve as the job action and it doesn't have any concept of a submission.
- `$func.previousSubmission().formDataMap.name` resolves a submission as it obtained the submission from the previous step.
- `$job.submission(0).name` resolves a submission as it obtained the first submission linked to the collaboration job.

For more information, see job functions.

properties

The `properties` property defines the job definition properties. The snippet of the `properties` property is shown below:

```
"properties": [
    { "name": "Task Assign Group",  "value": "$func.formProperty('Initial Review')" },
    { "name": "Task Type",  "value": "Review" },
]
```

Properties can be used to pass information into an instance of the Action Service. In this case, they are called action properties and can be defined at the action, step or collaboration job level. Properties with the same name defined at an action level override properties defined at the step level. Step override properties of the same name at the job level. This is illustrated in the snippet below:

```
{
  "jobDetails": { ... },
  "properties": [
    { "name": "Task Form Code", "value": "$func.startFormCode()" }
  ],
  "steps":[
    {
      "name": "Initial Review",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Text",  "value": "Your ${submission.formName}
application is at the Initial Review Step" },
            { "name": "Task Assign Group",  "value": "$func.formProperty('Initial
Review')" },
            { "name": "Task Message",  "value": "Please perform the initial review of the
${submission.formName} from ${formDataMap.firstName} ${formDataMap.lastName}." },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "Initial review of ${submission.formName}
from ${formDataMap.firstName} ${formDataMap.lastName}" },
            { "name": "Task Type",  "value": "Review" },
          ]
        },
        { ... }
      ],
      "properties": [
        { "name": "Task Form Code", "value": "$func.startFormCode()" }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" }
      ],
    },
    { ... }
  ]
}
```

The `properties` has the following attributes:

`name`

The name of the property. The predefined action services, such as the job task assign service, have a set number of action properties.

`value`

A property value that can include an Apache Velocity template. The action properties are templates that are evaluated when an action service is run. The templates can be:

- static text values, for example, `{ "name": "Task Email Message", "value": "An Email Message" }`
- substitute values from the form data extract, for example, `{ "name": "Process Message Text", "value": "Your ${job.submission.formName} application is at the Initial Review Step" }`

`template`

It evaluates the value as a velocity template and forces the return value to return a String. This can evaluate nested function calls. This is can be used with property values that start with "$func", these can return an object. For example, `$func.startUser()` will return a user.

```
{ "name": "Task Email Message", "value": "$func.getSubmission().formXml()", "tem-
plate": "true" }
```

routes

The `routes` property defines the route mapping between the route name and the next step, which are required for all steps except endpoints. An action can return a route result name, as a string, which is listed in each step. If a valid name is returned, the step routes the collaboration job to the next step. However, if the action returns an invalid route name, an error is thrown. The snippet of the `routes` property is shown below:

```
"routes": [
   { "name": "Approve",  "nextStep": "Application Delivery" },
   { "name": "Decline",  "nextStep": "Applicant Update" },
   { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display": "false"
},
   { "name": "Terminate",  "nextStep": "Terminated Initial" }
]
```

The `routes` has the following attributes:

`name`

The route name that can be any name except the following reserved names:

- Default - If there is no step found with a route name or if the route name is empty, the default route is used.
- Expiry - the route taken when the step has expired.

`nextStep`

The next step, which can be one of the following:

- A valid step name contained in the job definition.
- A special case is `##PREVIOUS_STEP` the job processing assign the next step as the previous step name.

`display`

The collaboration job provides a list of available routes to the form. However, the `"display": "false"` removes the route from the available routes. This is an optional attribute.

You can see the complete example of a job definition here.

Next, learn how to configure job definition.

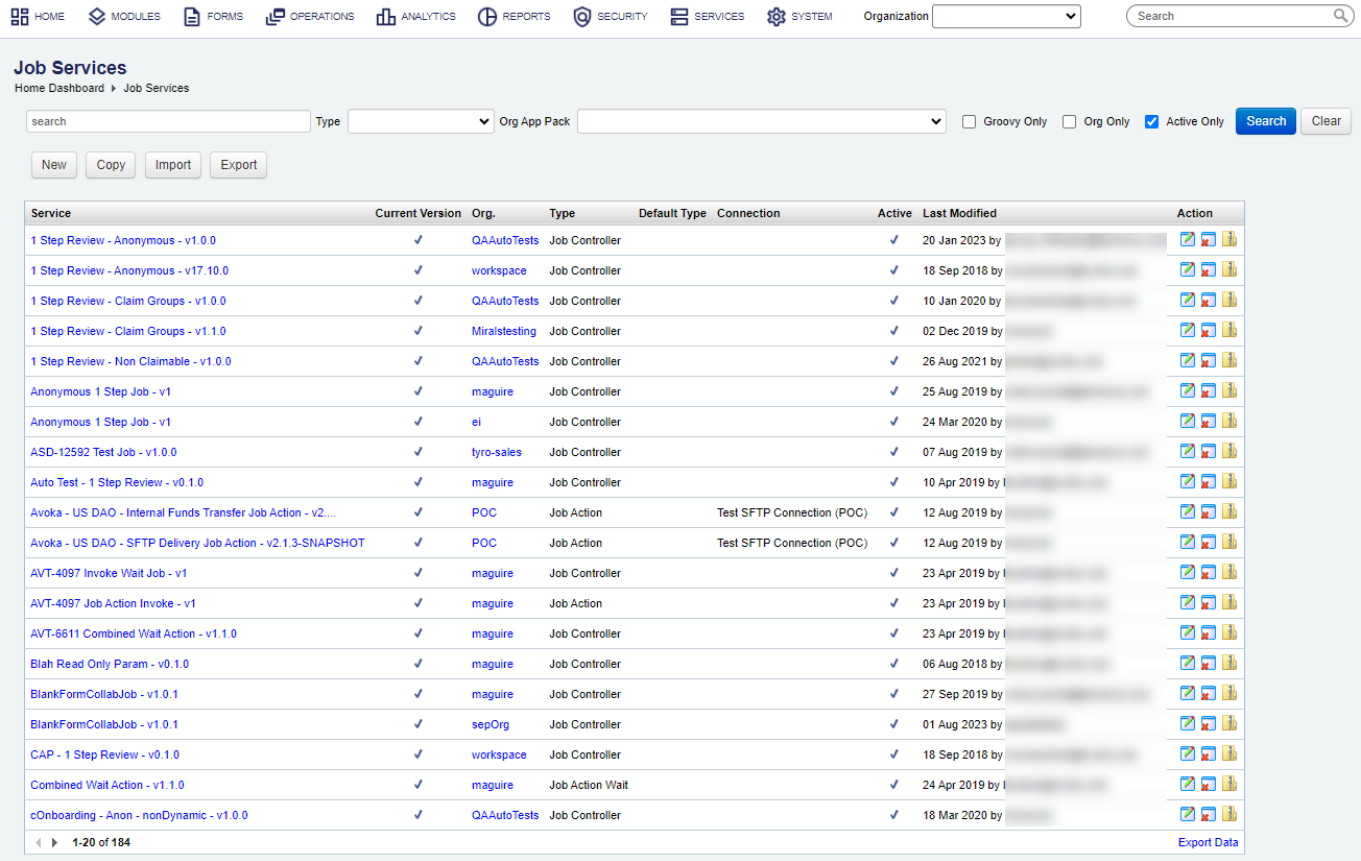
View Job Services

Manager allows you to view all job services, so you can easily find and modify them. The job services are used in collaboration jobs.

> TIP
>
> You can also check core global services using View Services UI.

To view job services, select Services > Job Services.

Job services are shown page by page, with the maximum number of records per page configured in the users preferences.



The list displays the following details:

- Service: a name of a service
- Current Version: whether a job service's version is current or not
- Org: Organization
- Type: a type of a job service
- Default Type: whether this service is the default service for this type of service.

- Connection: a service connection associated with this service, if it is configured.
- Active: whether this service is active or not. It must be active to be available.
- Last Modified: - a date when this service was last modified and by whom.

For more information, see View Services.

To filter or search the services by one or more criteria, specify the following settings and click Search:

- Service name or part of a name
- Service type
- Org App Pack
- Groovy Only to see only Groovy[1] services
- Org Only
- Active Only

Click Clear to reset the search criteria.

You can also perform the following actions:

- Create a job service
- View and configure a service
- Copy a service
- Remove a service
- Export services
- Import services

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

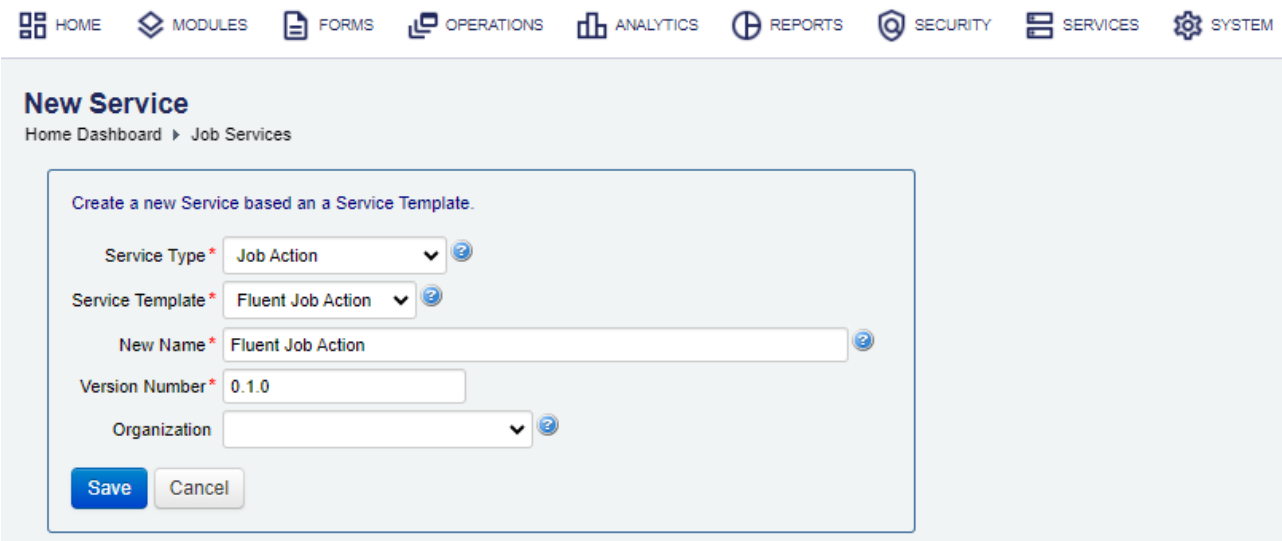
[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

Create a Job Service

Manager allows you to create and configure various job services, which are predominantly used in collaboration jobs.

To create a job service:

1. Select Services > Job Services.
2. Click New.



3. Select a service type of a new job service from the Service Type dropdown list.
4. Select a template from the Service Template dropdown list, which can be one of the following:
   - Fluent Job Action
   - Groovy Job Action
5. Select an organization from the Organization dropdown list or leave it blank to create a global job service.

> NOTE
>
> You must be a Journey Manager administrator to create or configure this functionality.

NOTE

You have to assign a job controller to an organization. If you have several organizations that would like to use the same job controller, you need to define one for each organization.

6. Click Save to update the changes.

Next, learn how to configure Job Definition.

Job Service Templates

Manager provides job service templates, which are enhanced job controller services. Templates allow you to customize collaboration jobs without modifying the job service definition. Templates are organization agnostic so you can't edit them. They are predominantly used for reviews and approvals. Manager comes with the following default templates:

- Review and Approval - 1 Step Group Review - v1.0.0
- Review and Approval - 1 Step User Review - v1.0.0
- Review and Approval - 2 Step Group Review - v1.0.0
- Review and Approval - Multi Step Group Review - v1.0.0

The job service templates are especially useful for:

- Business users and form builders to configure their forms without a developer.
- Greater reuse of a single job controller.

To use a job service template:

1. Locate a form version.
2. Click the Services tab to view services configured for this form.
3. Select a job service template from the Job Controller Service dropdown list, which has the following available templates:
    - Review and Approval - 1 Step Group Review - v1.0.0
    - Review and Approval - 1 Step User Review - v1.0.0
    - Review and Approval - 2 Step Group Review - v1.0.0
    - Review and Approval - Multi Step Group Review - v1.0.0
4. Click Save to update the changes.
5. Select the Job Properties tab to configure the collaboration job's properties.
6. Select the Job Reviewers from the Reviewer Group dropdown list.
7. Select the Job Example 1 from the Review Form dropdown list.
8. Click Save to update the changes.

A job service template has the Form Property Editor Definition service parameter, which is a JSON[1] file containing the form properties meta data.

---

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

Review and Approval - 1 Step Group Review - v1

Home Dashboard ▸ Job Services ▸ Service Definition

Service Definition | Job Definition | Job Services | Job Info | Parameters Edit | **Parameters** | Service Usage

Name	Type	Value	Description	Bind Parameter	Last Modified
actionRetryDelayMins	Number	5	Specify the default action retry delay in minutes for 'In Progress' and 'Error' actions.	✓	02 Sep 2015 by system
Form Property Editor Definition	JSON	[{ "name": "Step - Application Start", ...	Templated job form properties editor definition (JSON format).		02 Sep 2015 by system
jobControllerEnabled	Boolean	true	Specify whether the Job Controller service is enabled and can process jobs.	✓	02 Sep 2015 by system
jobDefinition	JSON	{ "jobDetails": { "name": "Review and Approval...	Job definition configuration (JSON format).	✓	02 Sep 2015 by system

Close

You can create a new custom template by adding the Form Property Editor Definition parameter to a standard job controller.

Next, learn about job controller services.

Job Definitions

Manager uses job services to process collaboration jobs. You can define and configure a job service as a single JSON[1] document, which is called a job definition. The job definition includes the list of steps that make up the collaboration job. Each of the individual steps has the list of actions to be performed when the step runs, and the routes property, which defines the next step transition by mapping the route name to the next step.

The job definition consists of the following properties:

- jobDetails
- jobGroups
- steps
- actions
- properties
- routes

jobDetails

The `jobDetails` property defines a collaboration job and how it is processed. The snippet of the `jobDetails` is shown below:

```
{
        "jobDetails": {
                "name": "1 Step Review",
                "processSubmitImmediate": "true",
                "allowEarlyCreation": true,
                "version": "0.1.0"
        },
        "steps":[
                { .. step definition ...},
                { .. step definition ...},
                ...
        ]
}
```

The `jobDetails` has the following attributes:

---

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

`name`

The name of the collaboration job, which is set by the job controller service when the collaboration job has been saved.

If defining multiple jobs, use distinct names for each job definition to ensure you can differentiate between them in the Journey Analytics scope selector.

`processSubmitImmediate`

The collaboration job processing mode. If this attribute is omitted or set to false, the job will use the standard mode. Otherwise, it will use the process immediate mode.

`allowEarlyCreation`

A boolean flag that allows you to create a collaboration job and associate it with a transaction that can only be in the Opened or Saved state.

This requires the Job Controller to be associated with a form version of a transaction and Job Controller service has the `allowEarlyCreation` flag set to true in its job definition. The `processJobAfterCreate` attribute will be ignored in this case.

| 24.10 This feature was introduced in 24.10

`version`

The version of Manager that the job was created in.

Let's look at a simple example of using the above configuration to create a job given the job controller service and version for a non submitted Txn as the initial Txn:

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
Job job3 = new JobBuilder()
   .setSvcName('1 Step Review')
   .setSvcVersion('0.1.0')
   .setReferenceNumber('ABC123')
   .setInitialTxnTrackingCode('ABC123')
   .build()
```

jobGroups

The `jobGroups` property lists groups, which job coordinators can use to manage a collaboration job. The `jobGroups` can be used to show collaboration jobs assigned to each group, for example, Journey Space Reviews. The snippet of the `jobGroups` is shown below:

```
"jobGroups" : [
      {
        "name" : "HL Melbourne",
        "description" : "Home Loan Customer Care Group for Vic"
      },
      {
        "name" : "HL Sydney",
        "description" : "Home Loan Customer Care Group for NSW"
      }
    ]
```

The `jobGroups` has the following attributes:

`name`
: The name of the job group, which will be displayed in the Job Groups dropdown from the Reviews tab of the form space. The name must be less than 100 characters.

`description`
: The description for the job group, which must be less than 200 characters.

The job groups can be created in one of the following ways:

1. If a job group defined in a job definition, as in above, has not been created, then the job group will be automatically created the first time a user submits a form associated to this job. This is the preferred way to create job groups.
2. Job groups can be manually created. By manually creating the job group, you must ensure that the job group names and descriptions match with the job definition. Any mismatch will not be subsequently updated to the job groups.

steps

The `steps` property is an JSON array that defines steps. The snippet of the `steps` property is shown below:

```
{
      "name": "Initial Review",
      "type": "",
      "expiryRule": "+5d"
      "expiryServiceName": "An Expiry Service"
      "actions": [
        { ... Action 1 Definition ... },
        { ... Action 2 Definition ... }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display":
"false" },
        { "name": "Terminate",  "nextStep": "Terminated Initial" }
      ]
},
```

The `steps` has the following attributes:

`name`

The name of the step, which is unique within a collaboration job.

`type`

The type of the step, that can be:

- `start`: the first step for the collaboration job. `"type":"start"`
- `endpoint`: the processing stops and the job completes. Endpoints do not specify actions, routes or `customProperties`. `"type":"endpoint"`
- `""`: empty string. `"type":""`

`expiryRule`

The step scheduled completion time, the step expiry, calculation rule. If specified, the `job_step.time_completion_scheduled` is set as the step creation time plus the expiry rule value, for example, `2014-03-05 + 5d = 2014-03-10`. The valid values are:

- `"expiryRule" : "+4h"` (hours)
- `"expiryRule" : "+5d"` (days)
- `"expiryRule" : "+4w` (weeks)
- `"expiryRule" : "+4m"`. (months)

This is an optional attribute.

> NOTE
>
> We recommend setting it up to some default value so an action doesn't stay in the queue forever, if an associated form never abandons and the task is never submitted, even as a transaction expires.

`expiryServiceName`

Specifies a job expiry service that is used to clean up some items, such as incomplete submissions. This service should specify a route.

> NOTE
>
> If the `expiryRule` is used, but the `expritysServiceName` is not specified, then `expiryRoute` must be provided.

actions

The `actions` property defines the list of actions for each step, which are executed sequentially. Actions are required for all steps except endpoints. The snippet of the `actions` property is shown below:

```
{
    "name": "Create Task",
    "type": "Job Task Assign",
    "properties": [
        { "name": "Process Message Send Email",  "value": "true" },
        { "name": "Process Message Submission Step",  "value": "Start" },
        { "name": "Process Message Text",  "value": "Your ${submission.formName} applic-
ation is at the Initial Review Step" },
}
```

The `actions` has the following attributes:

`name`

The name of the action, which is unique within a collaboration job.

`type`

The action service type, which can be one of the following:

- listed above in the Standard Actions Services.
- groovy action services can have a type of Job Action.

`properties`

The action-specific properties. For more information, see the `properties` object definition below. This is an optional attribute.

The standard Action Property classes (`com.avoka.fc.core.service.job.impl`) define a fixed set of Action Properties. The `JobTaskAssignService` defines a broad range of separate properties, so you can add them to your script when creating a new job action. This makes a job action re-useable across a collaboration job and the whole form application. For example, if the job definition has this property:

```
"properties": [
   { "name": "ABC Widget Name",  "value": "Aplle Blue Catapillar" },
   ...
 ]
```

Then, you can retrieve it using the Groovy code below:

```
def abcName = actionStepProperties.getProperty("ABC Widget Name")
```

However, you should consider the following:

- There is no submission associated with the job actions by default.
- `$formDataMap.name` doesn't resolve as the job action and it doesn't have any concept of a submission.
- `$func.previousSubmission().formDataMap.name` resolves a submission as it obtained the submission from the previous step.
- `$job.submission(0).name` resolves a submission as it obtained the first submission linked to the collaboration job.

For more information, see job functions.

properties

The `properties` property defines the job definition properties. The snippet of the `properties` property is shown below:

```
"properties": [
    { "name": "Task Assign Group",  "value": "$func.formProperty('Initial Review')" },
    { "name": "Task Type",  "value": "Review" },
]
```

Properties can be used to pass information into an instance of the Action Service. In this case, they are called action properties and can be defined at the action, step or collaboration job level. Properties with the same name defined at an action level override properties defined at the step level. Step override properties of the same name at the job level. This is illustrated in the snippet below:

```
{
  "jobDetails": { ... },
  "properties": [
    { "name": "Task Form Code", "value": "$func.startFormCode()" }
  ],
  "steps":[
    {
      "name": "Initial Review",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Text",  "value": "Your ${submission.formName}
application is at the Initial Review Step" },
            { "name": "Task Assign Group",  "value": "$func.formProperty('Initial
Review')" },
            { "name": "Task Message",  "value": "Please perform the initial review of the
${submission.formName} from ${formDataMap.firstName} ${formDataMap.lastName}." },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "Initial review of ${submission.formName}
from ${formDataMap.firstName} ${formDataMap.lastName}" },
            { "name": "Task Type",  "value": "Review" },
          ]
        },
        { ... }
      ],
      "properties": [
        { "name": "Task Form Code", "value": "$func.startFormCode()" }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" }
      ],
    },
    { ... }
  ]
}
```

The `properties` has the following attributes:

`name`

The name of the property. The predefined action services, such as the job task assign service, have a set number of action properties.

`value`

A property value that can include an Apache Velocity template. The action properties are templates that are evaluated when an action service is run. The templates can be:

- static text values, for example, `{ "name": "Task Email Message", "value": "An Email Message" }`
- substitute values from the form data extract, for example, `{ "name": "Process Message Text", "value": "Your ${job.submission.formName} application is at the Initial Review Step" }`

`template`

It evaluates the value as a velocity template and forces the return value to return a String. This can evaluate nested function calls. This is can be used with property values that start with "$func", these can return an object. For example, `$func.startUser()` will return a user.

```
{ "name": "Task Email Message", "value": "$func.getSubmission().formXml()", "tem-
plate": "true" }
```

routes

The `routes` property defines the route mapping between the route name and the next step, which are required for all steps except endpoints. An action can return a route result name, as a string, which is listed in each step. If a valid name is returned, the step routes the collaboration job to the next step. However, if the action returns an invalid route name, an error is thrown. The snippet of the `routes` property is shown below:

```
"routes": [
   { "name": "Approve",  "nextStep": "Application Delivery" },
   { "name": "Decline",  "nextStep": "Applicant Update" },
   { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display": "false"
},
   { "name": "Terminate",  "nextStep": "Terminated Initial" }
]
```

The `routes` has the following attributes:

`name`

The route name that can be any name except the following reserved names:

- Default - If there is no step found with a route name or if the route name is empty, the default route is used.
- Expiry - the route taken when the step has expired.

`nextStep`

The next step, which can be one of the following:

- A valid step name contained in the job definition.
- A special case is `##PREVIOUS_STEP` the job processing assign the next step as the previous step name.

`display`

The collaboration job provides a list of available routes to the form. However, the `"display": "false"` removes the route from the available routes. This is an optional attribute.

You can see the complete example of a job definition here.

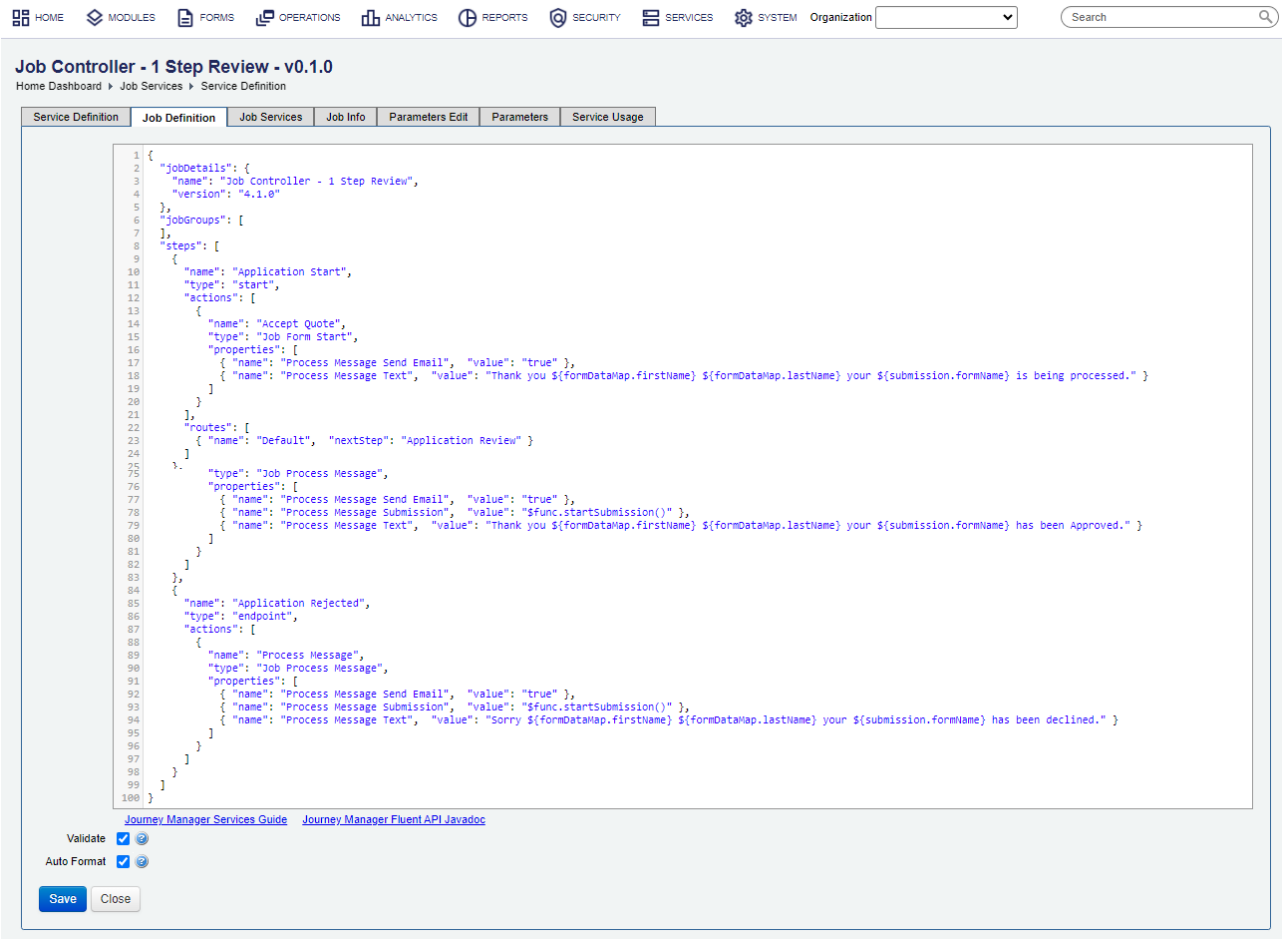
Next, learn how to configure job definition.

Configure Job Definition

Manager allows you to specify how a job service processes collaboration jobs by defining an execution flow using steps and actions in the job definition tab.

To configure a job definition of a service:

1. Locate an existing job service.
2. Select the Job Definition tab to review or update the JSON[1] file defining this service.



3. Select the Validate checkbox to validate action services, step properties, form codes and custom services defined in the JSON file. It is selected by default, but you can clear it, if you want to add a new action but you haven't created a form or a custom service for this, and you still need to save your changes.

---

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

4. Select the Auto Format checkbox to automatically format the job definition and strip unknown elements.
5. Click Save to update the changes..

Next, learn how to view all job services of a job controller.

View Job Info

Manager provides you with a read only view of job info of a selected job service.

To check job info:

1. Locate a job service.
2. Select the Job Info tab to view job info of this service that includes:
    - Job name
    - Job steps
    - Job Form Data Maps, which lists the names of the parameters used in the form XML data.
3. Click Close.

Next, learn how to configure service parameters.

Create a Job Controller Service

Manager uses job controller services to:

- Create a collaboration job instance when a user submits a form or the API is called.
- Process collaboration jobs by executing their steps and actions defined in a job definition.

Manager allows you to create and configure job controller services to implement your customer requirement.

To create a job controller service:

1. Select Services > Job Services and click New.
2. Select the Job Controller from the Service Type dropdown list. For more information, see service types.
3. Select a service template from the Service Template dropdown list, which can be one of the following:
   - Job Controller - 1 Step Review
   - Job Controller - 2 Step Review
4. Select an organization from the Organization dropdown list.

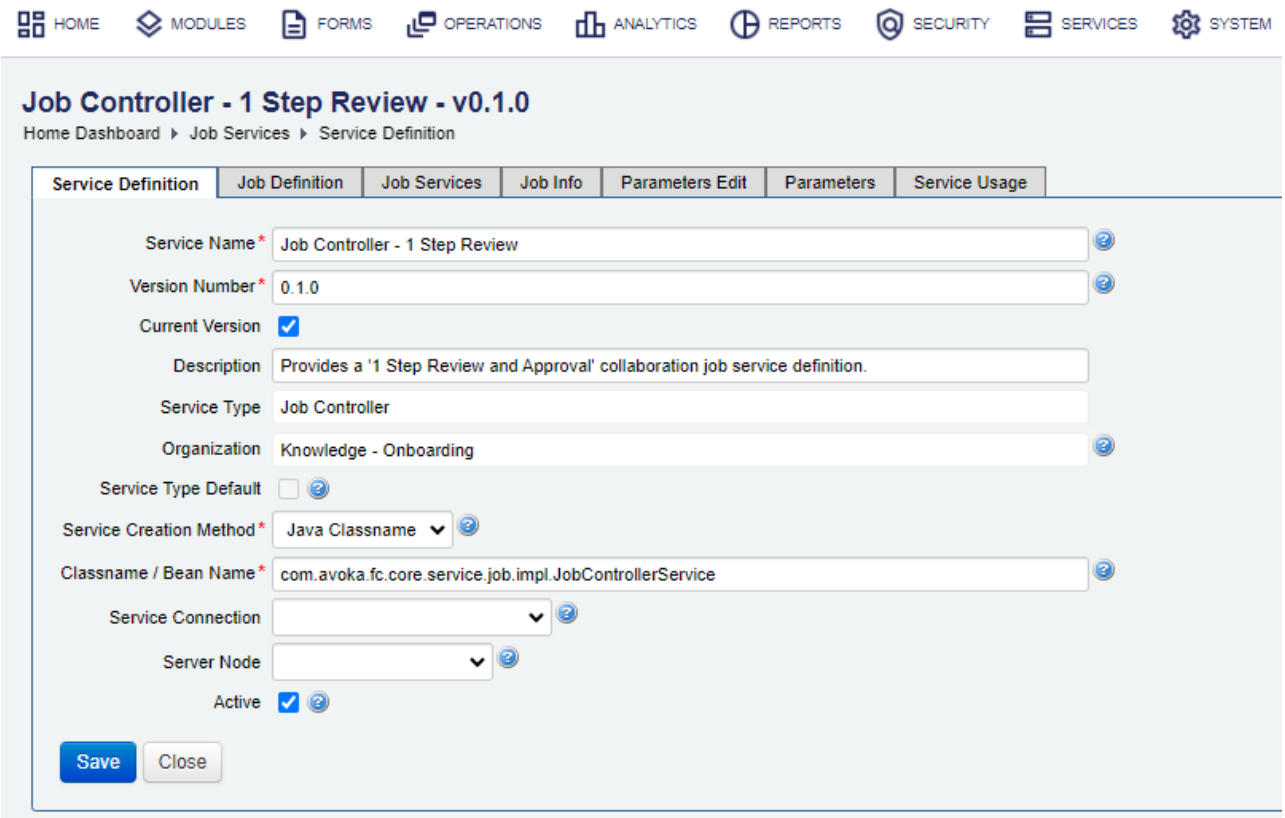
   NOTE

   You have to assign a job controller to an organization. If you have several organizations that would like to use the same job controller, you need to define one for each organization.

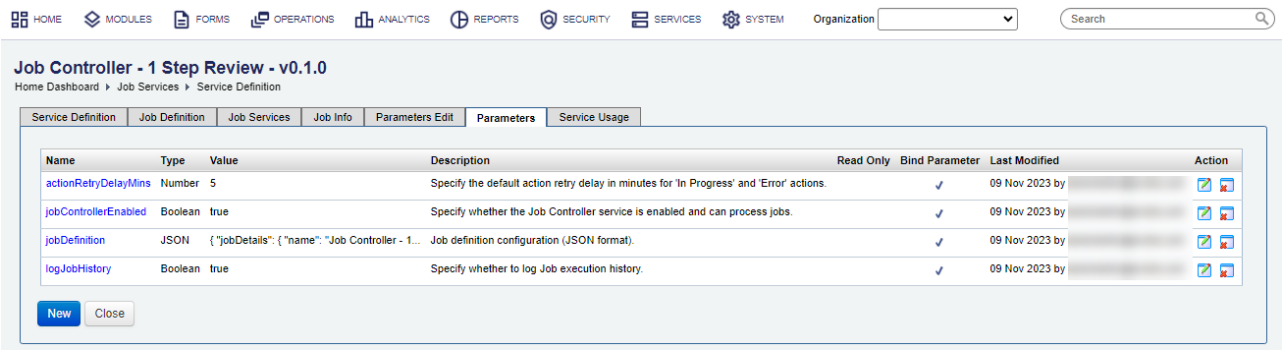
5. Click Save to update the changes.

To configure a job controller service:

1. Locate a job service.
2. Configure the service definition.



3. Configure the job definition.
4. Check the job services used in this job controller.
5. Check the job info.
6. Create service parameters required for this job controller to run.



The standard service parameters are:

- `actionRetryDelayMins` - specifies the default action retry delay in minutes for 'In Progress' and 'Error' actions. Default is 5. This is a bind parameter.

- `jobControllerEnabled` - specifies whether the job controller service is enabled and it can process collaboration jobs. This is useful for debugging and test in non-production, and selectively enabling certain collaboration jobs in production environments. Default is true. This is a bind parameter.
- `jobDefinition` - job definition configuration in JSON[1] format. This is a bind parameter.
- `logJobHistory` - specifies whether to write collaboration job execution history in a log file or not. Default is true. This is a bind parameter.

The optional service parameters are:

- `jobDiagram` - contains the collaboration job diagram format created by the external NextStep tool.
- `Form Property Editor Definition` - the job form properties editor definition in JSON format.

7. Check the job controller service usage.

Next, learn how to configure a job controller in a form.

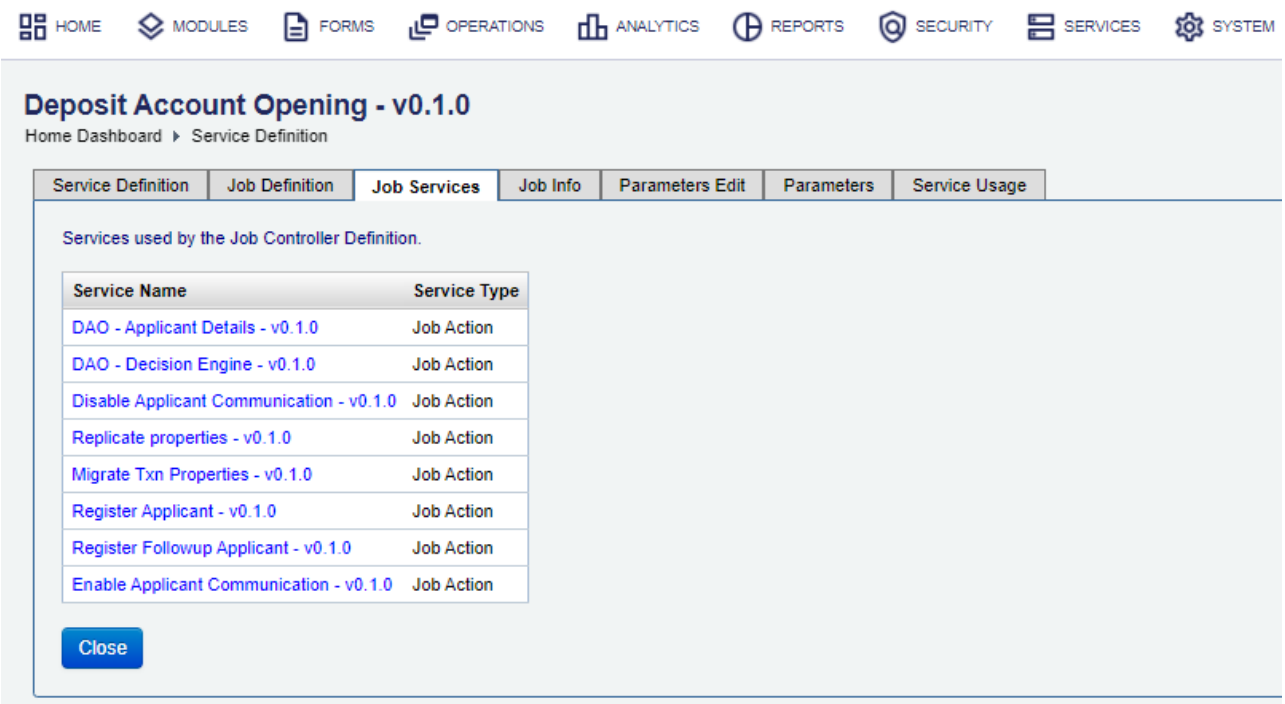
[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

View Job Controller Services

Manager allows you to viewjob services used by a job controller.

To view the list of job services:

1. Locate an existing job controller.
2. Select the Job Services tab to list all job services, which are configured in the job definition. The list is empty if there are no service configured in the JSON[1] file.



3. Click a service name to check its configuration.

Next, learn how to check job info.

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

Job Action Service

Manager allows you to create and configure job action services. When a collaboration job's action is processed, an action service is run. A collaboration job usually specifies an action service from one of the available inbuilt action services. This allows a developer to create their own job action and job expiry services using Groovy[1] scripts.

To create a job action service:

1. Select Services > Job Services.
2. Click New to create a job service.
3. Select the Job Action as a service type from the Service Type dropdown list.
4. Click Save to update the changes.

To configure a job action service:

1. Locate a job action service.
2. Configure the service definition.
3. Select the new service type to assign a Fluent or Groovy job action to a new type, such as Job Expiry Service or Job Task Wait.
4. Configure the job definition. You can configure an action to call a Groovy service from a job controller, as shown below.

```
"actions": [
  {
      "name": "Individual Tasks",
      "type": "Job Task Assign",
      "properties": [
         { "name": "Task Assign Repeating",  "value": "true" },
         { "name": "Task Assign Repeat Item",  "value": "$func.invoke('Repeating
Get Item', ${formDataMap.itemDelimited}, ${assignRepeatIndex})" },
         { "name": "Task Type",  "value": "Review" }
      ]
   }
```

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

The `$func.invoke` method executes the Repeating Get Item service (invoking it by the name) with two parameters.

If you want to invoke same job action multiple times as part of single step, give them different job action names, for example, `"name": "Individual Tasks 2"`, but define the same service name, for example,`"serviceName": "Repeating Claimable"`. If you do not define the service name explicitly, it expects the job action name to match the service name. It's not allowed to have 2 or more job actions with the same name in one job step.

5. Click the Job Services tab to view the Repeating Get Item service used in the action definition.

6. Click on the service name to open this service's Groovy script.

```
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import groovy.transform.TypeChecked
import javax.servlet.http.*
import org.apache.commons.lang3.StringUtils

@TypeChecked
class FluentGroovyService {

    // Injected at runtime
    public Logger logger

    /*
     * Perform a groovy service invocation
     *
     * return: the result object
     */
    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, Map params)
{
        JobAction jobAction = (JobAction) params.jobAction
        return invoke(svcDef, request, user, jobAction, params)
    }

    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, JobAction
jobAction, Map params) {
        logger.info params
        List<String> args = (List<String>) params.args

        def result = "Not Found"

        if( StringUtils.isNotEmpty(args[0]) &&
           StringUtils.isNotEmpty(args[1]) &&
           args[1].isNumber()) {
            def itemDelimited = args[0]
            // note repeat index from TM is 1 based not 0 based
            def repeatIndex = args[1].toInteger() - 1
            logger.info "itemDelimited: " + itemDelimited
            logger.info "repeatIndex: " + repeatIndex

            def items = itemDelimited.tokenize('|')
```

```
            logger.info "items: " + items

            result = items[repeatIndex.toInteger()]
            logger.info "result Item: " + result
        }

        return result

    }
}
```

7. Click Save to update the changes.

Download

You can download the complete code example here.

Next, learn how to view all job services.

Job Action Wait Service

Manager allows you to create a job action wait service, which waits until all the step's previous Invoked actions have been completed. For more information, see action statuses.

The job action wait service is an action service and it is a part of the collaboration job framework that you use to implement various form processing flows.

You can create and update a job action wait service using the Manager UI, as well as Fluent or Core API.

To create a job action wait service:

1. Select Services > Job Services and click New .
2. Select Job Action Wait from the Service Type dropdown list. For more information, see service types.
3. Select a template from the Service Template dropdown list, which can be one of the following:
   - Fluent Job Action Wait
   - Groovy Job Action Wait
   - Job Action Wait
4. Click Save to update the changes.

To configure a job action wait service:

1. Locate a job action service.
2. Configure the service definition. You can choose a new service type from the Service Type dropdown list to change the signature of the job service, for example, to Job Expiry Service or Job Task Wait. This won't remove the existing Groovy script or parameters.
3. Configure the job definition. You can configure an action to call a Groovy service from a job controller, as shown below.

```
"actions": [
  {
      "name": "Individual Tasks",
      "type": "Job Task Assign",
      "properties": [
         { "name": "Task Assign Repeating",  "value": "true" },
         { "name": "Task Assign Repeat Item",  "value": "$func.invoke('Repeating
Get Item', ${formDataMap.itemDelimited}, ${assignRepeatIndex})" },
         { "name": "Task Type",  "value": "Review" }
      ]
   }
```

The `$func.invoke` method executes the Repeating Get Item service (invoking it by the name) with two parameters.

If you want to invoke same job action multiple times as part of single step, give them different job action names, for example, `"name": "Individual Tasks 2"`, but define the same service name, for example,`"serviceName": "Repeating Claimable"`. If you do not define the service name explicitly, it expects the job action name to match the service name. It's not allowed to have 2 or more job actions with the same name in one job step.

4. Click the Job Services tab to view the Repeating Get Item service used in the action definition.

5. Click on the service name to open this service's Groovy script.

```
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import groovy.transform.TypeChecked
import javax.servlet.http.*
import org.apache.commons.lang3.StringUtils

@TypeChecked
class FluentGroovyService {

    // Injected at runtime
    public Logger logger

    /*
     * Perform a groovy service invocation
     *
     * return: the result object
     */
    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, Map params)
{
        JobAction jobAction = (JobAction) params.jobAction
        return invoke(svcDef, request, user, jobAction, params)
    }

    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, JobAction
jobAction, Map params) {
        logger.info params
        List<String> args = (List<String>) params.args

        def result = "Not Found"

        if( StringUtils.isNotEmpty(args[0]) &&
           StringUtils.isNotEmpty(args[1]) &&
           args[1].isNumber()) {
            def itemDelimited = args[0]
            // note repeat index from TM is 1 based not 0 based
            def repeatIndex = args[1].toInteger() - 1
            logger.info "itemDelimited: " + itemDelimited
            logger.info "repeatIndex: " + repeatIndex

            def items = itemDelimited.tokenize('|')
```

```
            logger.info "items: " + items

            result = items[repeatIndex.toInteger()]
            logger.info "result Item: " + result
        }

        return result

    }
}
```

6. Click Save to update the changes.

To create a Job Action Wait service using API:

Core API

`com.avoka.fc.core.service.job.impl.JobActionWaitService`
`com.avoka.fc.core.service.job.impl.GroovyJobActionService`

For more information, see Core API

Fluent API

`com.avoka.fc.core.service.fluent.FluentJobActionService`
For more information, see Fluent API

Next, learn how to view all job services.

Job Delivery Service

Manager allows you to create a job delivery service, which performs a delivery of form submitted data. Unlike some other service types like Fluent Function or Job Action, the job delivery doesn't require you to write any code. You simply give your service a name and indicate which delivery channel it's supposed to tickle when invoked. When your collaboration job has done all the things you need doing, finish off with a step that invokes your job delivery service.

You can create and update a job delivery service using the Manager UI, as well as Fluent or Core API.

To create a job delivery service:

1. Select Services > Job Services and click New .
2. Select job delivery from the Service Type dropdown list. For more information, see service types.
3. Select a template from the Service Template dropdown list, which can be one of the following:
    - Fluent Job Expiry
    - Groovy Job Expiry
4. Click Save to update the changes.

To configure a job expiry service:

1. Locate a job delivery service.
2. Configure the service definition.
3. Configure the job definition. You can configure an action to call a Groovy service from a job controller, as shown below.

```
{
  "jobDetails": {
    "name": "Basic Onboarding Job Controller",
    "version": "4.1.0"
  },
  "jobGroups": [
  ],
  "steps": [
    {
      "name": "Application Start",
      "type": "start",
      "actions": [
        {
          "name": "Handle Submission",
          "type": "Job Form Start",
          "properties": [
            { "name": "Process Message Send Email",  "value": "false" }
          ]
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Deliver" }
      ]
    },
    {
      "name": "Deliver",
      "type": "",
      "actions": [
        {
          "name": "Onboarding Job Delivery",
          "type": "Job Delivery"
        },
        {
          "name": "Application Delivery Wait",
          "type": "Job Delivery Wait"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Done" }
      ]
```

```
    },
    {
      "name": "Done",
      "type": "endpoint"
    }
  ]
}
```

You can see that the job step Deliver has two actions, one of type Job Delivery and another of type Job Delivery Wait. The reason for that is a bit technical. You see, Manager has a Collaboration Job Controller service that runs as a scheduled job and, as its name implies, manages collaboration jobs. It has a separate service called Transaction Processing, which does other stuff. Well, the job delivery is handled by the Transaction Processing service, not the Collaboration Job Controller service, so this wait business is just about the latter waiting for the former. You could think of the Onboarding Job Delivery action as instructing Job Controller to prepare this job for delivery by the Transaction Processor while the Application Delivery Wait action instructs the Job Controller to wait about until the delivery is complete.

4. Click the Job Services tab to view the Repeating Get Item service used in the action definition.
5. Click on the service name to open this service's Groovy script.
6. Click Save to update the changes.

To create a job delivery service using API:

Core API

`com.avoka.fc.core.service.job.impl.JobDeliveryService`
`com.avoka.fc.core.service.job.impl.GroovyJobDeliveryService`

For more information, see Core API

Fluent API

`com.avoka.fc.core.service.fluent.FluentJobDeliveryService`

For more information, see Fluent API

Next, learn how to view job services.

Job Task Wait Action Service

Manager allows you to create and configure job task wait action services, which are specialized job action services designed to block the execution of a collaboration job until a specified event takes place.

To create a job action service:

1. Select Services > Job Services.
2. Click New to create a job service.
3. Select the Job Task Wait Action as a service type from the Service Type dropdown list.
4. Select the Fluent Job Action template from the Service Template dropdown list.
5. Select an organization from the Organization dropdown list.
6. Click Save to update the changes.

To configure a job action service:

1. Locate a job action service.
2. Configure the service definition.
3. Select the new service type to assign a Fluent or Groovy job action to a new type, such as Job Expiry Service or Job Task Wait.
4. Configure the job definition. You can configure an action to call a Groovy service from a job controller, as shown below.

```
"actions": [
        {
          "name": "Individual Tasks",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Task Assign Repeating",  "value": "true" },
            { "name": "Task Assign Repeat Item",  "value": "$func.invoke('Repeating
Get Item', ${formDataMap.itemDelimited}, ${assignRepeatIndex})" },
            ...
            { "name": "Task Type",  "value": "Review" }
          ]
        }
```

The `$func.invoke` method executes the Repeating Get Item service (invoking it by the name) with two parameters.

5. Click the Job Services tab to view the Repeating Get Item service used in the action definition.

6. Click on the service name to open this service's Groovy script.

```
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import groovy.transform.TypeChecked
import javax.servlet.http.*
import org.apache.commons.lang3.StringUtils

@TypeChecked
class FluentGroovyService {

    // Injected at runtime
    public Logger logger

    /*
     * Perform a groovy service invocation
     *
     * return: the result object
     */
    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, Map params)
{
        JobAction jobAction = (JobAction) params.jobAction
        return invoke(svcDef, request, user, jobAction, params)
    }

    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, JobAction
jobAction, Map params) {
        logger.info params
        List<String> args = (List<String>) params.args

        def result = "Not Found"

        if( StringUtils.isNotEmpty(args[0]) &&
           StringUtils.isNotEmpty(args[1]) &&
           args[1].isNumber()) {
            def itemDelimited = args[0]
            // note repeat index from TM is 1 based not 0 based
            def repeatIndex = args[1].toInteger() - 1
            logger.info "itemDelimited: " + itemDelimited
            logger.info "repeatIndex: " + repeatIndex

            def items = itemDelimited.tokenize('|')
```

```
            logger.info "items: " + items

            result = items[repeatIndex.toInteger()]
            logger.info "result Item: " + result
        }

        return result

    }
}
```

7. Click Save to update the changes.

Next, learn how to view all job services.

Open the TaskUserWebService in SOAP UI

Manager allows you to open the TaskUserWebService in SOAP UI. You should read the section 4 of the transact_web_service_guide_v4.2.pdf documentation prior to doing so.

The TaskUserWebService and TaskUserWebServiceV2 WSDL in Manager are located in a secure context path. This wiki article shows the workaround steps to successfully import the WSDL's and call methods in SOAP-UI.

Task Web Services V2 requires you to authenticate using a username and password, It is possible to get the wsdl from the non secure context however you will be unable to call the secured services from the unsecured WSDL

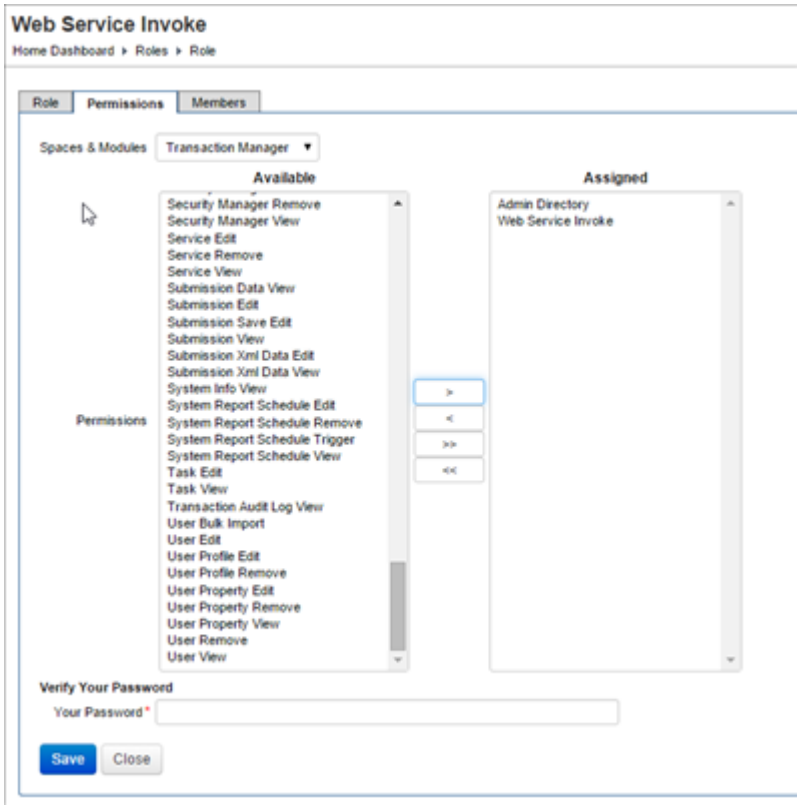
To open the service:

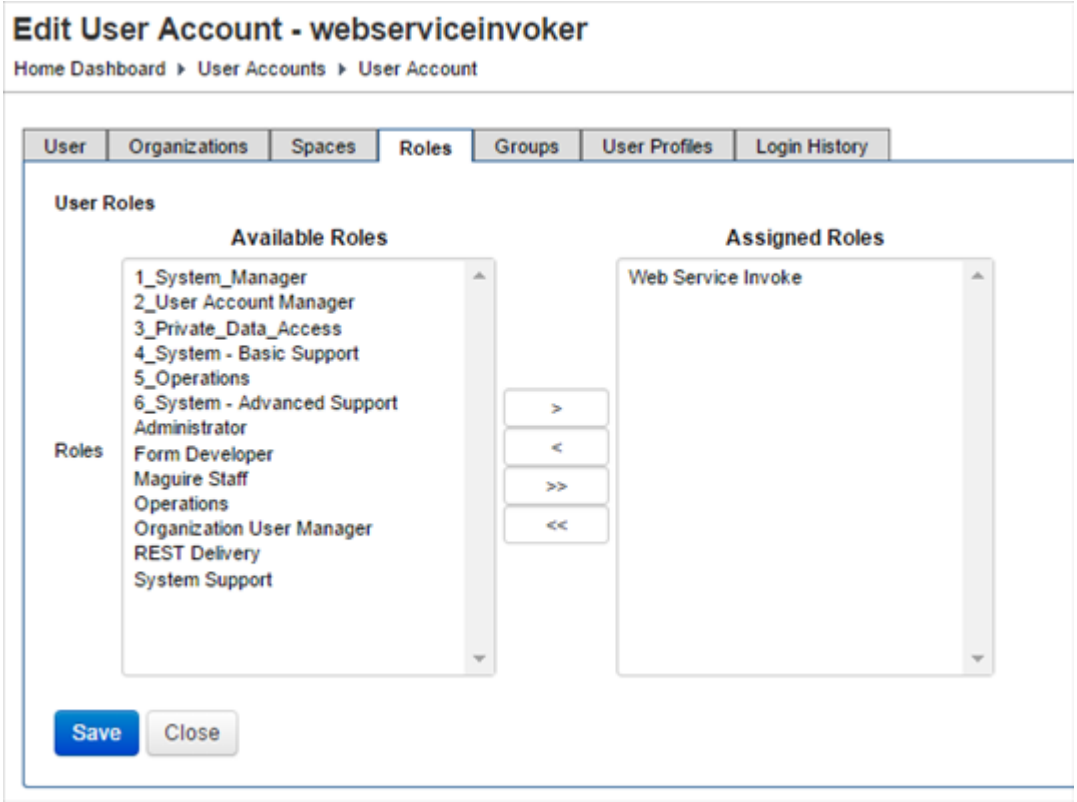
1. WSDL URL- https://{Server Name}/-manager/secure/services/TaskUserWebServiceV2?wsdl

   NOTE

   Do not use the unsecure URL https://{Server Name}/-manager/fcservices/TaskUserWebServiceV2?wsdl

2. Create a service user that has been assigned a Role that has the Web Service Invoke permission.

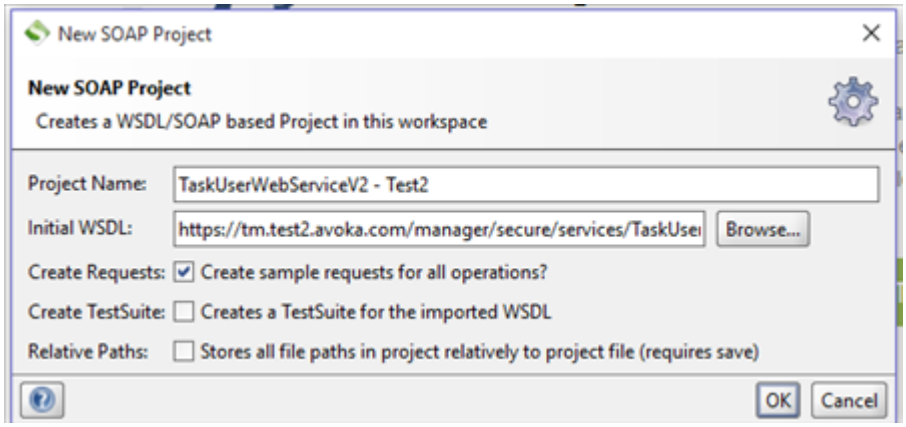

Web Service Invoke
Home Dashboard ▸ Roles ▸ Role
Role
Permissions
Members
Spaces & Modules
Transaction Manager
Available
Assigned
Permissions
Security Manager Remove
Security Manager View
Service Edit
Service Remove
Service View
Submission Data View
Submission Edit
Submission Save Edit
Submission View
Submission Xml Data Edit
Submission Xml Data View
System Info View
System Report Schedule Edit
System Report Schedule Remove
System Report Schedule Trigger
System Report Schedule View
Task Edit
Task View
Transaction Audit Log View
User Bulk Import
User Edit
User Profile Edit
User Profile Remove
User Property Edit
User Property Remove
User Property View
User Remove
User View
Admin Directory
Web Service Invoke
Verify Your Password
Your Password *
Save
Close


Edit User Account - webserviceinvoker
Home Dashboard ▸ User Accounts ▸ User Account
User
Organizations
Spaces
Roles
Groups
User Profiles
Login History
User Roles
Available Roles
Assigned Roles
Roles
1_System_Manager
2_User Account Manager
3_Private_Data_Access
4_System - Basic Support
5_Operations
6_System - Advanced Support
Administrator
Form Developer
Maguire Staff
Operations
Organization User Manager
REST Delivery
System Support
Web Service Invoke
Save
Close

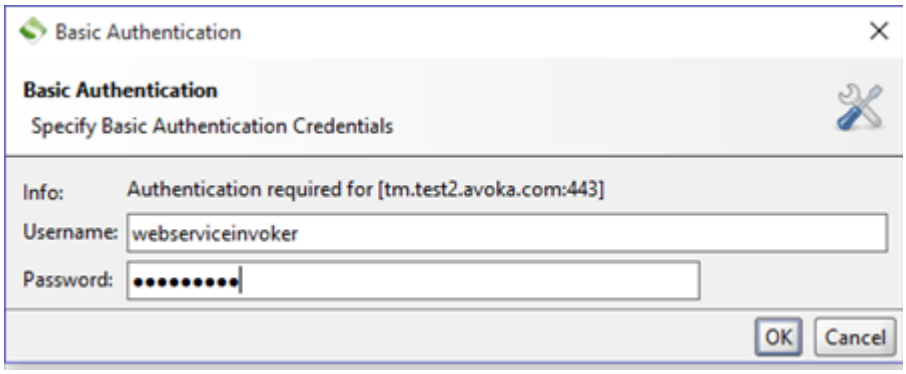
NOTE

Users with the Adminitrator role have the Web Service Invoke permission.

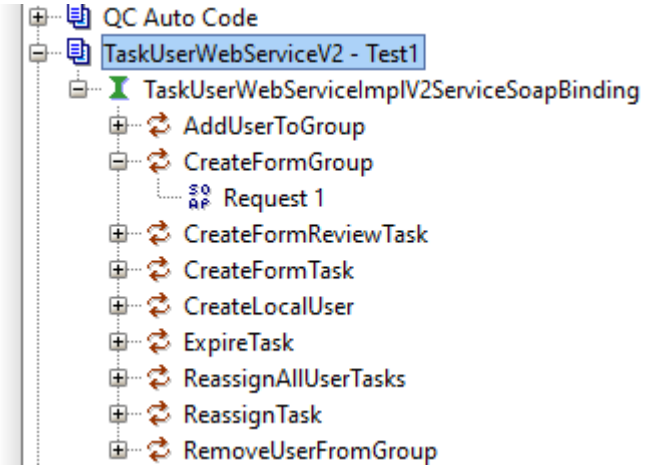
3. Create SOAP-UI project the wsdl in Step 1.



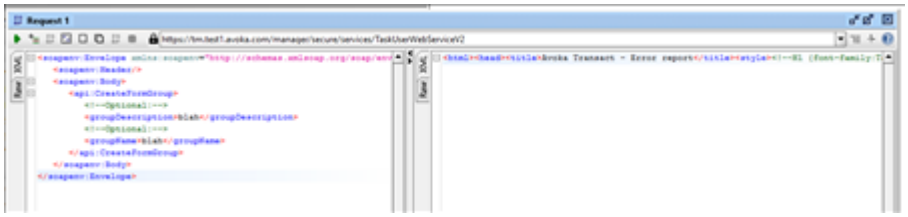
4. Soap-ui will prompt you with the Basic Authentication tab



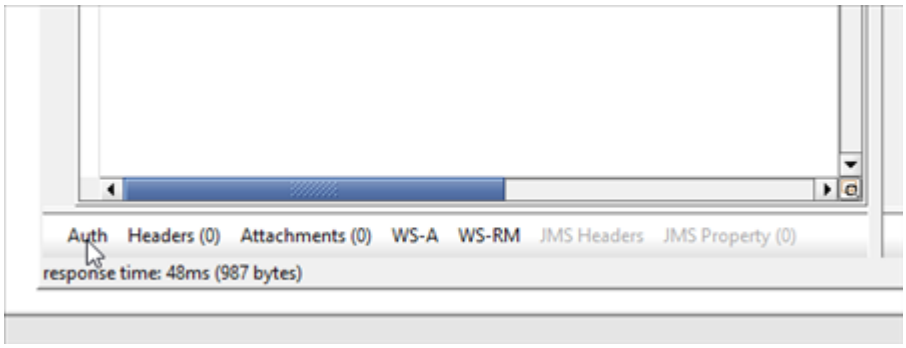
5. You should end up with the following



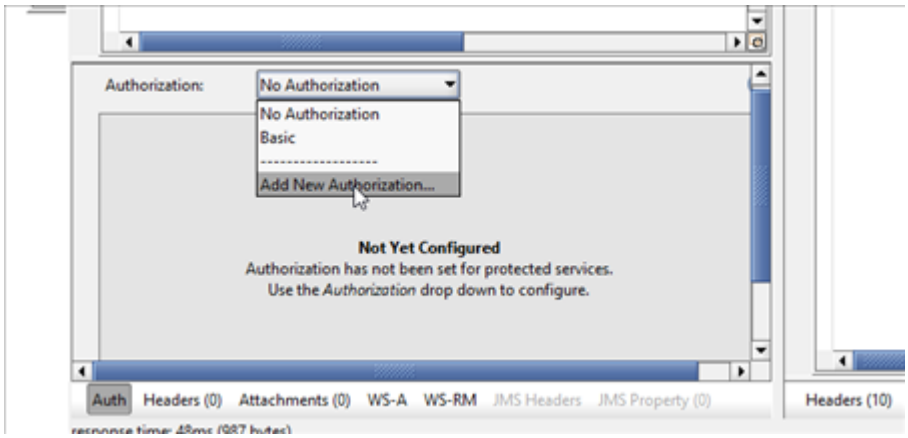
6. Try calling the web services with no authentication. Test Creating a Form Group. The response on the right tell you you are not authenticated.



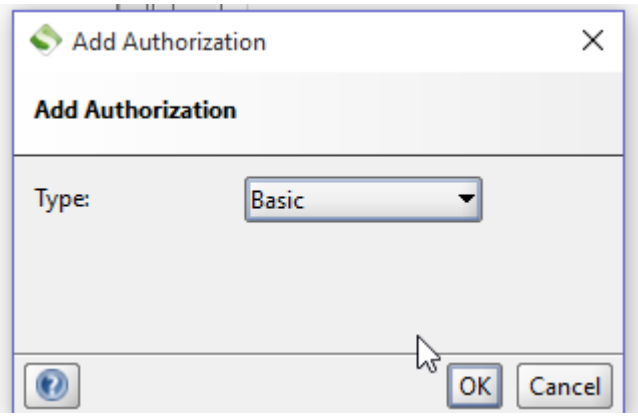
7. Click on the Auth buttong on the bottom left of the Request1 page to configure an Authentication.



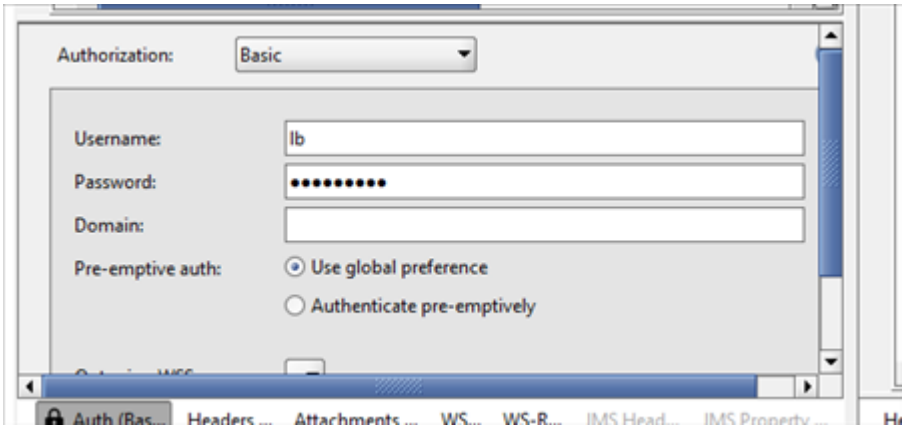
8. Select Add New Authentication.



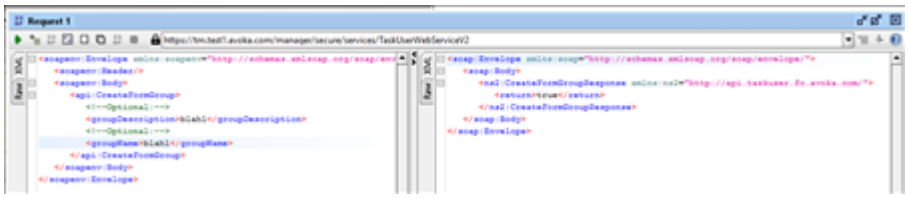
9. Select a type as Basic.



10. Specify a user name and a password.



11. Re-run the request.



Next, learn how to [view services](#).

Scheduled Jobs Overview

Journey Manager comes with the number of standard scheduled jobs that run regularly and perform routine operations, such as transaction processor schedule job that delivers transactions and purges data as per your environment data retention policies, generate receipts and temp file cleanup. A scheduled job invokes a schedule job service, which in turn calls other services that do the actual work. This increases flexibility by allowing you to swap out and reconfigure services. Unlike collaboration jobs, scheduled jobs are not necessarily related to one or more submissions. Instead of coming with a fixed set of hard-coded schedules, Manager allows you to manage existing scheduled jobs, modify their schedules, and control and monitor their execution.

Jobs have a schedule, which means they are automatically started by Manager at regular intervals.

One of the standard scheduled jobs is the collaboration job controller scheduled job, which processes standard collaboration jobs that are in the ready to be processed state.

In some cases, a particular solution may require custom scheduled jobs, so Manager allows you to create new scheduled jobs and configure them.

All scheduled jobs use a schedule that runs them automatically at regular intervals. You can pause and resume a scheduled job or trigger it manually.

Manager uses the Quartz Job Scheduling Framework to manage scheduled jobs. If Manager is configured to use multiple server nodes, the job scheduler runs only on a one node. You must log on to the Journey Manager Home Dashboard of that node to make changes to the scheduled jobs. Also, the Quartz configuration must be set to `org.quartz.jobStore.isClustered = “true”` to turn on clustering features.

Trigger Types

There are two trigger types: simple and cron. Let's look at each of them in more detail.

Simple

A simple trigger type defines how many times a scheduled job should be run and a time interval between each run. The interval can be set from 15 seconds up to 24 hours. This is sufficient for regular scheduled jobs but allows no control exactly when a scheduled job runs.

Cron

A cron[1] trigger type allows fine-grained control on when a scheduled job is executed. You must use cron expressions to specify both the interval between the runs and the time to execute the scheduled job. This is particularly useful for maintenance activities to perform overnight, outside the system peak usage times. This is particularly useful for maintenance jobs that need to run outside of peak times.

Next, learn about standard scheduled jobs.

[1]Cron is a time-based job scheduler in Unix-like computer operating systems. People who set up and maintain software environments use cron to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals.

Standard Scheduled Jobs

Manager comes with a number of preconfigured scheduled jobs that perform a range of important functions and shouldn't be paused or removed without good reasons.

Manager runs the following standard scheduled jobs:

Scheduled Job	Description
Collaboration Job Controller Job	Processes collaboration jobs by calling the job controller service associated with the collaboration jobs.
Database Integrity Checker Job	Pulls the database form center map and checks tables for integrity issues. | 23.04 This feature was introduced in 23.04
Data Keystore Rollover Job	Creates new keystores for organization, if automatic rollover is configured.
Data Retention Job	Calls the default Data Retention Management service, which is responsible for purging Manager data such as submissions and error logs. No other scheduled jobs will be run while the data retention job is executing to avoid data contention.
Delivery Escalation Job	Sends escalation emails for submissions delivered via secure email delivery.
Delivery Reminder Job	Sends reminder emails for submissions delivered via secure email delivery.
Email Queue Job	Sends emails placed into the email queue.
Event Outbox Processor Job	Retrieves unsent events from the DB table and sends them to Event Outbox Processor service for publishing into a Kafka topic. | 22.10 This feature was introduced in 22.10.
HTTP Invoker Job	Invokes a HTTP URL, which is configured as the context parameter of the job.

Payment Reminder Job	Loads the default payment reminder service and calls it. The service sends out a reminder email regarding incomplete payments For more information, see predefined groups to which emails are sent by default.
Run Reports Job	Loops through all report schedules setup in Manager. If it finds one that is due to be sent out, it will run the report, extract the result data and send out the report email.
Scheduled Service Job	Invokes a Scheduled Service Job. These services can be customized using a Groovy[1] script and are used to run custom logic on a schedule. The service the job will invoke can be selected when editing the job.
Security Policy Manager Job	Applies various security policies, such as: • User role and group expiry. • User password expiry. | 24.10 This feature was introduced in 24.10 • Removal of user accounts who never verified their email addresses.
Status Management Service	Changes form space's, form's, and form version's statuses that are scheduled using Notifications and Status Change. | 22.04 This feature was introduced in 22.04.
Submission Email Verification Reminder Job	Calls the default submission email verification service, which sends out reminders for submissions where the user must verify their email address.
System Monitoring Job	Monitors other important Manager system activities, such as the transaction processing and email queue. It also sends emails to the Receive System Alerts group if these scheduled jobs don't exist or have not been running as they should.
Template Version Deployment Job	Applies scheduled form version changes.

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

Transaction Licensing Job	Processes transaction data daily and generates licensing reports. NOTE You can't see the Transaction Licensing scheduled job in the list of standard scheduled jobs, but you can configure it and run manually from System > System Licensing.
Transaction History Publisher Job	Performs transaction history publishing.
Transaction Processing Job	Calls the default Transaction Processor service, which performs submission processing in the following order: • abandonment • receipts • delivery • task expiry • transaction history creation This scheduled job is typically configured to run every 5 minutes. This scheduled job is also used to perform manual data purging.
User Password Expiry Reminder	Sends reminder email for changing password when password is to expire within a configurable number of days, which you can modify in your Local Security Manager. | 24.10 This feature was introduced in 24.10
Virus Scan Job	Loads a list of all attachments that need to be virus scanned and calls the default virus scan service for each attachment, setting the attachment's virus scan status appropriately.

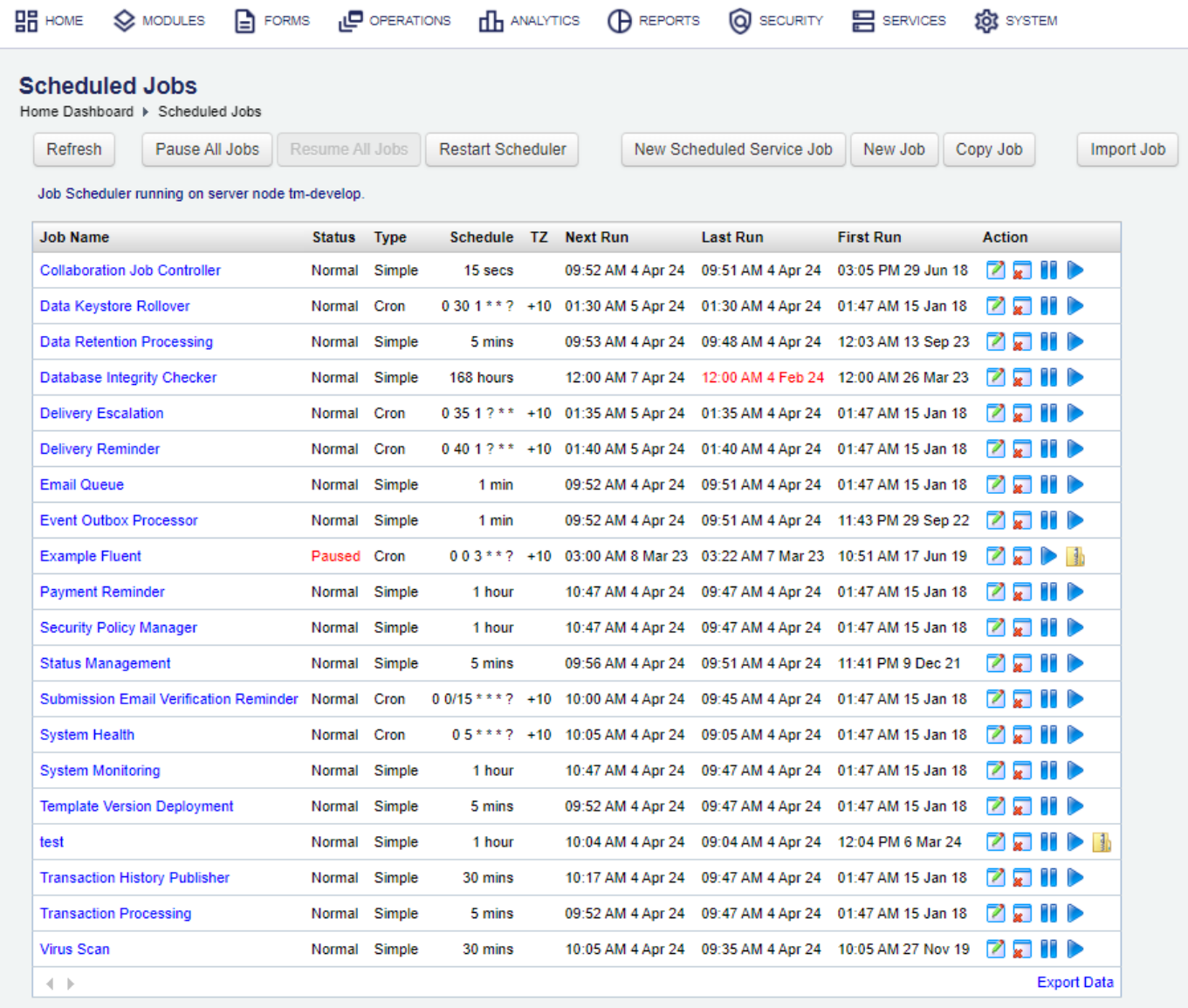
Next, learn how to manage scheduled jobs.

Manage Scheduled Jobs

Manager allows you to view and manage scheduled jobs, which can be standard scheduled job as well as custom ones that you create.

To view scheduled jobs, select System > Scheduled Jobs.

Scheduled jobs are shown page by page, with the maximum number of records per page configured in the users preferences.



Job Name	Status	Type	Schedule	TZ	Next Run	Last Run	First Run	Action
Collaboration Job Controller	Normal	Simple	15 secs		09:52 AM 4 Apr 24	09:51 AM 4 Apr 24	03:05 PM 29 Jun 18	
Data Keystore Rollover	Normal	Cron	0 30 1 * * ?	+10	01:30 AM 5 Apr 24	01:30 AM 4 Apr 24	01:47 AM 15 Jan 18	
Data Retention Processing	Normal	Simple	5 mins		09:53 AM 4 Apr 24	09:48 AM 4 Apr 24	12:03 AM 13 Sep 23	
Database Integrity Checker	Normal	Simple	168 hours		12:00 AM 7 Apr 24	12:00 AM 4 Feb 24	12:00 AM 26 Mar 23	
Delivery Escalation	Normal	Cron	0 35 1 ? * *	+10	01:35 AM 5 Apr 24	01:35 AM 4 Apr 24	01:47 AM 15 Jan 18	
Delivery Reminder	Normal	Cron	0 40 1 ? * *	+10	01:40 AM 5 Apr 24	01:40 AM 4 Apr 24	01:47 AM 15 Jan 18	
Email Queue	Normal	Simple	1 min		09:52 AM 4 Apr 24	09:51 AM 4 Apr 24	01:47 AM 15 Jan 18	
Event Outbox Processor	Normal	Simple	1 min		09:52 AM 4 Apr 24	09:51 AM 4 Apr 24	11:43 PM 29 Sep 22	
Example Fluent	Paused	Cron	0 0 3 * * ?	+10	03:00 AM 8 Mar 23	03:22 AM 7 Mar 23	10:51 AM 17 Jun 19	
Payment Reminder	Normal	Simple	1 hour		10:47 AM 4 Apr 24	09:47 AM 4 Apr 24	01:47 AM 15 Jan 18	
Security Policy Manager	Normal	Simple	1 hour		10:47 AM 4 Apr 24	09:47 AM 4 Apr 24	01:47 AM 15 Jan 18	
Status Management	Normal	Simple	5 mins		09:56 AM 4 Apr 24	09:51 AM 4 Apr 24	11:41 PM 9 Dec 21	
Submission Email Verification Reminder	Normal	Cron	0 0/15 * * * ?	+10	10:00 AM 4 Apr 24	09:45 AM 4 Apr 24	01:47 AM 15 Jan 18	
System Health	Normal	Cron	0 5 * * * ?	+10	10:05 AM 4 Apr 24	09:05 AM 4 Apr 24	01:47 AM 15 Jan 18	
System Monitoring	Normal	Simple	1 hour		10:47 AM 4 Apr 24	09:47 AM 4 Apr 24	01:47 AM 15 Jan 18	
Template Version Deployment	Normal	Simple	5 mins		09:52 AM 4 Apr 24	09:47 AM 4 Apr 24	01:47 AM 15 Jan 18	
test	Normal	Simple	1 hour		10:04 AM 4 Apr 24	09:04 AM 4 Apr 24	12:04 PM 6 Mar 24	
Transaction History Publisher	Normal	Simple	30 mins		10:17 AM 4 Apr 24	09:47 AM 4 Apr 24	01:47 AM 15 Jan 18	
Transaction Processing	Normal	Simple	5 mins		09:52 AM 4 Apr 24	09:47 AM 4 Apr 24	01:47 AM 15 Jan 18	
Virus Scan	Normal	Simple	30 mins		10:05 AM 4 Apr 24	09:35 AM 4 Apr 24	10:05 AM 27 Nov 19	

The list displays the following details:

- Job Name: a name of a A scheduled job.
- Status: a status of the scheduled job that can be Normal or Paused.

- Type: a type of the scheduled job that can be Simple or Cron.
- Schedule: time or interval, depending on a type, when the scheduled job is supposed to run.
- TZ: the time zone. Only cron scheduled jobs are associated with a timezone, which generally corresponds to the timezone used n your Manager server. Whenever the server is restarted, it adjusts the timezones for all cron scheduled jobs, if the system time has changed.

  > NOTE
  > If you see unexpected timezone values, check the system time settings of the server that Quartz is running on and restart Manager.

- Next Run: the time that this scheduled job will run next.
- Last Run: the time that this scheduled job was run last .
- First Run: the time that this scheduled job was run first .
- Action: an operation you can apply to a selected scheduled job, which can be one of the following:
  - Edit
  - Delete
  - Pause
  - Trigger
- A server node name is displayed above the main table. In this example, it is the `tm-develop` server node.

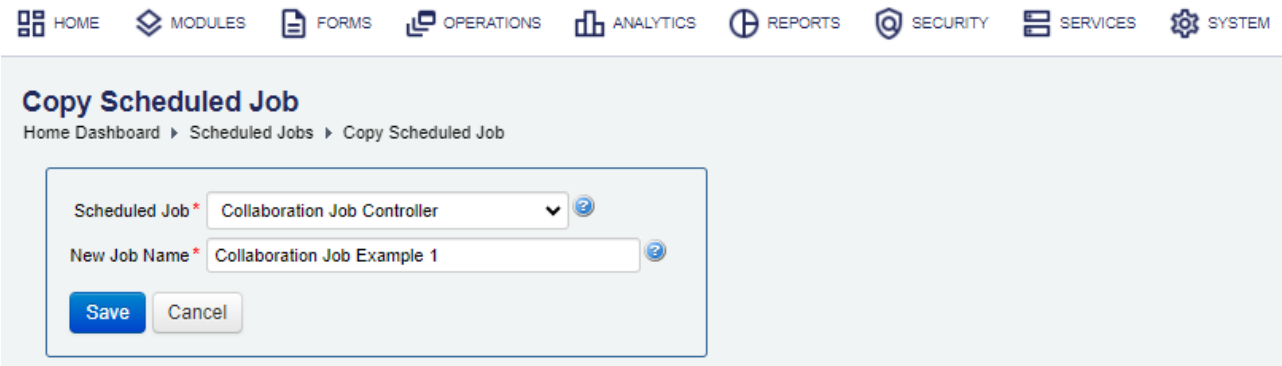
  > NOTE
  > If Manager is configured to use multiple server nodes, the job scheduler runs only on a one node. You must log on to the Journey Manager Home Dashboard of that node to make changes to the scheduled jobs.

You can also perform the following actions:

- Click Refresh to refresh the list of scheduled jobs.
- Click Pause All Jobs to temporarily suspend all scheduled jobs until you are ready to start them again.
- Click Resume All Jobs to resume all scheduled jobs.
- Click Restart Scheduler to restart the scheduler. If there are issues when running scheduled jobs, one option is to pause or trigger individual scheduled jobs for debugging pur-

poses. It is also possible to restart the entire job scheduler to reset it without restarting Manager.

- Click New Scheduled Service Job or New Job to create a new scheduled job.
- Click Copy Job to create a new scheduled job as a copy of the existing scheduled job.



- Click Edit Job to view or edit a scheduled job. An administrator would typically use this option to change the time interval or cron expression of a scheduled job.
- Click Delete Job to remove a scheduled job that is no longer needed.

> NOTE
>
> You can also disable a scheduled job by setting an end date after which will no longer be run by the scheduler.

- Click Pause Job to temporarily suspend a scheduled job until you are ready to start it again. You may need to pause it to make sure no transactions or emails queue being processed. The job status changes to Paused. Manager won't run the scheduled job until you resume it. Click Resume Job to start the job again.
- Click Trigger Job to run a job immediately. This is handy when the time for a scheduled job's next invocation has not come yet but you want it to run anyway. The job status changes to Running.
- Click Interrupt Running Job to stop the job. You may want to terminate a running scheduled job because of bugs in the job or data that the job can't handle. The job status changes to Normal.
- Click Export Archive to export a scheduled job configuration as a ZIP file. | 24.04 This feature was introduced in 24.04

> NOTE
>
> This action is only available for scheduled jobs that execute a service of the Scheduled Service Job type.

- Click Import Job to import a scheduled job as a ZIP file. | 24.04 This feature was introduced in 24.04

Next, learn how to create a scheduled job.

Create a Scheduled Job

Manager comes with several default scheduled job types , but you can also create a new scheduled job to suit your needs.

Each scheduled job requires a service that will be executed to perform a job, therefore you may need to create a scheduled service first if it doesn't exist.
To create a new scheduled service:

1. Select Services > All Services. and click New.
2. Select Scheduled Service from the Service Type dropdown list.
3. Select a service template from the Service Template dropdown list, which can be:
   - Fluent[1] Scheduled Service - this is the preferred service template.
   - Groovy[2] Scheduled Service
   - Scheduled Data Extract Service
4. Update the service name in the New Name field.
5. Update the service version in the Version Number field.
6. Select an optional organization from the Organization dropdown list.

   NOTE

   If organization is blank, the service will be global, otherwise it will be available to the selected organization only.

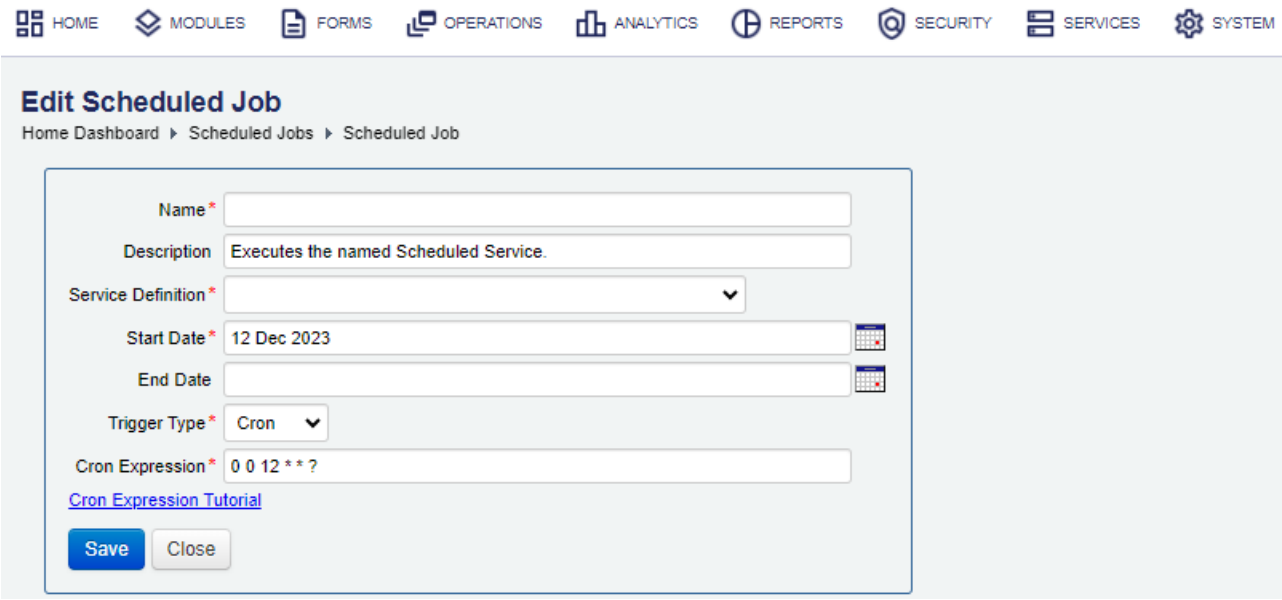
7. Click Save to update the changes.

To create a new scheduled job:

---

[1]Fluent is a programming style that can be applied to a wide range of programming languages. Across the Temenos Journey Manager platform, fluent is commonly associated with the Fluent SDK. The Fluent SDK is intended for use in developer integrated developer environment's (IDE) and contains libraries and project files for all the different services it supports.
[2]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

1. Select System > Scheduled Jobs and Click New Scheduled Service Job.



2. Enter the unique name of the scheduled job into the Name field.
3. Provide a description of the new scheduled job.
4. Select a service from the Service Definition dropdown list, which must be of the Scheduled Service type. If this service doesn't exist, create a new service.
5. Specify the Start Date and End Date.
6. Select a trigger type from the Trigger Type dropdown list, which can be Simple or Cron.
7. Select the repetition period from the Repeat dropdown list and repeat interval from the Repeat Interval for the simple trigger type.
8. Provide a cron job regular expression in the Cron Expression field, which the scheduler will use to know when and how often to invoke this job. For example, `0 0 1 * * ?` will cause the job to run every day at 1 am. However, much more sophisticated schedules are possible.
9. Click Save to update the changes.

After scheduled jobs are created, there are only a few configuration changes you can do to maintain them. Most frequently, you will need to update the repeat interval or cron expression of a scheduled job.

Next, learn how to configure a scheduled job.

Configure a Scheduled Job

Manager allows you to configure scheduled jobs, which can be the default ones or ones you have created to suit your needs. After scheduled jobs are created, there are only a few configuration changes you can do to maintain them. Most frequently, you will need to update the repeat interval or cron expression of a scheduled job.

Each scheduled job requires a service, which is executed to perform a job, so you may want to update it as well.

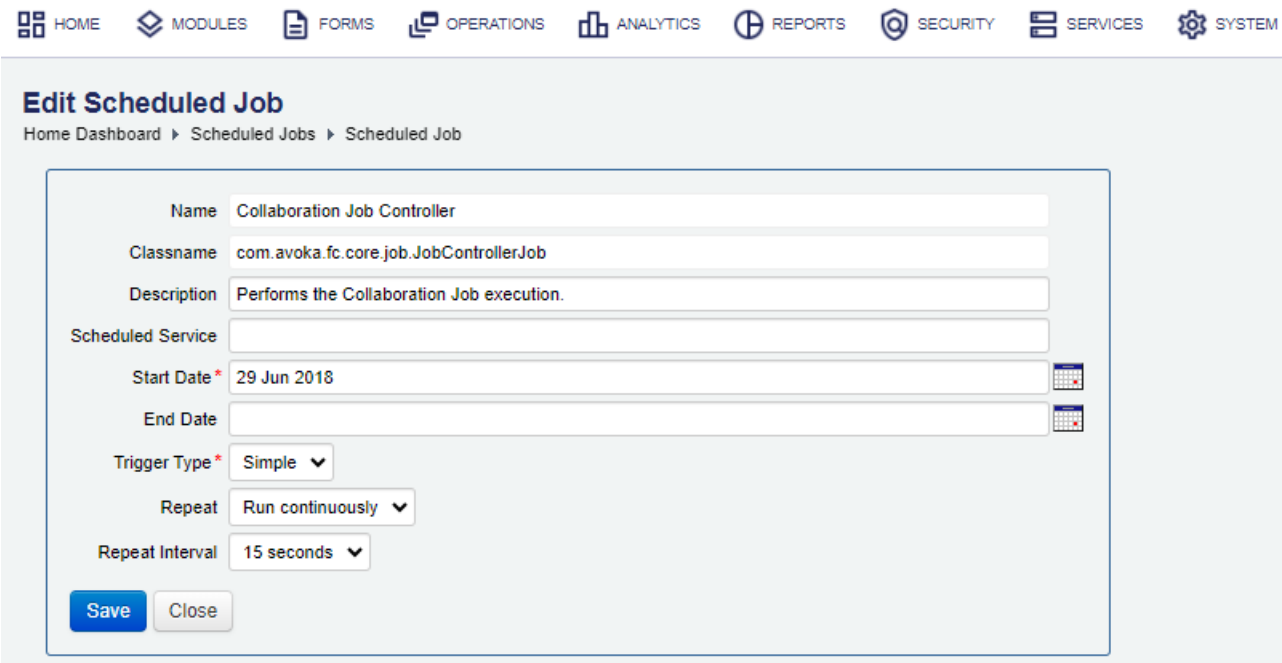
You can configure a scheduled job using one of following options:

- UI
- Configuration Service

Configure Using UI

To configure a scheduled job:

1. Select System > Scheduled Jobs
2. Locate a scheduled job and click Edit.



3. Update the Start Date or End Date.
4. Change a trigger type from the Trigger Type dropdown list, which can be one of the following:

    - Simple: the default option.
    - Cron:
5. Select a repeat from the Repeat dropdown list, which can be one of the following:
    - Run Continuously - This is the default option.
    - Repeat 1 - Repeats once
    - Repeat 2
    - Repeat 3
    - Repeat 5
    - Repeat 10
    - Repeat 20
6. Select a repeat interval from the Repeat Interval dropdown list, which can be one of the following:
    - 15 seconds
    - 30 seconds
    - 1 minute
    - 2 minutes
    - 5 minutes
    - 10 minutes
    - 15 minutes
    - 30 minutes
    - 1 hour - This is the default option
    - 2 hours
    - 3 hours
    - 6 hours
    - 12 hours
    - 24 hours
7. Update a cron job regular expression in the Cron Expression field.
8. Click Save to update the changes.

Configure Using a Configuration Service

You can also update one or several scheduled jobs using the Configuration Service. To do so, you need to add scheduled jobs' configuration to a JSON file, which must be based on a template version 2.1 and later, so it includes the following section | 24.04 This feature was introduced in 24.04:

```
"scheduledJobs": [
                {
                    "jobName": "Scheduled Job Example Cron",
                    "jobDescription": "This is a Cron scheduled job",
                    "serviceClientCode": "",
                    "serviceName": "Test Scheduled Service",
                    "serviceVersion": "0.1.0",
                    "startDate": "2024-05-06",
                    "endDate": "2024-05-15",
                    "triggerType": "Cron",
                    "cronExpression": "0 30 1 * * ?"
                },
                {
                    "jobName": "Scheduled Job Example Simple",
                    "jobDescription": "This is a sample scheduled job runs continouslly
every 15 seconds",
                    "serviceClientCode": "",
                    "serviceName": "Test Scheduled Service",
                    "serviceVersion": "0.1.0",
                    "startDate": "2024-05-06",
                    "endDate": "2024-05-15",
                    "triggerType": "Simple",
                    "repeat": "-1",
                    "repeatInterval": "15000"
                }
            ]
```

Next, learn how to export scheduled jobs.

Export Scheduled Jobs

Manager allows you to export scheduled jobs, so you can then import them.

> NOTE
>
> You can only export scheduled jobs, which execute services of the Scheduled Service Job type. You can't export scheduled jobs, which execute Core Services.

Use this functionality to:

- Roll back one or scheduled jobs.
- Re-create one or scheduled jobs in another Manager.
- Promote scheduled jobs to another environment.

To export a scheduled job:

1. Select System > Scheduled Jobs.
2. Click Export Archive to export a selected scheduled job as an archive ZIP file on your local PC. The ZIP file contains a `scheduled-job-config.xml` file with a scheduled job's definition. The example is shown below:

```
<?xml version="1.0" encoding="UTF-8"?><ScheduledServiceJobs databaseVersion="2085"
environmentName="my env" exportDate="2024-04-03" revisionNumber="00d9231" sfmVer-
sion="24.4.0-SNAPSHOT">
  <ScheduledServiceJob>
    <scheduledServiceJobName>Example Fluent</scheduledServiceJobName>
    <description>Executes the named Scheduled Service.</description>
    <startDate>1560732700000</startDate>
    <endDate/>
    <triggerType>Cron</triggerType>
    <cronExpression>0 0 3 * * ?</cronExpression>
    <serviceDefinition>
      <serviceName>Fluent Scheduled Service</serviceName>
      <serviceClientCode>maguire</serviceClientCode>
      <serviceVersion>0.1.0</serviceVersion>
    </serviceDefinition>
  </ScheduledServiceJob>
</ScheduledServiceJobs>
```

3. Click Export Data to export the displayed data as a Microsoft Excel (XLS) file. The XLS file has the same column names and described in view scheduled jobs.

Next, learn how to import scheduled jobs.

Import Scheduled Jobs

Manager allows you to import scheduled jobs, which were previously exported.

> NOTE
>
> You can only import scheduled jobs, which execute services of the Scheduled Service Job type. You can't import scheduled jobs, which execute Core Services.

Use this functionality to:

- Roll back one or scheduled jobs.
- Re-create one or scheduled jobs in another Manager.
- Promote scheduled jobs to another environment.

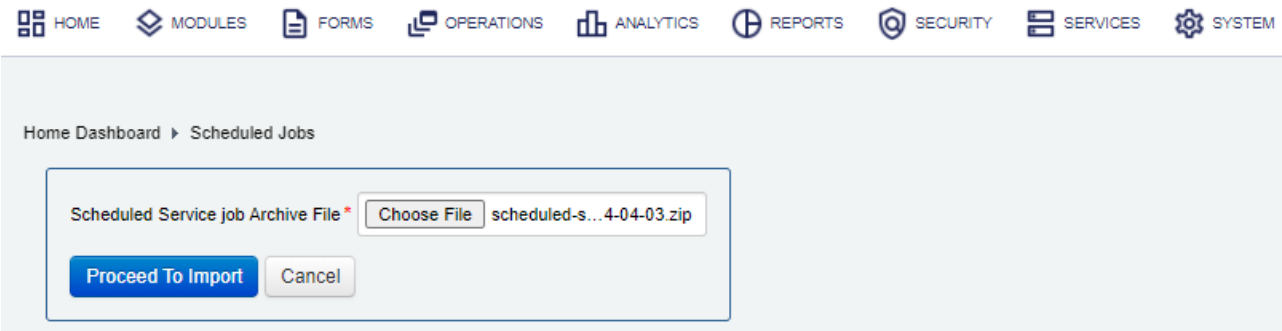
You can import a scheduled job using one of following options:

- UI
- SDK
- API

Import Using UI

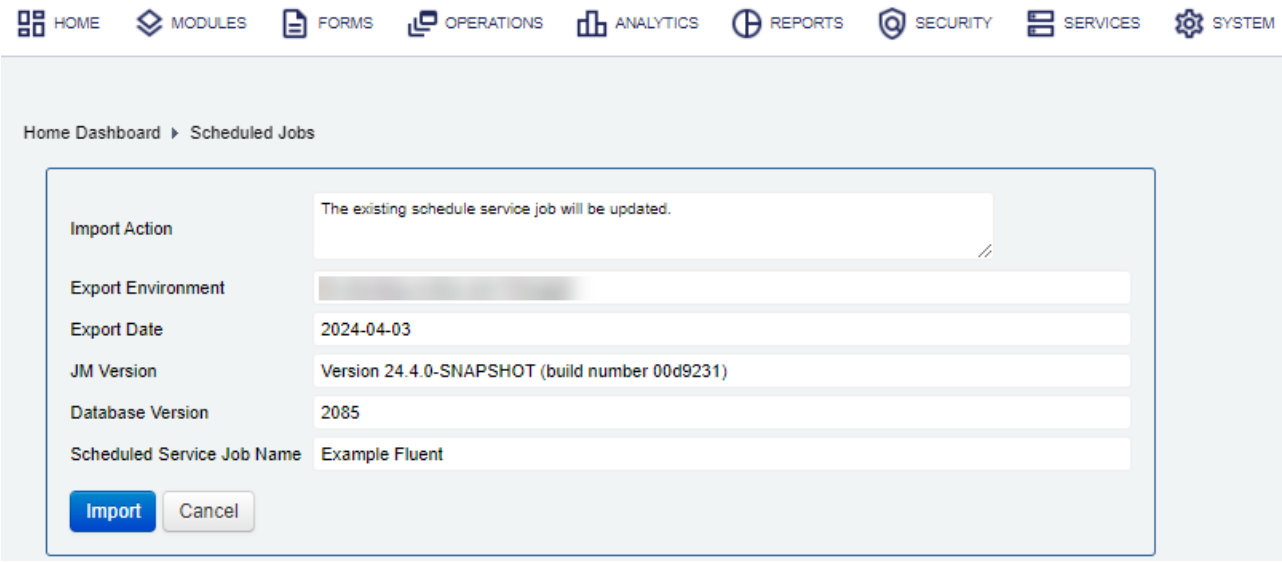
To import a scheduled job:

1. Select System > Scheduled Jobs.
2. Click Import Job.
3. Click Choose File and choose a scheduled job archive ZIP file.



4. Click Proceed To import.

5. Review the Import Action as shown below:



> NOTE
>
> There are no import options to select, so every import will override the existing scheduled job.

6. Click Import to complete the process.
7. Click Close and check the list of services to verify the imported services have been added.

> NOTE
>
> You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

Import Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:archive-deploy goal to import a scheduled job. For example, set `archive.file=my-scheduled-job.zip` in the `build.properties` file and run `mvn initialize tm-sdk:archive-deploy` to import a scheduled job.

Import Using API

The Journey Manager REST API provides the endpoint URL to import a scheduled job.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s)://<JM_SERVER>/manager/secure/rest/archive/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as Base64[1].

`--form ‘archiveFile=@“/Users/my-scheduled-job.zip”’` - a path to an archive file to be imported.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

Example

```
curl --location --request PUT ‘https://<JM_SERVER>/manager/secure/rest/archive/v1/’
\
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
--form ‘archiveFile=@“/Users/my-scheduled-job.zip”’
```

NOTE

You can also update an existing scheduled job using the Configuration Service API.

Next, learn how to view scheduled job history.

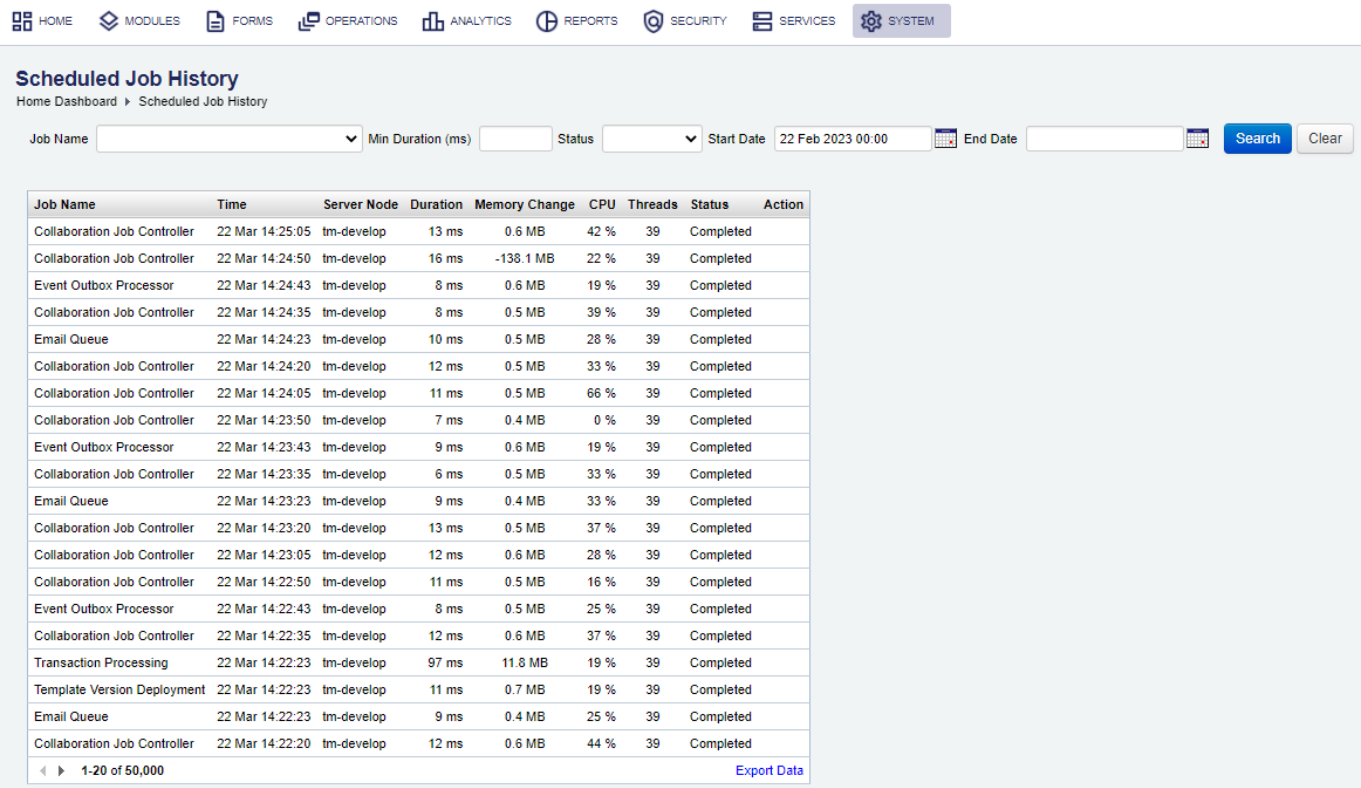
[1]Base64 is a binary to a text encoding scheme that represents binary data in an American Standard Code for Information Interchange (ASCII) string format.

View Scheduled Job History

Every time a scheduled job runs, Manager logs a history event recording which job ran, the status of the job and some performance metrics. Manager allows you to view and search scheduled jobs history.

Because Manager runs many scheduled jobs as part of normal operations, the scheduled job history can contain a large number of entries. Manager purges the scheduled job history as part of its data retention policies.

To view scheduled job history, select System > Scheduled Job History.



Job Name	Time	Server Node	Duration	Memory Change	CPU	Threads	Status	Action
Collaboration Job Controller	22 Mar 14:25:05	tm-develop	13 ms	0.6 MB	42 %	39	Completed	
Collaboration Job Controller	22 Mar 14:24:50	tm-develop	16 ms	-138.1 MB	22 %	39	Completed	
Event Outbox Processor	22 Mar 14:24:43	tm-develop	8 ms	0.6 MB	19 %	39	Completed	
Collaboration Job Controller	22 Mar 14:24:35	tm-develop	8 ms	0.5 MB	39 %	39	Completed	
Email Queue	22 Mar 14:24:23	tm-develop	10 ms	0.5 MB	28 %	39	Completed	
Collaboration Job Controller	22 Mar 14:24:20	tm-develop	12 ms	0.5 MB	33 %	39	Completed	
Collaboration Job Controller	22 Mar 14:24:05	tm-develop	11 ms	0.5 MB	66 %	39	Completed	
Collaboration Job Controller	22 Mar 14:23:50	tm-develop	7 ms	0.4 MB	0 %	39	Completed	
Event Outbox Processor	22 Mar 14:23:43	tm-develop	9 ms	0.6 MB	19 %	39	Completed	
Collaboration Job Controller	22 Mar 14:23:35	tm-develop	6 ms	0.5 MB	33 %	39	Completed	
Email Queue	22 Mar 14:23:23	tm-develop	9 ms	0.4 MB	33 %	39	Completed	
Collaboration Job Controller	22 Mar 14:23:20	tm-develop	13 ms	0.5 MB	37 %	39	Completed	
Collaboration Job Controller	22 Mar 14:23:05	tm-develop	12 ms	0.6 MB	28 %	39	Completed	
Collaboration Job Controller	22 Mar 14:22:50	tm-develop	11 ms	0.5 MB	16 %	39	Completed	
Event Outbox Processor	22 Mar 14:22:43	tm-develop	8 ms	0.5 MB	25 %	39	Completed	
Collaboration Job Controller	22 Mar 14:22:35	tm-develop	12 ms	0.6 MB	37 %	39	Completed	
Transaction Processing	22 Mar 14:22:23	tm-develop	97 ms	11.8 MB	19 %	39	Completed	
Template Version Deployment	22 Mar 14:22:23	tm-develop	11 ms	0.7 MB	19 %	39	Completed	
Email Queue	22 Mar 14:22:23	tm-develop	9 ms	0.4 MB	25 %	39	Completed	
Collaboration Job Controller	22 Mar 14:22:20	tm-develop	12 ms	0.6 MB	44 %	39	Completed	

The list displays the following details:

- Job Name: a name of a scheduled job, which can be one of the standard jobs.
- Time: when the scheduled jobwas run.
- Server Node: a server node of a Manager environment where the scheduled job was run on.
- Duration: how long it took to run the scheduled job.
- Memory Change: the amount of the server's memory used to run the scheduled job.
- CPU: percentage of the server's CPU used to run the scheduled job.
- Threads: number of threads allocated to run the scheduled job.

- Status: the status of this scheduled job, which can be one of the following:
  - Completed: a scheduled job has completed.
  - Error: a scheduled job has failed.

To filter or search the scheduled jobs by one or more criteria, specify the following settings and click Search:

- Job name or part of the name.
- Min Duration (ms) it took the server to run a job.
- Status of the job, which can be one of the following:
  - Completed: a scheduled job has completed.
  - Error: a scheduled job has failed.
- Start Date and End Date of the scheduled job.

Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Click Error Log to view the detailed error information.

> NOTE
>
> The Error Log action is only available if a scheduled job has the status as Error.

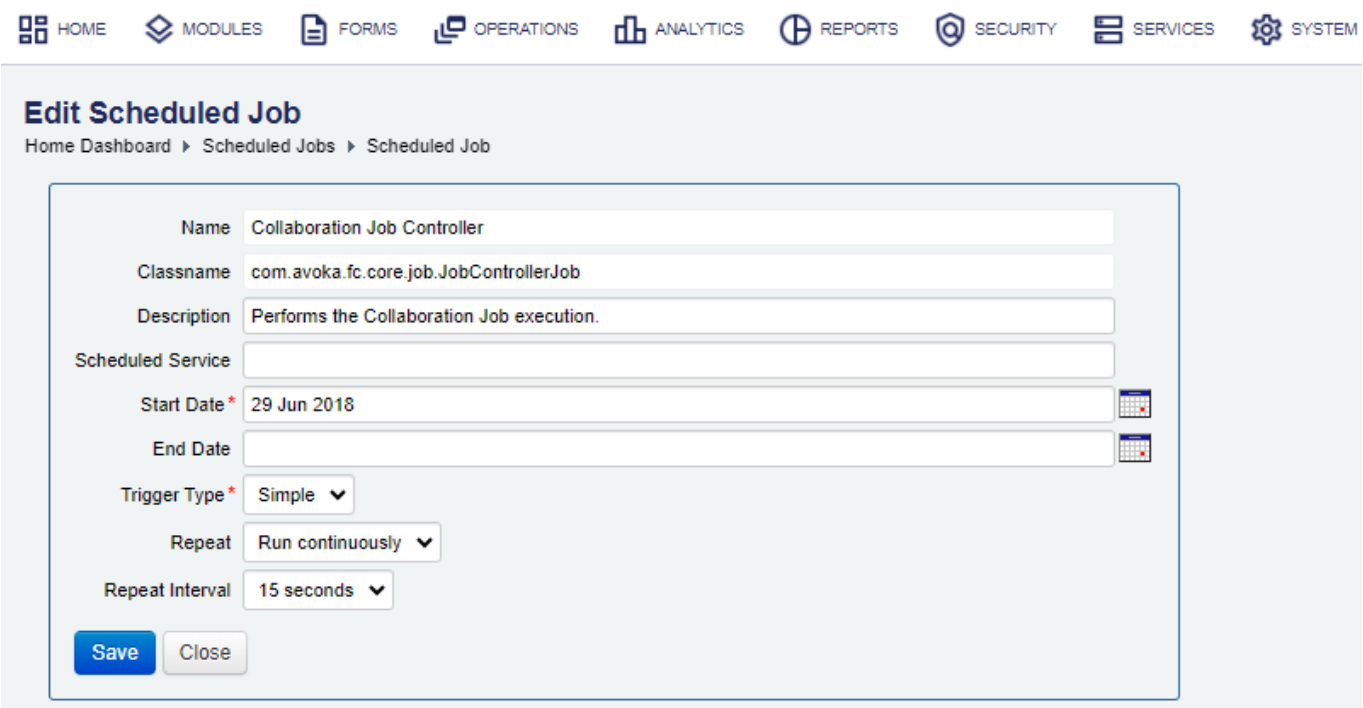
Next, learn how to view scheduled job.

Collaboration Job Controller

The collaboration job controller is the scheduled job service, which runs as a single thread[1] to processes collaboration jobs. First, it selects a list of collaboration jobs that are available for processing.Then, it goes through each collaboration job in the list and calls an associated job controller service. The job controller service definition is used to process collaboration job's actions specified in each step and go (route) to the next step, until an endpoint is reached. Each collaboration job runs until a wait action is encountered, or an error is thrown during the collaboration job action or the collaboration job successfully completes. The scheduled job terminates when there are no more collaboration jobs in the list to process.

The scheduler only starts one collaboration job thread at a time. A large backlog of work may take more than a minute for the thread to complete. A new collaboration job thread won't be started until the previous one has completed.

The default collaboration job controller configuration is shown below.



The default interval is 1 minute, which is the value we recommend for production environments. However, it is useful to set this interval to 15 seconds in on a development environment to improve development and testing turnaround.

---

[1]A thread of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler, which is typically a part of the operating system.

Next, learn how to manage scheduled jobs

Database Integrity Checker Job

Manager provides the Database Integrity Checker job that periodically executes a Database Integrity Checker service to check the integrity of a database used by the server. This scheduled job runs weekly.

The Database Integrity check consists of the following validation rules:

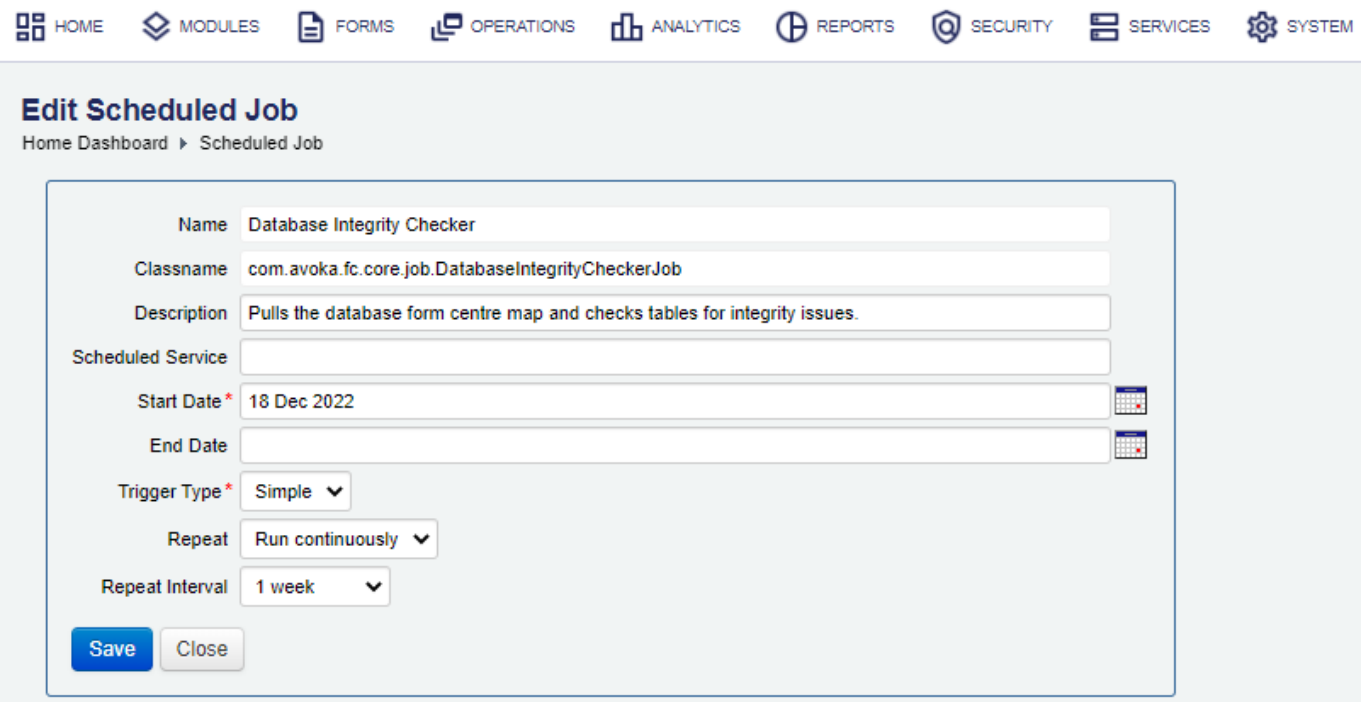
1. Check tables for orphaned records based on entity relations.
2. Remove invalid orphaned records, if the Delete Orphaned Records checkbox is selected in the Database Integrity Checker service. | 24.10This feature was updated in 24.10
3. Check tables' indexes.
4. Check services' default values.
5. Check forms' delivery method configuration.

For each validation rule, the Database Integrity Checker job generates a report, including the following information:

1. An entity, target entity and the number of invalid records in a table.
2. An entity name and columns which we expect to have an index on.

Then, the Database Integrity Checker job sends the report as an email to the Receive System Alerts - Database Integrity alert group. You need to assign this group to a user to receive the Database Integrity notifications.

The default Database Integrity Checker job configuration is shown below.

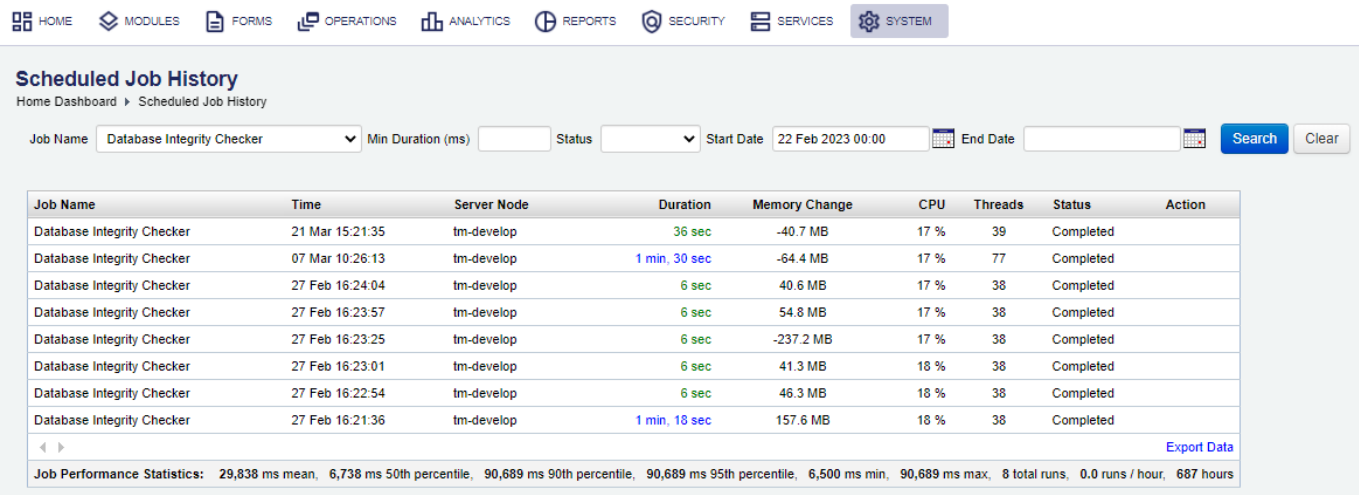


The Database Integrity Checker job adds an entry of the Database Integrity type to the Event Log. This allows you to find all Database Integrity issues easily.

The Event Log entry example is illustrated below: | 24.10This feature was updated in 24.10

```
Database Integrity Checker Job found 2 invalid/orphaned groups:
1 record(s) for Entity: submission_property_search Field: submission_property_oid Tar-
getEntity: submission_property
2 record(s) for Entity: submission_property Field: submission_oid TargetEntity: sub-
mission
```

You can also view the history of scheduled jobs and use Database Integrity Checker as a Job Name to narrow down your search, as shown below.



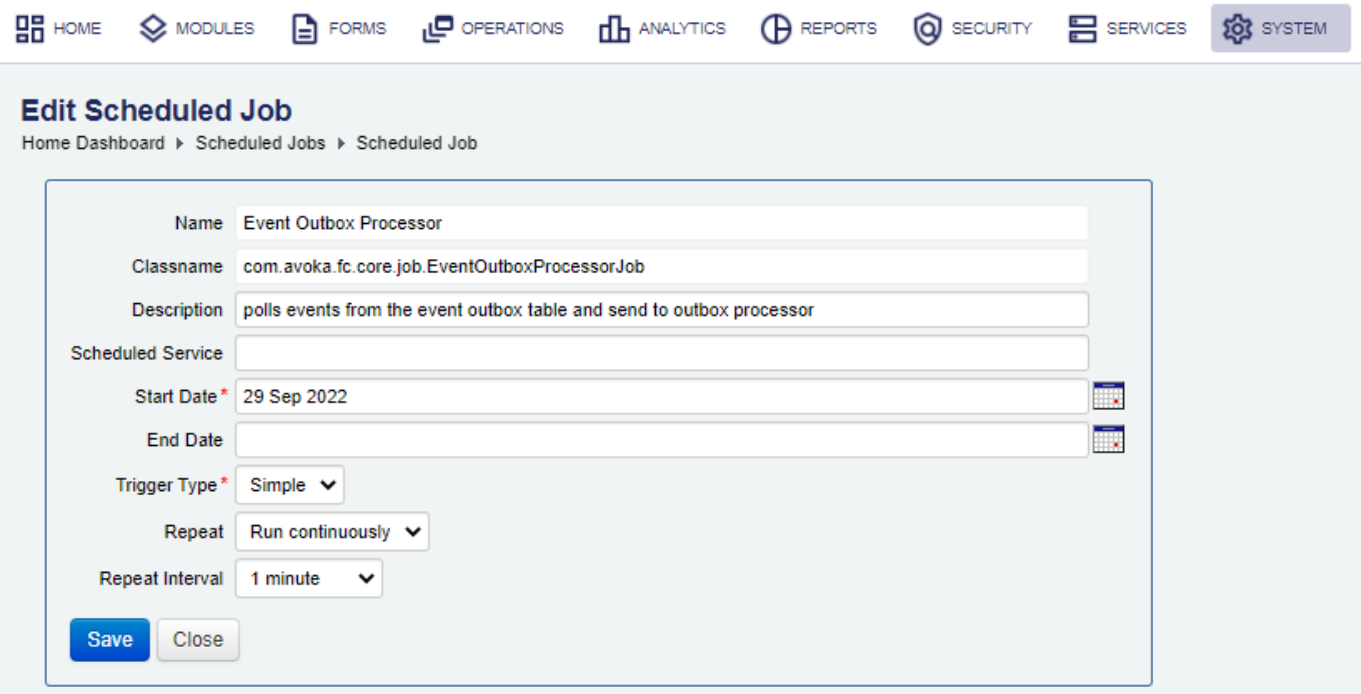
Job Name	Time	Server Node	Duration	Memory Change	CPU	Threads	Status	Action
Database Integrity Checker	21 Mar 15:21:35	tm-develop	36 sec	-40.7 MB	17 %	39	Completed	
Database Integrity Checker	07 Mar 10:26:13	tm-develop	1 min, 30 sec	-64.4 MB	17 %	77	Completed	
Database Integrity Checker	27 Feb 16:24:04	tm-develop	6 sec	40.6 MB	17 %	38	Completed	
Database Integrity Checker	27 Feb 16:23:57	tm-develop	6 sec	54.8 MB	17 %	38	Completed	
Database Integrity Checker	27 Feb 16:23:25	tm-develop	6 sec	-237.2 MB	17 %	38	Completed	
Database Integrity Checker	27 Feb 16:23:01	tm-develop	6 sec	41.3 MB	18 %	38	Completed	
Database Integrity Checker	27 Feb 16:22:54	tm-develop	6 sec	46.3 MB	18 %	38	Completed	
Database Integrity Checker	27 Feb 16:21:36	tm-develop	1 min, 18 sec	157.6 MB	18 %	38	Completed	

Next, learn how to manage scheduled jobs

Event Outbox Processor Job

Manager provides the Event Outbox Processor job that retrieves unsent events from the DB table and sends them to Event Outbox Processor service for publishing into a Kafka topic. This scheduled job runs every 1 min.

The default Event Outbox Processor job configuration is shown below.



Next, learn how to manage scheduled jobs

Receipts Overview

Journey Manager provides a form user with a receipt that is read-only statement acknowledging the data submitted by the user. Receipts are an essential part of the form submission process. Receipts are PDF forms that are generated, based on form content, for all completed transactions. Each receipt contains a unique receipt number that you can use to identify the corresponding transaction. Maestro allows you to create custom receipts for Maestro forms to optimize them for printing or to implement specific customer requirements. In this case, Manager uses the receipt design to issue the PDF receipt.

> NOTE
>
> The Maestro editor allows you to build a form with a separate receipt design when you select the the Use Separate Receipt Design checkbox.

Receipts are primarily delivered by email. There are two options available to send receipts to users via email:

- Email a PDF receipt directly to the user as an attachment
- Email a secure link that allows the user to directly download the PDF receipt

> NOTE
>
> Manager still supports downloading PDF receipts from form confirmation pages. However, this method is not recommended due to performance scaling considerations.

Manager allows you to use the following orchestrations for receipt generation:

- PhantomJS[1]
- Static PDF

---

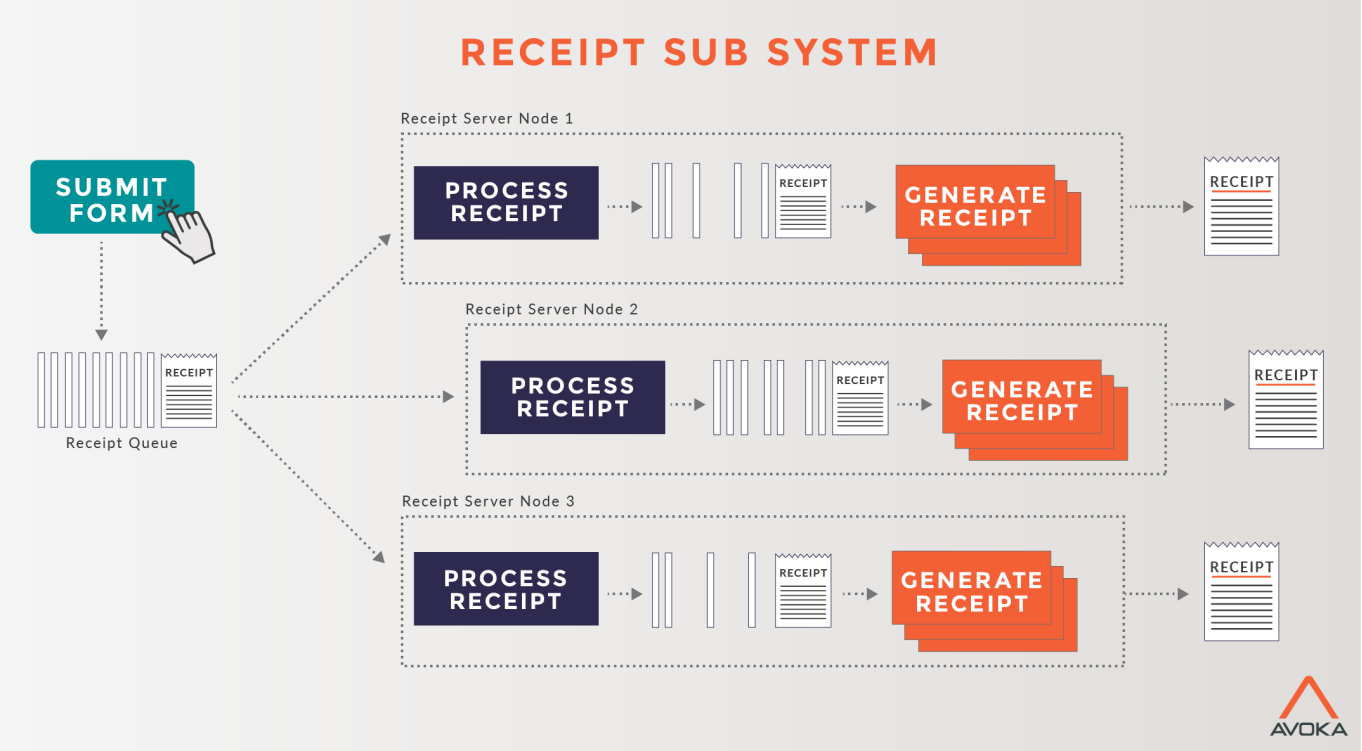
[1]PhantomJS is a scripted, headless browser used for automating web page interaction. PhantomJS provides a JavaScript API enabling automated navigation, screenshots, user behavior and assertions making it a common tool used to run browser-based unit tests in a headless system like a continuous integration environment. PhantomJS is based on WebKit making it a similar browsing environment to Safari and Google Chrome (before Chrome's fork of WebKit evolved into Blink). It is open-source software released under the BSD License.

- iText
- Puppeteer[1] - | 19.05 This feature was introduced in 19.05.

Manager generates user receipts in the background by using dedicated PDF receipt server nodes. This approach offers the following improvements:

- PDF receipt generation can be performed on dedicated PDF receipt generation servers that can be scaled independently.
- PDF receipts should only be generated by a background thread to improve server efficiently. This is the default option.

The receipt generation process consists of the sub-systems as shown below.



Each of the sub-system uses the following core global services:

- The Generate Receipt service renders the PDF receipt.
- The Dynamic PDF Receipt service generates a receipt as it appears on a form. This is the legacy receipt render service.

---

[1]Puppeteer is a Node library which provides a high-level API to control headless Chrome or Chromium over the DevTools Protocol. It can also be configured to use full (non-headless) Chrome or Chromium.

- The Dynamic PDF Receipt 2 service provides more receipt customization options. | 17.10
  This feature was introduced in 17.10.
- The Process Receipt service submits the receipt in the background. This is achieved by using a load balancing mechanism that queues the receipt to a chosen node (in the above diagram, it's a Node 1, Node 2 or Node 3), which can then be processed by one or more threads, depending on the Manager configuration.

Next, learn how to configure receipts server node.

Configure Receipt Numbers and Tracking Codes

Manager lets you identify form transactions by using a receipt number or a tracking code. A receipt number is used to identify a submitted transaction in Manager. A tracking code is a unique alphanumerical code that is generated when a transaction is created and is used to track this transaction that is yet to be submitted. Tracking codes are generally shorter than receipt numbers and by their nature are identified before a receipt number is generated. Manager allows you to use the same value for both the receipt number and tracking code of a submission.

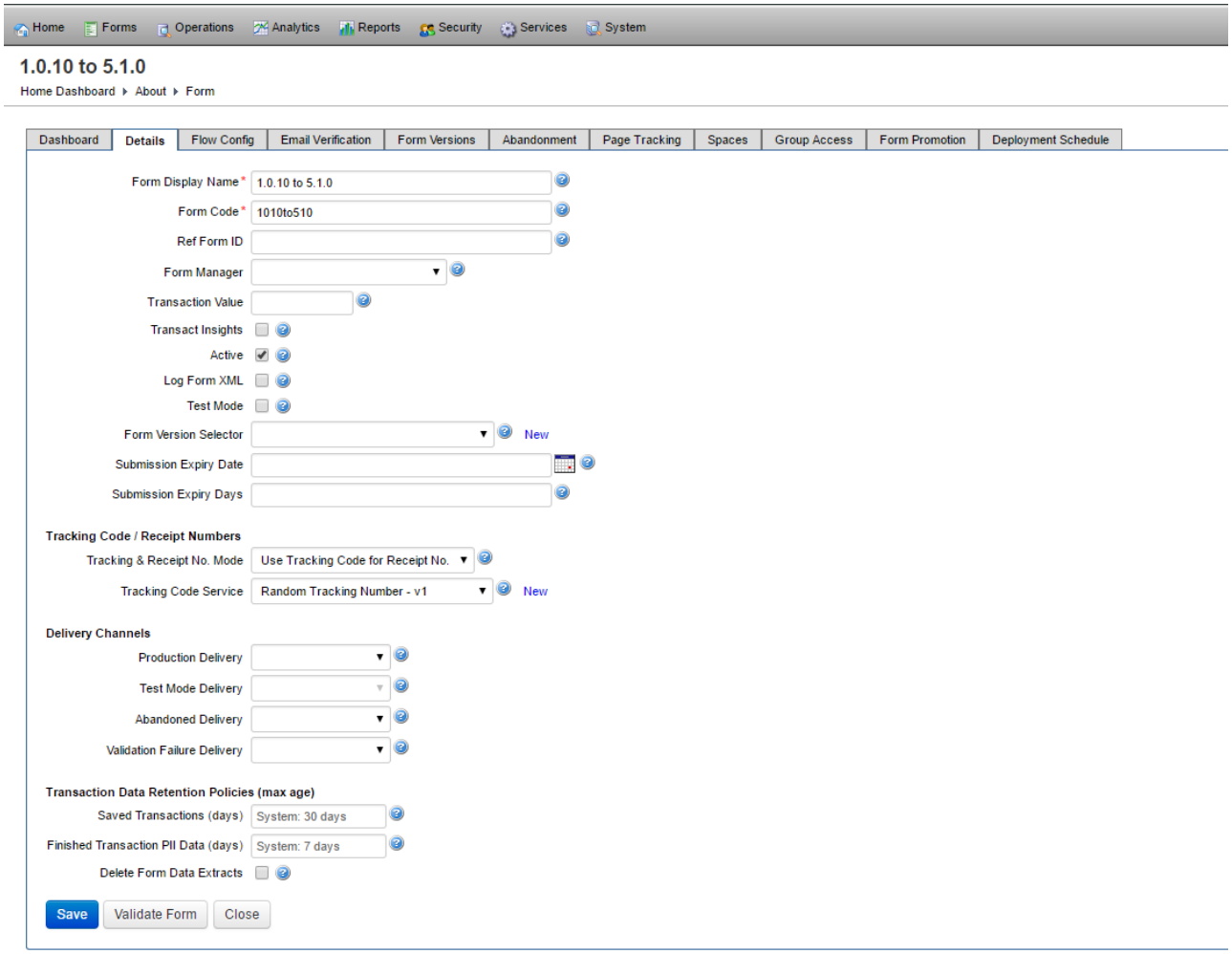
Tracking codes are generated using the `TrackingCodeBuilder` class and are 29 characters long. They are stored in a database table, called `tracking_number`, to ensure every new tracking code is unique. The table is never purged to make sure the codes are unique for as long as the Manager server runs. In the unlikely event of the table growing too large so it needs to be purged, the uniqueness of tracking codes can be compromised. In reality, it won't happen because transactions are not that long-lived and once a transaction is purged, that tracking code is technically available and can be reused.

> NOTE
>
> Check Top Table Sizes to see a number of records that the `tracking_number` table has. In some installations, this table is not listed.

Tracking codes and receipt numbers are only unique to the environment and hence may be duplicated across systems, such as Prod, Dev, UAT etc. If you need a really unique key, you should use a submission key (`submitKey`) which is unique globally. Alternatively, you can prefix or suffix a tracking code with some other piece of unique information, such as day, month and year time stamp.

To configure a form's tracking code or receipt number, follow these steps:

1. Select Forms > Forms.
2. Locate a form and click the Details tab to view form details.



3. Specify whether to have separate Tracking Code and Receipt Numbers generated for your transactions, or use the same value for both. Select one of the following options from the Tracking & Receipt No Mode dropdown list:
   - Use Tracking Code for Receipt No – the receipt number will be identical to the tracking code. It's recommended to use the Tracking Code to improve performance and scalability.
   - Use Receipt No for Tracking Code – the tracking code will be the same as the receipt number. This option overrides the previously generated tracking code.
4. Select a service to generate form tracking codes from the Tracking Code Service drop-down list. You have the following options:
   - Blank - the default Tracking Code service is used
   - Fluent Tracking Number
   - Random Tracking Number
5. Click Save to update the changes.

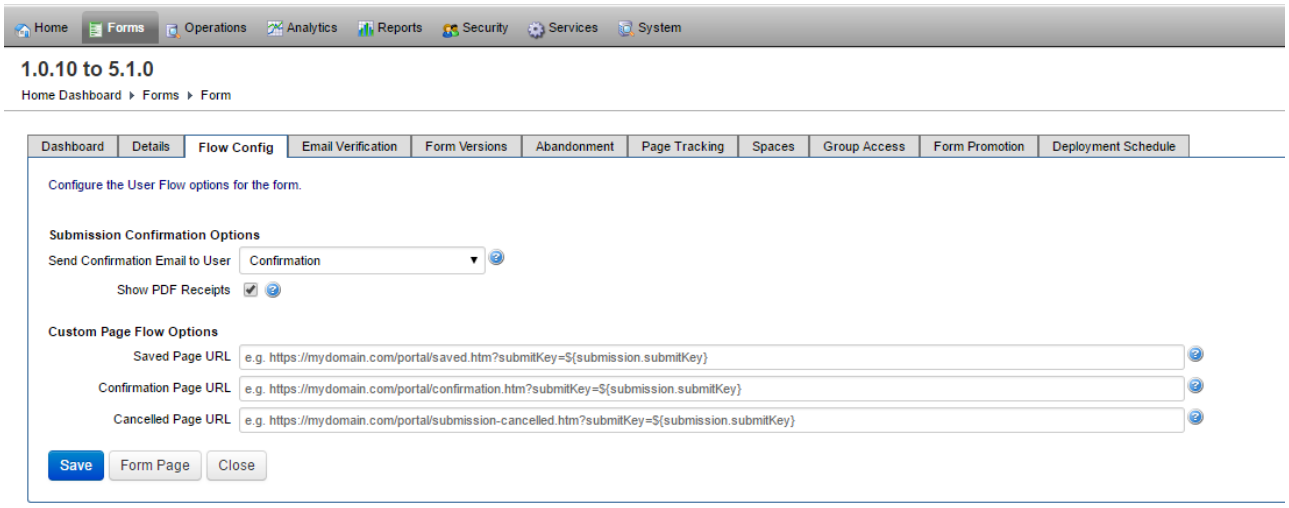
Next, learn how to configure receipt headers and footers.

Configure Receipts Delivery Mode

Manager automatically sends users a confirmation email with a receipt when they complete submissions.

To customize the receipt delivery:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select the Flow Config tab.



4. Select one of the following options from the Send Confirmation Email to User dropdown:
   - **None** – doesn't send a confirmation email to the user. Use this option to turn off the auto-confirmation email.
   - **Confirmation** – sends a confirmation email with no attachment or link to the PDF receipt.
   - **Confirmation and Receipt Attachment** – sends a confirmation email including the receipt PDF file.
   - **Confirmation and Receipt Link** – sends a confirmation email containing a secure link to the receipt PDF file.
5. Select the Show PDF Receipts checkbox to offer users of the form the opportunity to see the PDF receipt on the Confirmation Page that can be set up to appear as soon as the user submits the form. If no Confirmation Page is set up, then this setting is ignored. If Skip Confirmation Page is selected, this setting is ignored as well.
6. Click Save to update the changes.

Next, learn how to configure form flow.

Configure a Receipt Server Node

Manager can utilize several server nodes to generate receipts. To optimize receipt generation, create at least one server node that is solely dedicated for receipt generation by selecting the Transact PDF Receipt Server checkbox only.

Manager has a Transact Receipts module, which can be deployed on a dedicated receipt server node.

You can tune a receipt server node performance by:

- Creating more dedicated receipt server nodes that will process the receipt queue quicker.
- Creating more threads on each server node to increase concurrent processing. This is configured in the Receipt Core Global Service.
- The maximum queue length of submissions to process for each node.
- The maximum number of attempts for rendering any PDF receipt before abandoning.

For low throughput systems, receipt rendering can be performed on server nodes that are shared as opposed to dedicated. However, on high throughput systems with complex receipts, dedicating server nodes for receipt generation may be a better option.

With this approach, each receipt server node can be configured to have multiple threads running. The queue for these threads is updated every minute. This provides the ability to spread the receipt rendering over several nodes and dedicating some server nodes to just rendering receipts, effectively building a server farm for receipt rendering. The use of a farm for receipt rendering means these nodes are not affected by user web traffic and other system processes. As a result, they can be tuned to maintain an optimum level of throughput.

Next, learn how to configure receipt delivery mode.

Receipt Headers and Footers Overview

When Manager creates a PDF receipt, it uses `phantomjs` to render each header and footer in a separate HTML[1] DOM prior to generating the receipt. This has several consequences:

- The pagination process requires a fixed header and footer height in order to correctly size the pages. This means that every page will reserve a fixed amount of space for the header and footer even if there is no content present. You can set this fixed header and footer height by setting the `paperSize.header.height` or `paperSize.footer.height` properties in the JSON[2] options file. Content that exceeds this height will be clipped and not show on the receipt.
- Because each header and footer is rendered in a separate HTML DOM, CSS[3] classes defined in the form itself will not ordinarily be available to use, but instead, styling must be applied using in-line styles. Render Receipt services, such as Dynamic PDF Receipt 2 Service, allow you to apply these styles with ease.
- You can use the `config.customCSS` property to add additional CSS style definitions to the form.
- If the `config.supportHeaderFooterCSSClasses` is set to true then any CSS classes used in a header or footer template are converted automatically to inline styles. This also allows the use of CSS classes already defined in the form itself, not just those defined in the `config.customCSS` property.

---

[1]Maestro forms are fundamentally built using HTML 5. The structure of the form and the text displayed in the form are all configured and established using HTML5. HTML is the standard markup language for creating Web pages and stands for Hypertext Markup Language. HTML 5 describes the structure of a Maestro form.

[2]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

[3]CSS stands for Cascading Styles Sheets and is used to define styles for a Maestro Form, including the design, layout, and variations in the form's display on different devices and screen sizes. The CSS on a Maestro form is primarily taken care of by the Template Designer. The Template Designer will implement the branding and/or styling of the form's assets.CSS can be applied to the template of the form, individual components, and any other asset associated with a Maestro form.

Header and Footer Templates

The `headers` and `footers` properties, stored in a JSON file, allow you to specify template rules that a receipt service runs to determine the content of a header and a footer. Click Receipt Properties Json to view a JSON file of your render receipt service, for example Dynamic PDF Receipt 2 Service. Each rule has a `pageFilter` and `template` property.

The `template` property contains the HTML that defines the header and footer content. The tokens `${page}`and `${numPages}` will be replaced with the current page number and total number of pages respectively. Page numbering will start from 1. The content must be a single HTML element so if you need multiple elements, be sure to wrap them in an appropriate `<div>` element.

The `pageFilter` property is a string that contains either the page number or a * to match all pages not otherwise filtered.

You can enable logging of the templates used by setting the `config.logHeadersFooters` property to true.

Custom Headers and Footers In a Form

Forms can optionally use JavaScript[1] to control the content and styling of headers and footers. You may want to do this for example to inject calculated or dynamic content in the header or footer. To allow a form to control the header and footer behavior first you will need to make sure the `config.disableFormOverrides` property of the JSON options file is set to false.

You will then need to inject a function into the window object of your form using JavaScript inside your form as in the example below.

[1]Javascript (JS) is a scripting languages, primarily used on the Web. It is used to enhance HTML pages and is commonly found embedded in HTML code. JavaScript is an interpreted language. Thus, it doesn't need to be compiled.

```
window.phantomReceiptCallback = function(actionName,data) {
if (actionName=="orientation") {
       // return either "landscape" or "portrait" or fall
       // this action is only invoked once per form
}
if (actionName=="format") {
       // return the paper format eg "A4","letter",etc or fall
       // this action is only invoked once per form
}
if (actionName=="margin") {
       // return a JSON object defining the margin eg {"top": "0.5cm","left": "0.5cm","bottom":
"0.5cm","right":"0.5cm"} or fall
       // this action is only invoked once per form
}
if (actionName=="headerHeight") {
       // return the header height  eg "1cm" or fall
       // this action is only invoked once per form
}
if (actionName=="footerHeight") {
       // return the footer height  eg "1cm" or fall
       // this action is only invoked once per form
}
if (actionName=="header") {
       // return the header template string eg "&lt;p&gt;This is the header&lt;/p&gt;" or fall
       // you can specify the tokens ${page} and ${numPages} in the template you are returning
and they will be replaced appropriately
       // you can access the current page and number of pages programmatically by using
data.pageNum and data.numPages respectively
       // this action is invoked for every page on the form
}
if (actionName=="footer") {
       // return the footer template string eg "&lt;p&gt;Page ${page} of ${numPages}&lt;/p&gt;"
or fall
       // you can specify the tokens ${page} and ${numPages} in the template you are returning
and they will be replaced appropriately
       // you can access the current page and number of pages programmatically by using
data.pageNum and data.numPages respectively
       // this action is invoked for every page on the form
}
// NOTE: always return the value defined in Manager if not explicitly overriden above
```

```
return data.defaultValue; // this is the value from the JSON options file
};
```

Here is an example of using a form function to inject the tracking code into the form footer on page 1 for a Composer form. You can for example add this JavaScript to a composer form by dropping a Business Rule [General Purpose] with the below script content.

```
window.phantomReceiptCallback = function(actionName,data) {
	if (actionName=="footer") {
		if (data.pageNum==1) {
			var trackingCode = sfc.getPhaseValue("TrackingCode");
			return "&lt;p class='phantom-footer'&gt;Page ${page} of ${numPages} - "+track
ingCode+"&lt;/p&gt;";
		}
		return "&lt;p class='phantom-footer'&gt;Page ${page} of ${numPages}&lt;/p&gt;";
	}
	// NOTE: always return the value defined in Manager if not explicitly overriden above
	return data.defaultValue; // this is the value from the JSON options file
};
```

Next, learn how to configure receipt headers and footers.

Receipt Headers and Footers Customization

Manager allows you to add and customize headers and footers of a form's receipt, which can be a PDF file. The headers and footers may include static content, such as HTML, as well as dynamic data extracted from the form, such as tracking codes. There are several steps you need to follow to enable and customize the receipt.

Enable a Render Receipt Service

To customize the headers and footers for a receipt, you must select a relevant Render Receipt service, such as the Dynamic PDF Receipt 2 service, for the form version you're working with. This is not to be confused with the legacy Dynamic PDF Receipt service which has fewer customization options.

To enable the Render Receipt service:

1. Open a form version in Manager.
2. Navigate to the Services tab and select Dynamic PDF Receipt 2 from the Receipt Render Service dropdown list. For more information, see Form Version Services.
3. Click Save to update the changes.

Now, all PDF receipts of this form will be generated using this service. You won't, however, notice any difference from the default receipt generation until the receipt's properties are modified.

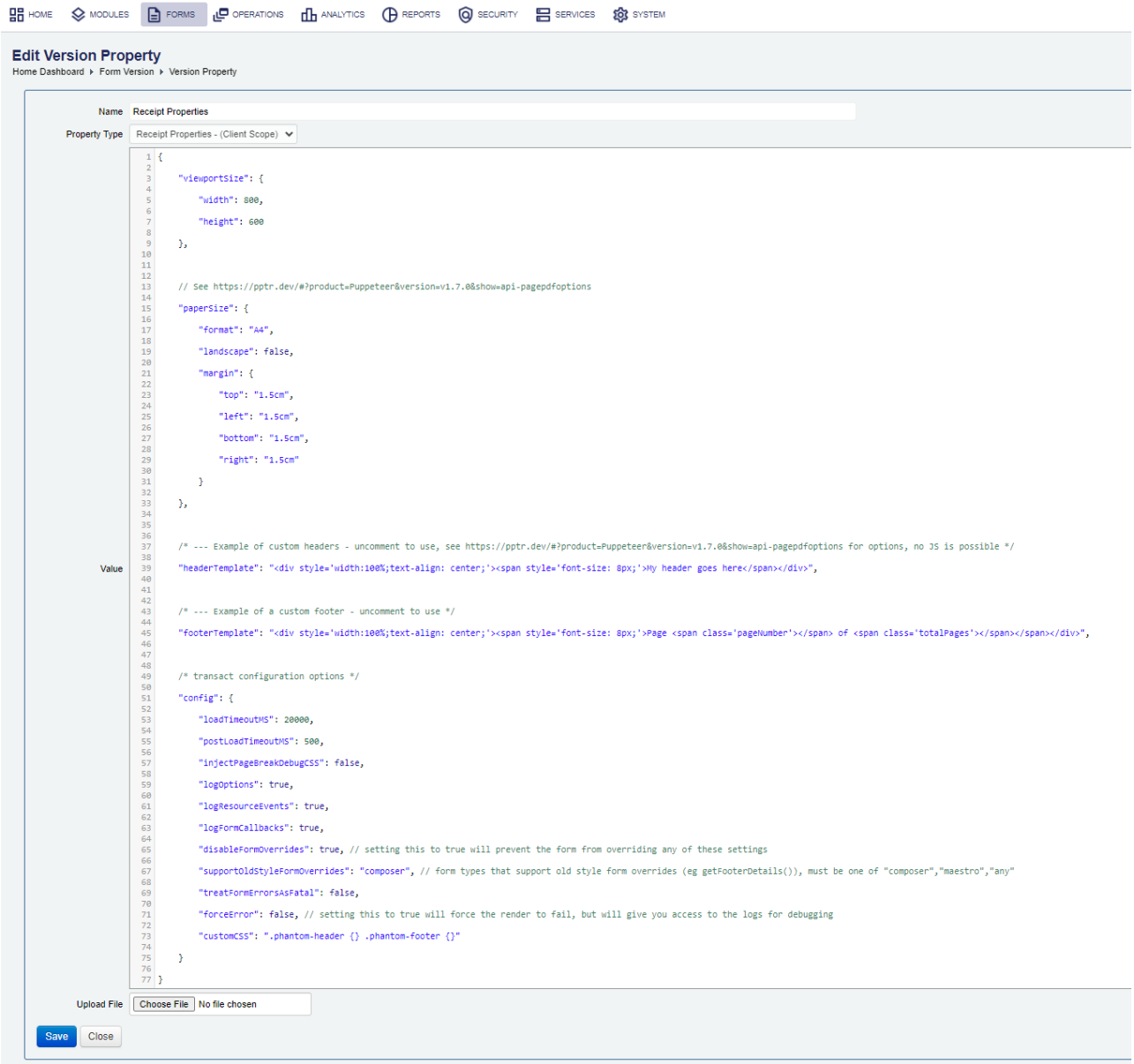
Update Receipt Properties

When the Dynamic PDF Receipt 2 service is enabled, there are several default properties defining the default behavior of the service. However, you can always overrride them to implement your specific design of headers and footers.

To override the properties:

1. Navigate to a form version's Properties taband Click New.
2. From the Property Type dropdown, select Receipt Properties - (Client Scope). For more information, see Form Version Properties.
3. Edit the JSON file defining the form's receipt configuration in the Value field depending on

your requirements.



Let's look at the specific example of how to enable custom headers and footers for a receipt:

1. Update the `paperSize` object's `height` properties. By default, these property's values are set to "0cm", which means the header and footer won't be visible. Change this to a larger value, such as "3cm":

```
"header": {
        "height": "3cm"
},
"footer": {
        "height": "3cm"
}
```

2. Uncomment the `headers` and `footers` properties:

```
"headers": [
        {
                "pageFilter" : "*",
                "template": "<h1 class='phantom-header'>Default Header</h1>"
        },{
                "pageFilter" : "1",
                "template": "<h1 class='phantom-header'>First Page Header</h1>"
        }
],

"footers": [
        {
                "pageFilter" : "*",
                "template": "<p class='phantom-footer'>Page ${page} of ${numPages}</p>"
        }
],
```

These properties are used to define the default content inside the headers and footers. The `pageFilter` properties can be used to include headers or footers on all pages, or on specific pages.

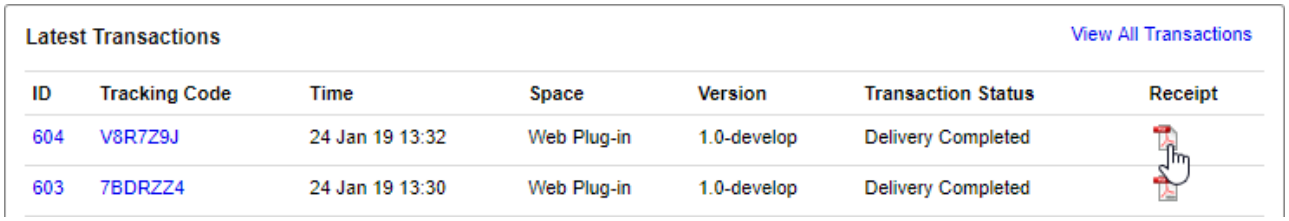
NOTE

Due to a PhantomJS bug, it won't render any `<img>` tags in the header or footer unless the same image is present in the main receipt area. The image can be hidden on the main receipt area; it just needs to be there in Maestro for the header and footer image to render.

After making changes to these properties:

1. Click Save.
2. Return to the form's Dashboard tab.
3. Click the Form Test Render icon.

4. Submit some test data via the form.
5. Return to the Dashboard tab.
6. Click the View PDF Receipt icon for the latest transaction.



You should see that the generated receipt contains the customized header and footer that we defined via the Receipt Properties. Here, for instance, we can see the `h1` element that has been added to the first page of the receipt:

Customize Receipts with JavaScript

All of the properties available via the Receipt Properties can be overwritten by the form that generates the receipt. You might, for instance, like to include the reference code of a form in the footer of a receipt. This requires the form to affect how the receipt is generated. Forms can

optionally use JavaScript to control the content and styling of headers and footers. You may want to do this for example to inject calculated or dynamic content in the header or footer.

To enable this functionality:

1. In Manager, return to the form version's Properties tab.
2. Click Receipt Properties.
3. Within the Value field, change the `disableFormOverrides` property to `false`:

   ```
   "disableFormOverrides": false
   ```

4. Click Save to update the changes.

Then, to use this feature, create a form load rule that contains the following function:

```
window.phantomReceiptCallback = function(actionName, data) {
       // code goes here
};
```

This function executes multiple times while a receipt is generated and provides us with two parameters:

1. `actionName`
2. `data`

Each time the function executes, the `actionName` parameter contains the name of one of the customizable properties, such as `format` or `margin`. Using a conditional, we can return a new value, thereby overriding the value of the property.

```
 value from the JSON options file
```

A few examples are shown below:

- Change the receipt orientation to "landscape":

  ```
  if (actionName === "orientation") {
         return "landscape";
  }
  ```

- Change the height of the header to "3cm":

```
if (actionName === "headerHeight") {
       return "3cm";
}
```

- Add the form's tracking code to the footer:

```
if (actionName === "footer") {
       // Get the tracking code from a Composer form
       // var trackingCode = sfc.getPhaseValue("TrackingCode");
       // Get the tracking code from a Maestro form
       var trackingCode = Form.getSystemData("TrackingCode");
       return "<p>Tracking Code: " + trackingCode + "</p>";
}
```

- More options:

```
window.phantomReceiptCallback = function(actionName,data) {
if (actionName=="orientation") {
       // return either "landscape" or "portrait" or fall
       // this action is only invoked once per form
}
if (actionName=="format") {
       // return the paper format eg "A4","letter",etc or fall
       // this action is only invoked once per form
}
if (actionName=="margin") {
       // return a JSON object defining the margin eg {"top": "0.5cm","left":
"0.5cm","bottom": "0.5cm","right":"0.5cm"} or fall
       // this action is only invoked once per form
}
if (actionName=="headerHeight") {
       // return the header height  eg "1cm" or fall
       // this action is only invoked once per form
}
if (actionName=="footerHeight") {
       // return the footer height  eg "1cm" or fall
       // this action is only invoked once per form
}
if (actionName=="header") {
       // return the header template string eg "<p>This is the header</p>" or fall
       // you can specify the tokens ${page} and ${numPages} in the template you are
returning and they will be replaced appropriately
       // you can access the current page and number of pages programmatically by using
data.pageNum and data.numPages respectively
       // this action is invoked for every page on the form
}
if (actionName=="footer") {
       // return the footer template string eg "<p>Page ${page} of ${numPages}</p>" or
fall
       // you can specify the tokens ${page} and ${numPages} in the template you are
returning and they will be replaced appropriately
       // you can access the current page and number of pages programmatically by using
data.pageNum and data.numPages respectively
       // this action is invoked for every page on the form
}
// NOTE: always return the value defined in Manager, if not explicitly overriden as
above
```

```
return data.defaultValue;
// this is the value from the JSON options file
```

The `data` parameter is an object with the following structure:

```
{
        "pageNum": 1,
        "numPages": 2,
        "defaultValue": "portrait"
}
```

The `pageNum` property contains the current page number of the receipt. One way to use this property is to display different content on the first page of a receipt:

```
if (actionName === "footer") {
        if (data.pageNum === 1) {
                return "<p>This appears in the footer on the first page.</p>";
        } else {
                return "<p>This appears in the footer on every other page.</p>";
        }
}
```

The `numPages` property contains the total number of pages in the receipt. This is commonly used in the footer, in conjunction with the `pageNum` property:

```
if (actionName === "footer") {
        return "<p>Page " + data.pageNum + " of " + data.numPages + "</p>";
}
```

The `defaultValue` property contains the value of the property that was defined in Manager, via the Receipt Properties. If, for instance, you give the `orientation` property a value of "portrait" via the Receipt Properties, the `defaultValue` property will contain that same value.

This property should always be returned at the bottom of the function:

```
window.phantomReceiptCallback = function(actionName, data) {
	if (actionName === "orientation") {
		return "landscape";
	}
	return data.defaultValue;
};
```

This ensures that the function never returns `undefined`.

For all of the properties, the function expects a string to be returned, except for the margin property, which expects an object with string values:

```
if (actionName === "margin") {
	return {
		top: "1.5cm",
		left: "1.5cm",
		bottom: "1.5cm",
		right: "1.5cm"
	};
}
```

Default Receipt Properties

A typical receipt has the following default properties:

Property	Default Value
format	a4
orientation	portrait
margin	`{ top: "1.5cm", left: "1.5cm", bottom: "1.5cm", right: "1.5cm" }`
headerHeight	0cm

footerHeight	0cm
header	<h1 class='phantom-header'>Default Header</h1>
footer	Page ${page} of ${numPages}

After making changes to this function: save the code, build the form, and update the form via Manager. Any submissions from this form will use this function when generating the receipt.

Next, learn more about receipt's headers and footers.

Using Header and Footer Templates

The headers and footers properties of the JSON options file allow you to specify an array of templating rules that are run to determine the content of the header or footer respectively. Each rule has a pageFilter and template property.

The template property contains the HTML that defines the header/footer content. The tokens ${page} and ${numPages} will be replaced with the current page number and total number of pages respectively. Page numbering will start from 1. The content must be a single HTML element so if you need multiple elements, be sure to wrap them in an appropriate element.

The pageFilter property is a string that contains either the page number or a * to match all pages not otherwise filtered.

You can enable logging of the templates used by setting the config.logHeadersFooters property to true.

Field Overlay Receipts Using AcroForms

By default, Manager generates a PDF receipt for every completed transaction. This receipt, however, is based on the design of the completed Maestro form, which might not meet your organization's design or legal requirements. Your organization, for instance, might prefer to use PDF forms they've already designed, such as AcroForms[1]. If this is the case, you should consider using Field Overlay Receipts.

Field Overlay Receipts allow organizations to use a PDF form as a receipt template. Then, when a transaction is completed, the customer's data is inserted into the fields of the PDF.

The advantage of this approach is that receipts will look identical to the original PDF. This means:

- They won't need to be re-approved by design or legal departments.
- No one will need to be retrained on how to process the receipts.
- The same form can be used if there is a need to partially complete it, print it out, and complete it with pen and ink.

There are, however, some potential draw-backs:

- Additional development is required to overlay the data fields over the existing PDF.
- Hide and show rules are not honored. All fields are always displayed.
- Repeats are only honored to the extent that they're repeated on the original form. The Form Designer must be careful not to allow more repeats than are presented on the PDF.
- Field lengths must be limited to ensure they don't overflow the space allocated on the PDF.
- The ability to re-design the data entry experience is limited. No new fields can be added, since there is nowhere to place them.

Despite these draw-backs, Field Overlay Receipts are an excellent option for organizations that need to move quickly and integrate Manager with their current processes, rather than adapting to Manager's way of doing things.

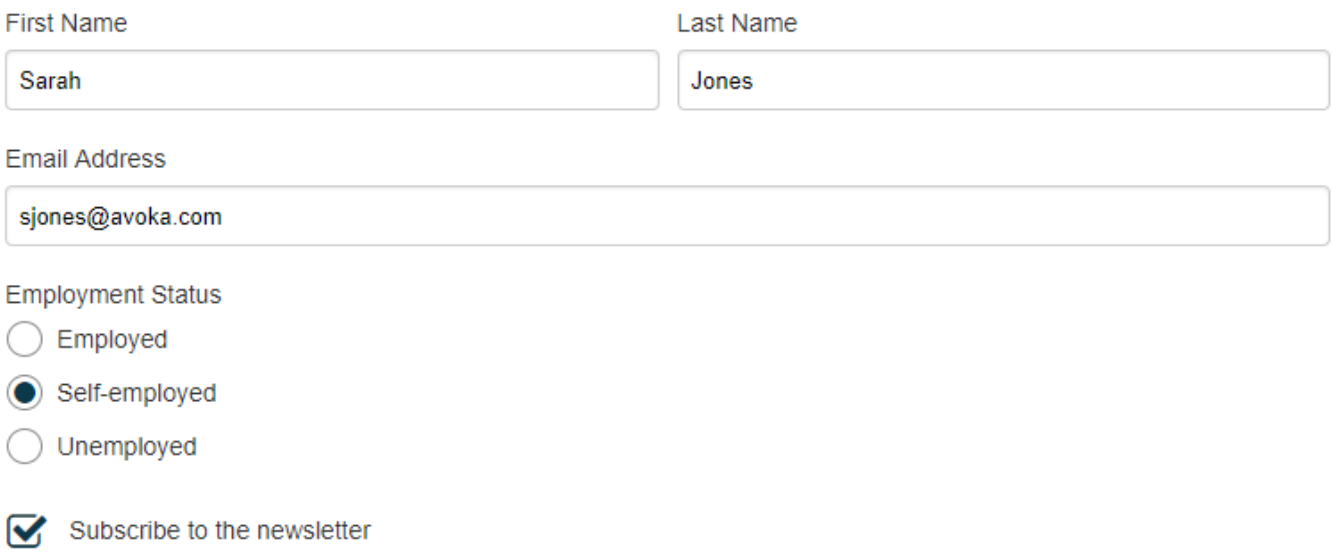
Getting Started

Before we continue:

[1]AcroForms are the original PDF forms technology that accepts input in both Forms Data Format (FDF) and XML Forms Data Format (XFDF). Many 3rd party vendors support AcroForms.

1. Download Adobe Acrobat Pro DC. This is the easiest software for converting a PDF form into an interactive form, which is what Manager needs to insert data into a PDF. The software requires a monthly subscription, but there is a 7-day free trial.
2. Choose a PDF form to use as a receipt template. If you don't have a PDF form available, either use this example PDF from Adobe or create your own PDF in Microsoft Word.
3. Create a form in Maestro.

In this topic, we'll be using this Maestro form as an example:



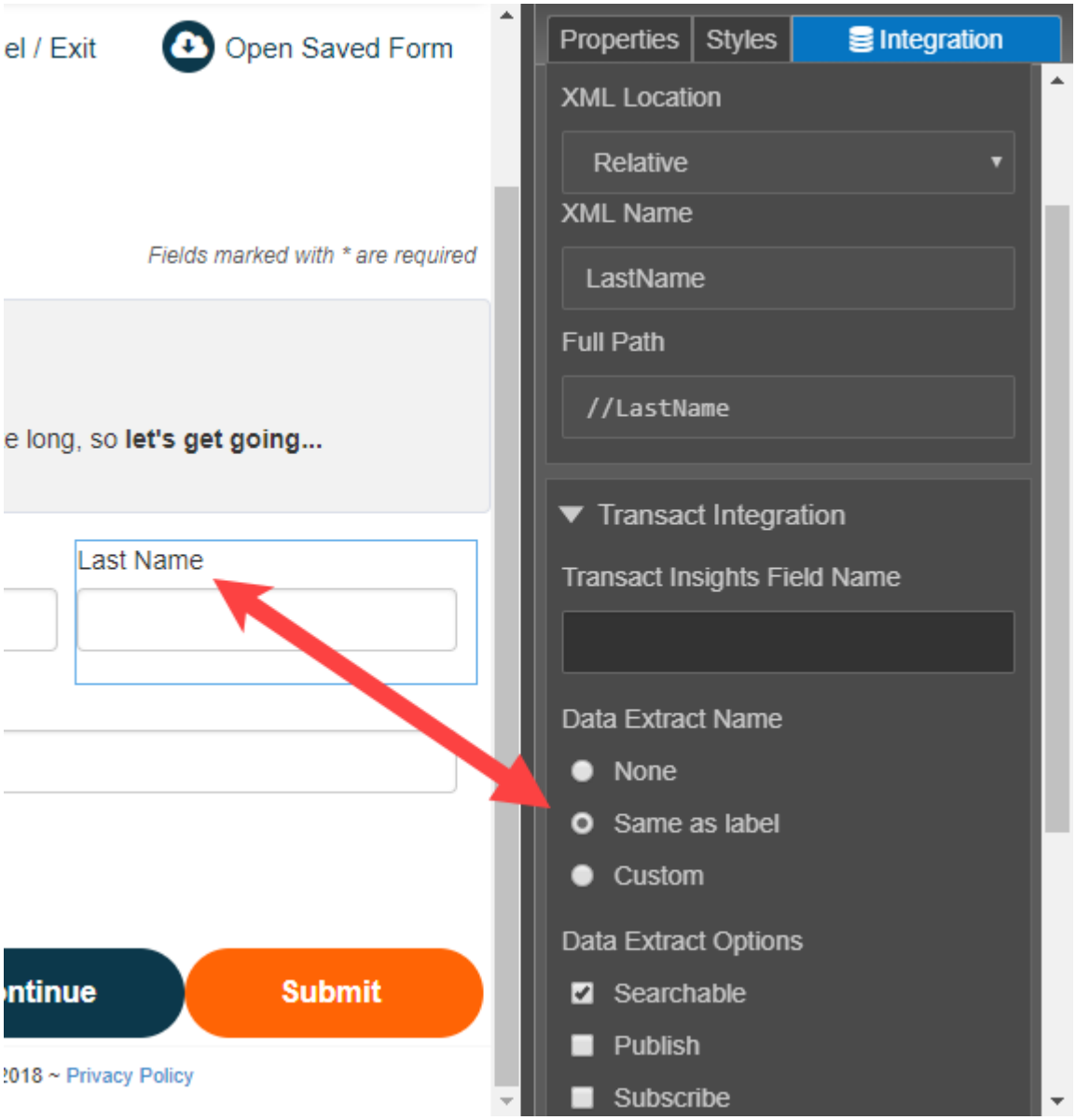
As we can see, this form has various field types, including text fields, a radio button group, and a checkbox. This will allow us to see how different types of data can be represented in a receipt.

Step 1: Add Data Extracts to form fields

Data Extracts allow you to extract data from a completed transaction and integrate that data with external systems. One use of this feature is to transfer data from a form into a PDF file.

To implement Data Extracts:

1. Login to Maestro.
2. Open a form.
3. Select a form field.
4. Open the **Integrations** tab.
5. Expand the **Transaction Integration** panel.
6. For the **Data Extract Name** field, select **Same as Label**.



7. Repeat the previous step for every field that needs to be inserted into the PDF form.
8. Click Save to update the changes.

Step 2: Export the form from Maestro

1. Click **Build**.
2. In the modal that appears, click **Build**.
3. Click **Download TM Form Version**.

Step 3: Publish the form to Manager

1. Login to Manager.
2. Navigate to **Forms**.
3. Click **Import Form**.

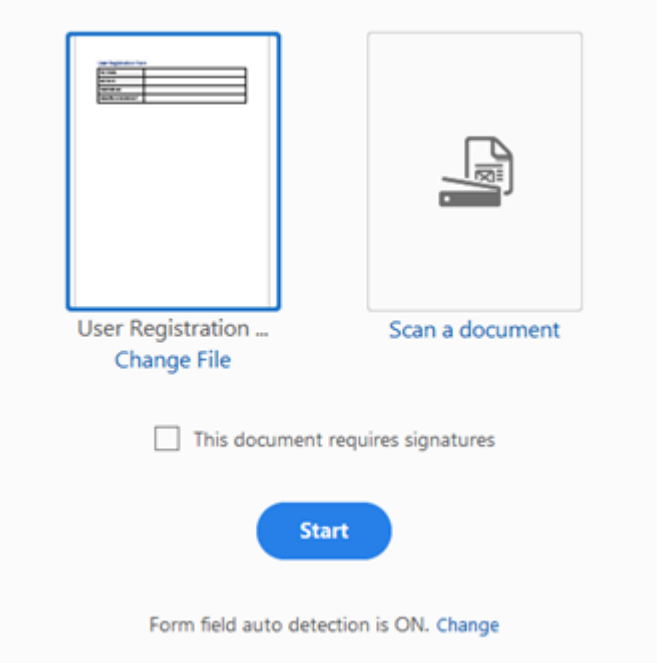
4. Upload the **Form Version Archive File**. This is the **TM Form Version** downloaded in step 2.
5. Select **Create a Form**.
6. Click **Proceed to Import**.
7. Click **Import**.

Step 4: Create a fillable PDF with Adobe Acrobat Pro DC

In this step, we're going to convert a PDF into what Adobe calls a fillable PDF. This will allow Manager to insert data into the PDF file.

To do this:

1. Open Adobe Acrobat Pro DC.
2. Go to **Tools > Prepare Form**.
3. Click **Select a file**.
4. Select the PDF file.
5. Click **Start**.



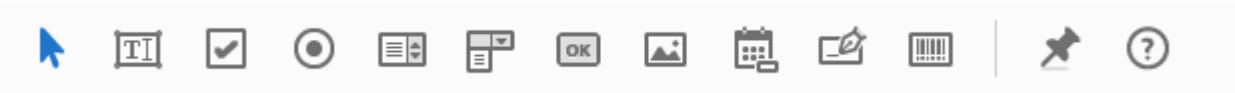
During this process, Acrobat will attempt to detect and create fillable form fields inside the PDF. This isn't a seamless process though, so manual adjustments may be necessary.

In this example, Acrobat recognized three text fields:

User Registration Form

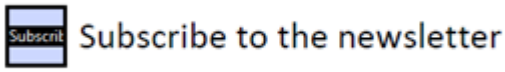
First Name	First Name
Last Name	Last Name
Email Address	Email Address

But the radio buttons and checkboxes need to be manually inserted via the toolbar.

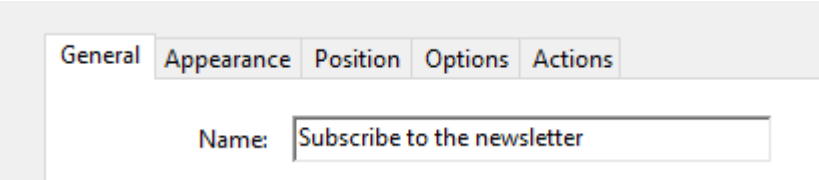


Step 4.1: Insert a checkbox

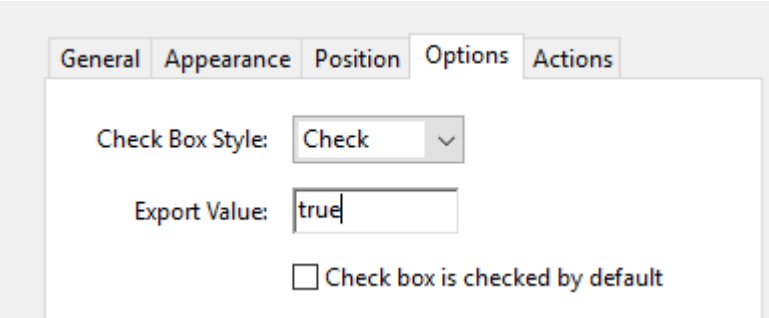
1. Click the **Check Box** icon in the toolbar.
2. Click inside the PDF to place the checkbox.



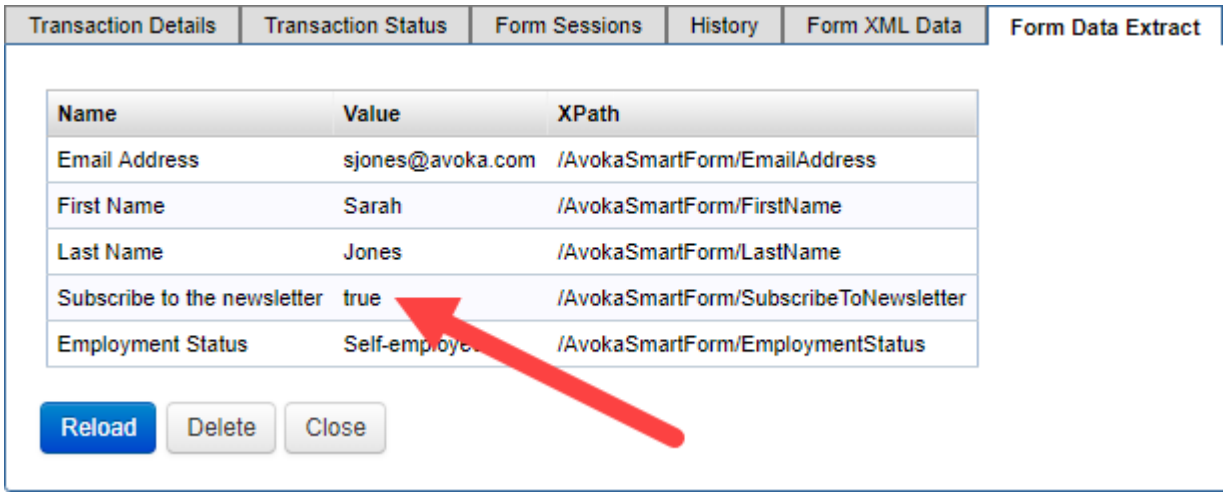
3. Double-click the checkbox to open the **Check Box Properties** window.
4. On the **General** tab, update the **Name** field. This name should match the field's label that was defined in Maestro. In this example, the label is "Subscribe to the newsletter".



5. On the **Options** tab, set the **Export Value** as "true".

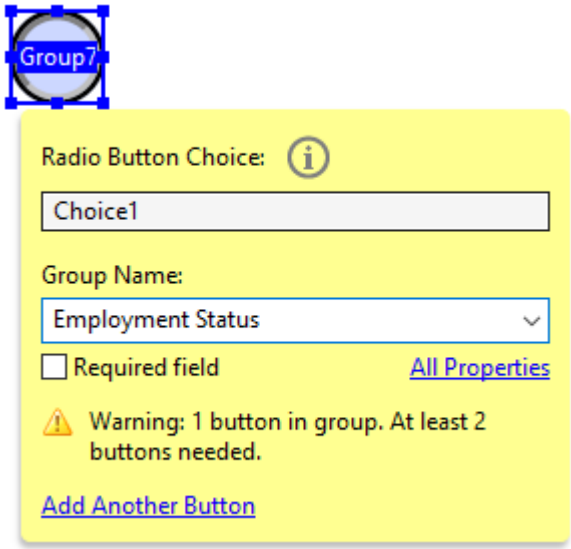


We need to set the value to "true" because this is how the value is stored in Manager. When the **Export Value** matches the value in Manager, the checkbox will be checked in the PDF.

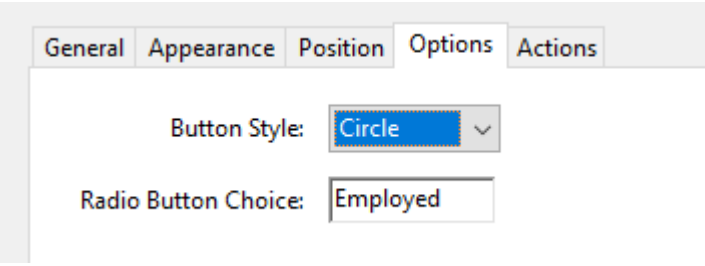


Step 4.2: Insert a radio button

1. Click the **Radio Button** icon in the toolbar.
2. Click inside the PDF to place the radio button.
3. Set the **Group Name** field to the radio button group's label that was defined in Maestro. In this example, the radio button group's label is "Employment Status".

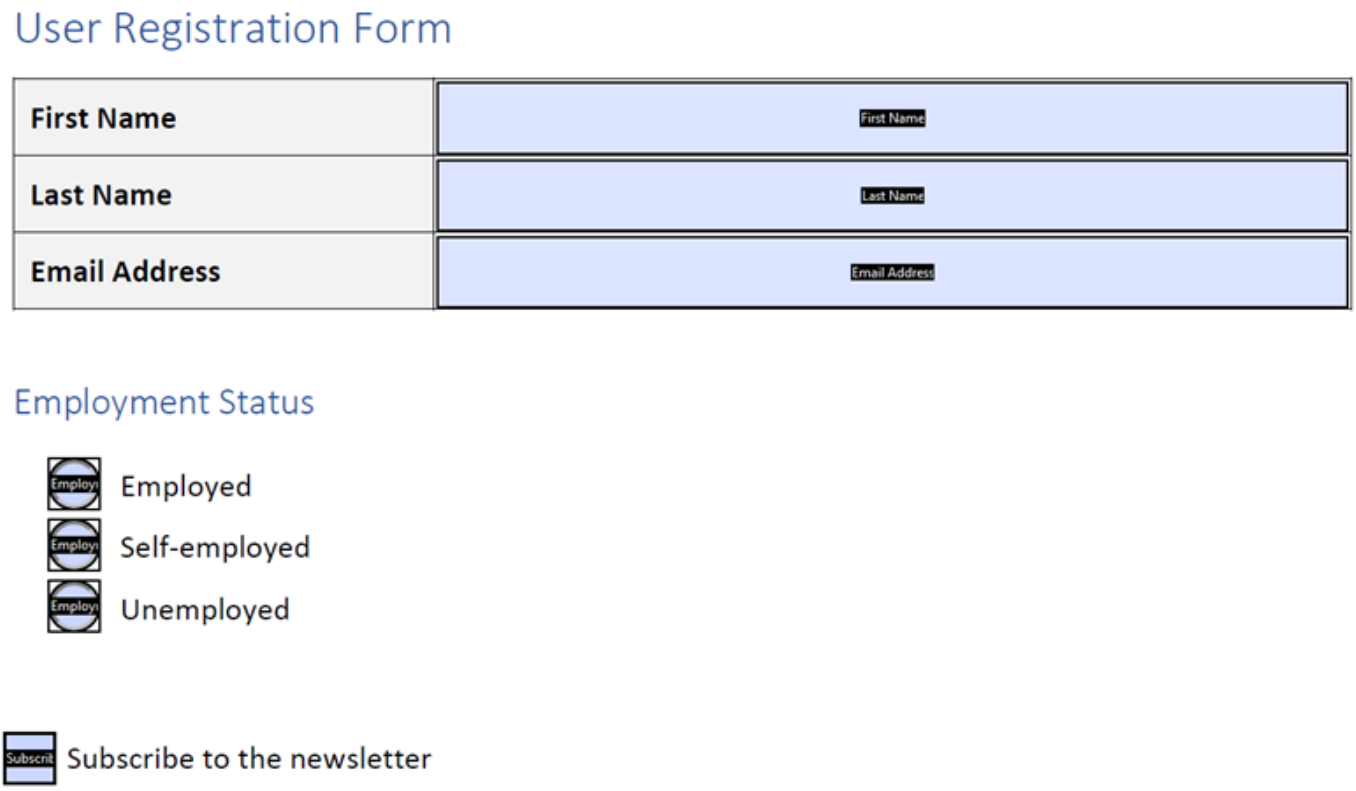


4. Double-click the radio button to open the **Radio Button Properties** window.
5. On the **Options** tab, set the **Radio Button Choice** field to the value of an individual radio button that was defined in Maestro. In this example, the first radio button value is "Employed".

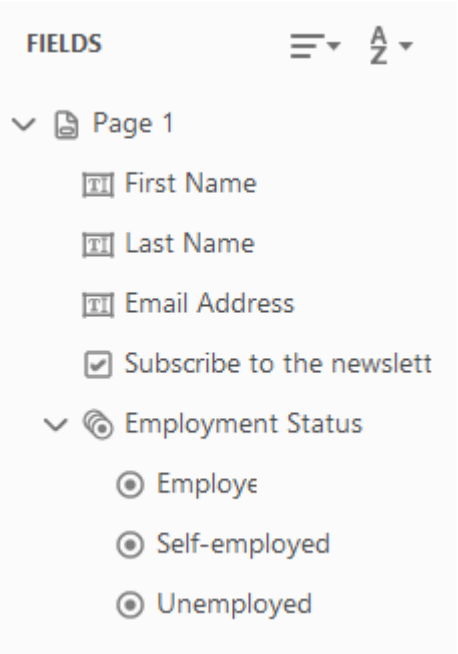


6. Repeat this process for all of the form's radio buttons. In this example, the remaining radio button values are "Self-employed" and "Unemployed".

You should end up with a form that resembles the following:



The structure of the form should also be visible on the right side of the screen:

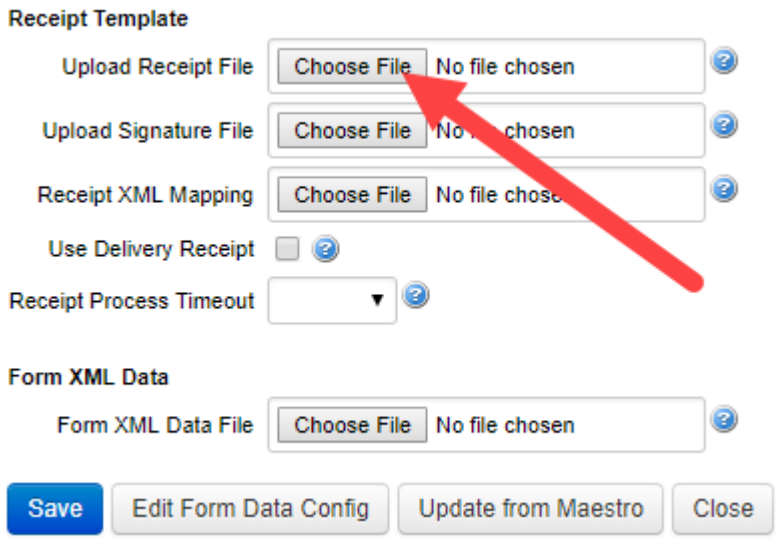


If any of the labels in this list don't match the labels in Maestro, fix them before continuing. If you have any other issues preparing the form, read the official documentation for Acrobat Pro DC.

To end this section, navigate to **File > Save As...** and save the form.

Step 5: Upload the fillable PDF to Manager

1. Return to Manager.
2. Navigate to **Forms**.
3. Open the published form. If you have a lot of published forms, you may need to use the search feature.
4. Click the **Form Versions** tab.
5. Click the **Edit Form Version** icon.
6. Upload the fillable PDF file to the **Upload Receipt File** field.



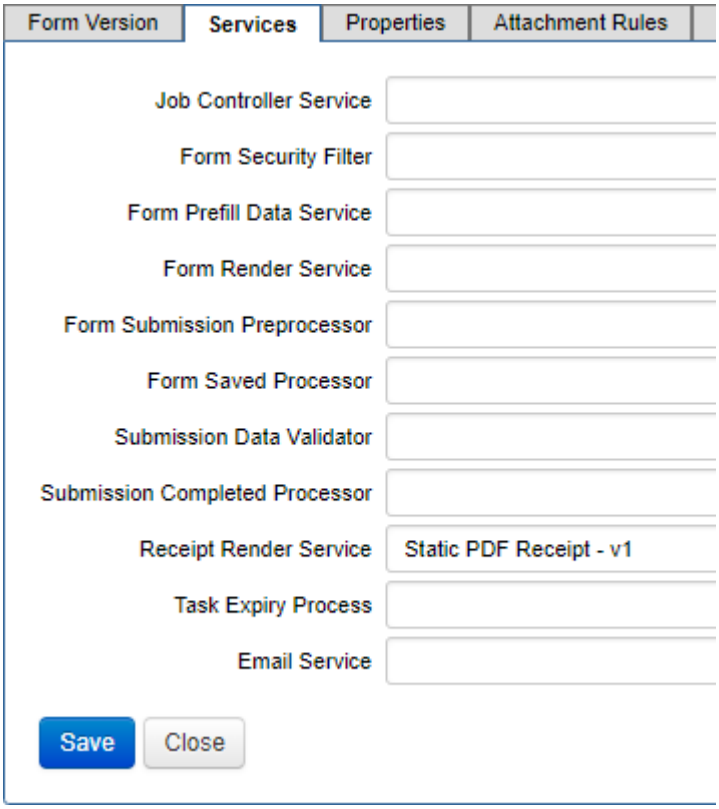
7. Click Save to update the changes.

Step 6: Configure the Receipt Render Service

Within Manager, services are used to access a range of business-related functionality on demand. One such service is the Receipt Render Service, which is used to generate PDF receipts.

To enable this service:

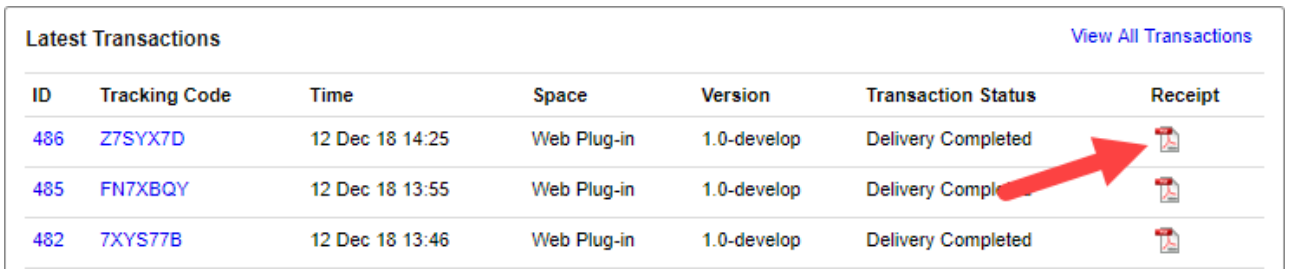
1. On the **Form Version** page, click the **Services** tab.
2. From the **Receipt Render Service** dropdown, select **Static PDF Receipt v1**.
3. Click **Save**.



Step 7: Verify the results

1. Navigate to the **Form Dashboard**.
2. Click the **Test Render** icon for the latest form version.

3. After the form loads, submit some sample data.
4. Refresh the **Form Dashboard**.
5. Click the **View PDF Receipt** icon.



The receipt should contain data from the completed transaction:

User Registration Form

First Name	Sarah
Last Name	Jones
Email Address	sjones@avoka.com

Employment Status

- ○ Employed
- ◉ Self-employed
- ○ Unemployed

☒ Subscribe to the newsletter

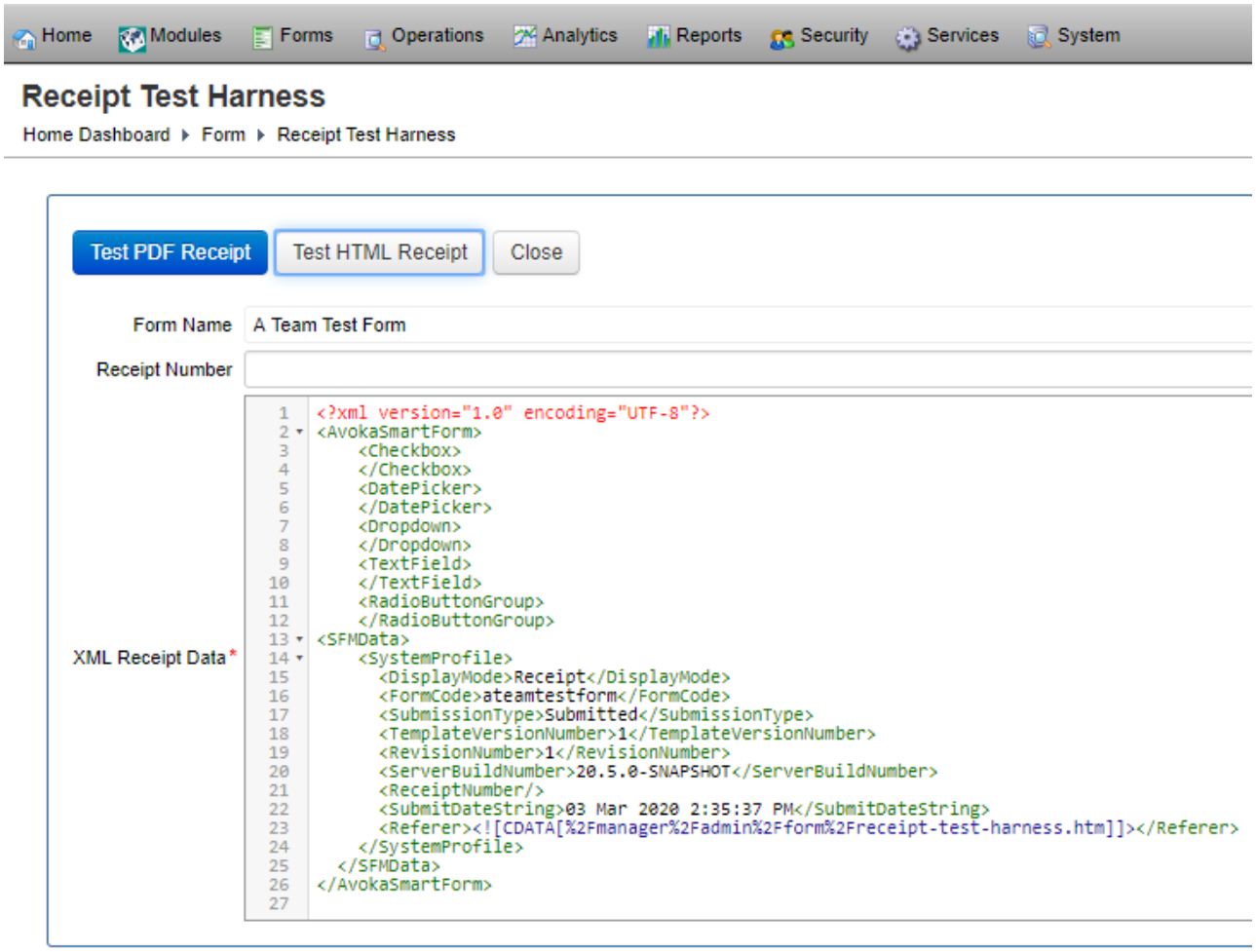
Next, learn how headers and footers are rendered.

Test a Receipt with Receipt Test Harness

Manager allows you to manually render a form as a PDF receipt, so you can preview it and debug it. You can generate a receipt as PDF or HTML, which is determined by the form type you have used to create the form. The PDF receipt is suitable for more traditional AcroForms[1], whereas the HTML receipt refers to Composer or Maestro forms. However, in either case, the form is rendered as PDF. We recommend you render the form as HTML to test your receipt in a browser and debug issues.

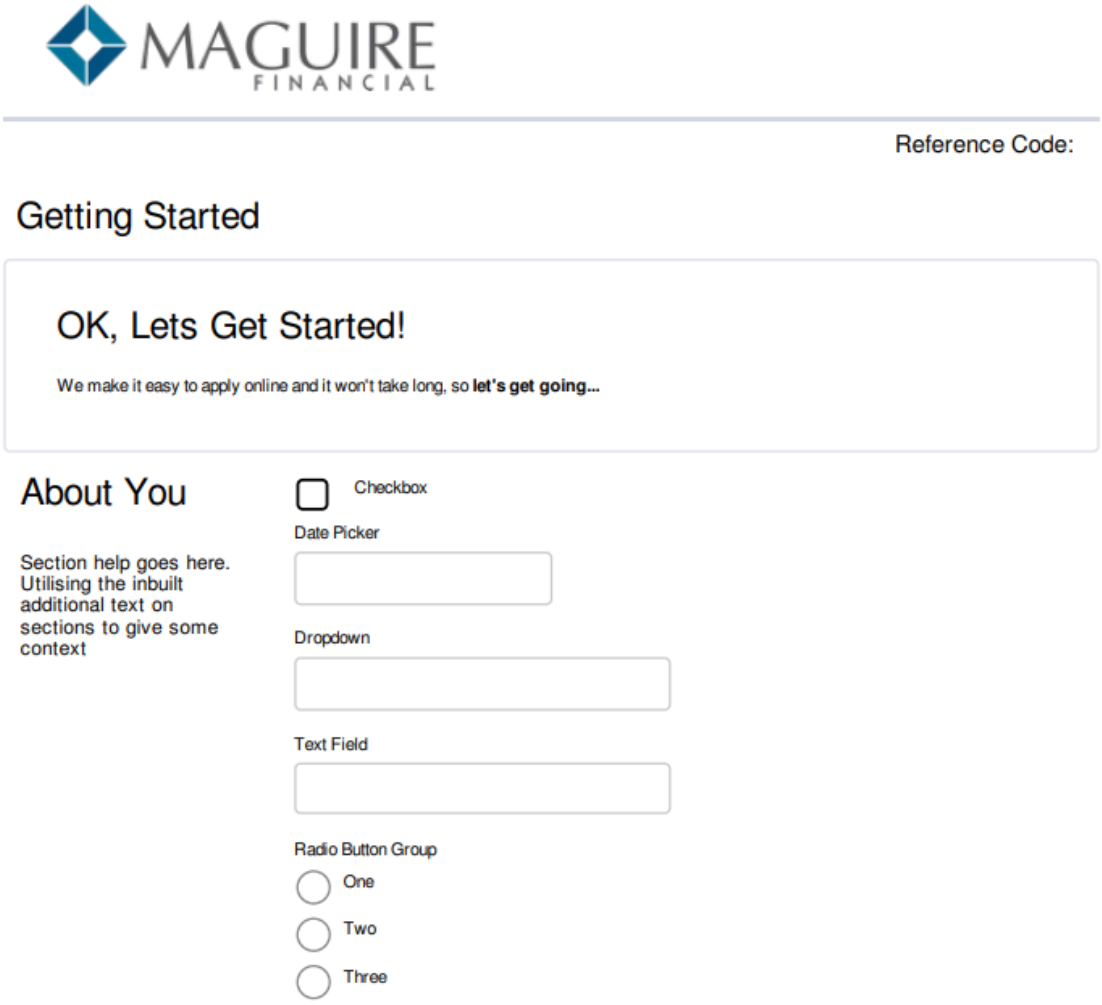
To render a form as a receipt:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Click Receipt Test Harness under the Form Spaces section.

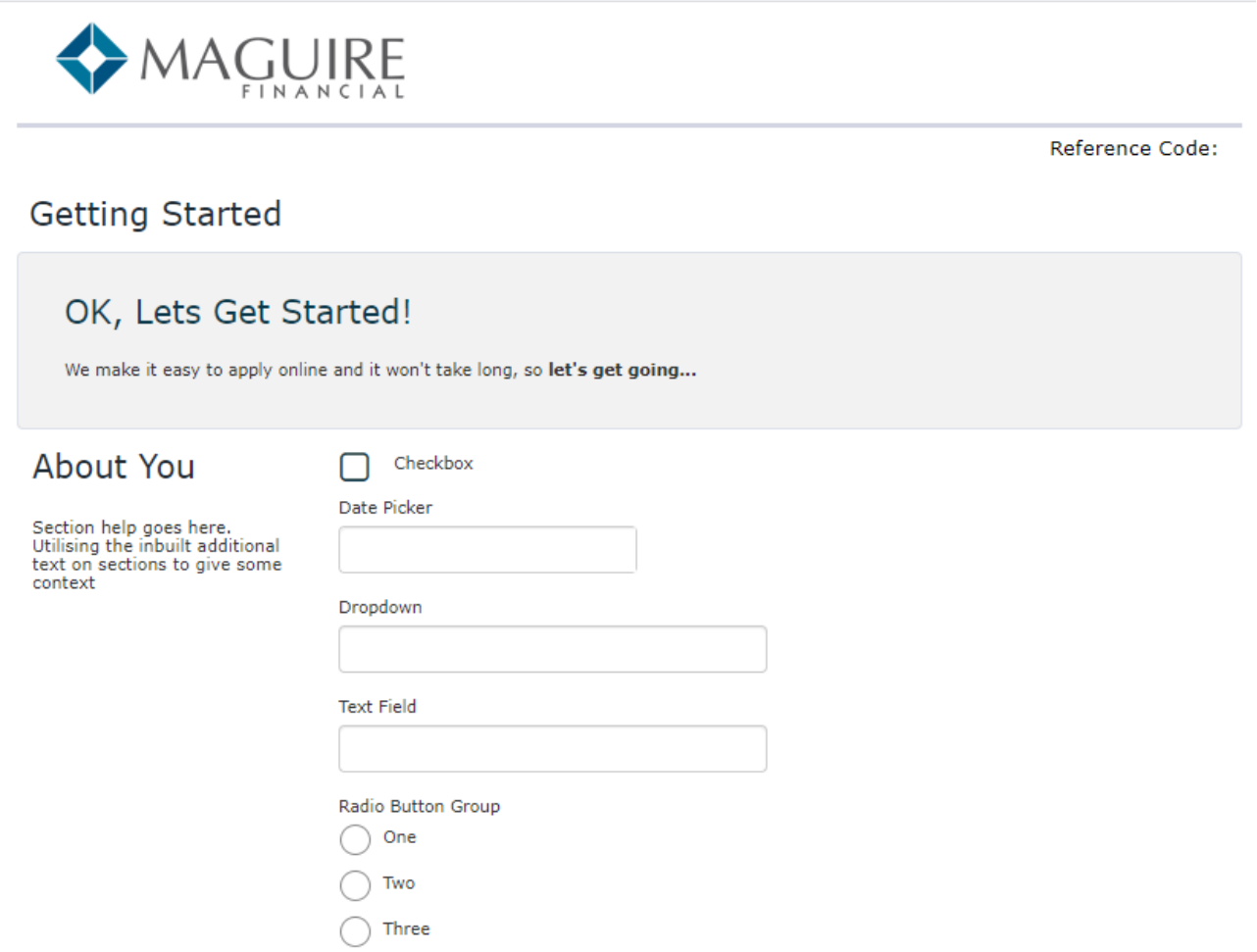


[1]AcroForms are the original PDF forms technology that accepts input in both Forms Data Format (FDF) and XML Forms Data Format (XFDF). Many 3rd party vendors support AcroForms.

4. Enter a new receipt number in the Receipt Number field.
5. Update the XML data in the XML Receipt Data field to test the new changes.
6. Click Test PDF Receipt to render a PDF receipt in a new window.



7. Click Test HTML Receipt to render a form using PhantomJS[1] to produce a receipt as a PDF file. The way this works is by first rendering the Maestro HTML code then 'printing' the result to PDF and returning the PDF data. You should use this functionality to test your receipt in the browser and debug issues.



8. Click Close.

Next, learn about the form dashboard.

---

[1]PhantomJS is a scripted, headless browser used for automating web page interaction. PhantomJS provides a JavaScript API enabling automated navigation, screenshots, user behavior and assertions making it a common tool used to run browser-based unit tests in a headless system like a continuous integration environment. PhantomJS is based on WebKit making it a similar browsing environment to Safari and Google Chrome (before Chrome's fork of WebKit evolved into Blink). It is open-source software released under the BSD License.

PDF Receipt Options

Manager gives you a number of options to generate a PDF receipt or document of records. The options are:

- Transact PDF Receipts
- XFA Receipts
- Field Overlay Receipts
  - using Acrobat
  - using LiveCycle Designer | 19.11 This feature was removed in 19.11
  - using Texcel
- Custom Receipt Generation

We not recommend one of these approaches over another - the choice will depend on the requirements of each situation. Each solution has its own pros and cons. In addition, each tool used has a learning curve, and productivity will be determined by the skills of those using it.

> NOTE
>
> Adobe LiveCycle[1] support has been completely removed in 19.11, so you should use other solutions.

Manager PDF Receipts

Manager is able to create a PDF receipt directly from the form design UI. This receipt is very similar to the interactive form, but with a number of modifications to optimize it for printing. These modifications include:

- Reducing the use of color (users often complain that it uses up too much printer ink)
- Removal of the navigational menu, so that the form becomes a single long multi-page doc-

[1]Adobe LiveCycle Enterprise Suite (ES4) is an service-oriented architecture Java EE server software product from Adobe Systems used to build applications that automate a broad range of business processes for enterprises and government agencies. LiveCycle ES4 is an enterprise document and form platform that allows capturing and processing information, delivering personalized communications, and protecting and tracking sensitive information. It is used for purposes such as account opening, services and benefits enrollment, correspondence management, request for proposal processes, and other manual based workflows.

ument.
- Removal of all interactive features such as buttons and drop-down lists.

The exact optimizations used can be modified by adjusting a Composer skin.

Pros

- Requires no additional configuration or effort. (It is built into the product.)
- Honors all business rules in the form, such as hide-show logic and repeats.

Cons

- Does not generate PDF/A compatible PDFs. This may be a requirement for some organizations.
- Has limited support for headers, footers, and pagination options.
- Some problems have been reported with layouts, particularly character spacing on Linux.
- Since interactive forms tend to be quite spaced out, this can result in lengthy receipts with multiple pages and low field densities.
- Has a different layout to existing paper forms. (This may be an issue for downstream systems or people. This may also result in additional approval steps.)
- Not suitable for the case where a customer partially completes a form interactively, prints it out, and then completes the rest of the form using ink. (This can be achieved with additional effort in the form design.)
- Since the length and layout of the document may vary based on hide-show rules and repeats, the location of fields on the page cannot be guaranteed. This may impact downstream systems such as OCR, imaging, or digital signatures.

XFA Receipts

Composer is able to generate a PDF in XFA format. This is a format suited to high-fidelity PDF documents.

Pros

- Has strong capabilities for producing high-fidelity PDF documents, including headers, footers, and pagination options.
- Text formatting and layout is good.
- Produces PDF/A receipts.
- Honors all business rules in the form, such as hide-show logic and repeats.

Cons

- Requires Adobe LiveCycle. This incurs additional costs, and additional effort to provision and maintain.
- Since interactive forms tend to be quite spaced out, this can result in lengthy receipts with multiple pages and low field densities.
- Has a different layout to existing paper forms. (This may be an issue for downstream systems or people. This may also result in additional approval steps.)
- Not suitable for the case where a customer partially completes a form interactively, prints it out, and then completes the rest of the form using ink. (This can be achieved with additional effort in the form design.)
- Since the length and layout of the document may vary based on hide-show rules and repeats, the location of fields on the page cannot be guaranteed. This may impact downstream systems such as OCR, imaging, or digital signatures.

Field Overlay Receipts

Instead of generating a complete form, it is possible instead to import an existing (flat) PDF form, and lay the data fields over the top of this existing form. Regardless of the technology used to do this field overlay, there are some pros and cons.

Pros

- High fidelity - the receipt will look the same as the existing PDF.
- No need for additional approval steps, as the receipt is the same as the existing document.
- Information density is high, because the existing PDF has probably already been designing to minimize the number of pages.
- This type of form can be used if there is a need to partially complete the form, print it out, and then complete it with pen and ink.
- Receipt PDF is identical to the existing PDF, which will minimize downstream implications, including retraining. It also eliminates the need for additional approvals.
- PDF/A supported.

Cons

- Additional development effect is required to perform the task of overlaying the data fields over the top of the existing PDF.
- Additional development effort is required to map the XML data fields into a flat field naming structure. This can be done either within Composer, or using a mapping file in Manager.
- Hide or show rules are not honored. All fields are always displayed.

- Repeats are only honored to the extent that they are repeated on the original form. The form designer must be careful not to allow more repeats that are presented on the PDF. Mapping of repeats can be complicated.
- Field lengths must be limited in the interactive form to ensure that they don't overflow the space allocated on the PDF.
- Multi-line text must be controlled carefully to ensure they they don't overflow the space allowed in the PDF.
- Requires very detailed work to accurately size and place fields so that they match the underlying PDF.
- To some extent, the ability to re-design the data entry experience is limited. In particular, no new fields can be added, since there is nowhere to place them.

Field Overlay Receipts Using Adobe Acrobat

Adobe Acrobat has the ability to overlay fields over the top of an existing flat PDF. This uses a technology known as AcroForms.

Pros

- Relatively inexpensive per-seat development license.
- Makes no modifications to underlying PDF, and so the receipt maintains 100% fidelity with the original PDF.
- Manager supports AcroForms technology with no additional provision or configuration.
- Has a field recognition feature, which allows fields to be automatically recognized. This can save a lot of time, particularly when large numbers of PDF forms need to be converted. The effectiveness of this feature varies widely, based on the specifics of particular documents. It is suggested that this feature be tested against a representative sample of documents before being used on large scale.

Cons

- The field creation and editing environment within Acrobat is considered by many to be quite rudimentary, particularly for larger forms, though this is subject to personal preferences.
- No ability to modify the underlying PDF document. For example, to remove explanatory text that may be inappropriate, or to make minor modifications to the form.
- If a new version of the underlying PDF document is produced, then the entire process needs to be repeated on the new document. Copying and pasting fields may help in some cases.

Field Overlay Receipts Using Adobe LiveCycle Designer

Adobe LiveCycle Designer has the ability to overlay fields over the top of an existing PDF document.

Pros

- Relatively inexpensive per-seat development license.
- Makes no modifications to underlying PDF, and so the receipt maintains 100% fidelity with the original PDF.
- The field editing and creation environment is reportedly better than Acrobat, though this is subject to personal preferences.

Cons

- Requires Adobe LiveCycle. This incurs additional costs, and additional effort to provision and maintain.
- No ability to modify the underlying PDF document. For example, to remove explanatory text that may be inappropriate, or to make minor modifications to the form.
- If a new version of the underlying PDF document is produced, then the entire process needs to be repeated on the new document. Copying and pasting fields may help in some cases.

Field Overlay Receipts Using Texcel

Texcel FormBridge is a third-party product that can intelligently recognize fields in an existing PDF document. In many cases, this results in a very quick conversion of existing PDF documents into AcroForms which can then be imported into Manager. Some manual cleanup is almost always required. This is particularly useful when large numbers of PDFs need to be converted.

In addition, Texcel will also produce a list of fields and their attributes. These field definitions can be imported into Composer, and uses as the basis for the form design in Composer. This can accelerate the production of forms in Composer, as well as automate the mapping of fields between Composer and the generated AcroForm.

The effectiveness of the conversion is dependent on the structure of the original PDF documents. It is suggested that this feature be tested against a representative sample of documents before being used on large scale.

Pros

- Automated recognition of fields can save a lot of time.
- The recognized fields are used both to generate a AcroForm PDF, and to import into Composer as a starting point for building the interactive form.
- In addition, the boilerplate text such as instructions, product disclosures and legal copy are also able to be imported into Composer, reducing the amount of effort in doing this.
- Strong and capable editing environment.
- The underlying PDF document is converted into lines and text. This means that the PDF itself is editable, if this is required.

Cons

- Texcel incurs additional cost to convert each page of the original PDF.
- Because the underlying PDF is converted, there may be some differences between the original PDF and the generated one. The differences, if any, will depend on how the original PDF was constructed. Some configuration options in Texcel can be tuned, but ultimately, Texcel should be tested on sample documents.

Custom Receipt Generation

Any third-party tools or systems can be used to generate PDF receipts, as long as these systems have an interface that can be invoked from Manager.

Pros

- No limits as to what can be achieved.

Cons

- The third-party tools systems must be purchased, provisioned and managed.
- A Manager service must be developed that can invoke the receipt service.
- PDF receipts can be quite large. If the system is not hosted close to Manager, this may incur processing delays or bandwidth costs.

Next, learn how to configure receipts server node.

Generate Multiple PDF Receipts from a Single Submission

Sometimes it is necessary to generate multiple different PDF receipts from a single submission. There are several use cases for this, for example:

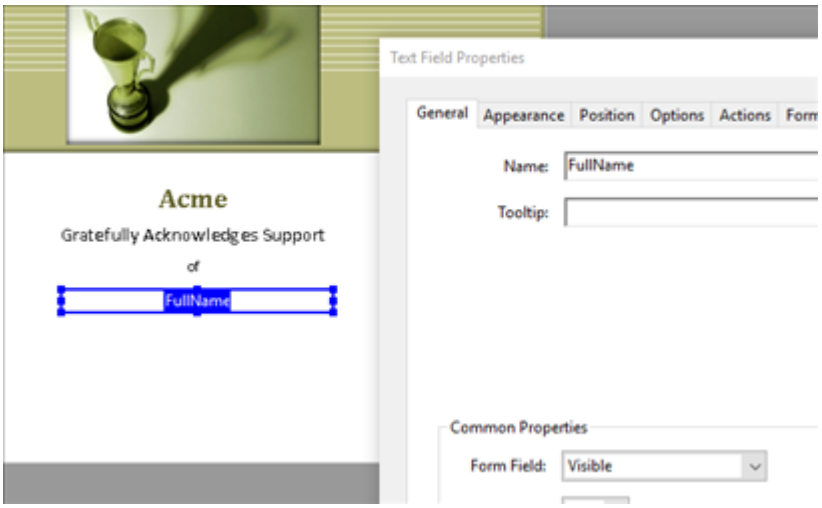
- You may want a different PDF receipt depending on some data in the form. For example, a slightly different variation of the receipt depending on which state the applicant is from.
- You may want to generate multiple separate documents, and concatenate them together. For example, a single application form may result in both a credit card and a check account PDF to be generated. You may want to concatenate these two documents together, or perhaps just provide the two separate documents to downstream business processes.

For this particular example, we are going to generate a certificate for a course attendee. When the user fills in the form, they will be able to choose between different PDF receipts, and green receipt, and a receipt with a picture of flowers on it.

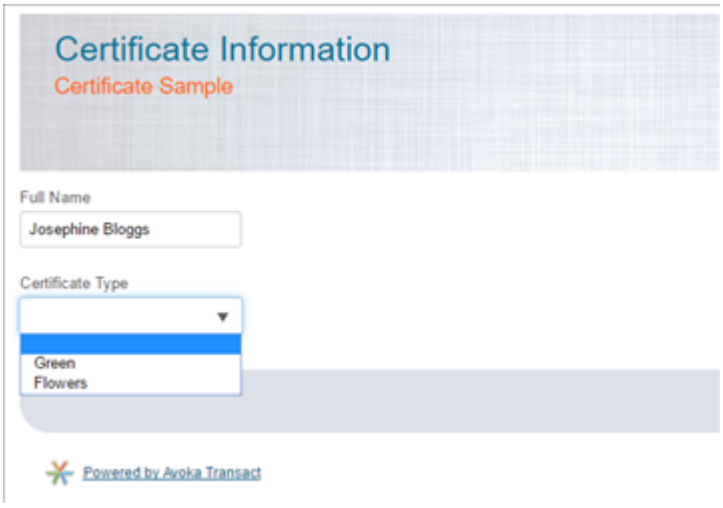
In this case, we are going to generate “pixel perfect” or AcroForm receipts. Please refer to this article for more information about AcroForm Receipts: How to import an Acrobat PDF form as a Transact receipt template (AcroForms).

The code is slightly different if you want to generate other types of receipts, but the principles are similar.

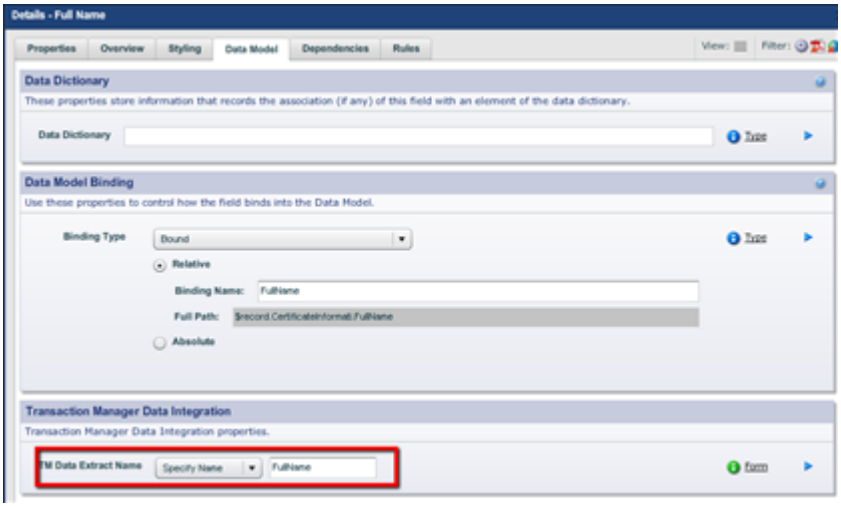
Each of the Acroforms has a single Acroform field placed on it – this field is named “FullName”. For consistency, both forms have the identical field name – but in a real scenario there can be a combination of common fields and form-specific fields. You just need to make sure that the data entry form provides all the necessary fields.



The data entry form has two fields, a field for the person’s name, and a dropdown containing the values “Green” and “Flowers”.



Each of these fields has a data extract name. We will use the data extract values for binding to the Acroform, as well as for the logic in our script.



The techniques for creating and generating an AcroForm receipt is described in the AcroForm article. But for the multiple receipt example, we have to solve two additional problems.

- We need somewhere to store the two (or more) PDF receipt template files. By default, Transact Manager only stores one receipt template per form.
- We need to create a custom receipt rendering service. By default, the various PDF rendering receipt services that Transact Manager provides only generate a single receipt.

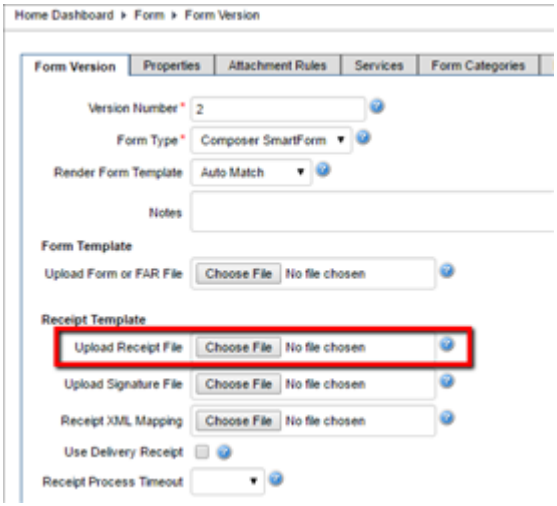
Storing multiple PDF receipts

In order to store multiple receipts, we will create a “dummy” form in Transact Manager for each receipt. This form is not intended for data entry, and it will never be visible externally. It’s simply used as a place to store the receipt.

There are two ways to do this. Either:

- Deploy your data entry form twice, under two different form codes. Or
- Deploy your data entry form once. Then go into Transact Manager, and manually create a new form. You will need to upload a data entry form when you create this form, because Transact will not allow a form to be created without a data entry template. However, this can be any form, since the data entry form will never be visible.

Once you have defined your data entry form plus your additional “dummy” forms, you can upload the AcroForm PDF receipts to each of these form codes as shown below:



Custom Rendering Service

The Groovy code is shown below. This is an example of how you might write this script, and of course your needs may vary. The code is hopefully reasonably self-evident for Groovy programmers. You may want to refer to http://itextpdf.com/ for more details about the iText PDF library which does the work of merging the field data into the AcroForm.

```
/* Render Receipt Groovy script.
   Script parameters include:
       form : com.avoka.fc.core.entity.Form
       request : javax.servlet.http.HttpServletRequest
       submission : com.avoka.fc.core.entity.Submission
       serviceDefinition : com.avoka.fc.core.entity.ServiceDefinition
       serviceParameters : Map<String, String>
   Script return:
       a PDF receipt document of type com.avoka.fc.form.service.DataDocument
*/
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.fc.core.dao.DaoFactory
import com.avoka.fc.form.service.DataDocument
import com.itextpdf.text.pdf.AcroFields
import com.itextpdf.text.pdf.AcroFields.Item
import com.itextpdf.text.pdf.PdfReader
import com.itextpdf.text.pdf.PdfStamper
import org.apache.commons.lang.Validate
logger.info "Start render"

final DATA_EXTRACT_CERTIFICATE_TYPE = "Certificate Type"
final FLOWER_FORM = "certificate-flowers"
final GREEN_FORM = "certificate-green"

// Get the Data Extract
Map<String, String> dataExtractMap = submission.dataExtractMap
logger.info "dataExtractMap=" + dataExtractMap

// Using the Certificate Type field data to determine which form code to use
def certificateType = dataExtractMap.get(DATA_EXTRACT_CERTIFICATE_TYPE)
def formCode = ""
if (certificateType == "Green") {
  formCode = GREEN_FORM
} else {
  formCode = FLOWER_FORM
}
logger.info "formCode=" + formCode

// Get the appropriate form object
def form = DaoFactory.getFormDao().getFormByFormCode(formCode.toLowerCase())
```

```
// Call the function to render the AcroForm and return the result
def receiptDocument = renderAcroForm(form, dataExtractMap)
logger.info "End render"
return receiptDocument

// This function uses iText to inject the fields from the data extract map into the
AcroForm receipt
def renderAcroForm(form, dataExtractMap) {
    final CONTENT_TYPE_PDF = "application/pdf"
    Validate.notNull(form, "form parameter is null")
    Validate.notEmpty(dataExtractMap, "dataExtractMap parameter is null")

    def templateVersion = form.getCurrentVersion()
    Validate.notNull(templateVersion, "form.currentVersion parameter is null")

    byte [] templateData = templateVersion.getTemplateVersionData().getReceiptFileData()
    Validate.notNull(templateData, "form.-
currentVersion.templateVersionDate.receiptFileData parameter is null")

    // Create the output stream
    ByteArrayOutputStream stream = new ByteArrayOutputStream(256 * 1024)

    // Setup the PDF receipt template
    PdfReader templatePdfReader = new PdfReader(templateData)
    PdfStamper pdfStamper = new PdfStamper(templatePdfReader, stream)

    // Bind the AcroForm field values
    AcroFields acroFields = pdfStamper.getAcroFields()
    Map<String, Item> fieldMap = acroFields.getFields()
    for (String fieldName : fieldMap.keySet()) {
        if (dataExtractMap.containsKey(fieldName)) {
            acroFields.setField(fieldName, dataExtractMap.get(fieldName))
        }
    }

    // "Flatten" the AcroForm with the bound field data - the fields will no longer be
editable
    pdfStamper.setFormFlattening(true)
    pdfStamper.close()
    return new DataDocument(stream.toByteArray(), CONTENT_TYPE_PDF)
```

Next, learn how to configure receipts server node.

Import an Acrobat PDF Form as a Receipt Template (AcroForms)

Some organizations have invested heavily in creating glossy PDF forms which are designed to be printed, filled, and posted in, or alternatively opened, filled electronically, and saved and emailed. Sometimes these assets can be re-used to ensure that:

- downstream manual processing processes do not need to be modified, or
- customers are provided with a copy of their application which meets marketing guidelines
- Wet signature processes can be implemented, using copy which has already been ratified by legal

This guide describes how an existing "print & fill" PDF forms can be used by Manager as a pixel-perfect, "artwork PDF" template, without writing any code. This technique requires Adobe Acrobat (additional licenses); Other techniques exist and are not covered by this guide.

What is an AcroForm?

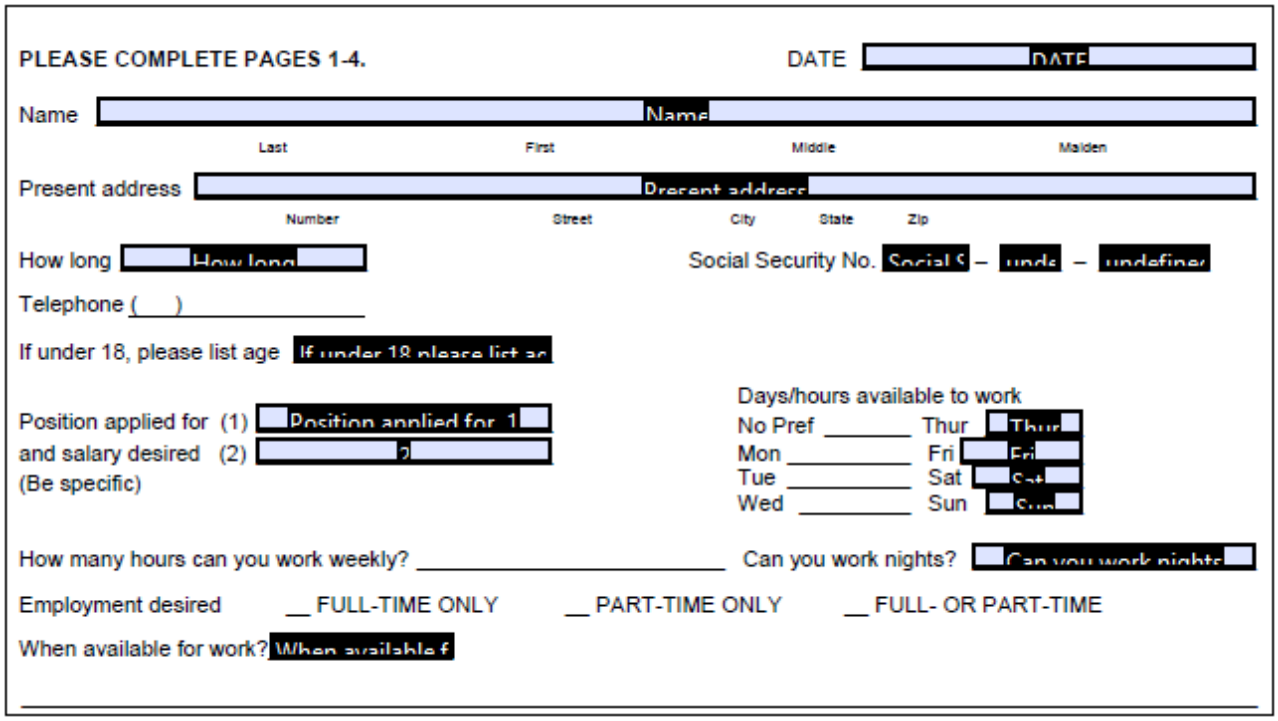
Forms which are created in Acrobat are called "AcroForms"; some PDF readers (Including Acrobat Reader) are capable of rendering these files as interactive PDFs, with features such as fields and radio buttons, and allows saving of data within the file. These should not be confused with interactive XFA forms which function in a specialized Adobe ecosystem (Adobe LiveCycle Designer ES & only Adobe Reader).

Manager re-purposes these PDFs as Templates associated with HTML5 forms on Manager, where HTML5 field data is merged with the template after submission for use as a Document of Record.

High Level Overview of steps required

At a high level, you need to follow these steps

1. Create your form in Composer, using the new form wizard. Drag and drop pages, sections, and fields, and add business rules from the palette, as you normally would. This produces your default dynamic PDF receipt, which will be discarded later. Publish your form to Manager, as you normally would.
2. Open the PDF form in Acrobat, and run the "Prepare Form" wizard. This will create something which looks like this:



PLEASE COMPLETE PAGES 1-4. DATE

Name

Last First Middle Maiden

Present address

Number Street City State Zip

How long Social Security No. –

Telephone ( )

If under 18, please list age

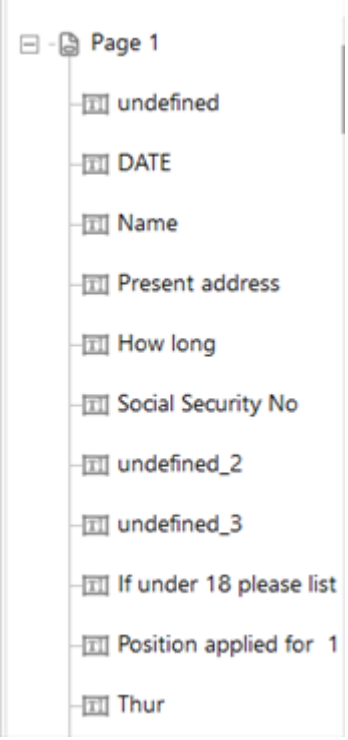
Position applied for (1)
and salary desired (2)
(Be specific)

Days/hours available to work
No Pref _____ Thur
Mon _____ Fri
Tue _____ Sat
Wed _____ Sun

How many hours can you work weekly? _____ Can you work nights?

Employment desired __ FULL-TIME ONLY __ PART-TIME ONLY __ FULL- OR PART-TIME

When available for work?



3. In Acrobat, correct any issues. You may need to manually add fields, or rename them to make more sense. If you have authorization to make changes to the form, now would be a good time to replace "comb" fields with regular fields, and remove any character place-holders such as parentheses in a telephone field, or slash characters in a date field.

4. In Composer, add Data Extracts to each field. The name of the data extract must match the name of the field declared in Acrobat. Additionally you will need to add some invisible fields (Data Fields) with calculation rules to reformat any dates or other fields which need conversion before printing.
5. In Manager, open the form version, and upload a receipt file. Then under Services, set the Receipt Render Service to "Static PDF Receipt".

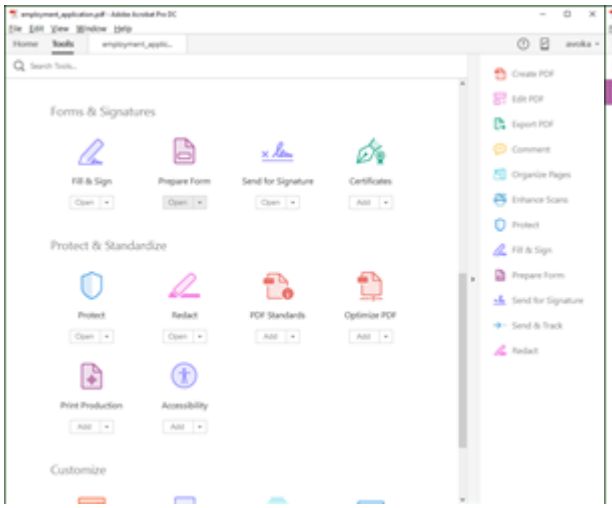
Each of the steps 2-5 is described in detail below:

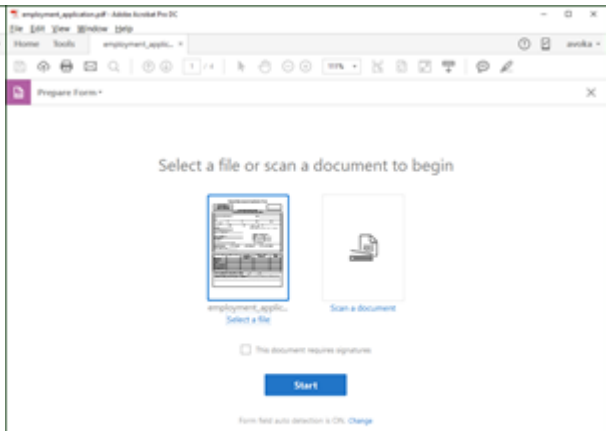
Step 2: Creating AcroForms

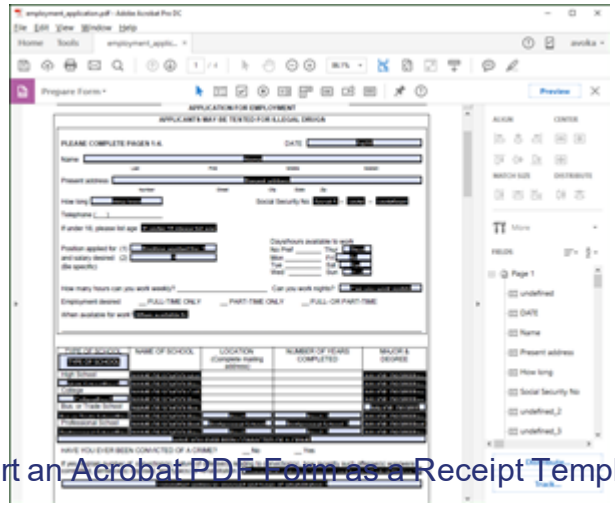
Adobe Acrobat Pro DC (2015 Release)

In order to create your AcroForm using Adobe Acrobat Pro DC (2015 Release) follow these steps:

1. Open the PDF form in Acrobat
2. Go to "Tools", and select "Prepare Form" (Open), under the "Forms & Signatures" heading.
3. Select "Start" (take note that "Form field auto detection is ON" is displayed under the button).
4. You can now "Save As..." and continue to the next section "Prepare Form" in Adobe Acrobat Pro DC.



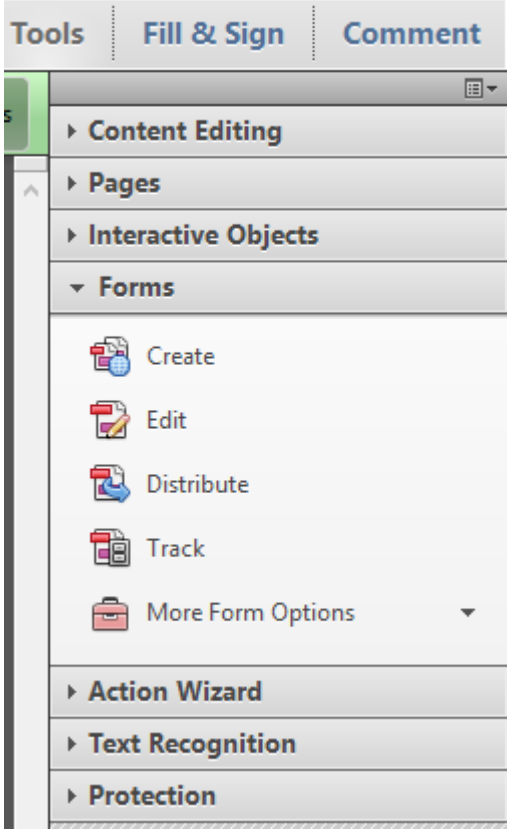


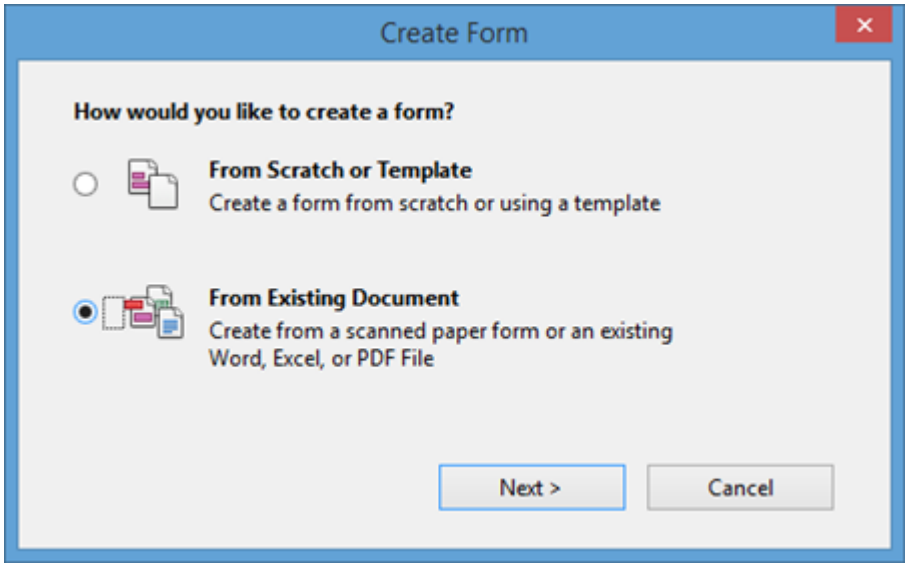


Acrobat XI and below

Acrobat 9 or Acrobat XI users can follow these steps:

1. Start, by accessing the "Forms" menu (under Tools, on the right-hand pane).
2. Select "Create", to start the wizard
3. You can now "Save As..." and continue to the next section.



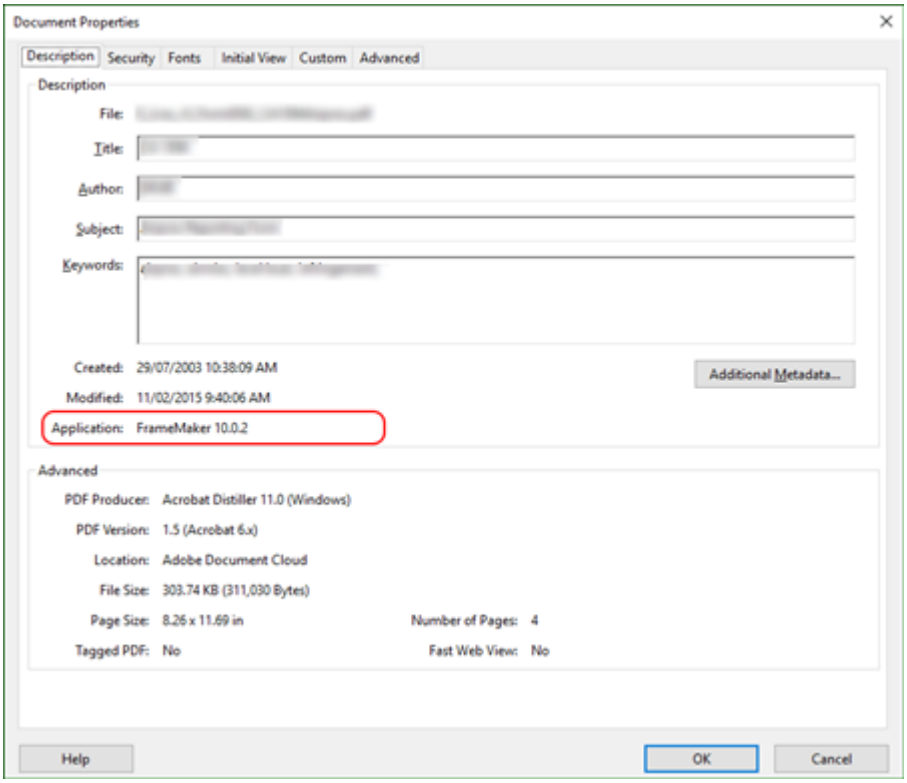


Possible Issues & workarounds

If you have any issues with the above steps, you may need to check what type of PDF you have. There are many types of PDF, some of them may cause some issues. For example:

- Secured PDFs cannot be manipulated in any way without a password.
- XFA PDFs may have a .pdf file extension, but can only be edited with certain Adobe products (Adobe LiveCycle, not covered in this guide)
- Some PDFs may already be converted to an AcroForm, in which case you have less work to do.

The best way to get around any issues is to print the form to PDF using a non-Adobe product such as PDFcreator, CutePDF or your browser's built-in PDF engine. You may need to try some different products to see what works best for you. You can check which application was used to produce it by opening it in Adobe Acrobat (or Reader), and selecting "File" -> "Properties".



Inspecting the file properties in Acrobat

Another way around PDF security or unsupported file types is to either physically print them, or take screenshots, and then use Acrobat's OCR capabilities to scan in the pages.

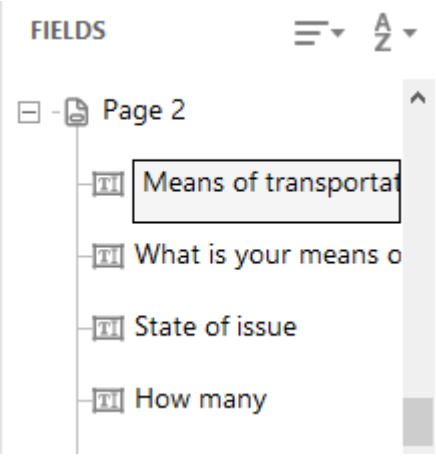
Step 3: Correcting automatic conversion issues

Automatically Detected Fields

For each automatically detected field, you will need to inspect to check:

- Field size, alignment & centring
- Font (Acrobat often chooses large and ugly fonts)
- Field names (it will become helpful to correct any auto-naming of fields to be something that is human readable)

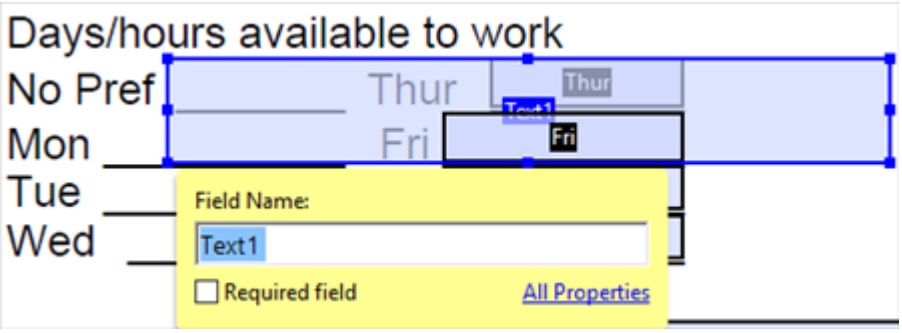
The toolbar on the right will help you quickly rename and reorganise all the interactive objects on the page. You can also bulk edit by selecting more than one field and right-clicking to access the properties panel.

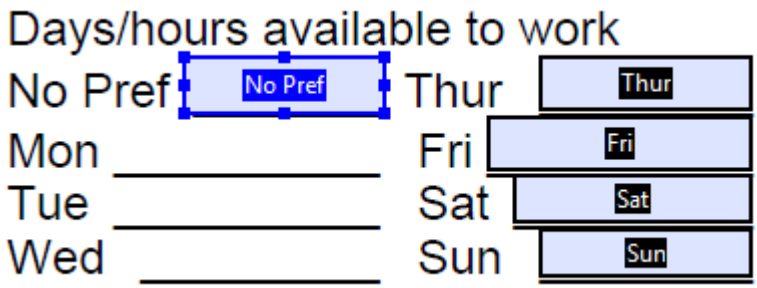


Acrobat Pro DC - Fields Toolbar

Missed Fields

Sometimes the Acrobat forms wizard misses fields, and you will be required to add them manually. In Acrobat DC, there's a toolbar on the top of the page to add fields, checkboxes and radio buttons. Dropdown lists should be added as plain text field.





Adding a missing text field

When you are done, please think about "global" properties such as fonts. You can select all the fields on the form, and edit them in bulk, to ensure consistency.

Dealing with character-separated fields (Dates / phone numbers with area code)

One of the issues that you will encounter is how to deal with character separated fields, such as dates: " / / ". You have two options on how to deal with this: *Override* the PDF design or *re-factor* the data in Composer.

The easiest option and the option with the best result is to add fields with an opaque (colored) background that cover up the character separators in the original PDF design. This may not be permitted in all circumstances, as the business may mandate that no changes are made to the layout (EG: OCR in downstream processing may fail if you change the exact layout).

If you cannot override the field layout, then you will be required to add 3 separate text boxes for the date (Day, Month, Year), ensure that all values fit in the spaces created.

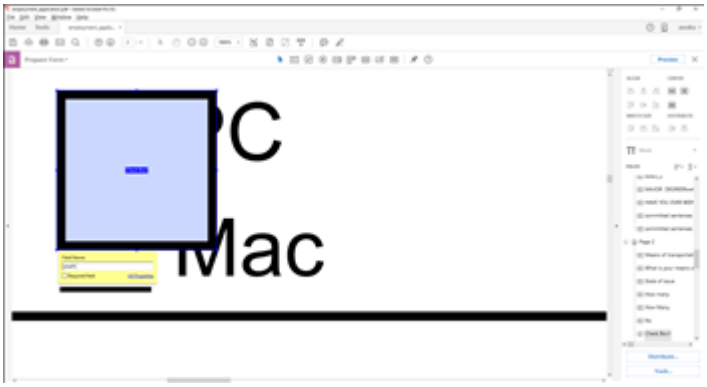
Re-factoring the format of field values

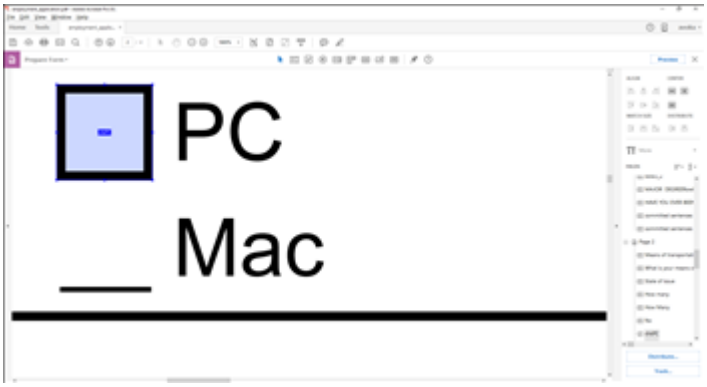
With some fields, you will be required to do some re-factoring of the field values. For example, Manager stores dates in a format of YYYY-MM-DD, and you will want to re-order this to be

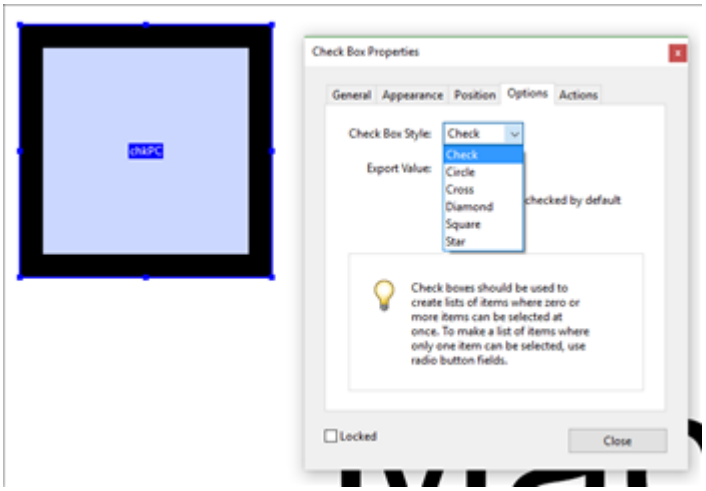
human readable, or to be displayed in 3 separate fields (pre-formatted character separators). This configuration is performed in Composer, and is described in the next step.

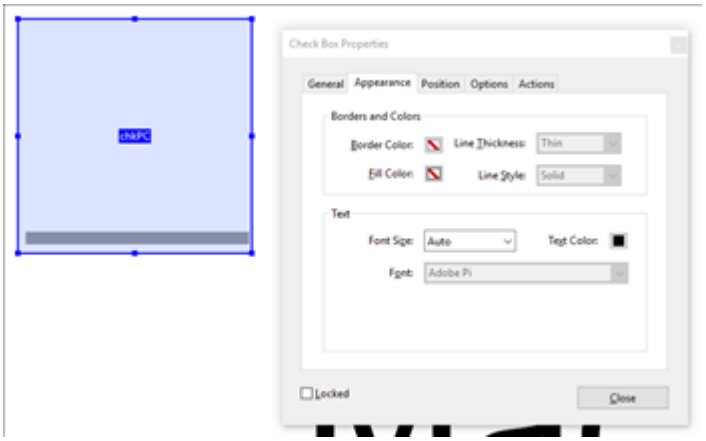
Check Boxes

Check Boxes are quite straightforward. Simply add them, style them, and name them appropriately (Choose a convention which makes sense, such as prefixing the name with "chk"). You will need to resize them to fit in the correct area, and the best way to achieve a neat and accurate layout is to zoom right in to the part of the page where you're going to place it. Acrobat allows you to align, centre, match size and distribute objects for perfect layout. By default, Acrobat uses a check (tick) character within a box, but you can change this to a cross by editing the Check Box Properties, Options tab: Change the "Check Box Style" configuration (You may want to "Bulk edit" this setting, by selecting all your check boxes to edit). There are many layout options including border and fill color.





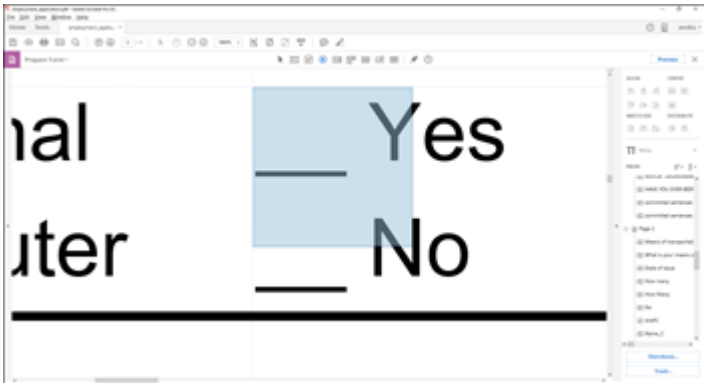


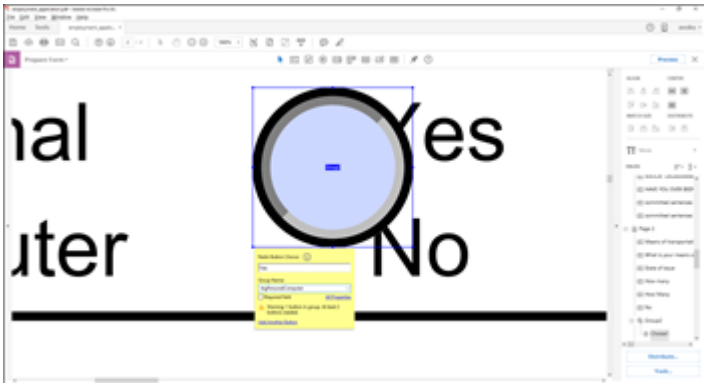


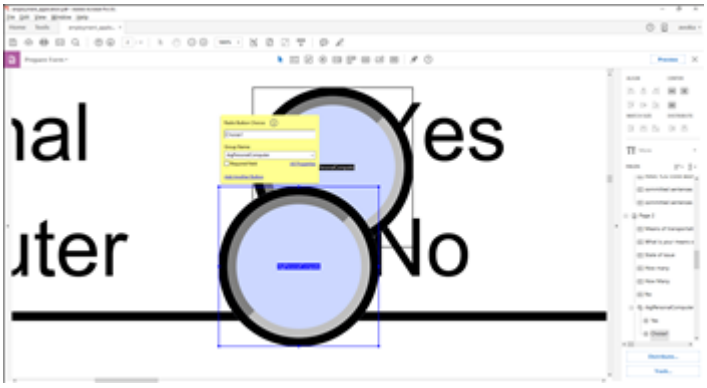
Adding Check Boxes to a form & configuring the appearance

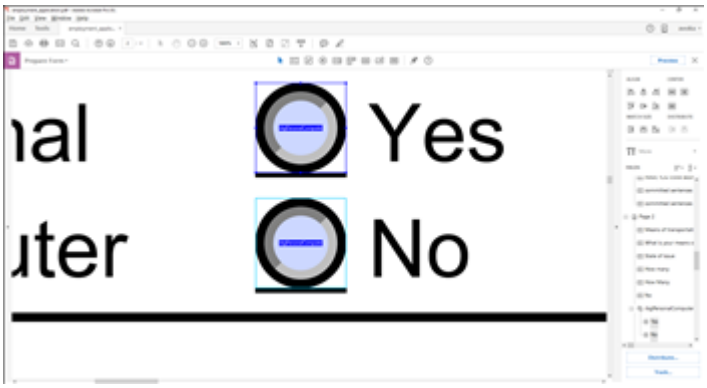
Radio Buttons

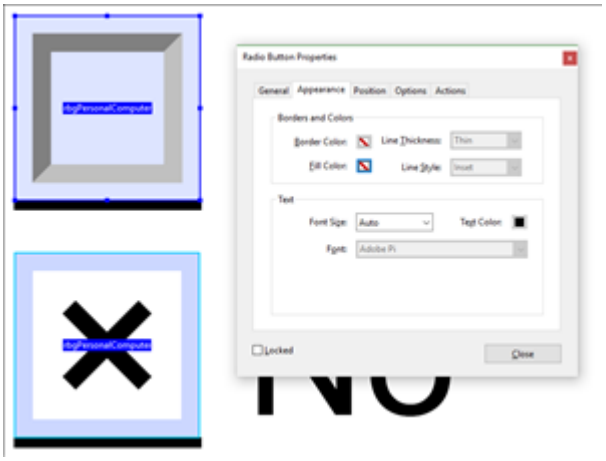
Radio buttons require membership in a radio button group. Acrobat attempts to help you with this by providing a wizard which helps you add multiple options in a list, but you will most likely need to re-factor the configuration to fit into a structure in the "Fields" editor. There is a limitation here: The name of the Radio Button Group should not contain spaces in order to be compatible with Composer.


Yes
No



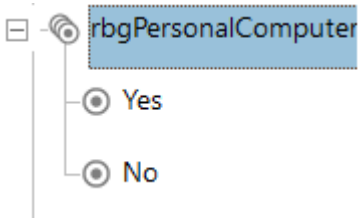



Yes
No



Adding a radio button group and configuring appearance

When you are done, your Fields tab should look something like this:

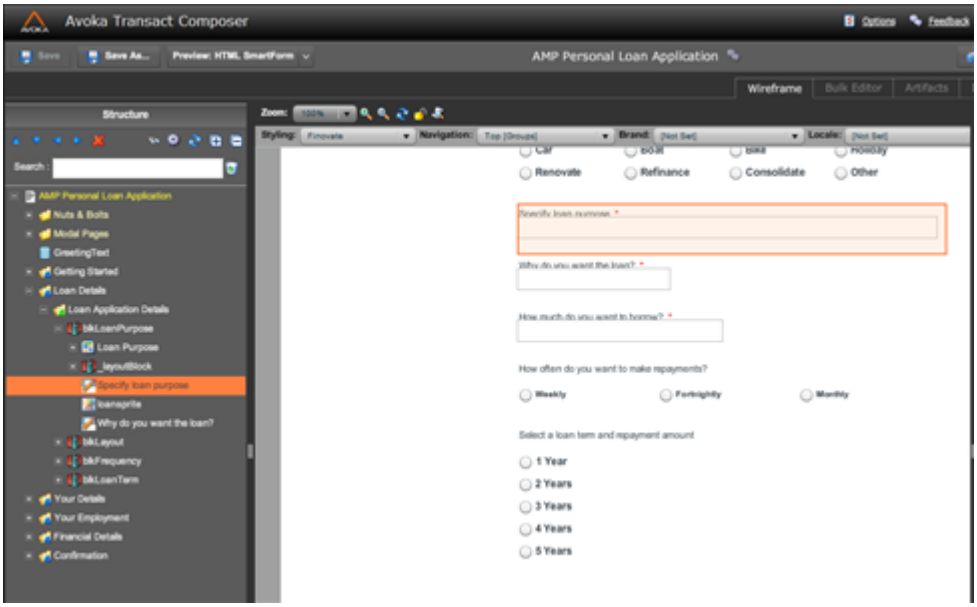


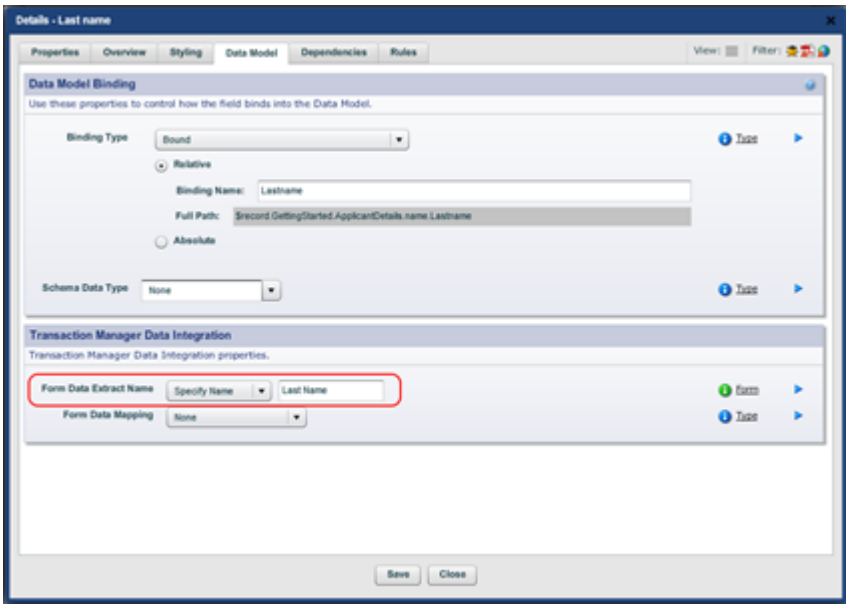
Step 4: Composer Configuration

In Composer, you will be required to perform two tasks:

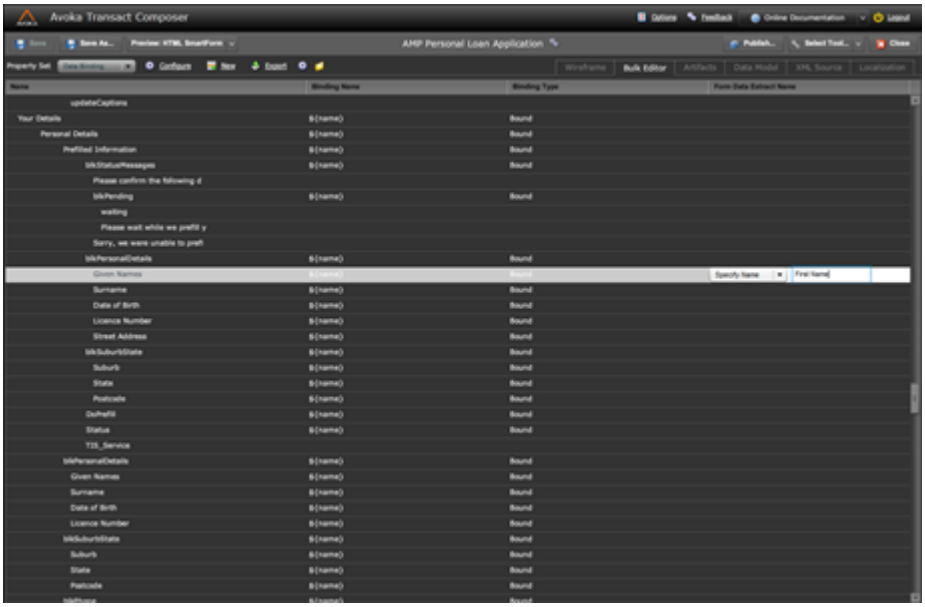
1. Bind Composer fields to PDF fields using Data Extracts
2. Re-factor any formatting of field values (such as dates)

To configure a data extract, you can either inspect the form wireframe/structure, or you can use the bulk editor, which displays the form fields in a table. The bulk editor is a great way of setting Data Extract properties in a tabular interface. To enable it, select the "Bulk Editor" tab, then change the property set to "Data Binding". Scroll down past all the internal fields (at least 10 screen pages), then click in an empty space for the drop-down list to appear.





Setting the Data Extract Name using the wireframe and property editor



Setting the Data Extract Name using the Bulk Editor

To configure data extracts from the wireframe/structure, double click on the field, then go to "Data Model", and set the "Form Data Extract Name".

Whichever process you choose, it is important to:

- Select "Specify Name" in the drop-down list
- Type in the name of the field. This name must match the name defined in the PDF form defined in Step 3 above.

Re-factoring the format of field values

With some fields, you will be required to do some re-factoring of the field values. For example, Manager stores dates in a format of YYYY-MM-DD, and you will want to re-order this to be human readable, or to be displayed in separate fields to be printed in pre-formatted fields with character separators. Follow these steps to ensure data is in the format expected by the PDF form.

1. Add a "Data Field" to your form, adjacent to the field in question. A data field is just an invisible field.
2. Add a scripted calculation rule to the data field.
3. Add some JavaScript to perform to parse and format the date or alternatively add string operations on the dependent field (See sample code below)
4. Add the Data Extract to the Data Field instead of the original user interface element

Reformatting using the Composer library

```
return sfc.parseDate({date},"DD-MM-YYYY").format("DD/MM/YYYY")
```

The above line of code encapsulates several operations in one line. If you are new to JavaScript, you may wish to analyse the sample code below, which achieves the same outcome using string operations

Example code (for beginners)

```
var date = {date};                  // Link to date field as a dependency
var dateArray = date.split("-");    // "YYYY-MM-DD"  ==>  Array [ "YYYY", "MM", "DD" ]

var day   = dateArray[2];           // The 3rd element in the array "DD"
var month = dateArray[1];           // The 2nd element in the array "MM"
var year  = dateArray[0];           // The 1st element in the array "YYYY"

return day + "/" + month + "/" + year;  // "DD/MM/YYYY" (adding strings will concatenate
them)
```

You may wish to extend this example with other types of fields. E.G.: you can use the same technique to re-factor phone numbers (use str.substring(0, 1); to get the first 2 digits in the string).

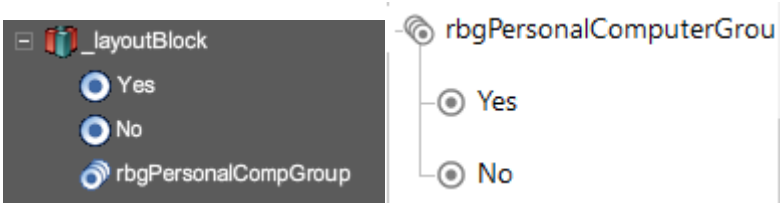
Radio Buttons & Check Boxes

Check boxes in Composer should be configured the same way as text fields: with the Data Extract name matching the AcroForm field name.

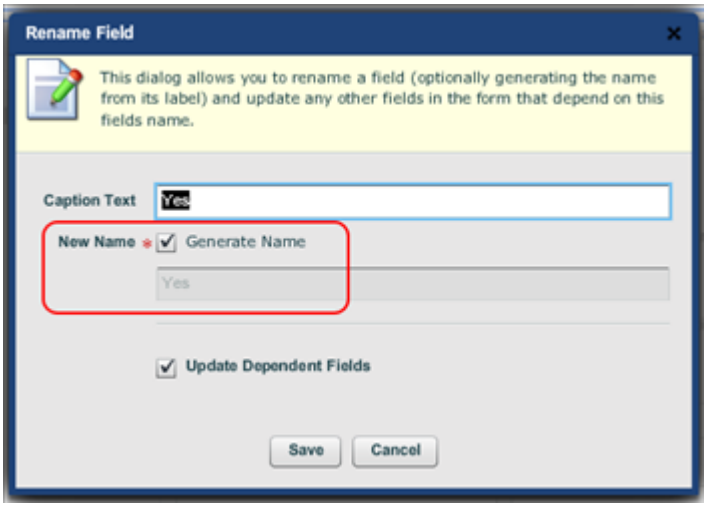
Radio buttons require a little extra care, ensuring that:

- The Composer Radio Button Group **data extract** matches the AcroForm Radio Button Group **name**
- Each Composer Radio Button **name** (not caption) matches the AcroForm Radio Button **names** exactly (check this for each radio button)

It's usually easier to perform modifications in Acrobat, conforming to Composer design.



Composer structure and Acrobat structure with matching names

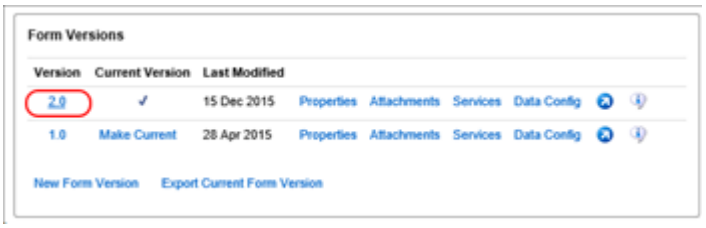


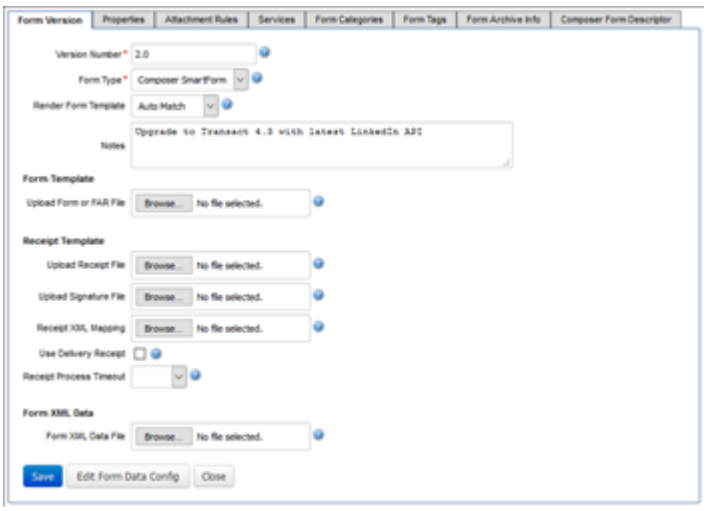
*Note that the **name** of the field does not necessarily match the **caption***

Step 5: Manager Configuration

When you have finished editing the AcroForm, it can be uploaded into Manager for use as a PDF Template for the receipt. First though, please ensure that you have published any changes out of Composer.

In the form dashboard, you can upload a receipt file using the form version dialog as shown below.



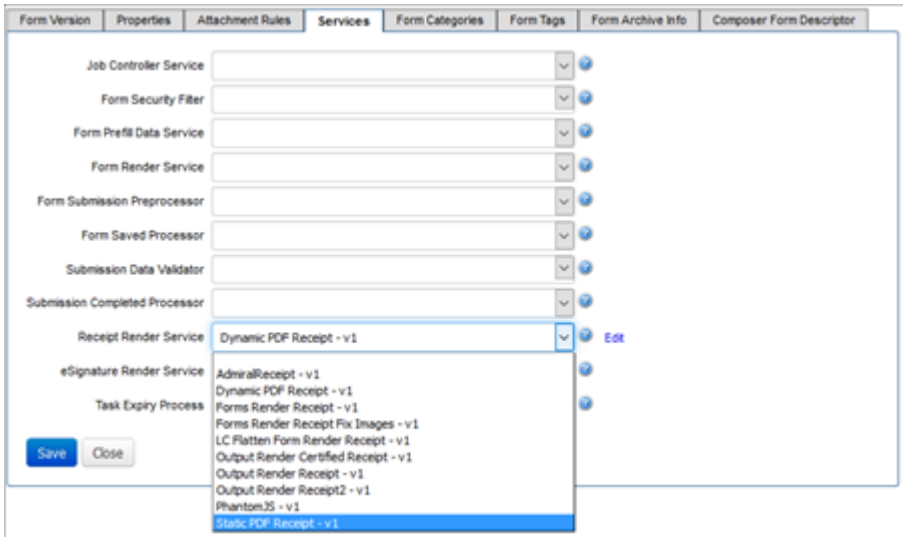


To import the file:

1. Log into Manager.
2. Under Forms, find the form you wish to update.
3. When you select the form you will see the Form Dashboard. On the top-right corner, you will see the Form Versions dashboard element (above)
4. Select the current version (highlighted in red above)
5. This will display the "Form Version" dialog. Under "Receipt Template", select the browse button adjacent to "Upload Receipt File".
6. Attach the AcroForm PDF file and save

Now that the file has been uploaded, it needs to be used by the receipt service:

1. On the form dashboard, in the form versions dashboard area, select the "Services" link adjacent to the latest version.
2. Change the Receipt Render Service to "Static PDF Receipt"



Extending this solution

A code-free solution can be great for rapid prototyping or building business-oriented solutions quickly, without relying on developers to contribute to the project. If you have access to Groovy developers, you can skip step #3, and and build server-side scripts to link form data Xpaths to AcroForm fields. Any data value re-factoring can then be performed in the custom script, rather than in Composer. Developers then have access to an extended suite of capabilities including:

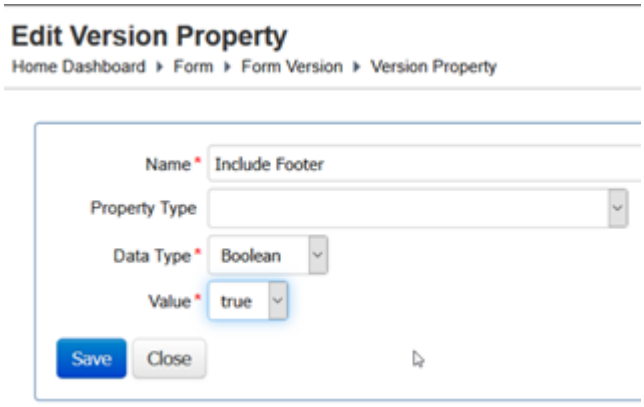
- Concatenation of receipt with other static files (EG: PDS)
- Handling of attachments (as PDF attachments or pages)
- Enrichment of content (adding data from external services)

Next, learn how to configure receipts server node.

Insert a Watermark in a PDF Receipt

You can creating a watermark on every rendered receipt by running a Groovy service, which will render the standard PDF receipt using iText to overlay HTML.

1. Publish your form to Manager, which will perform all the work, so no additional effort is needed in Composer.
2. Go to Services > Service Definitions
3. Import this Groovy Service.
4. Add two form version properties called Include Footer and Include Header i.e. Form dashboard > Properties as shown below:



5. Go to the Form Version > Services and include the Receipt Render Service.
6. Update the following Groovy code to change the position of your watermark:

```
def addHeader = addOverlayContent.curry(
 getTemplate("Header HTML Template"),
   [envName: tmEnv],
   245, 40, 600, 832
);
```

   The coordinates above are: lower left X, lower right Y, upper right X, upper left Y
7. Render and submit your form to test the generated receipt or use the receipt test harness. If it is not working, check the system error log.

If you get this error, check that you have created the two form properties.

Next, learn how to configure receipts server node.

Receipt Generator Service

Manager provides a Receipt Generator service for creating PDF receipts based only on a form design and a form XML data files. The service doesn't require any form submission data, as opposed to the Dynamic PDF Receipt service and the Chromium Puppeteer Receipt service, so you can conveniently create a receipt on demand. This is usually to render Maestro forms. Manager doesn't store generated receipts so they are discarded after use.

The Receipt Generator service supports only the Puppeteer[1] and the PhantomJS[2] headless internet browsers.

To run the service against a form, the form must belong to an organization and a user must have that organization or form assigned to them to call the service.

You can generate a receipt using one of following options:

- Fluent API
- REST API

Generate Using Fluent API

You can use any rendering service or create a new one using Fluent API.

The example below illustrates how to generate a receipt as an array of bytes:

---

[1]Puppeteer is a Node library which provides a high-level API to control headless Chrome or Chromium over the DevTools Protocol. It can also be configured to use full (non-headless) Chrome or Chromium.

[2]PhantomJS is a scripted, headless browser used for automating web page interaction. PhantomJS provides a JavaScript API enabling automated navigation, screenshots, user behavior and assertions making it a common tool used to run browser-based unit tests in a headless system like a continuous integration environment. PhantomJS is based on WebKit making it a similar browsing environment to Safari and Google Chrome (before Chrome's fork of WebKit evolved into Blink). It is open-source software released under the BSD License.

```
import com.avoka.tm.svc.ReceiptGenerator

byte[] receiptBytes = new ReceiptGenerator()
        .setFormCode("formCode")
        .setFormTemplateVersion("1.0.0")
        .setSvcName("serviceName")
        .setSvcVersion("1.0.0")
        .setXml("xmlData")
        .generateReceipt()
```

Generate Using API

The Journey Manager REST API provides the endpoint URL to generate a receipt. The endpoint uses the Basic Auth.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s)://{JM_SERVER}/-
manager/secure/rest/fluentapi/receiptgenerator/generateReceipt
```

Parameters

`setXml`: a form content as XML that is used to generate a PDF receipt of the form.

`setFormCode`: a form code.

`setFormTemplateVersion` - a form template version.

`setSvcName`: a service name.

`setSvcVersion`: a version of a service

Response

If successful, this operation returns a response with a receipt encoded as Base64 and a HTTP `200 OK` status code.

If an error occurs, the Can't resolve the form, Invalid XML (a form XML data must include AvokaSmartForm) or Unauthorized message is logged into the Error Log.

Example

```
curl --location --request POST ‘http(s)://{JM_SERVER}/-
manager/secure/rest/fluentapi/receiptgenerator/generateReceipt’ \
        "setXml": "<?xml version=.....", \
```

```
"setFormCode": "RECEIPT-GENERATOR", \
"setFormTemplateVersion": "0.1.0", \
"setSvcName": "Dynamic PDF Receipt", /
"setSvcVersion": "1.0.0"
```

Next, learn how to regenerate a receipt.

Regenerate a Receipt

Journey Manager allows you to re-generate a receipt after it has been generated and stored in a database, either with the Azure or AWS Storage services, by using the `ReceiptSvc` Fluent API methods. The receipt generation is a background process, and it may take some time to execute, so you can't assume that the receipt will be generated immediately.

You can set the receipt status to Ready, so the receipt generation background process will start and create the receipt. This is shown below:

```
new TxnUpdater(param.txn)
   .setReceiptStatusReady()
   .update()
```

To obtain at the generated receipt (as an array of bytes), you should use the following code:

```
Txn updatedTxn = new TxnQuery()
   .setId(submission_oid)
   .withReceiptPdf()
   .firstValue()

byte[] receiptBytes = new ReceiptSvc()
   .setTxn(updatedTxn)
   .renderReceipt();
```

If you use Azure Submission Data Storage service to store receipts in a database, you need to select the Overwrite Receipt On Upload checkbox (`overwriteReceiptOnUpload(true)`) to allow receipt overwrite. If this is not set, an error will be thrown.

To simplify the process and make it generic, you can use the `setForceUpdate(boolean)` method to regenerate a receipt for either type of storage services. The method takes a Boolean value, which you use to retrieve a previous rendered receipt (false), or a new receipt with the newly added form data (true). | 24.04 This feature was introduced in 24.04

The typical example is shown below:

```
...
byte[] receiptBytes = new ReceiptSvc()
   .setTxn(updatedTxn)
   .setForceUpdate(true);
```

TIP

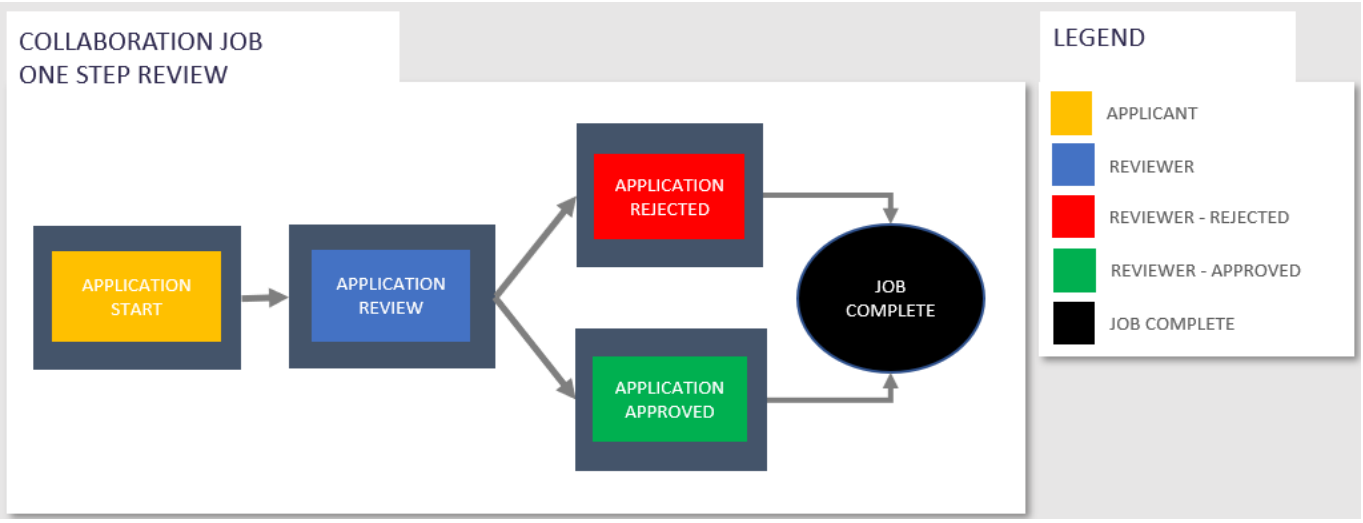
You can use the Receipt Generator service to generate (and regenerate) a receipt on demand.

Next, learn how to configure receipts server node.

Collaboration Jobs Overview

Journey Manager comes with the collaboration job framework for building complex and robust workflows[1] without the need to use Business Process Modeling[2] tools. The framework is easy to use, so the collaboration jobs can be configured by non-developers.

A typical workflow can be an application review and approval process, which enable several people to work collaboratively on a user submitted form to either approve or reject it, as shown below.

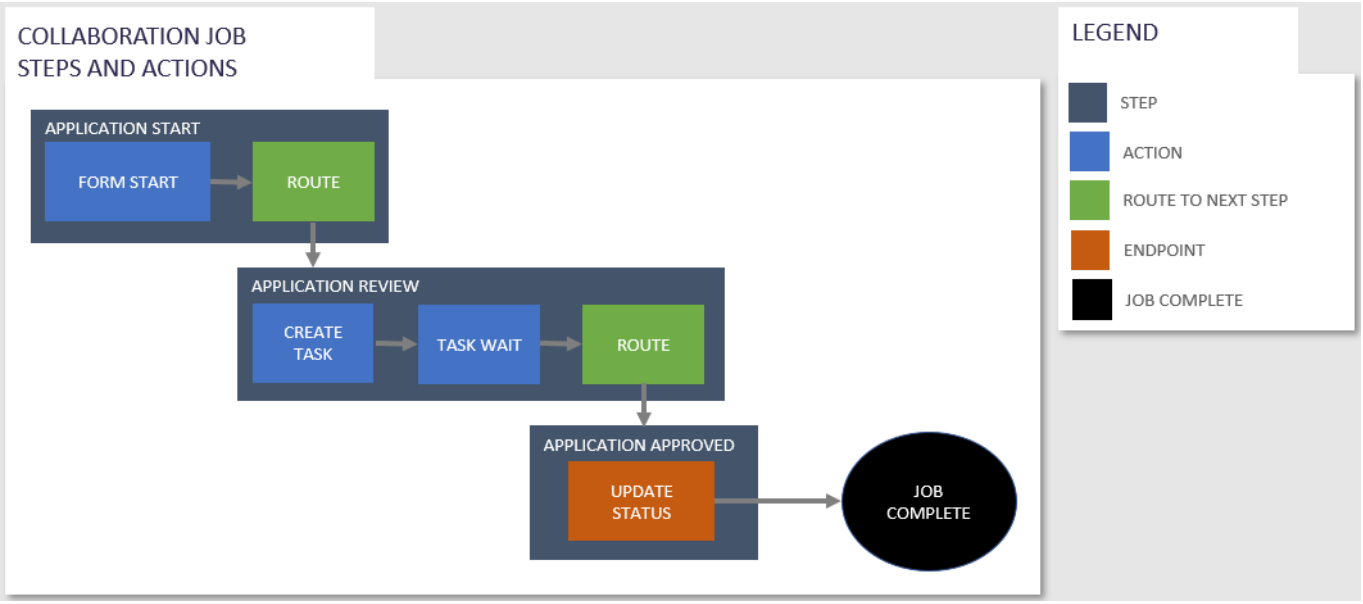


A collaboration job controls a workflow of transactions that represents user interaction with a form. A collaboration job can be considered as a series of tasks that Manager creates for users to complete, review or reject a form. A collaboration job consists of several steps with each step containing actions or endpoints. Steps are not ordered, but actions are sequential within each step. The sequence of steps and actions defining a form approval workflow scenario is shown in the diagram below.

---

[1]A workflow is progression of steps (tasks, events, interactions) that comprise a work process, involve two or more persons, and create or add value to the organization's activities. In a sequential workflow, each step is dependent on occurrence of the previous step; in a parallel workflow, two or more steps can occur concurrently.

[2]Business Process Modeling Notation (BPMN) is a method of illustrating business processes in the form of a diagram similar to a flowchart.



This diagram represents the happy path[1] scenario where the instance of a collaboration job is approved. The collaboration job is routed through three of the four steps shown above. An applicant fills and submits a form which starts the collaboration job. The first step being the Application Start, which has one action - the Form Start - and it links this initial submission made by the applicant to the collaboration job. The collaboration job processing completes this action and immediately routes to the next step - the Application Review step - which has two actions - the Create Task, which creates a group task assigned to a review group, and the Task Wait which waits for the task to be submitted by a user. A user in the review group opens the task, review the submission, selects the Approve option to approve it, and submits this approval. This completes the Task Wait action which takes the Approve route to the next step - the Application Approved. This step is an endpoint which has one action, the Update Status, that notifies the applicant that there submission has been approved. Once all the actions have completed in an endpoint step, the collaboration job completes.

A collaboration job's workflow is specified in the job definition of a job controller service as a JSON[2] document. You can configure the job definition to change its execution flow. The usual way to initiate a collaboration job is to setup a form with the corresponding job controller service. When a form user submits the initial form, it will start a collaboration job, which in turn creates tasks and assigns them to various users, groups or email addresses (for anonymous users) in a

---

[1]A happy path is a default scenario featuring no exceptional or error conditions.
[2]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

review and approval flow. Alternatively, tasks can be assigned back to the original submitter to have them fill in new forms which is a form bundles flow.

> NOTE
>
> A collaboration job can be started from an API call without requiring an initial form submission.

For submissions in the Open or Saved states, you can start a collaboration job earlier in the initial journey flow, that is before a user submits a form. Using the `JobBuilder` Fluent API - `setInitialTxnTrackingCode()` - you create a job linked to such submissions any time after the initial form request, which provides better flexibility in orchestrating user's journeys. For more information, see Job Details. | 24.10 This feature was introduced in 24.10

A collaboration job has two types of execution mode. The standard mode, which is used for reviews and approvals, has the processing kicked off by a standard scheduled job called the Collaboration Job Controller - Scheduled Job, which runs every minute by default. The other mode is used for form bundles and has the job processing kicked of at the end of a submission processing. This is allows for collaboration jobs to run immediately and then present the a newly created form task back in the users browser.

Maestro forms have a number of custom widgets that interact with collaboration jobs. In addition, form fields can be mapped in Maestro to Data Extracts. Collaboration jobs use data extracts in preconditions to determine run a step or an action and also in the displaying of tasks details to the user. Finally, when a task is opened it contains job context information in the form XML, such as Job Ref Number, Current Step and Available routes. This populates widgets as well as controlling the visibility of form sections.

Manager allows you to track collaboration jobs, manage their details and check related transactions. You can also create custom job services.

Typically, there are two groups of Manager users that will work with collaboration jobs - developers and system administrators. You should configure and grant them the following permissions:

- Collaboration Job View
- Collaboration Job Edit
- Collaboration Job Remove

These permissions let developers to view collaboration jobs and enable the Event and Error tabs, which are useful for testing collaboration jobs. It allows them to search for the test job and drill down to review each step and action.

System administrators should be also able to move forms and collaboration jobs between environments, to maintain and assign users to the correct groups and roles. This will let them to:

- Search on a job and see what step it is up to and then drill down into it for more detail.
- Look at a action task and see who the task is assigned to.
- View form receipts, form XML data for completed submissions.
- Review error status.

Next, learn about collaboration job modes.

Collaboration Job Modes

Manager has two processing modes for collaboration jobs: standard and process immediate. Depending on the mode, the form bundles, reviews and approvals workflows are processed differently and they also have different licensing implications. The processing modes are configured in a job definition of each collaboration job's job service.

Let's take a look at these processing modes.

Standard Mode

The standard mode is used to execute the review and approval collaboration jobs. When a user completes and submits a form, Manager starts the standard form submission processing, which calls a job and runs a callback that marks it as ready for processing. This collaboration job runs in a thread, with a delay up to 60 seconds, that executes the callback.

When selecting this mode, consider the following:

- Pausing a scheduler job stops the collaboration job controller from processing all collaboration jobs it has been running.
- Form submissions and scheduled jobs are processed in separate threads so user experience is not affected.
- There is a delay, up to 1 minute, from the time when the submission is completed and the time the submission processing is started. This delay can cause some issues for form bundles when tasks are immediately assigned back to the same user, or during review and approval demo in the non-production environment.
- Easier development and debugging process. If a scheduled job fails during execution of an action, you can change this action configuration and retry it until the action is completed.
- Supports automatic retry and recovery.
- Use it when there are many participants in the collaboration job.

Process Immediate Mode

The process immediate mode is used to execute form bundles, such as customer onboarding forms. For example, a new banking customer fills out an initial form which has some common fields and a list of products. They select the products and submit the form. After that, they are presented with a list of forms to complete, one for each product. A scheduled job runs as a user thread that performs the following:

- The standard submission processing.
- The callback to make the job ready for processing.
- The next part of the collaboration job instead of the collaboration job controller.

When selecting this mode, consider the following:

- Pausing a scheduled job stops the collaboration job controller from processing all collaboration jobs it has been running, but it doesn't stop processing of collaboration jobs running in the immediate mode, so they will be completed.
- Using form bundles where there is a single participant completing all tasks.
- Using collaboration jobs with form chaining, but with only one form per each step.
- Running a demo of review and approval collaboration jobs in non-production environments.
- Debugging is difficult, because a transaction is associated with the whole form submission, which is the user's thread. If an error occurs, steps, actions and collaboration job event logs are rolled back.
- The collaboration job processing adds time to the submission completion. The designer needs to be mindful of how long it takes to run the next part of the job to prevent long waiting times for the form user. Form bundles may take an extra 3 - 10 seconds, because they create form tasks, which is an important step before the tasks can be chained to the next available form.

Next, learn how to view collaboration jobs.

Job Services Overview

Job services are specialized services used in Journey Manager to create and process collaboration jobs.

Manager provides different job service types and comes with pre-configured job services and job service templates to help you create new job services.

Job services are defined per an organization. Each organization must have at least one job service configured. There is no limit on the number of job services an organization can have. Each form within the organization can use these job services. You can create new job controllers and job actions to implement your requirements.

Manager provides job service templates, which are enhanced job controller services. Templates allow a user to customize collaboration jobs without having to modify the job service definition. Templates are organization agnostic and you can't edit them.

It's important to understand a difference between a job service and a Transact function. A Transact function is usually used to fetch some data or make a payment. A job service, on the other hand, is used to create a workflow when a form application requires a workflow, for example, for back office employees to verify and review customers' applications. You can't replace a job service with a Transact function. You can create a form that triggers a job service and at the same time the form can trigger a number of fluent functions.

Next, learn how to view job services.

Job Definitions

Manager uses job services to process collaboration jobs. You can define and configure a job service as a single JSON[1] document, which is called a job definition. The job definition includes the list of steps that make up the collaboration job. Each of the individual steps has the list of actions to be performed when the step runs, and the routes property, which defines the next step transition by mapping the route name to the next step.

The job definition consists of the following properties:

- jobDetails
- jobGroups
- steps
- actions
- properties
- routes

jobDetails

The `jobDetails` property defines a collaboration job and how it is processed. The snippet of the `jobDetails` is shown below:

```
{
        "jobDetails": {
                "name": "1 Step Review",
                "processSubmitImmediate": "true",
                "allowEarlyCreation": true,
                "version": "0.1.0"
        },
        "steps":[
                { .. step definition ...},
                { .. step definition ...},
                ...
        ]
}
```

The `jobDetails` has the following attributes:

---

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

`name`

The name of the collaboration job, which is set by the job controller service when the collaboration job has been saved.

If defining multiple jobs, use distinct names for each job definition to ensure you can differentiate between them in the Journey Analytics scope selector.

`processSubmitImmediate`

The collaboration job processing mode. If this attribute is omitted or set to false, the job will use the standard mode. Otherwise, it will use the process immediate mode.

`allowEarlyCreation`

A boolean flag that allows you to create a collaboration job and associate it with a transaction that can only be in the Opened or Saved state.

This requires the Job Controller to be associated with a form version of a transaction and Job Controller service has the `allowEarlyCreation` flag set to true in its job definition. The `processJobAfterCreate` attribute will be ignored in this case.

| 24.10 This feature was introduced in 24.10

`version`

The version of Manager that the job was created in.

Let's look at a simple example of using the above configuration to create a job given the job controller service and version for a non submitted Txn as the initial Txn:

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
Job job3 = new JobBuilder()
   .setSvcName('1 Step Review')
   .setSvcVersion('0.1.0')
   .setReferenceNumber('ABC123')
   .setInitialTxnTrackingCode('ABC123')
   .build()
```

jobGroups

The `jobGroups` property lists groups, which job coordinators can use to manage a collaboration job. The `jobGroups` can be used to show collaboration jobs assigned to each group, for example, Journey Space Reviews. The snippet of the `jobGroups` is shown below:

```
"jobGroups" : [
      {
        "name" : "HL Melbourne",
        "description" : "Home Loan Customer Care Group for Vic"
      },
      {
        "name" : "HL Sydney",
        "description" : "Home Loan Customer Care Group for NSW"
      }
    ]
```

The `jobGroups` has the following attributes:

`name`
: The name of the job group, which will be displayed in the Job Groups dropdown from the Reviews tab of the form space. The name must be less than 100 characters.

`description`
: The description for the job group, which must be less than 200 characters.

The job groups can be created in one of the following ways:

1. If a job group defined in a job definition, as in above, has not been created, then the job group will be automatically created the first time a user submits a form associated to this job. This is the preferred way to create job groups.
2. Job groups can be manually created. By manually creating the job group, you must ensure that the job group names and descriptions match with the job definition. Any mismatch will not be subsequently updated to the job groups.

steps

The `steps` property is an JSON array that defines steps. The snippet of the `steps` property is shown below:

```
{
      "name": "Initial Review",
      "type": "",
      "expiryRule": "+5d"
      "expiryServiceName": "An Expiry Service"
      "actions": [
        { ... Action 1 Definition ... },
        { ... Action 2 Definition ... }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display":
"false" },
        { "name": "Terminate",  "nextStep": "Terminated Initial" }
      ]
},
```

The `steps` has the following attributes:

`name`

The name of the step, which is unique within a collaboration job.

`type`

The type of the step, that can be:

- `start`: the first step for the collaboration job. `"type":"start"`
- `endpoint`: the processing stops and the job completes. Endpoints do not specify actions, routes or `customProperties`. `"type":"endpoint"`
- `""`: empty string. `"type":""`

`expiryRule`

The step scheduled completion time, the step expiry, calculation rule. If specified, the `job_step.time_completion_scheduled` is set as the step creation time plus the expiry rule value, for example, `2014-03-05 + 5d = 2014-03-10`. The valid values are:

- `"expiryRule" : "+4h"` (hours)
- `"expiryRule" : "+5d"` (days)
- `"expiryRule" : "+4w` (weeks)
- `"expiryRule" : "+4m"`. (months)

This is an optional attribute.

> NOTE
>
> We recommend setting it up to some default value so an action doesn't stay in the queue forever, if an associated form never abandons and the task is never submitted, even as a transaction expires.

`expiryServiceName`

Specifies a job expiry service that is used to clean up some items, such as incomplete submissions. This service should specify a route.

> NOTE
>
> If the `expiryRule` is used, but the `expritysServiceName` is not specified, then `expiryRoute` must be provided.

actions

The `actions` property defines the list of actions for each step, which are executed sequentially. Actions are required for all steps except endpoints. The snippet of the `actions` property is shown below:

```
{
    "name": "Create Task",
    "type": "Job Task Assign",
    "properties": [
        { "name": "Process Message Send Email",  "value": "true" },
        { "name": "Process Message Submission Step",  "value": "Start" },
        { "name": "Process Message Text",  "value": "Your ${submission.formName} applic-
ation is at the Initial Review Step" },
}
```

The `actions` has the following attributes:

`name`

The name of the action, which is unique within a collaboration job.

`type`

The action service type, which can be one of the following:

- listed above in the Standard Actions Services.
- groovy action services can have a type of Job Action.

`properties`

The action-specific properties. For more information, see the `properties` object definition below. This is an optional attribute.

The standard Action Property classes (`com.avoka.fc.core.service.job.impl`) define a fixed set of Action Properties. The `JobTaskAssignService` defines a broad range of separate properties, so you can add them to your script when creating a new job action. This makes a job action re-useable across a collaboration job and the whole form application. For example, if the job definition has this property:

```
"properties": [
   { "name": "ABC Widget Name",  "value": "Aplle Blue Catapillar" },
   ...
 ]
```

Then, you can retrieve it using the Groovy code below:

```
def abcName = actionStepProperties.getProperty("ABC Widget Name")
```

However, you should consider the following:

- There is no submission associated with the job actions by default.
- `$formDataMap.name` doesn't resolve as the job action and it doesn't have any concept of a submission.
- `$func.previousSubmission().formDataMap.name` resolves a submission as it obtained the submission from the previous step.
- `$job.submission(0).name` resolves a submission as it obtained the first submission linked to the collaboration job.

For more information, see job functions.

properties

The `properties` property defines the job definition properties. The snippet of the `properties` property is shown below:

```
"properties": [
    { "name": "Task Assign Group",  "value": "$func.formProperty('Initial Review')" },
    { "name": "Task Type",  "value": "Review" },
]
```

Properties can be used to pass information into an instance of the Action Service. In this case, they are called action properties and can be defined at the action, step or collaboration job level. Properties with the same name defined at an action level override properties defined at the step level. Step override properties of the same name at the job level. This is illustrated in the snippet below:

```
{
  "jobDetails": { ... },
  "properties": [
    { "name": "Task Form Code", "value": "$func.startFormCode()" }
  ],
  "steps":[
    {
      "name": "Initial Review",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Text",  "value": "Your ${submission.formName}
application is at the Initial Review Step" },
            { "name": "Task Assign Group",  "value": "$func.formProperty('Initial
Review')" },
            { "name": "Task Message",  "value": "Please perform the initial review of the
${submission.formName} from ${formDataMap.firstName} ${formDataMap.lastName}." },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "Initial review of ${submission.formName}
from ${formDataMap.firstName} ${formDataMap.lastName}" },
            { "name": "Task Type",  "value": "Review" },
          ]
        },
        { ... }
      ],
      "properties": [
        { "name": "Task Form Code", "value": "$func.startFormCode()" }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" }
      ],
    },
    { ... }
  ]
}
```

The `properties` has the following attributes:

`name`

The name of the property. The predefined action services, such as the job task assign service, have a set number of action properties.

`value`

A property value that can include an Apache Velocity template. The action properties are templates that are evaluated when an action service is run. The templates can be:

- static text values, for example, `{ "name": "Task Email Message", "value": "An Email Message" }`
- substitute values from the form data extract, for example, `{ "name": "Process Message Text", "value": "Your ${job.submission.formName} application is at the Initial Review Step" }`

`template`

It evaluates the value as a velocity template and forces the return value to return a String. This can evaluate nested function calls. This is can be used with property values that start with "$func", these can return an object. For example, `$func.startUser()` will return a user.

```
{ "name": "Task Email Message", "value": "$func.getSubmission().formXml()", "tem-
plate": "true" }
```

routes

The `routes` property defines the route mapping between the route name and the next step, which are required for all steps except endpoints. An action can return a route result name, as a string, which is listed in each step. If a valid name is returned, the step routes the collaboration job to the next step. However, if the action returns an invalid route name, an error is thrown. The snippet of the `routes` property is shown below:

```
"routes": [
   { "name": "Approve",  "nextStep": "Application Delivery" },
   { "name": "Decline",  "nextStep": "Applicant Update" },
   { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display": "false"
},
   { "name": "Terminate",  "nextStep": "Terminated Initial" }
]
```

The `routes` has the following attributes:

`name`

The route name that can be any name except the following reserved names:

- Default - If there is no step found with a route name or if the route name is empty, the default route is used.
- Expiry - the route taken when the step has expired.

`nextStep`

The next step, which can be one of the following:

- A valid step name contained in the job definition.
- A special case is `##PREVIOUS_STEP` the job processing assign the next step as the previous step name.

`display`

The collaboration job provides a list of available routes to the form. However, the `"display": "false"` removes the route from the available routes. This is an optional attribute.

You can see the complete example of a job definition here.

Next, learn how to configure job definition.

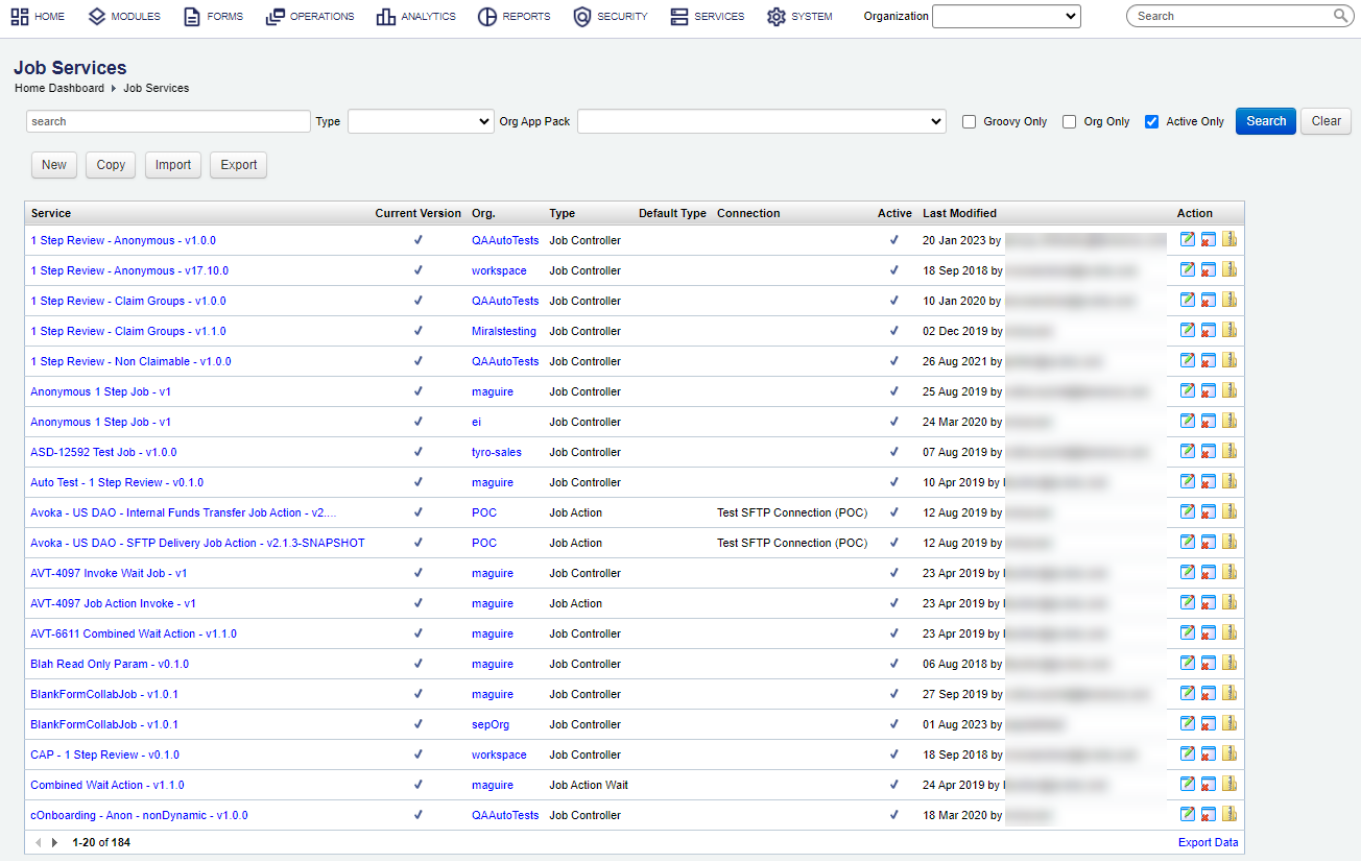
View Job Services

Manager allows you to view all job services, so you can easily find and modify them. The job services are used in collaboration jobs.

> TIP
>
> You can also check core global services using View Services UI.

To view job services, select Services > Job Services.

Job services are shown page by page, with the maximum number of records per page configured in the users preferences.



The list displays the following details:

- Service: a name of a service
- Current Version: whether a job service's version is current or not
- Org: Organization
- Type: a type of a job service
- Default Type: whether this service is the default service for this type of service.

- Connection: a service connection associated with this service, if it is configured.
- Active: whether this service is active or not. It must be active to be available.
- Last Modified: - a date when this service was last modified and by whom.

For more information, see View Services.

To filter or search the services by one or more criteria, specify the following settings and click Search:

- Service name or part of a name
- Service type
- Org App Pack
- Groovy Only to see only Groovy[1] services
- Org Only
- Active Only

Click Clear to reset the search criteria.

You can also perform the following actions:

- Create a job service
- View and configure a service
- Copy a service
- Remove a service
- Export services
- Import services

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

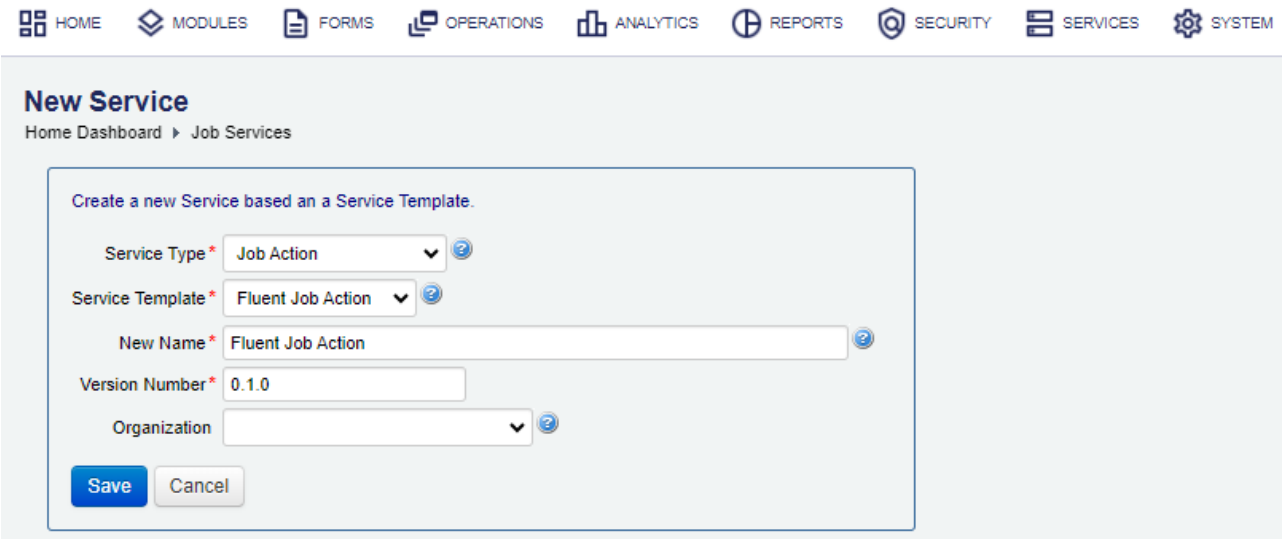
[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

Create a Job Service

Manager allows you to create and configure various job services, which are predominantly used in collaboration jobs.

To create a job service:

1. Select Services > Job Services.
2. Click New.



3. Select a service type of a new job service from the Service Type dropdown list.
4. Select a template from the Service Template dropdown list, which can be one of the following:
   - Fluent Job Action
   - Groovy Job Action
5. Select an organization from the Organization dropdown list or leave it blank to create a global job service.

> NOTE
>
> You must be a Journey Manager administrator to create or configure this functionality.

NOTE

You have to assign a job controller to an organization. If you have several organizations that would like to use the same job controller, you need to define one for each organization.

6. Click Save to update the changes.

Next, learn how to configure Job Definition.

Job Service Templates

Manager provides job service templates, which are enhanced job controller services. Templates allow you to customize collaboration jobs without modifying the job service definition. Templates are organization agnostic so you can't edit them. They are predominantly used for reviews and approvals. Manager comes with the following default templates:

- Review and Approval - 1 Step Group Review - v1.0.0
- Review and Approval - 1 Step User Review - v1.0.0
- Review and Approval - 2 Step Group Review - v1.0.0
- Review and Approval - Multi Step Group Review - v1.0.0

The job service templates are especially useful for:

- Business users and form builders to configure their forms without a developer.
- Greater reuse of a single job controller.

To use a job service template:

1. Locate a form version.
2. Click the Services tab to view services configured for this form.
3. Select a job service template from the Job Controller Service dropdown list, which has the following available templates:
    - Review and Approval - 1 Step Group Review - v1.0.0
    - Review and Approval - 1 Step User Review - v1.0.0
    - Review and Approval - 2 Step Group Review - v1.0.0
    - Review and Approval - Multi Step Group Review - v1.0.0
4. Click Save to update the changes.
5. Select the Job Properties tab to configure the collaboration job's properties.
6. Select the Job Reviewers from the Reviewer Group dropdown list.
7. Select the Job Example 1 from the Review Form dropdown list.
8. Click Save to update the changes.

A job service template has the Form Property Editor Definition service parameter, which is a JSON[1] file containing the form properties meta data.

---

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

Review and Approval - 1 Step Group Review - v1

Home Dashboard ▸ Job Services ▸ Service Definition

Service Definition | Job Definition | Job Services | Job Info | Parameters Edit | **Parameters** | Service Usage

Name	Type	Value	Description	Bind Parameter	Last Modified
actionRetryDelayMins	Number	5	Specify the default action retry delay in minutes for 'In Progress' and 'Error' actions.	✓	02 Sep 2015 by system
Form Property Editor Definition	JSON	[{ "name": "Step - Application Start", ...	Templated job form properties editor definition (JSON format).		02 Sep 2015 by system
jobControllerEnabled	Boolean	true	Specify whether the Job Controller service is enabled and can process jobs.	✓	02 Sep 2015 by system
jobDefinition	JSON	{ "jobDetails": { "name": "Review and Approval...	Job definition configuration (JSON format).	✓	02 Sep 2015 by system

Close

You can create a new custom template by adding the Form Property Editor Definition parameter to a standard job controller.

Next, learn about job controller services.

Job Definitions

Manager uses job services to process collaboration jobs. You can define and configure a job service as a single JSON[1] document, which is called a job definition. The job definition includes the list of steps that make up the collaboration job. Each of the individual steps has the list of actions to be performed when the step runs, and the routes property, which defines the next step transition by mapping the route name to the next step.

The job definition consists of the following properties:

- jobDetails
- jobGroups
- steps
- actions
- properties
- routes

jobDetails

The `jobDetails` property defines a collaboration job and how it is processed. The snippet of the `jobDetails` is shown below:

```
{
        "jobDetails": {
                "name": "1 Step Review",
                "processSubmitImmediate": "true",
                "allowEarlyCreation": true,
                "version": "0.1.0"
        },
        "steps":[
                { .. step definition ...},
                { .. step definition ...},
                ...
        ]
}
```

The `jobDetails` has the following attributes:

---

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

`name`

The name of the collaboration job, which is set by the job controller service when the collaboration job has been saved.

If defining multiple jobs, use distinct names for each job definition to ensure you can differentiate between them in the Journey Analytics scope selector.

`processSubmitImmediate`

The collaboration job processing mode. If this attribute is omitted or set to false, the job will use the standard mode. Otherwise, it will use the process immediate mode.

`allowEarlyCreation`

A boolean flag that allows you to create a collaboration job and associate it with a transaction that can only be in the Opened or Saved state.

This requires the Job Controller to be associated with a form version of a transaction and Job Controller service has the `allowEarlyCreation` flag set to true in its job definition. The `processJobAfterCreate` attribute will be ignored in this case.

| 24.10 This feature was introduced in 24.10

`version`

The version of Manager that the job was created in.

Let's look at a simple example of using the above configuration to create a job given the job controller service and version for a non submitted Txn as the initial Txn:

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*
Job job3 = new JobBuilder()
   .setSvcName('1 Step Review')
   .setSvcVersion('0.1.0')
   .setReferenceNumber('ABC123')
   .setInitialTxnTrackingCode('ABC123')
   .build()
```

jobGroups

The `jobGroups` property lists groups, which job coordinators can use to manage a collaboration job. The `jobGroups` can be used to show collaboration jobs assigned to each group, for example, Journey Space Reviews. The snippet of the `jobGroups` is shown below:

```
"jobGroups" : [
      {
        "name" : "HL Melbourne",
        "description" : "Home Loan Customer Care Group for Vic"
      },
      {
        "name" : "HL Sydney",
        "description" : "Home Loan Customer Care Group for NSW"
      }
    ]
```

The `jobGroups` has the following attributes:

`name`
: The name of the job group, which will be displayed in the Job Groups dropdown from the Reviews tab of the form space. The name must be less than 100 characters.

`description`
: The description for the job group, which must be less than 200 characters.

The job groups can be created in one of the following ways:

1. If a job group defined in a job definition, as in above, has not been created, then the job group will be automatically created the first time a user submits a form associated to this job. This is the preferred way to create job groups.
2. Job groups can be manually created. By manually creating the job group, you must ensure that the job group names and descriptions match with the job definition. Any mismatch will not be subsequently updated to the job groups.

steps

The `steps` property is an JSON array that defines steps. The snippet of the `steps` property is shown below:

```
{
      "name": "Initial Review",
      "type": "",
      "expiryRule": "+5d"
      "expiryServiceName": "An Expiry Service"
      "actions": [
        { ... Action 1 Definition ... },
        { ... Action 2 Definition ... }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display":
"false" },
        { "name": "Terminate",  "nextStep": "Terminated Initial" }
      ]
},
```

The `steps` has the following attributes:

`name`

The name of the step, which is unique within a collaboration job.

`type`

The type of the step, that can be:

- `start`: the first step for the collaboration job. `"type":"start"`
- `endpoint`: the processing stops and the job completes. Endpoints do not specify actions, routes or `customProperties`. `"type":"endpoint"`
- `""`: empty string. `"type":""`

`expiryRule`

The step scheduled completion time, the step expiry, calculation rule. If specified, the `job_step.time_completion_scheduled` is set as the step creation time plus the expiry rule value, for example, `2014-03-05 + 5d = 2014-03-10`. The valid values are:

- `"expiryRule" : "+4h"` (hours)
- `"expiryRule" : "+5d"` (days)
- `"expiryRule" : "+4w` (weeks)
- `"expiryRule" : "+4m"`. (months)

This is an optional attribute.

> NOTE
>
> We recommend setting it up to some default value so an action doesn't stay in the queue forever, if an associated form never abandons and the task is never submitted, even as a transaction expires.

`expiryServiceName`

Specifies a job expiry service that is used to clean up some items, such as incomplete submissions. This service should specify a route.

> NOTE
>
> If the `expiryRule` is used, but the `expritysServiceName` is not specified, then `expiryRoute` must be provided.

actions

The `actions` property defines the list of actions for each step, which are executed sequentially. Actions are required for all steps except endpoints. The snippet of the `actions` property is shown below:

```
{
    "name": "Create Task",
    "type": "Job Task Assign",
    "properties": [
        { "name": "Process Message Send Email",  "value": "true" },
        { "name": "Process Message Submission Step",  "value": "Start" },
        { "name": "Process Message Text",  "value": "Your ${submission.formName} applic-
ation is at the Initial Review Step" },
}
```

The `actions` has the following attributes:

`name`

The name of the action, which is unique within a collaboration job.

`type`

The action service type, which can be one of the following:

- listed above in the Standard Actions Services.
- groovy action services can have a type of Job Action.

`properties`

The action-specific properties. For more information, see the `properties` object definition below. This is an optional attribute.

The standard Action Property classes (`com.avoka.fc.core.service.job.impl`) define a fixed set of Action Properties. The `JobTaskAssignService` defines a broad range of separate properties, so you can add them to your script when creating a new job action. This makes a job action re-useable across a collaboration job and the whole form application. For example, if the job definition has this property:

```
"properties": [
   { "name": "ABC Widget Name",  "value": "Aplle Blue Catapillar" },
   ...
 ]
```

Then, you can retrieve it using the Groovy code below:

```
def abcName = actionStepProperties.getProperty("ABC Widget Name")
```

However, you should consider the following:

- There is no submission associated with the job actions by default.
- `$formDataMap.name` doesn't resolve as the job action and it doesn't have any concept of a submission.
- `$func.previousSubmission().formDataMap.name` resolves a submission as it obtained the submission from the previous step.
- `$job.submission(0).name` resolves a submission as it obtained the first submission linked to the collaboration job.

For more information, see job functions.

properties

The `properties` property defines the job definition properties. The snippet of the `properties` property is shown below:

```
"properties": [
    { "name": "Task Assign Group",  "value": "$func.formProperty('Initial Review')" },
    { "name": "Task Type",  "value": "Review" },
]
```

Properties can be used to pass information into an instance of the Action Service. In this case, they are called action properties and can be defined at the action, step or collaboration job level. Properties with the same name defined at an action level override properties defined at the step level. Step override properties of the same name at the job level. This is illustrated in the snippet below:

```
{
  "jobDetails": { ... },
  "properties": [
    { "name": "Task Form Code", "value": "$func.startFormCode()" }
  ],
  "steps":[
    {
      "name": "Initial Review",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Text",  "value": "Your ${submission.formName}
application is at the Initial Review Step" },
            { "name": "Task Assign Group",  "value": "$func.formProperty('Initial
Review')" },
            { "name": "Task Message",  "value": "Please perform the initial review of the
${submission.formName} from ${formDataMap.firstName} ${formDataMap.lastName}." },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "Initial review of ${submission.formName}
from ${formDataMap.firstName} ${formDataMap.lastName}" },
            { "name": "Task Type",  "value": "Review" },
          ]
        },
        { ... }
      ],
      "properties": [
        { "name": "Task Form Code", "value": "$func.startFormCode()" }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" }
      ],
    },
    { ... }
  ]
}
```

The `properties` has the following attributes:

`name`

The name of the property. The predefined action services, such as the job task assign service, have a set number of action properties.

`value`

A property value that can include an Apache Velocity template. The action properties are templates that are evaluated when an action service is run. The templates can be:

- static text values, for example, `{ "name": "Task Email Message", "value": "An Email Message" }`
- substitute values from the form data extract, for example, `{ "name": "Process Message Text", "value": "Your ${job.submission.formName} application is at the Initial Review Step" }`

`template`

It evaluates the value as a velocity template and forces the return value to return a String. This can evaluate nested function calls. This is can be used with property values that start with "$func", these can return an object. For example, `$func.startUser()` will return a user.

```
{ "name": "Task Email Message", "value": "$func.getSubmission().formXml()", "tem-
plate": "true" }
```

routes

The `routes` property defines the route mapping between the route name and the next step, which are required for all steps except endpoints. An action can return a route result name, as a string, which is listed in each step. If a valid name is returned, the step routes the collaboration job to the next step. However, if the action returns an invalid route name, an error is thrown. The snippet of the `routes` property is shown below:

```
"routes": [
   { "name": "Approve",  "nextStep": "Application Delivery" },
   { "name": "Decline",  "nextStep": "Applicant Update" },
   { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display": "false"
},
   { "name": "Terminate",  "nextStep": "Terminated Initial" }
]
```

The `routes` has the following attributes:

`name`

The route name that can be any name except the following reserved names:

- Default - If there is no step found with a route name or if the route name is empty, the default route is used.
- Expiry - the route taken when the step has expired.

`nextStep`

The next step, which can be one of the following:

- A valid step name contained in the job definition.
- A special case is `##PREVIOUS_STEP` the job processing assign the next step as the previous step name.

`display`

The collaboration job provides a list of available routes to the form. However, the `"display": "false"` removes the route from the available routes. This is an optional attribute.

You can see the complete example of a job definition here.

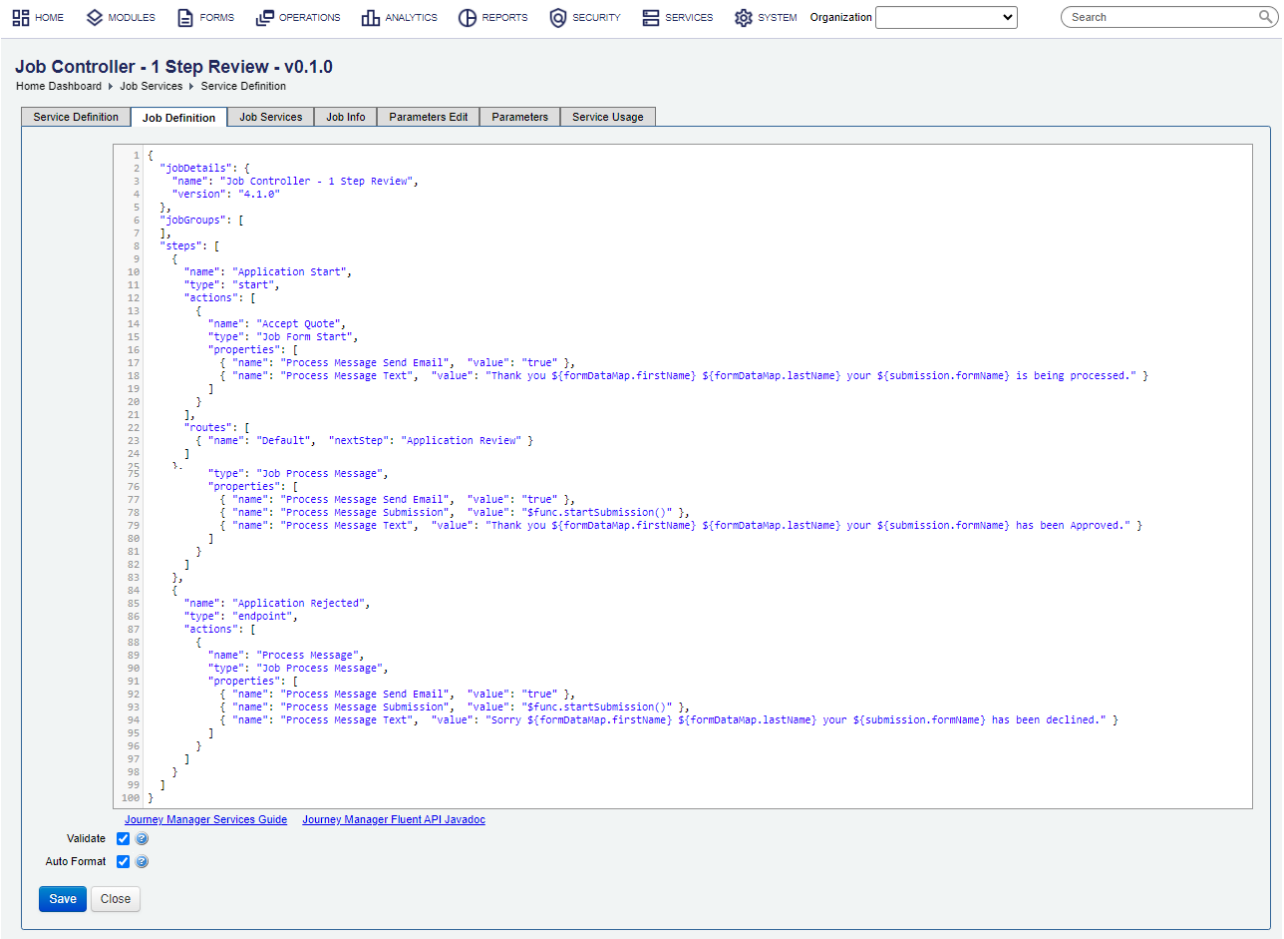
Next, learn how to configure job definition.

Configure Job Definition

Manager allows you to specify how a job service processes collaboration jobs by defining an execution flow using steps and actions in the job definition tab.

To configure a job definition of a service:

1. Locate an existing job service.
2. Select the Job Definition tab to review or update the JSON[1] file defining this service.



3. Select the Validate checkbox to validate action services, step properties, form codes and custom services defined in the JSON file. It is selected by default, but you can clear it, if you want to add a new action but you haven't created a form or a custom service for this, and you still need to save your changes.

---

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

4. Select the Auto Format checkbox to automatically format the job definition and strip unknown elements.
5. Click Save to update the changes..

Next, learn how to view all job services of a job controller.

View Job Info

Manager provides you with a read only view of job info of a selected job service.

To check job info:

1. Locate a job service.
2. Select the Job Info tab to view job info of this service that includes:
    - Job name
    - Job steps
    - Job Form Data Maps, which lists the names of the parameters used in the form XML data.
3. Click Close.

Next, learn how to configure service parameters.

Create a Job Controller Service

Manager uses job controller services to:

- Create a collaboration job instance when a user submits a form or the API is called.
- Process collaboration jobs by executing their steps and actions defined in a job definition.

Manager allows you to create and configure job controller services to implement your customer requirement.

To create a job controller service:

1. Select Services > Job Services and click New.
2. Select the Job Controller from the Service Type dropdown list. For more information, see service types.
3. Select a service template from the Service Template dropdown list, which can be one of the following:
   - Job Controller - 1 Step Review
   - Job Controller - 2 Step Review
4. Select an organization from the Organization dropdown list.

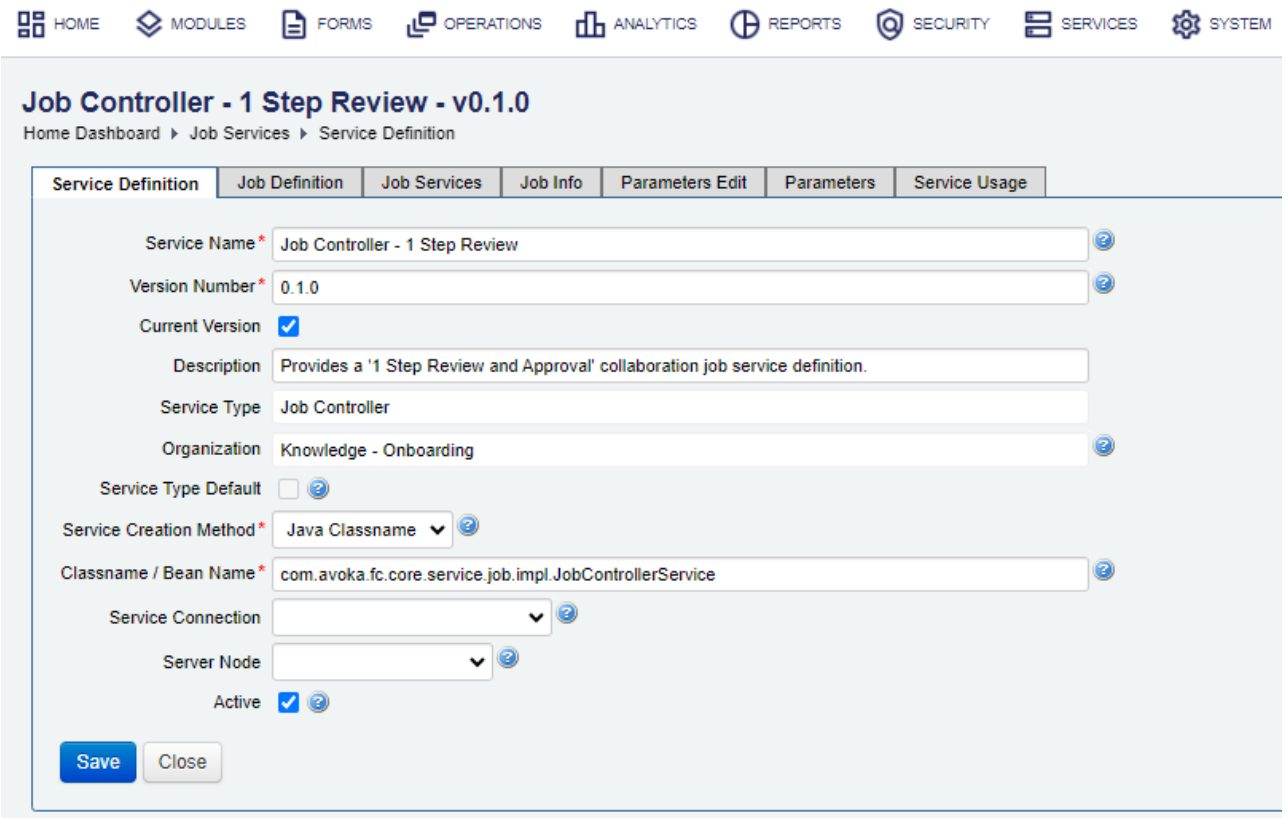
   NOTE

   You have to assign a job controller to an organization. If you have several organizations that would like to use the same job controller, you need to define one for each organization.

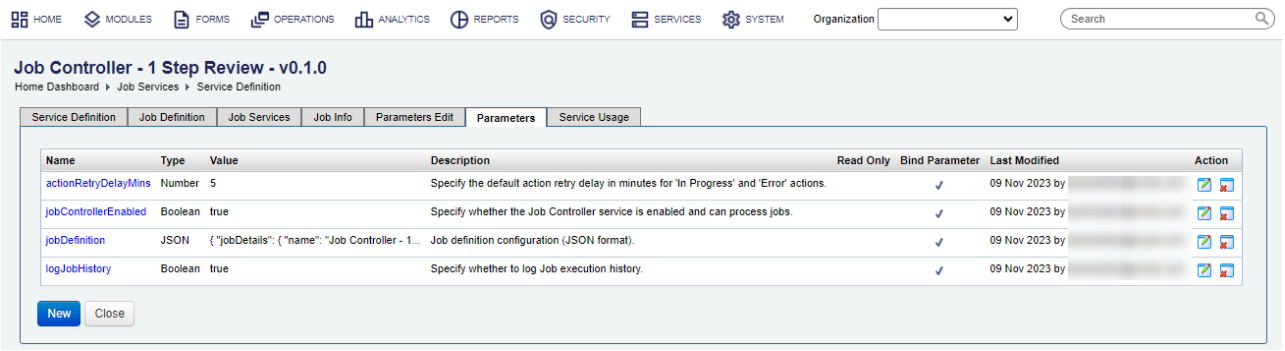
5. Click Save to update the changes.

To configure a job controller service:

1. Locate a job service.
2. Configure the service definition.



3. Configure the job definition.
4. Check the job services used in this job controller.
5. Check the job info.
6. Create service parameters required for this job controller to run.



The standard service parameters are:

- `actionRetryDelayMins` - specifies the default action retry delay in minutes for 'In Progress' and 'Error' actions. Default is 5. This is a bind parameter.

- `jobControllerEnabled` - specifies whether the job controller service is enabled and it can process collaboration jobs. This is useful for debugging and test in non-production, and selectively enabling certain collaboration jobs in production environments. Default is true. This is a bind parameter.
- `jobDefinition` - job definition configuration in JSON[1] format. This is a bind parameter.
- `logJobHistory` - specifies whether to write collaboration job execution history in a log file or not. Default is true. This is a bind parameter.

The optional service parameters are:

- `jobDiagram` - contains the collaboration job diagram format created by the external NextStep tool.
- `Form Property Editor Definition` - the job form properties editor definition in JSON format.

7. Check the job controller service usage.

Next, learn how to configure a job controller in a form.

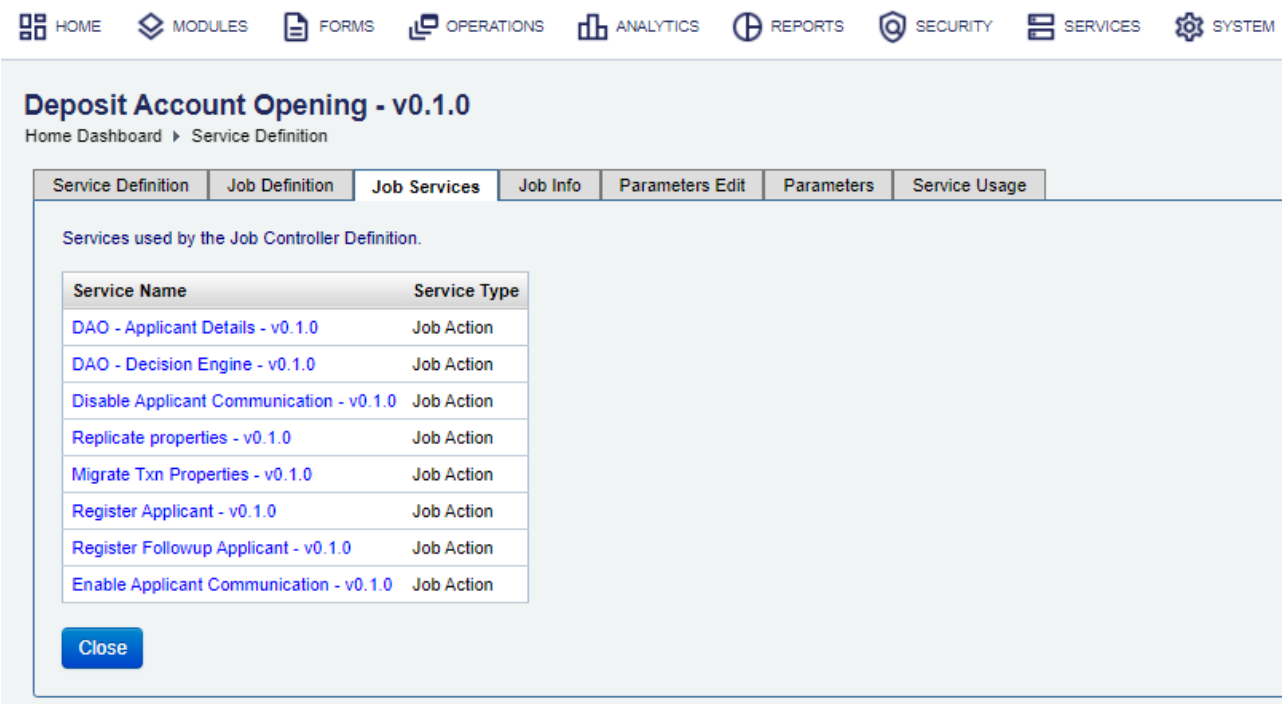
[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

View Job Controller Services

Manager allows you to viewjob services used by a job controller.

To view the list of job services:

1. Locate an existing job controller.
2. Select the Job Services tab to list all job services, which are configured in the job definition. The list is empty if there are no service configured in the JSON[1] file.



3. Click a service name to check its configuration.

Next, learn how to check job info.

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

Job Action Service

Manager allows you to create and configure job action services. When a collaboration job's action is processed, an action service is run. A collaboration job usually specifies an action service from one of the available inbuilt action services. This allows a developer to create their own job action and job expiry services using Groovy[1] scripts.

To create a job action service:

1. Select Services > Job Services.
2. Click New to create a job service.
3. Select the Job Action as a service type from the Service Type dropdown list.
4. Click Save to update the changes.

To configure a job action service:

1. Locate a job action service.
2. Configure the service definition.
3. Select the new service type to assign a Fluent or Groovy job action to a new type, such as Job Expiry Service or Job Task Wait.
4. Configure the job definition. You can configure an action to call a Groovy service from a job controller, as shown below.

```
"actions": [
  {
      "name": "Individual Tasks",
      "type": "Job Task Assign",
      "properties": [
         { "name": "Task Assign Repeating",  "value": "true" },
         { "name": "Task Assign Repeat Item",  "value": "$func.invoke('Repeating
Get Item', ${formDataMap.itemDelimited}, ${assignRepeatIndex})" },
         { "name": "Task Type",  "value": "Review" }
      ]
   }
```

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

The `$func.invoke` method executes the Repeating Get Item service (invoking it by the name) with two parameters.

If you want to invoke same job action multiple times as part of single step, give them different job action names, for example, `"name": "Individual Tasks 2"`, but define the same service name, for example,`"serviceName": "Repeating Claimable"`. If you do not define the service name explicitly, it expects the job action name to match the service name. It's not allowed to have 2 or more job actions with the same name in one job step.

5. Click the Job Services tab to view the Repeating Get Item service used in the action definition.

6. Click on the service name to open this service's Groovy script.

```
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import groovy.transform.TypeChecked
import javax.servlet.http.*
import org.apache.commons.lang3.StringUtils

@TypeChecked
class FluentGroovyService {

    // Injected at runtime
    public Logger logger

    /*
     * Perform a groovy service invocation
     *
     * return: the result object
     */
    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, Map params)
{
        JobAction jobAction = (JobAction) params.jobAction
        return invoke(svcDef, request, user, jobAction, params)
    }

    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, JobAction
jobAction, Map params) {
        logger.info params
        List<String> args = (List<String>) params.args

        def result = "Not Found"

        if( StringUtils.isNotEmpty(args[0]) &&
           StringUtils.isNotEmpty(args[1]) &&
           args[1].isNumber()) {
            def itemDelimited = args[0]
            // note repeat index from TM is 1 based not 0 based
            def repeatIndex = args[1].toInteger() - 1
            logger.info "itemDelimited: " + itemDelimited
            logger.info "repeatIndex: " + repeatIndex

            def items = itemDelimited.tokenize('|')
```

```
            logger.info "items: " + items

            result = items[repeatIndex.toInteger()]
            logger.info "result Item: " + result
        }

        return result

    }
}
```

7. Click Save to update the changes.

Download

You can download the complete code example here.

Next, learn how to view all job services.

Job Action Wait Service

Manager allows you to create a job action wait service, which waits until all the step's previous Invoked actions have been completed. For more information, see action statuses.

The job action wait service is an action service and it is a part of the collaboration job framework that you use to implement various form processing flows.

You can create and update a job action wait service using the Manager UI, as well as Fluent or Core API.

To create a job action wait service:

1. Select Services > Job Services and click New .
2. Select Job Action Wait from the Service Type dropdown list. For more information, see service types.
3. Select a template from the Service Template dropdown list, which can be one of the following:
   - Fluent Job Action Wait
   - Groovy Job Action Wait
   - Job Action Wait
4. Click Save to update the changes.

To configure a job action wait service:

1. Locate a job action service.
2. Configure the service definition. You can choose a new service type from the Service Type dropdown list to change the signature of the job service, for example, to Job Expiry Service or Job Task Wait. This won't remove the existing Groovy script or parameters.
3. Configure the job definition. You can configure an action to call a Groovy service from a job controller, as shown below.

```
"actions": [
  {
      "name": "Individual Tasks",
      "type": "Job Task Assign",
      "properties": [
         { "name": "Task Assign Repeating",  "value": "true" },
         { "name": "Task Assign Repeat Item",  "value": "$func.invoke('Repeating
Get Item', ${formDataMap.itemDelimited}, ${assignRepeatIndex})" },
         { "name": "Task Type",  "value": "Review" }
      ]
   }
```

The `$func.invoke` method executes the Repeating Get Item service (invoking it by the name) with two parameters.

If you want to invoke same job action multiple times as part of single step, give them different job action names, for example, `"name": "Individual Tasks 2"`, but define the same service name, for example,`"serviceName": "Repeating Claimable"`. If you do not define the service name explicitly, it expects the job action name to match the service name. It's not allowed to have 2 or more job actions with the same name in one job step.

4. Click the Job Services tab to view the Repeating Get Item service used in the action definition.

5. Click on the service name to open this service's Groovy script.

```
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import groovy.transform.TypeChecked
import javax.servlet.http.*
import org.apache.commons.lang3.StringUtils

@TypeChecked
class FluentGroovyService {

    // Injected at runtime
    public Logger logger

    /*
     * Perform a groovy service invocation
     *
     * return: the result object
     */
    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, Map params)
{
        JobAction jobAction = (JobAction) params.jobAction
        return invoke(svcDef, request, user, jobAction, params)
    }

    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, JobAction
jobAction, Map params) {
        logger.info params
        List<String> args = (List<String>) params.args

        def result = "Not Found"

        if( StringUtils.isNotEmpty(args[0]) &&
           StringUtils.isNotEmpty(args[1]) &&
           args[1].isNumber()) {
            def itemDelimited = args[0]
            // note repeat index from TM is 1 based not 0 based
            def repeatIndex = args[1].toInteger() - 1
            logger.info "itemDelimited: " + itemDelimited
            logger.info "repeatIndex: " + repeatIndex

            def items = itemDelimited.tokenize('|')
```

```
            logger.info "items: " + items

            result = items[repeatIndex.toInteger()]
            logger.info "result Item: " + result
        }

        return result

    }
}
```

6. Click Save to update the changes.

To create a Job Action Wait service using API:

Core API

`com.avoka.fc.core.service.job.impl.JobActionWaitService`
`com.avoka.fc.core.service.job.impl.GroovyJobActionService`

For more information, see Core API

Fluent API

`com.avoka.fc.core.service.fluent.FluentJobActionService`

For more information, see Fluent API

Next, learn how to view all job services.

Job Delivery Service

Manager allows you to create a job delivery service, which performs a delivery of form submitted data. Unlike some other service types like Fluent Function or Job Action, the job delivery doesn't require you to write any code. You simply give your service a name and indicate which delivery channel it's supposed to tickle when invoked. When your collaboration job has done all the things you need doing, finish off with a step that invokes your job delivery service.

You can create and update a job delivery service using the Manager UI, as well as Fluent or Core API.

To create a job delivery service:

1. Select Services > Job Services and click New .
2. Select job delivery from the Service Type dropdown list. For more information, see service types.
3. Select a template from the Service Template dropdown list, which can be one of the following:
    - Fluent Job Expiry
    - Groovy Job Expiry
4. Click Save to update the changes.

To configure a job expiry service:

1. Locate a job delivery service.
2. Configure the service definition.
3. Configure the job definition. You can configure an action to call a Groovy service from a job controller, as shown below.

```
{
  "jobDetails": {
    "name": "Basic Onboarding Job Controller",
    "version": "4.1.0"
  },
  "jobGroups": [
  ],
  "steps": [
    {
      "name": "Application Start",
      "type": "start",
      "actions": [
        {
          "name": "Handle Submission",
          "type": "Job Form Start",
          "properties": [
            { "name": "Process Message Send Email",  "value": "false" }
          ]
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Deliver" }
      ]
    },
    {
      "name": "Deliver",
      "type": "",
      "actions": [
        {
          "name": "Onboarding Job Delivery",
          "type": "Job Delivery"
        },
        {
          "name": "Application Delivery Wait",
          "type": "Job Delivery Wait"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Done" }
      ]
```

```
    },
    {
      "name": "Done",
      "type": "endpoint"
    }
  ]
}
```

You can see that the job step Deliver has two actions, one of type Job Delivery and another of type Job Delivery Wait. The reason for that is a bit technical. You see, Manager has a Collaboration Job Controller service that runs as a scheduled job and, as its name implies, manages collaboration jobs. It has a separate service called Transaction Processing, which does other stuff. Well, the job delivery is handled by the Transaction Processing service, not the Collaboration Job Controller service, so this wait business is just about the latter waiting for the former. You could think of the Onboarding Job Delivery action as instructing Job Controller to prepare this job for delivery by the Transaction Processor while the Application Delivery Wait action instructs the Job Controller to wait about until the delivery is complete.

4. Click the Job Services tab to view the Repeating Get Item service used in the action definition.
5. Click on the service name to open this service's Groovy script.
6. Click Save to update the changes.

To create a job delivery service using API:

Core API

`com.avoka.fc.core.service.job.impl.JobDeliveryService`
`com.avoka.fc.core.service.job.impl.GroovyJobDeliveryService`

For more information, see Core API

Fluent API

`com.avoka.fc.core.service.fluent.FluentJobDeliveryService`

For more information, see Fluent API

Next, learn how to view job services.

Job Task Wait Action Service

Manager allows you to create and configure job task wait action services, which are specialized job action services designed to block the execution of a collaboration job until a specified event takes place.

To create a job action service:

1. Select Services > Job Services.
2. Click New to create a job service.
3. Select the Job Task Wait Action as a service type from the Service Type dropdown list.
4. Select the Fluent Job Action template from the Service Template dropdown list.
5. Select an organization from the Organization dropdown list.
6. Click Save to update the changes.

To configure a job action service:

1. Locate a job action service.
2. Configure the service definition.
3. Select the new service type to assign a Fluent or Groovy job action to a new type, such as Job Expiry Service or Job Task Wait.
4. Configure the job definition. You can configure an action to call a Groovy service from a job controller, as shown below.

```
"actions": [
        {
          "name": "Individual Tasks",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Task Assign Repeating",  "value": "true" },
            { "name": "Task Assign Repeat Item",  "value": "$func.invoke('Repeating
Get Item', ${formDataMap.itemDelimited}, ${assignRepeatIndex})" },
            ...
            { "name": "Task Type",  "value": "Review" }
          ]
        }
```

The `$func.invoke` method executes the Repeating Get Item service (invoking it by the name) with two parameters.

5. Click the Job Services tab to view the Repeating Get Item service used in the action definition.

6. Click on the service name to open this service's Groovy script.

```
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import groovy.transform.TypeChecked
import javax.servlet.http.*
import org.apache.commons.lang3.StringUtils

@TypeChecked
class FluentGroovyService {

    // Injected at runtime
    public Logger logger

    /*
     * Perform a groovy service invocation
     *
     * return: the result object
     */
    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, Map params)
{
        JobAction jobAction = (JobAction) params.jobAction
        return invoke(svcDef, request, user, jobAction, params)
    }

    Object invoke(SvcDef svcDef, HttpServletRequest request, User user, JobAction
jobAction, Map params) {
        logger.info params
        List<String> args = (List<String>) params.args

        def result = "Not Found"

        if( StringUtils.isNotEmpty(args[0]) &&
           StringUtils.isNotEmpty(args[1]) &&
           args[1].isNumber()) {
            def itemDelimited = args[0]
            // note repeat index from TM is 1 based not 0 based
            def repeatIndex = args[1].toInteger() - 1
            logger.info "itemDelimited: " + itemDelimited
            logger.info "repeatIndex: " + repeatIndex

            def items = itemDelimited.tokenize('|')
```

```
            logger.info "items: " + items

            result = items[repeatIndex.toInteger()]
            logger.info "result Item: " + result
        }

        return result

    }
}
```

7. Click Save to update the changes.

Next, learn how to view all job services.

Open the TaskUserWebService in SOAP UI

Manager allows you to open the TaskUserWebService in SOAP UI. You should read the section 4 of the transact_web_service_guide_v4.2.pdf documentation prior to doing so.

The TaskUserWebService and TaskUserWebServiceV2 WSDL in Manager are located in a secure context path. This wiki article shows the workaround steps to successfully import the WSDL's and call methods in SOAP-UI.

Task Web Services V2 requires you to authenticate using a username and password, It is possible to get the wsdl from the non secure context however you will be unable to call the secured services from the unsecured WSDL

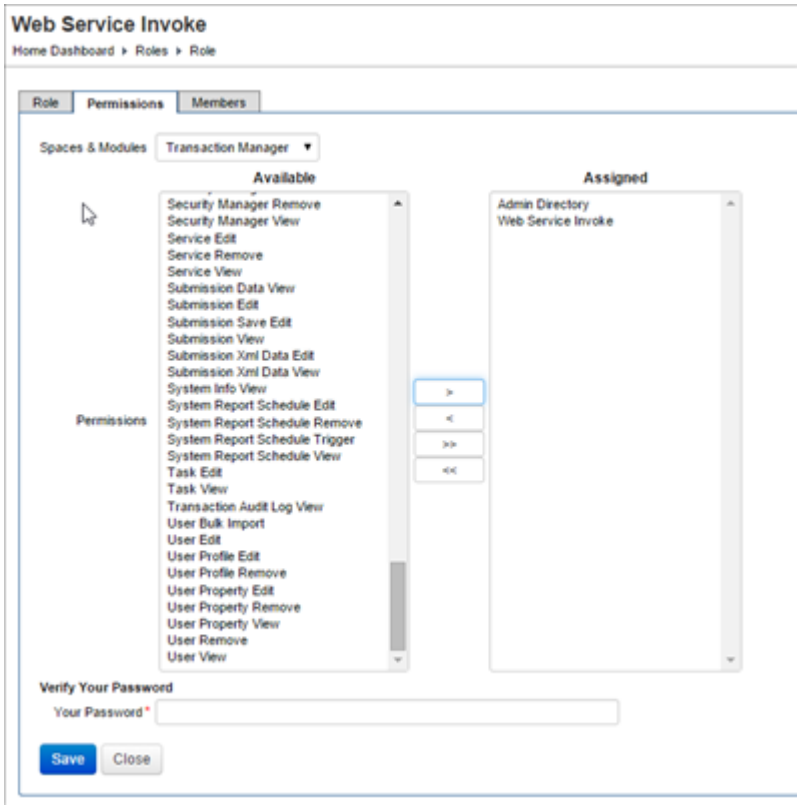
To open the service:

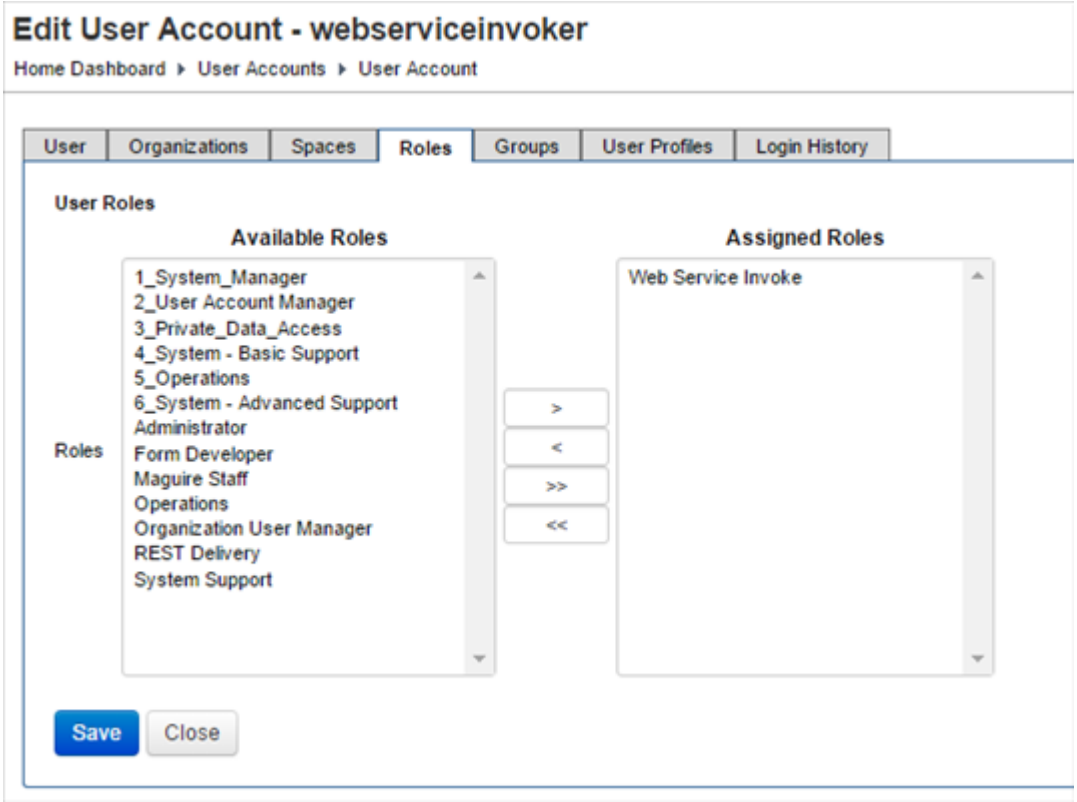
1. WSDL URL- https://{Server Name}/-manager/secure/services/TaskUserWebServiceV2?wsdl

   NOTE

   Do not use the unsecure URL https://{Server Name}/-manager/fcservices/TaskUserWebServiceV2?wsdl

2. Create a service user that has been assigned a Role that has the Web Service Invoke permission.

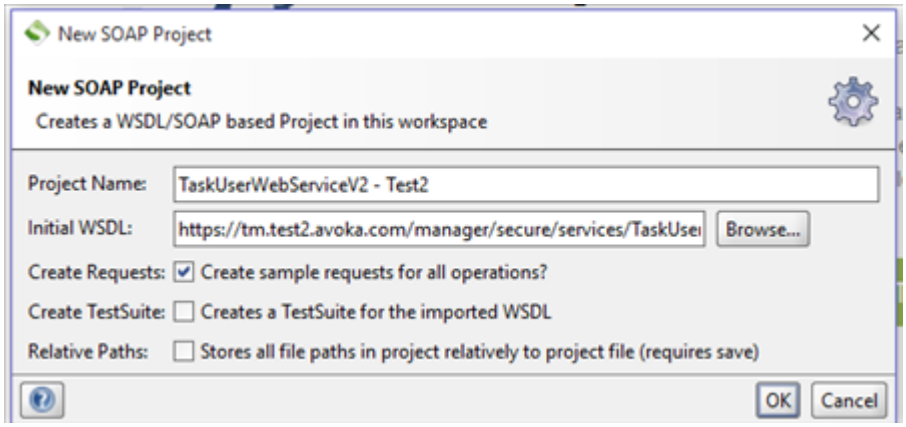

Web Service Invoke
Home Dashboard ▸ Roles ▸ Role
Role
Permissions
Members
Spaces & Modules
Transaction Manager
Available
Assigned
Permissions
Security Manager Remove
Security Manager View
Service Edit
Service Remove
Service View
Submission Data View
Submission Edit
Submission Save Edit
Submission View
Submission Xml Data Edit
Submission Xml Data View
System Info View
System Report Schedule Edit
System Report Schedule Remove
System Report Schedule Trigger
System Report Schedule View
Task Edit
Task View
Transaction Audit Log View
User Bulk Import
User Edit
User Profile Edit
User Profile Remove
User Property Edit
User Property Remove
User Property View
User Remove
User View
Admin Directory
Web Service Invoke
Verify Your Password
Your Password *
Save
Close


Edit User Account - webserviceinvoker
Home Dashboard ▸ User Accounts ▸ User Account
User
Organizations
Spaces
Roles
Groups
User Profiles
Login History
User Roles
Available Roles
Assigned Roles
Roles
1_System_Manager
2_User Account Manager
3_Private_Data_Access
4_System - Basic Support
5_Operations
6_System - Advanced Support
Administrator
Form Developer
Maguire Staff
Operations
Organization User Manager
REST Delivery
System Support
Web Service Invoke
Save
Close

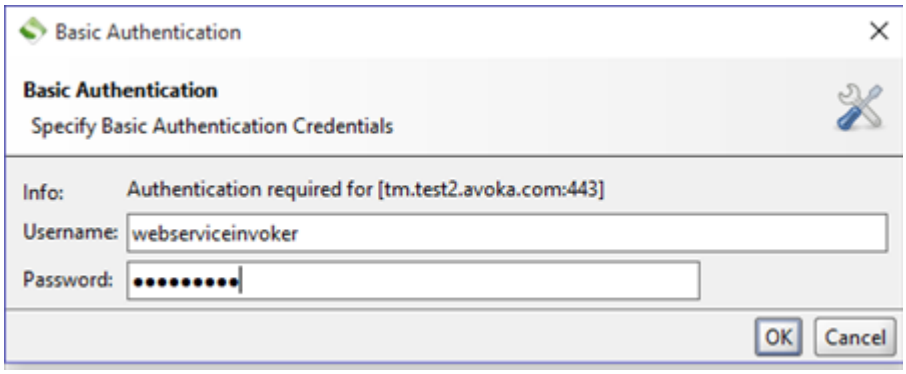
NOTE

Users with the Adminitrator role have the Web Service Invoke permission.

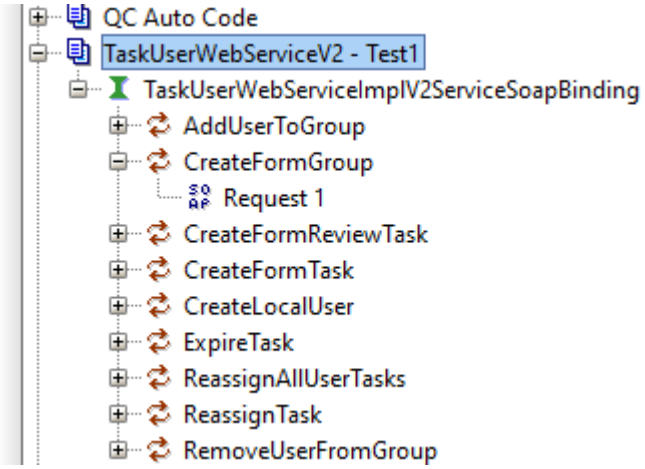
3. Create SOAP-UI project the wsdl in Step 1.



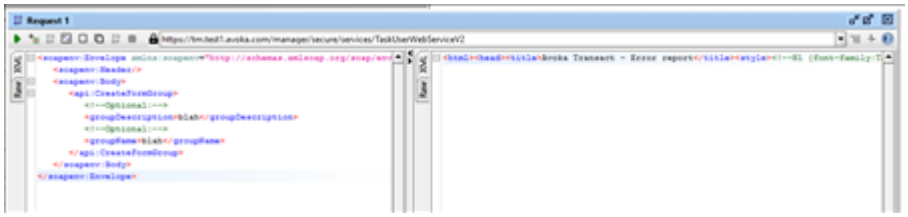
4. Soap-ui will prompt you with the Basic Authentication tab



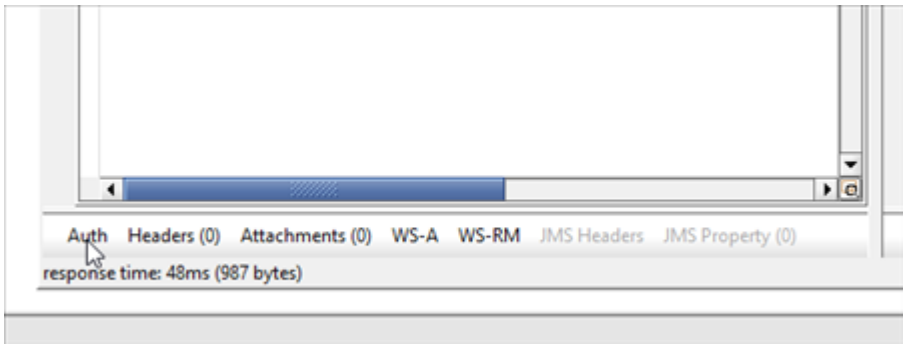
5. You should end up with the following



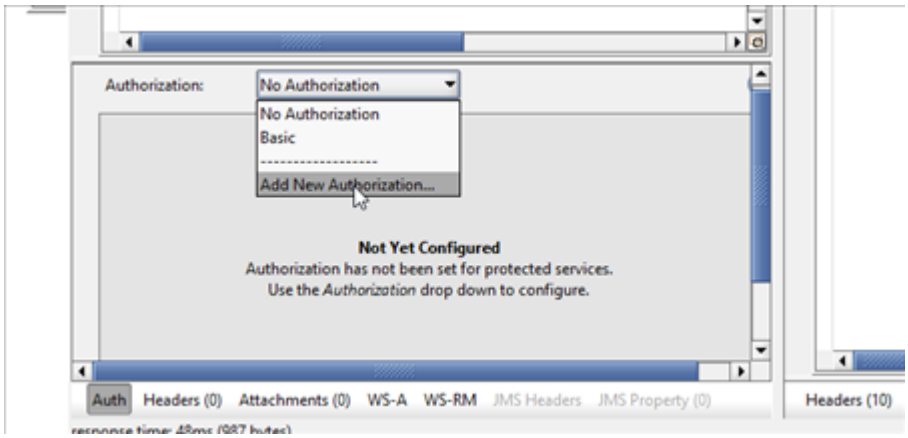
6. Try calling the web services with no authentication. Test Creating a Form Group. The response on the right tell you you are not authenticated.



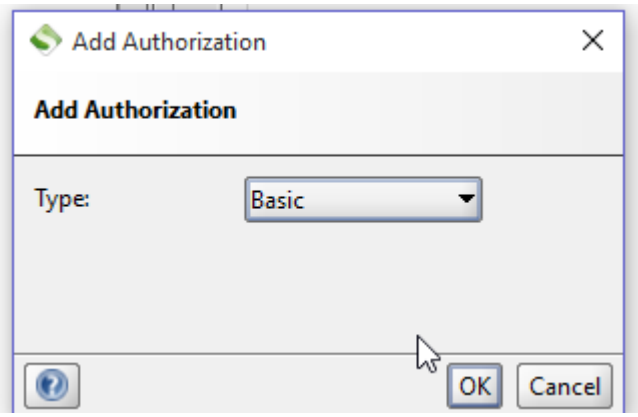
7. Click on the Auth buttong on the bottom left of the Request1 page to configure an Authentication.



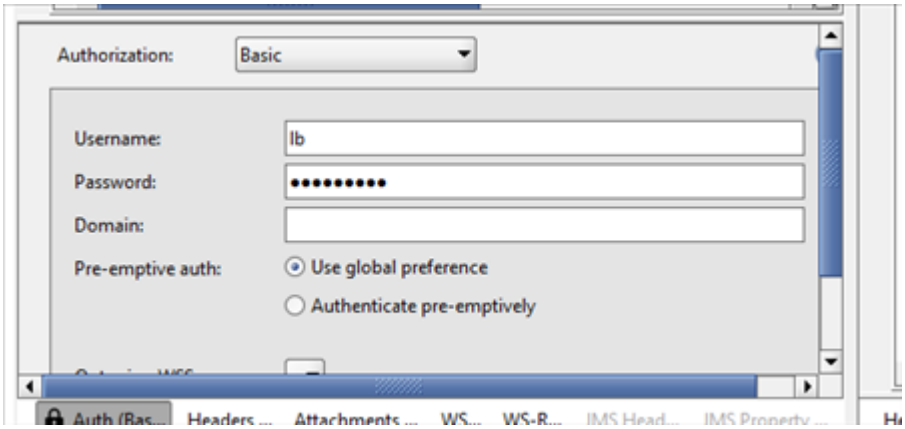
8. Select Add New Authentication.



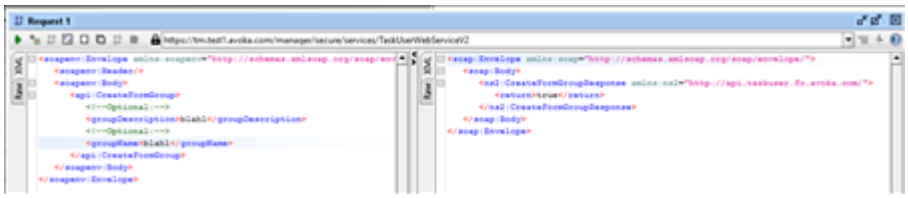
9. Select a type as Basic.



10. Specify a user name and a password.



11. Re-run the request.



Next, learn how to [view services](#).

Steps

A collaboration job consists of individual steps executed by Journey Manager to complete the collaboration job. The definition of what steps are used and how they are linked together resides in the job definition of the job controller service, which processes collaboration jobs. The order of steps within the job definition is not important because they are linked with each other using the `nextStep` instruction of the `routes` element. Each step contains one or more actions. Each action completes a part of the workflow, usually by calling a service that does the work. Actions are sequential within each step.

> NOTE
>
> The order of steps are not important, but the order of actions is.

A step has several pre-defined statuses depending on the completion of actions.

A job step status can be one of the following:

- In Progress- the step is in progress now.
- Completed - the step is completed.
- Expired - the step is expired.
- Cancelled - the step has been manually cancelled by an administrator.

A step is completed when all its actions have executed in specified order without any exceptions, so the next step can be run. The next step is defined by either:

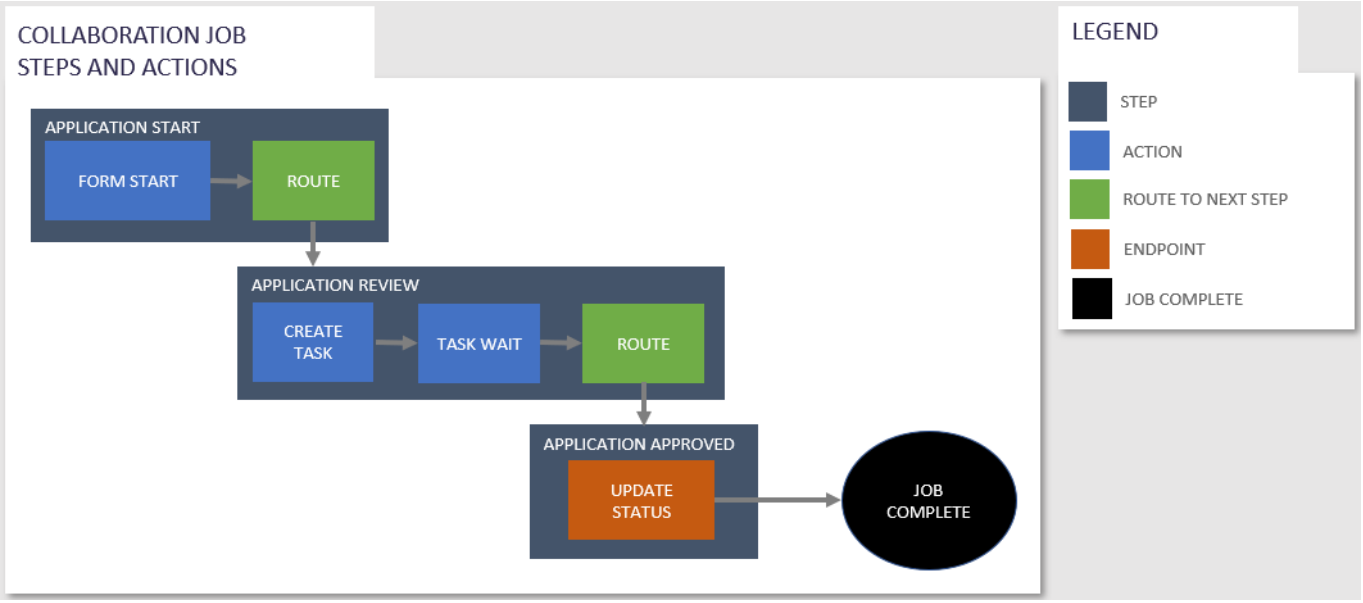
- The last action that returns a route name, which, in many cases, is selected by a user while completing a form.
- A step's `routes` element pointing to the next step.

A step can be a standard or a start step. A standard step consists of:

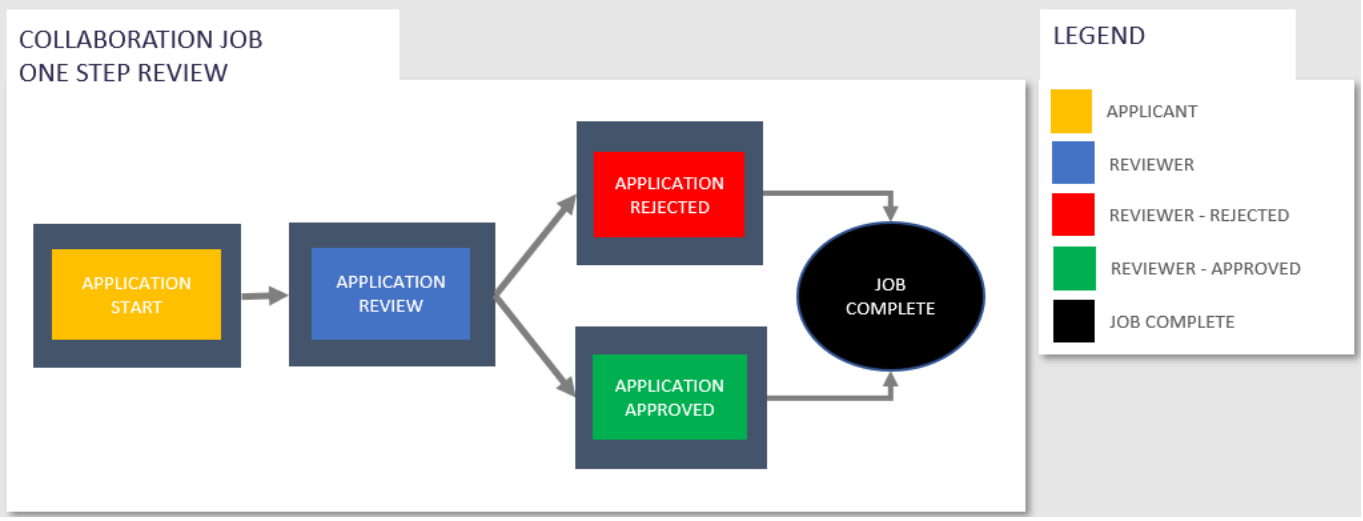
- A list of actions which are executed in the specified order.
- An optional expiry action.
- One or more routes.

> NOTE
>
> Once a step reaches the Completed status, it can't be reverted back to any particular step in the flow, for example, In Progress.

A start step is a first standard step that is executed during a collaboration job processing. The start step has the same elements as the standard step, with an extra start type assigned to it. You must define one of the steps in the collaboration job as the start step. Usually, a collaboration job is started by a form submission, so the first action in the start step is a Job Form Start action.



A collaboration job can have one or more endpoint steps. Each endpoint can execute none, one or more actions, such as update a status or send an email. When all actions complete, this makes this step complete, so the collaboration job can complete as well. For example, the Application Approved and Application Rejected steps are the two endpoints that make the collaboration job complete.



A step definition for the One Step Review is shown below.

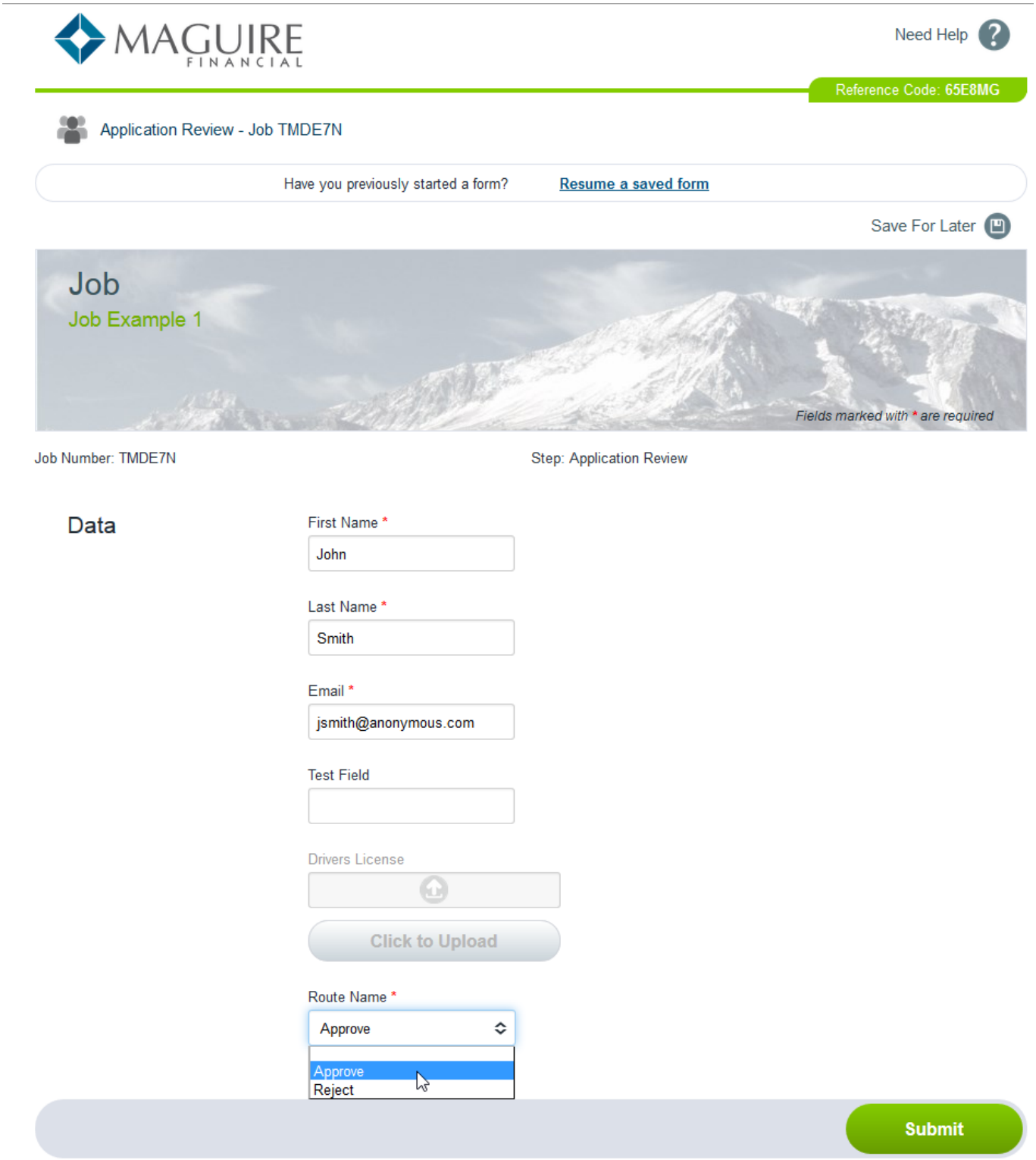
```
{
 "$schema": "http://json-schema.org/draft-07/schema#",
 "$id": "http://example.com/product.schema.json",
 "jobDetails": {
  "name": "1 Step Review",
  "processSubmitImmediate": false,
  "version": "4.1.0"
 },
 "jobGroups": {},
 "properties": {},
 "steps": {
  "Application Start": {
   "type": "start",
   "actions": {
    "Accept Quote": {
     "name": "Accept Quote",
     "type": "Job Form Start",
     "properties": {
         "Process Message Send Email": {
          "value": "true",
          "template": false
         },
         "Process Message Text": {
          "value": "Thank you ${formDataMap.firstName} ${formDataMap.lastName} your ${sub-
mission.formName} is being processed.",
          "template": false
         }
        }
    }
  },
  "routes": {
   "Default": {
    "nextStep": "Application Review"
   }
  },
  "properties": {}
  },
  "Application Review": {
   "type": "",
   "actions": {
    "Assign Review": {
```

```
        "name": "Assign Review",
        "type": "Job Task Assign",
        "properties": {
         "Task Assign Groups": {
          "value": "Job Reviewers, Job Managers",
          "template": false
         },
         "Task Form Code": {
          "value": "$func.startFormCode()",
          "template": false
         },
         "Task Message": {
          "value": "Please review the ${submission.formName} by ${formDataMap.firstName}
${formDataMap.lastName}.",
          "template": false
         },
         "Task Review Previous Step": {
          "value": "true",
          "template": false
         },
         "Task Subject": {
          "value": "Review ${submission.formName} by ${submission.contactEmailAddress}.",
          "template": false
         },
         "Task Type": {
          "value": "Review",
          "template": false
         }
        }
       },
       "Review Wait": {
        "name": "Review Wait",
        "type": "Job Task Wait",
        "properties": {}
       }
   },
   "routes": {
    "Approve": {
       "nextStep": "Application Delivery"
   },
   "Reject": {
```

```
    "nextStep": "Application Rejected"
   }
  },
  "properties": {}
 },
 "Application Delivery": {
             "type": "",
             "actions": {
                     "Application Delivery": {
                             "name": "Application Delivery",
                             "type": "Job Delivery",
                             "properties": {}
                     },
                     "Application Delivery Wait": {
                             "name": "Application Delivery Wait",
                             "type": "Job Delivery Wait",
                             "properties": {}
                     }
             },
             "routes": {
                     "Default": {
                             "nextStep": "Application Completed"
                     }
             },
             "properties": {}
     },
     "Application Completed": {
             "type": "endpoint",
             "actions": {
                     "Process Message": {
                             "name": "Process Message",
                             "type": "Job Process Message",
                             "properties": {
                                     "Process Message Send Email": {
                                             "value": "true",
                                             "template": false
                                     },
                                     "Process Message Submission": {
                                             "value": "$func.startSubmission()",
                                             "template": false
                                     },
```

```
                                        "Process Message Text": {
                                                "value": "Thank you ${formDataMap.firstName} ${formDataM
mission.formName} has been Approved.",
                                                "template": false
                                        }
                                }
                        }
                },
                "properties": {}
        },
        "Application Rejected": {
                "type": "endpoint",
                "actions": {
                        "Process Message": {
                                "name": "Process Message",
                                "type": "Job Process Message",
                                "properties": {
                                        "Process Message Send Email": {
                                                "value": "true",
                                                "template": false
                                        },
                                        "Process Message Submission": {
                                                "value": "$func.startSubmission()",
                                                "template": false
                                        },
                                        "Process Message Text": {
                                                "value": "Sorry ${formDataMap.firstName} ${formDataMap.f
mission.formName} has been declined.",
                                                "template": false
                                        }
                                }
                        }
                },
                "properties": {}
                }
        }
}
```

The Application Review step has two possible next steps: Application Approved and Application Rejected. The first action of the Application Review step creates and assigns a Group Task. A user who is a member of that group can open the form associated with the task as shown below.



The user has an option of approving or rejecting the form by selecting a corresponding route from the Route Name dropdown list before submitting the form. If the user selects Approve, the step is routed to the Application Approved step. If the user selects Reject , it is routed to the Application Rejected step.

You can view the list of all steps for each collaboration job.

Next, learn about collaboration job actions.

Actions

A collaboration job consists of a number of steps and actions. Each action executes an instance of an action service or a job function. There are a number of predefined actions that you can use. You can also create Groovy[1] actions and expiry services to implement customer requirements. You can view all actions associated with the collaboration job. A Action can have one or more action properties.

An action can be assigned to a different user. However, more often a single user is assigned to work on several actions. To achieve a seamless transition between the actions, the `"redirectNext": true` property is used to show next action to the same user upon the completion of the previous action.

An action can be anonymous. In this case, you should set the `"name": "Task Type", "value": "Anonymous"` property.

- In Progress - the action is active and in progress. The action will be retried (looping), if needed.
- Completed - the action has been completed.

- Expired - the action is expired when a step is automatically expired by the system. Any actions that are In Progress (Pending, Assigned, Invoked, Ready or In Progress) will be marked as Expired. This include action associated with tasks that are in the Assigned state or job actions that are in the Invoked state awaiting a callback from a 3rd party system.
- Cancelled - the action is manually canceled by an administrator, who cancelled a job or transaction, or a user, who clicked cancel on a form task.
- Error - the action has a critical or unrecoverable error and requires an operator's or administrator's intervention before it can be run again.
- Invoked - the action that is a part of an Asynchronous execution where you have a Job Action and a Job Action Wait pair. The Job Actions processing (invoking a 3rd party service) has completed successfully, so it is awaiting a callback from a 3rd party system and, therefore, it can move to the completed status.
- Pending - or paused, the action is waiting for an external event to move it forward.

---

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

- Assigned - the Job Task Assign Action process has successfully created and assigned a task to a user, group or email (anonymous). The task is awaiting user submission to be completed which then executes a callback on the collaboration job.
- Ready - the action will be executed next time the scheduled job runs.

Manager allows you to use various actions, such as a simple, auto retry, or wait actions, that are processed differently.

Simple Actions

A job controller processes a simple action as described below:

Normal Case

An action service completes an action and sets the its status as Completed. Then, a job controller proceeds to the next action or a step.

Error Case

An action service may stop processing an action, depending on errors:

- Transient Errors - the errors that can be retried automatically, for example, in case a web service is down, by configuring the retry time interval of action services.
- Hard Errors - the errors that can't be fixed by retrying, for example, a bug in the source code. In this case, you may want to stop a collaboration job until the error is fixed and then run the collaboration job manually.

Polling Case

An action service can implement polling or looping logic to check if a certain event has occurred, for example, whether a receipt has been generated. If the event has occurred, the service sets the action status as Completed and the job controller proceeds to the next action or step. If the event has not occurred, the service sets the action status as In Progress.

Expiry Case

You can configure an action to expire by setting an expiry time, so the action status will be set as Expired. You can also configure a step to expire.

Auto Retry Actions

An action with the Error or In Progress statuses can implement the retry (looping) behavior, when it is retried each time a job controller service runs. By default, the job controller service tries to

retry an action after a five minute delay, but you can change it by updating the `actionRetryDelayMins` service parameter. To illustrate it with an example, let's assume your action calls a web service. If the web service is down for a couple of minutes, the action instance status will be set as Error and the next time the action will be run is set as the current time plus 5 minutes. The next time your action is executed, the web service is running, so the action completes and its status is set as Completed. This concludes the life cycle of your action.

You can always write a custom job action in Groovy script to specify the next time it runs.

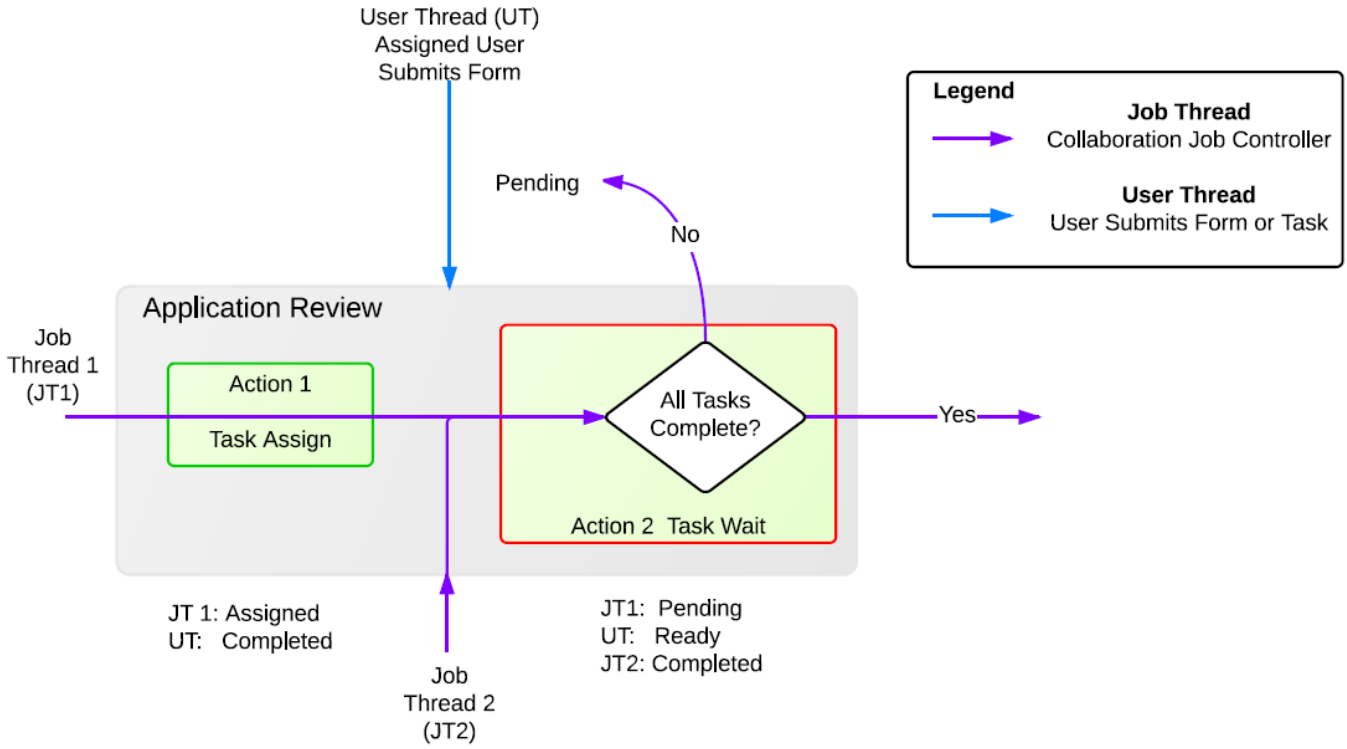
Wait Actions

A wait action implements a callback to handle asynchronous events, such as a user completing a task. A Job Task Wait Action service is paired with a single step having one or more Job Task Assign Action. The Job Task Assign Action creates a task and assigns it to a user or a group. The user completes the task by submitting it, which could be within hours, days or even weeks after the task was assigned. The submission triggers a callback, which enables the collaboration job to complete.

The examples below show how to use different actions within various steps.

Example 1 - Task Step - Standard Mode Processing

The state changes associated with the actions in a task step are illustrated in the diagram below.



Job Thread 1

1. Runs the Job Task Assign action immediately.
2. Creates and assigns the task submission.
3. Links the task submission to the action instance.
4. Sets the Task Assign Action status to Assigned.
5. Runs the Task Wait processing. As the Action 1 task isn't complete the Action Status of the Task Wait to Pending and Job Thread 1 terminates. No processing is done on the collaboration job where the current action is in Pending status.

Assigns User Action 1 - Task Submission Thread

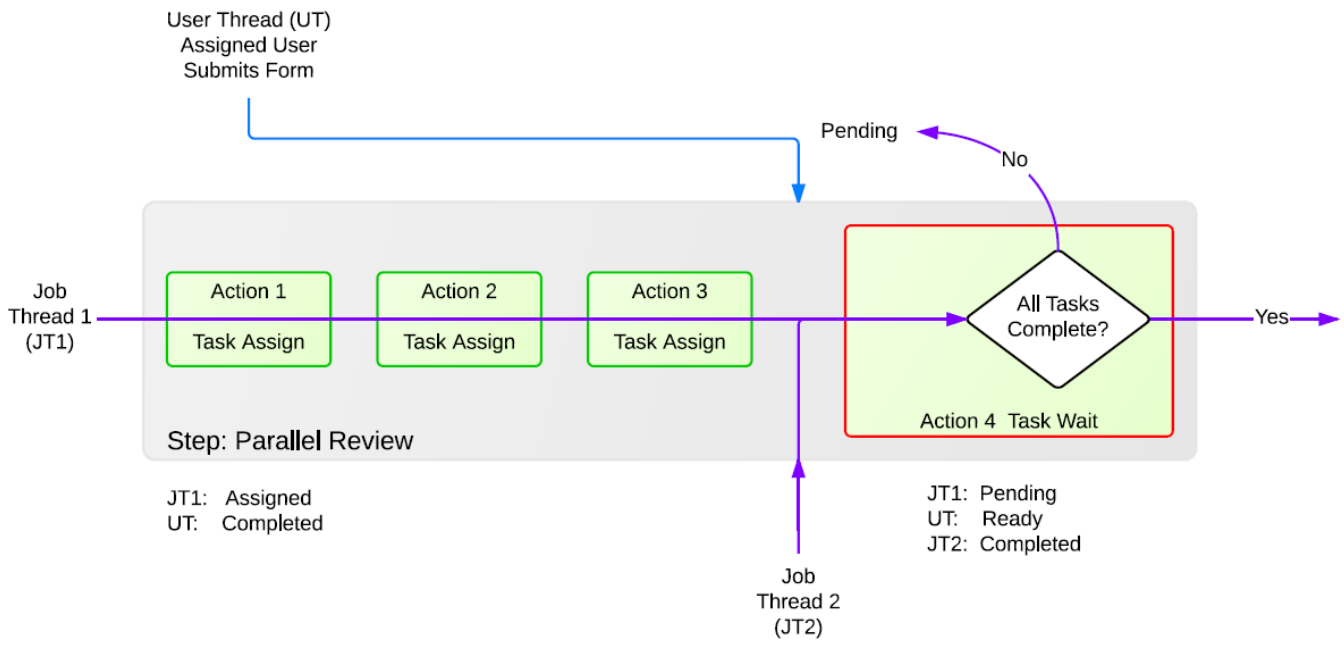
1. It generates a callback.
2. Changes the Task Assign action status from Assigned to Completed.
3. Changes the Task Wait action status from Pending to Ready.
4. As a result, the Task Wait is now available for processing.

Job Thread 2

1. Completes the Task Wait processing.
2. Changes the Task Wait status to Completed.
3. Completes the application review step.
4. Routes to the next step to be processed.

Example 2 - Multiple Tasks Step - Standard Mode Processing

The Job Task Wait Action service can wait on more than one Job Task Assign action services to complete. The diagram below shows a step with three assigned tasks that run in parallel. This can be created using the same form assigned to different users or groups.



Job Thread 1

1. Creates all of the action instances and the 3 associated tasks are created and are given an action status of Assigned.
2. Stops when it gets to the Task Wait and sets the status to Pending.

Assigned User Action 1 - Task Submission Thread

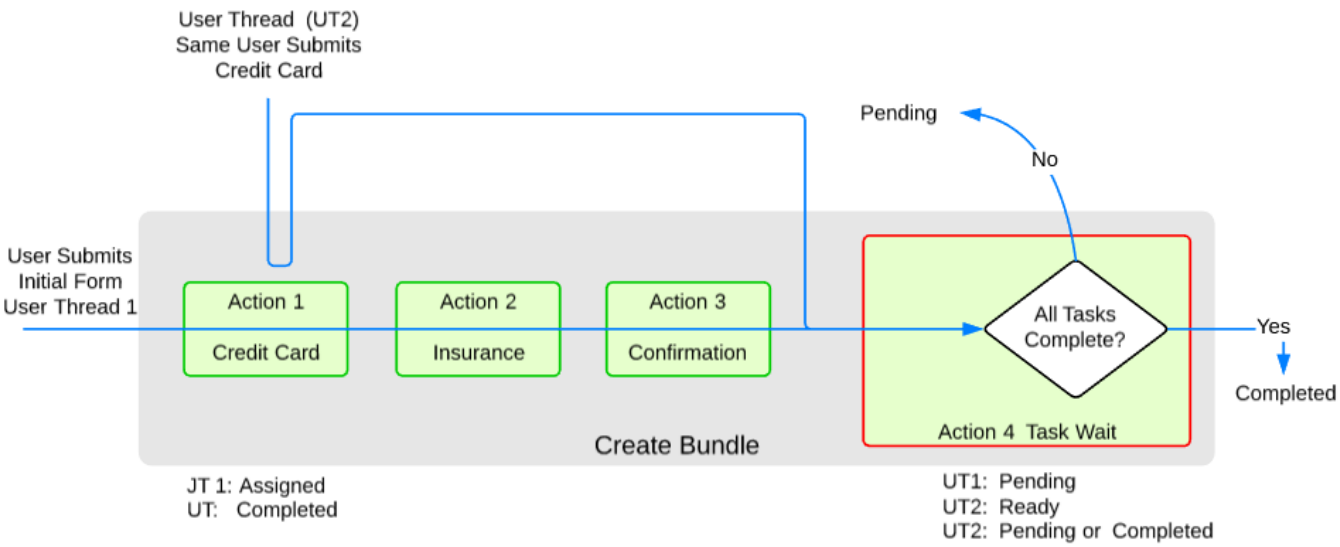
1. Callback triggered that changes the Action 1 status to Completed and Changes the Task Wait status to Ready

Job Thread 2

1. The Task Wait Processing is run which check to check whether Tasks Complete or not.
2. If complete, the Task Wait action status is set to Completed. The step is completed and processing the next step is done.
3. If not complete, the Task Wait action status is set to Pending.

Example 3 - Form Bundle - Process Immediate Mode

Form bundles can create multiple Task Assign action instances that are assigned back to the original submitter.



User Thread 1 - User Submits Initial Form

1. Creates all of the action instances and the 3 associated tasks are created and assigned to the same user, they are given an action status of Assigned.
2. Stops when it gets to the Task Wait and sets the status to Pending.

User Thread 2 - Same User Submits Credit Card:

1. Callback triggered that changes the Action 1 status to Completed and Changes the Task Wait status to Ready
2. Immediately Runs Task Wait processing is run which check to see Are Tasks Complete:
3. No: The action status for the Task Wait is set to Pending.
4. Yes: The action status for the Task Wait is set to Completed. The step is completed and processing the next step starts.

Actions can assign a group to a collaboration job. The example below shows how to assign the Job Reviewers group to a collaboration job.

```
"actions": [
    {
        "name": "Create Task",
        "type": "Job Task Assign",
        "properties": [
            ...
            { "name": "Task Assign Group",  "value": "Job Reviewers" },
            ...
        ]
    }
]
```

Next, learn how to view collaboration job steps.

Action Properties

An Action can have a set of properties, such as the Job Task Assign service properties, which are associated with:

Status Updates and Email
: This service can add a submission processing status messages to the specified Submission. These appear in the user's submission history. It can also send a status update via email.

Task Assignment
: These properties are used to create and assign user and group tasks. Tasks can be categorized as:

- Standard Tasks - are for an authenticated user or group.
- Review Tasks - are the similar to the standard tasks with attachments copied from a previous submission.
- Anonymous Save - when a submission record is saved but not assigned to a user in the Manager database. They are usually assigned to a email address, so the user is sent a notification email with a link. They click on a link and enter a challenge response before they have access to the task form.

Task Types

The Task Type action properties specify either the Anonymous, Form or Review task. Form and Review tasks are both require a user to be an authenticated user, a user Task Assign User or group(s) Task Assign Group or Task Assign Groups must be specified.

Anonymous tasks don't require a user authentication, but an email address, such as Task Assign Email, and a portal, such as Task Assign Portal. When specifying the form XML and Attachments for a Review task reference has to be given to a previous submission. The Task creation process will automatically copy the XML and Attachments for you from the previous submission. However, for Form and Anonymous tasks, we don't know about previous tasks, so the XML and attachments needs to be explicitly specified.

The XML requires either the Task Form XML Data or Task Input XML Prefill to be configured.

The attachments can be copied from a previous submission using the Task Attachments Previous step or the Task Attachments Submission step

The following table summarizers the action property required by a task type. Where a single letter is repeated in two cells, only one of the action properties should be used.

Action Property Name	Form	Review	Anonymous
Task Form XML Data	A		B
Task Input XML Prefill	A		B
Task Review Previous Step		C	
Task Review Submission Step		C	
Task Attachments Previous Step	D		D
Task Attachments Submission Step	D		D
Task Assign User	E*	F*	
Task Assign Group	E	F	
Task Assign Groups	E	F	
Task Assign Email			G
Task Assign Portal	Optional	Optional	H

Note: * A user can be assigned at the same time as a claimable group. This is the same as a user claiming the Task. The user must belong to one of the assigned groups.

A Form Task job definition is shown below.

```
{
  "jobDetails": {
    "name": "1 Step Review - Form Task with Attach",
    "version": "4.2.0"
  },
  "steps": [
    {
      "name": "Application Start",
      "type": "start",
      "actions": [
        {
          "name": "Accept Quote",
          "type": "Job Form Start"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Review" }
      ]
    },
    {
      "name": "Application Review",
      "type": "",
      "actions": [
        {
          "name": "Assign Review",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Task Assign User",  "value": "roger" },
            { "name": "Task Form Code",  "value": "$func.startFormCode()" },
            { "name": "Task Message",  "value": "blah" },
            { "name": "Task Attachments Previous Step",  "value": "true" },
            { "name": "Task Form XML Data",  "value": "$func.stepOrStartSubmissionXml()"
},
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Send Email",  "value": false },
            { "name": "Task Subject",  "value": "Review this" },
            { "name": "Task Type",  "value": "Form" }
          ]
        },
        {
          "name": "Review Wait",
```

```
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Completed" },
        { "name": "Reject",  "nextStep": "Application Rejected" }
      ]
    },
    {
      "name": "Application Completed",
      "type": "endpoint"
    },
    {
      "name": "Application Rejected",
      "type": "endpoint"
    }
  ]
}
```

An Anonymous Task job definition is shown below.

```
{
  "jobDetails": {
    "name": "1 Step Review - Form Task with Attach",
    "version": "4.2.0"
  },
  "steps": [
    {
      "name": "Application Start",
      "type": "start",
      "actions": [
        {
          "name": "Accept Quote",
          "type": "Job Form Start"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Review" }
      ]
    },
    {
      "name": "Application Review",
      "type": "",
      "actions": [
        {
          "name": "Assign Review",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Task Assign Email",  "value": "${formDataMap.email}" },
            { "name": "Task Form Code",  "value": "$func.startFormCode()" },
            { "name": "Task Message",  "value": "Please review the ${submission.formName}
by ${formDataMap.firstName} ${formDataMap.lastName}." },
            { "name": "Task Subject",  "value": "Review ${submission.formName} by ${sub-
mission.contactEmailAddress}." },
            { "name": "Task Assign Portal",  "value": "Maguire" },
            { "name": "Task Form XML Data",  "value": "$func.stepOrStartSubmissionXml()"
},
            { "name": "Task Attachments Submission Step",  "value": "Application Start"
},
            { "name": "Task Send Email",  "value": "true" },
            { "name": "Task Email Message",  "value": "$func.formProperty('Anonymous Task
Email Message')" },
```

```
            { "name": "Task Email Subject",  "value": "$func.formProperty('Anonymous Task
Email Subject')" },
            { "name": "Task Type",  "value": "Anonymous" }
          ]
        },
        {
          "name": "Review Wait",
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Reject",  "nextStep": "Application Rejected" }
      ]
    }    {
      "name": "Application Completed",
      "type": "endpoint"
    },
    {
      "name": "Application Rejected",
      "type": "endpoint"
    }
  ]
}
```

Job Functions

You can use `$func` as an Action Property value. For more information, see Job Functions Core API.

For example, if you need to fetch a submission property value within a collaboration job controller JSON definition, which to be injected into a job action property value, you should use the `$func.startSubmission.getSubmissionPropertyValue('propertyName')` call.

Next, learn about job functions.

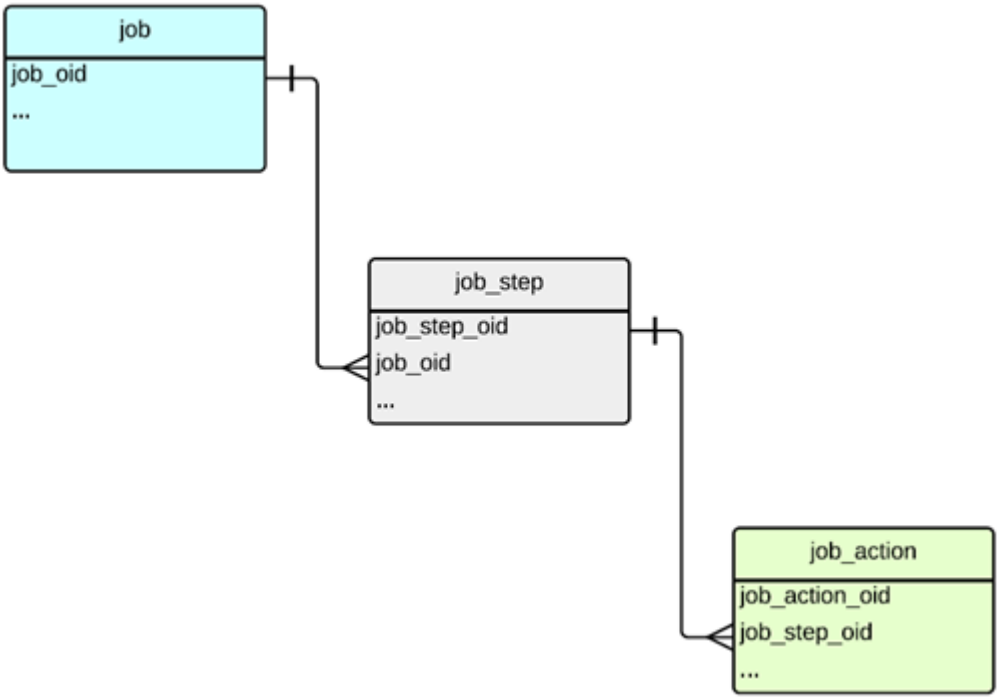
Collaboration Job Basics

A collaboration job contains of two or more steps, each step contains one or more actions. This hierarchy flows down into the design of the job database entities and the structure of a job definition.

Manager stores collaboration jobs' entities in the following database tables:

- job - job instances
- job_step - step instances
- job_action- action instances

The tables relationship is shown in the diagram below.



A collaboration job has a status.

- In Progress - the collaboration job is active and in progress.
- Completed - the collaboration job has been completed. This happens when a endpoint step has completed.
- Expired - this happens when an endpoint expiry collaboration job has been automatically expired by Journey Manager.
- Cancelled - the collaboration job has been manually canceled by an administrator.
- Error - the collaboration job has a critical or unrecoverable error that requires an admin-

istrator intervention before it can be run again.

- Paused - the collaboration job has been paused temporarily.

Next, learn how to.

View Collaboration Jobs

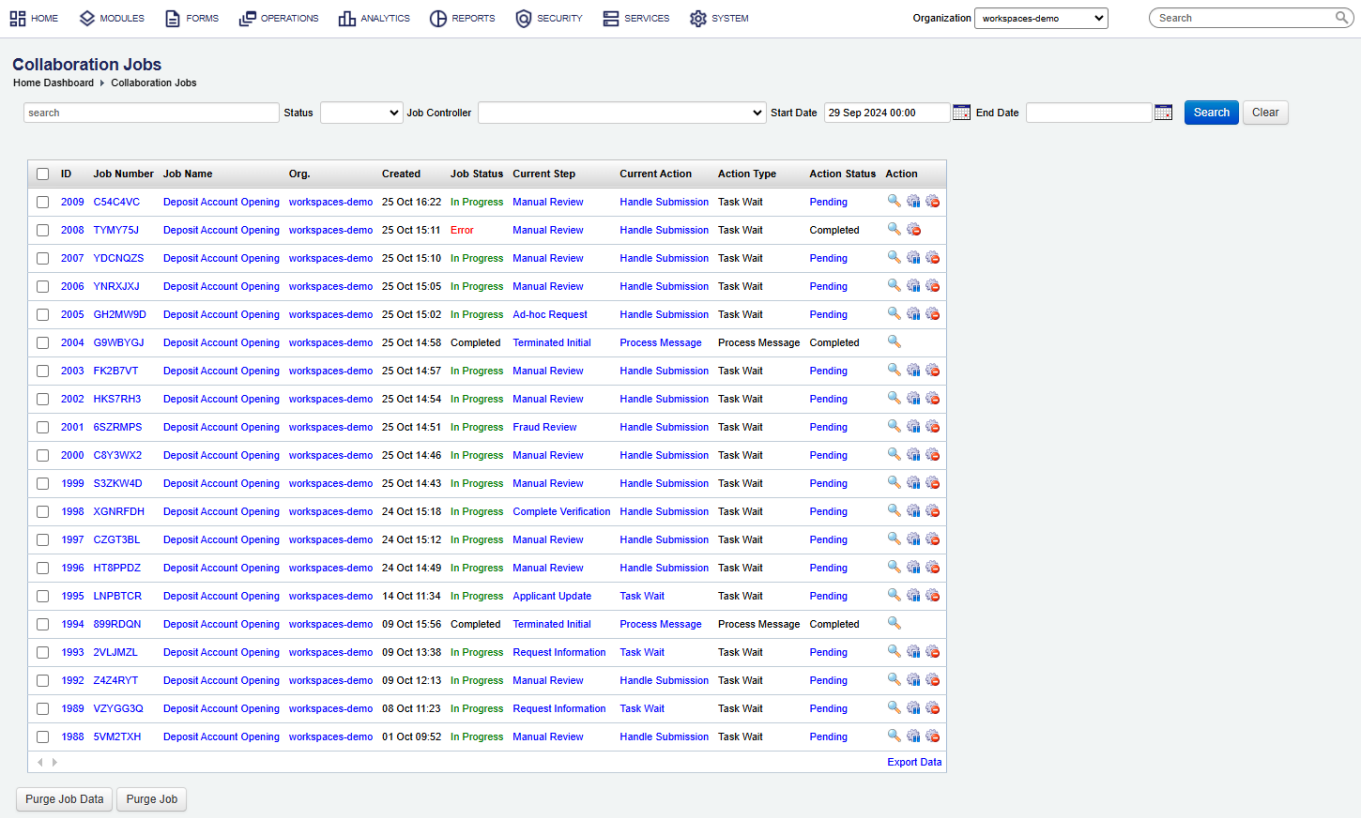
Manager allows you to search for and view collaboration jobs that are in various states of onboarding flows but have not been purged yet. You can then drill down into a collaboration job to see which step and action it is at. This is especially useful in troubleshooting transactions.

This UI also provides an option to manually purge a single or multiple collaboration jobs. This will mark the collaboration jobs for immediate deletion so they will be removed from the system during the next run of the Transaction Processor service, which is usually run every 5 minutes by the Transaction Processing scheduled job. This is the addition to the existing Manual Data Purge functionality.

> NOTE
>
> You have to be a Job Coordinator or an Administrator to use this functionality.

To view collaboration jobs, select Operations > Collaboration Jobs.

Collaboration jobs are shown page by page, with the maximum number of records per page configured in the users preferences.



ID	Job Number	Job Name	Org.	Created	Job Status	Current Step	Current Action	Action Type	Action Status	Action
2009	C54C4VC	Deposit Account Opening	workspaces-demo	25 Oct 16:22	In Progress	Manual Review	Handle Submission	Task Wait	Pending	
2008	TYMY75J	Deposit Account Opening	workspaces-demo	25 Oct 15:11	Error	Manual Review	Handle Submission	Task Wait	Completed	
2007	YDCNQZS	Deposit Account Opening	workspaces-demo	25 Oct 15:10	In Progress	Manual Review	Handle Submission	Task Wait	Pending	
2006	YNRXJXJ	Deposit Account Opening	workspaces-demo	25 Oct 15:05	In Progress	Manual Review	Handle Submission	Task Wait	Pending	
2005	GH2MW9D	Deposit Account Opening	workspaces-demo	25 Oct 15:02	In Progress	Ad-hoc Request	Handle Submission	Task Wait	Pending	
2004	G9WBYGJ	Deposit Account Opening	workspaces-demo	25 Oct 14:58	Completed	Terminated Initial	Process Message	Process Message	Completed	
2003	FK2B7VT	Deposit Account Opening	workspaces-demo	25 Oct 14:57	In Progress	Manual Review	Handle Submission	Task Wait	Pending	
2002	HKS7RH3	Deposit Account Opening	workspaces-demo	25 Oct 14:54	In Progress	Manual Review	Handle Submission	Task Wait	Pending	
2001	6SZRMPS	Deposit Account Opening	workspaces-demo	25 Oct 14:51	In Progress	Fraud Review	Handle Submission	Task Wait	Pending	
2000	C8Y3WX2	Deposit Account Opening	workspaces-demo	25 Oct 14:46	In Progress	Manual Review	Handle Submission	Task Wait	Pending	
1999	S3ZKW4D	Deposit Account Opening	workspaces-demo	25 Oct 14:43	In Progress	Manual Review	Handle Submission	Task Wait	Pending	
1998	XGNRFDH	Deposit Account Opening	workspaces-demo	24 Oct 15:18	In Progress	Complete Verification	Handle Submission	Task Wait	Pending	
1997	CZGT3BL	Deposit Account Opening	workspaces-demo	24 Oct 15:12	In Progress	Manual Review	Handle Submission	Task Wait	Pending	
1996	HT8PPDZ	Deposit Account Opening	workspaces-demo	24 Oct 14:49	In Progress	Manual Review	Handle Submission	Task Wait	Pending	
1995	LNPBTCR	Deposit Account Opening	workspaces-demo	14 Oct 11:34	In Progress	Applicant Update	Task Wait	Task Wait	Pending	
1994	899RDQN	Deposit Account Opening	workspaces-demo	09 Oct 15:56	Completed	Terminated Initial	Process Message	Process Message	Completed	
1993	2VLJMZL	Deposit Account Opening	workspaces-demo	09 Oct 13:38	In Progress	Request Information	Task Wait	Task Wait	Pending	
1992	Z4Z4RYT	Deposit Account Opening	workspaces-demo	09 Oct 12:13	In Progress	Manual Review	Handle Submission	Task Wait	Pending	
1989	VZYGG3Q	Deposit Account Opening	workspaces-demo	08 Oct 11:23	In Progress	Request Information	Task Wait	Task Wait	Pending	
1988	5VM2TXH	Deposit Account Opening	workspaces-demo	01 Oct 09:52	In Progress	Manual Review	Handle Submission	Task Wait	Pending	

The list displays the following details:

- ID: an internal table ID of a collaboration job, which uniquely identifies it in Manager.
- Job Number (Job Reference Number): a unique code, which is generated on the first submission of a form and assigned to a collaboration job using:
  - A Job Number, which is the default behavior of a collaboration job created on a form submission, The job number is shown on a form as a reference code (a transaction tracking code), so a form user (the form applicant) can use to reference the status of their application.
  - A Job Reference Number, when a collaboration job is created using the Fluent API, such as `JobBuilder`.

> NOTE
>
> Each subsequent job task has a different tracking code to the initial submitted form. However, they may show the Job Number in a separate field on the form.

- Job Name: the name of the job controller service this collaboration job use.
- Org: an organization this collaboration job belongs to.
- Created: day and time when this collaboration job was created.
- Job Status: a status of a collaboration job, which can be in one of the following depending on execution of its steps and actions.
  - In Progress - the collaboration job is active and in progress.
  - Completed - the collaboration job has been completed. This happens when a endpoint step has completed.
  - Expired - this happens when an endpoint expiry collaboration job has been automatically expired by Journey Manager.
  - Cancelled - the collaboration job has been manually canceled by an administrator.
  - Error - the collaboration job has a critical or unrecoverable error that requires an administrator intervention before it can be run again.
  - Paused - the collaboration job has been paused temporarily.
- Current Step: the step the collaboration job is at.
- Current Action: the action the collaboration job is at.
- Action Type: a type of the action, which can be one of the following:
  - Action
  - Input
  - Output
  - Task Assign
  - Task Wait

- Action Status: a status of the action, which can be one of the following:
  - In Progress - the action is active and in progress. The action will be retried (looping), if needed.
  - Completed - the action has been completed.

  - Expired - the action is expired when a step is automatically expired by the system. Any actions that are In Progress (Pending, Assigned, Invoked, Ready or In Progress) will be marked as Expired. This include action associated with tasks that are in the Assigned state or job actions that are in the Invoked state awaiting a callback from a 3rd party system.
  - Cancelled - the action is manually canceled by an administrator, who cancelled a job or transaction, or a user, who clicked cancel on a form task.
  - Error - the action has a critical or unrecoverable error and requires an operator's or administrator's intervention before it can be run again.
  - Invoked - the action that is a part of an Asynchronous execution where you have a Job Action and a Job Action Wait pair. The Job Actions processing (invoking a 3rd party service) has completed successfully, so it is awaiting a callback from a 3rd party system and, therefore, it can move to the completed status.
  - Pending - or paused, the action is waiting for an external event to move it forward.
  - Assigned - the Job Task Assign Action process has successfully created and assigned a task to a user, group or email (anonymous). The task is awaiting user submission to be completed which then executes a callback on the collaboration job.
  - Ready - the action will be executed next time the scheduled job runs.

To filter or search the collaboration jobs by one or more criteria, specify the following settings and click Search:

- Enter a collaboration job's ID, job number, job name or part of a name, for example, claimable.
- Select a collaboration job status from the Status dropdown list.
- Select a job controller from the Job Controller dropdown list.

  NOTE

  The Job Controller dropdown list includes only job controllers for organizations that the administrator has access to, which means you can only search for jobs linked to those organizations.

- Specify Start Date to filter collaboration jobs from that date.

- Specify End Date to filter collaboration jobs up to that date.
- Select an organization from the Organization dropdown list. This only applies to a global administrator, or an organizational administrator who has been assigned to more than one organization.

Click Clear to reset the search criteria.

You can also perform the following actions:

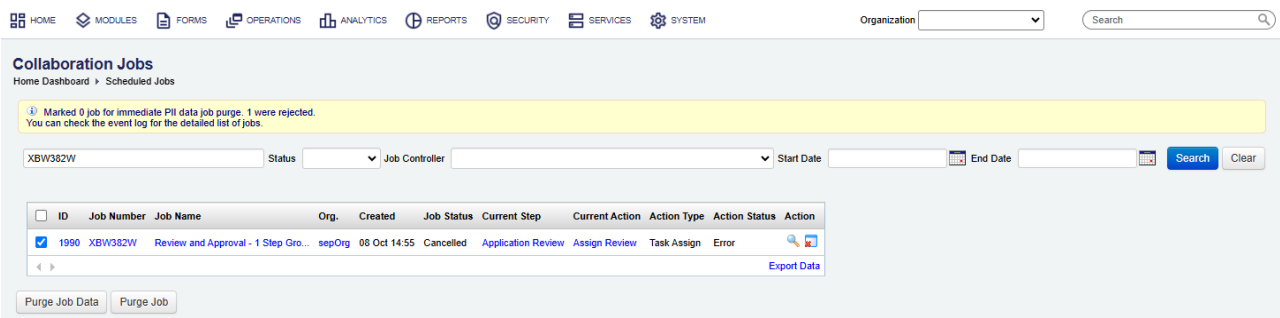
> NOTE
>
> Depending on a status of a selected collaboration job, a different set of actions is available for you to perform.

- Click View to view collaboration job details.
- Click Pause Job to temporarily pause collaboration job processing.
- Click Process Job to start collaboration job processing immediately, which is useful for developers to troubleshoot an action service as they can make a change to the service and execute it to see if it works or not right away. An administrator can re-run a failed action once it is fixed.
- Click Cancel Job to permanently cancel a job that is in progress.
- Click Remove Job to permanently delete a collaboration job. This option is useful for cleaning out test collaboration jobs created during form development.

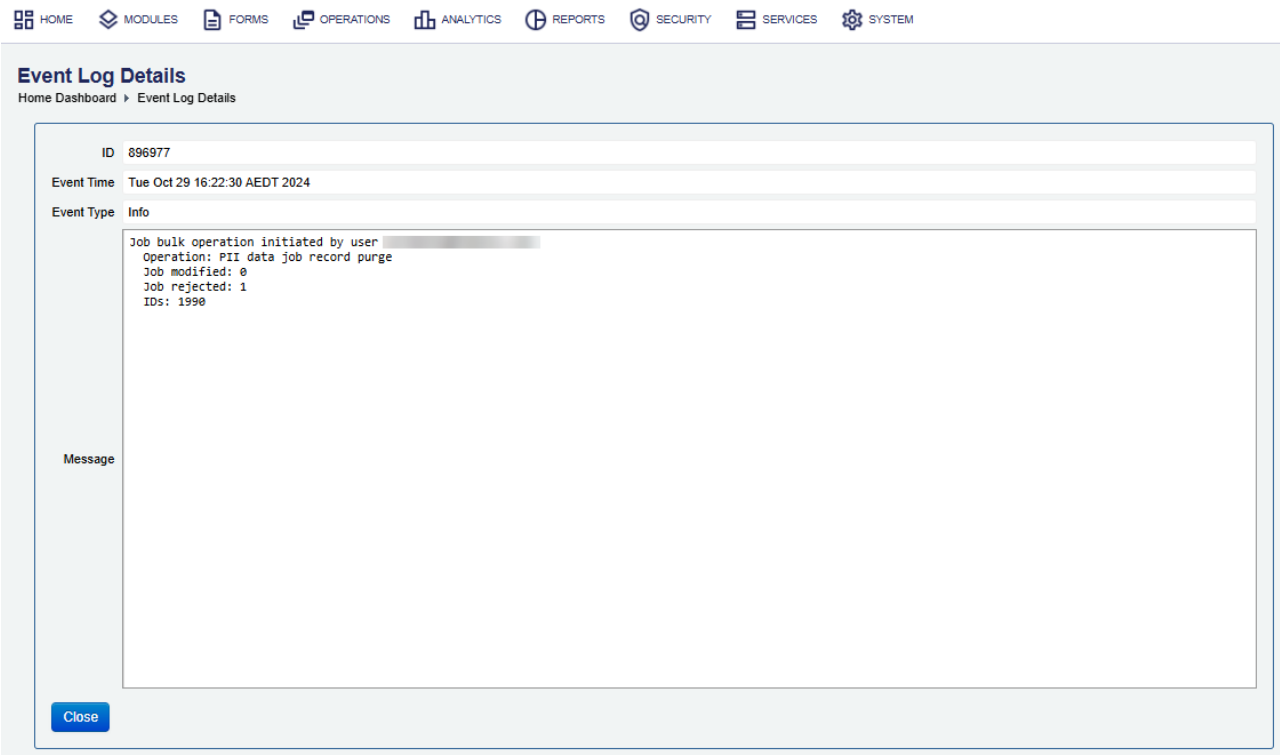
  > NOTE
  >
  > You must cancel a collaboration job first, so its status changes to Cancelled and the Remove Job option becomes available.

- Click Purge Job Data to delete PII data of selected collaboration jobs that are in the Completed state. This will schedule them for immediate deletion. | 24.10 This feature was introduced in 24.10
- Click Purge Job to delete selected collaboration jobs, including their PII data, that are in the Completed state. This will schedule them for immediate deletion. | 24.10 This feature was introduced in 24.10

  For both actions, the status of manual purging is displayed in UI, as shown below.



You can check the details in Event Log, which is illustrated below.



In this case the collaboration job was not purged because it was not in the Completed state.

Also, if the collaboration job's data has already been deleted, the following message is displayed:

```
Marked 0 job for immediate PII data job purge. 1 were rejected.
You can check the event log for the detailed list of jobs.
```

NOTE

Open collaboration job details and see if the Job Data Deleted checkbox is selected to confirm the PII data was purged.

The successful purging messages are:

- For the View Collaboration Jobs UI:

```
Marked 1 job for immediate job record purge.
You can check the event log for the detailed list of jobs.
```

- For the Event Log:

```
Job bulk operation initiated by user user@mycompany.com:
  Operation: Job record purge
  Job modified: 1
  IDs: 2004
  Job rejected: 0
```

Manager doesn't provide UI to create a collaboration job manually, because it is based on a form's submission where a form is associated with a job controller. However, for submissions in the Open or Saved states, you can start a collaboration job earlier in the initial journey flow, that is before a user submits a form. Using the `JobBuilder`Fluent API - `setInitialTxnTrackingCode()` - you create a job linked to such submissions any time after the initial form request. For more information, see Job Details. | 24.10 This feature was introduced in 24.10

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

TIP

You can also view a collaboration job workflow in Journey Analytics.

Next, learn how to check collaboration job details.

Collaboration Job Details

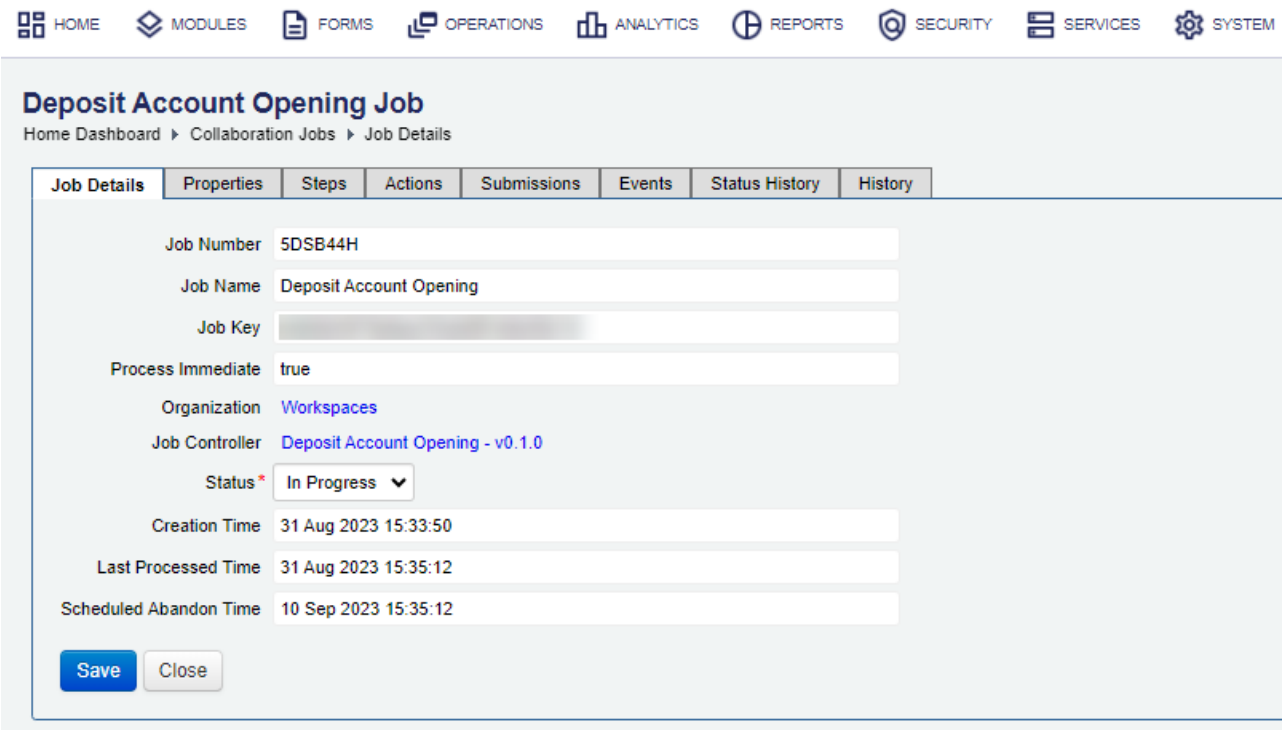
Manager allows you to check details of a collaboration job and update its status, which is useful when you need to resolve some issues related to a form workflow when the collaboration job has been assigned a wrong status.

To view the details of a collaboration job:

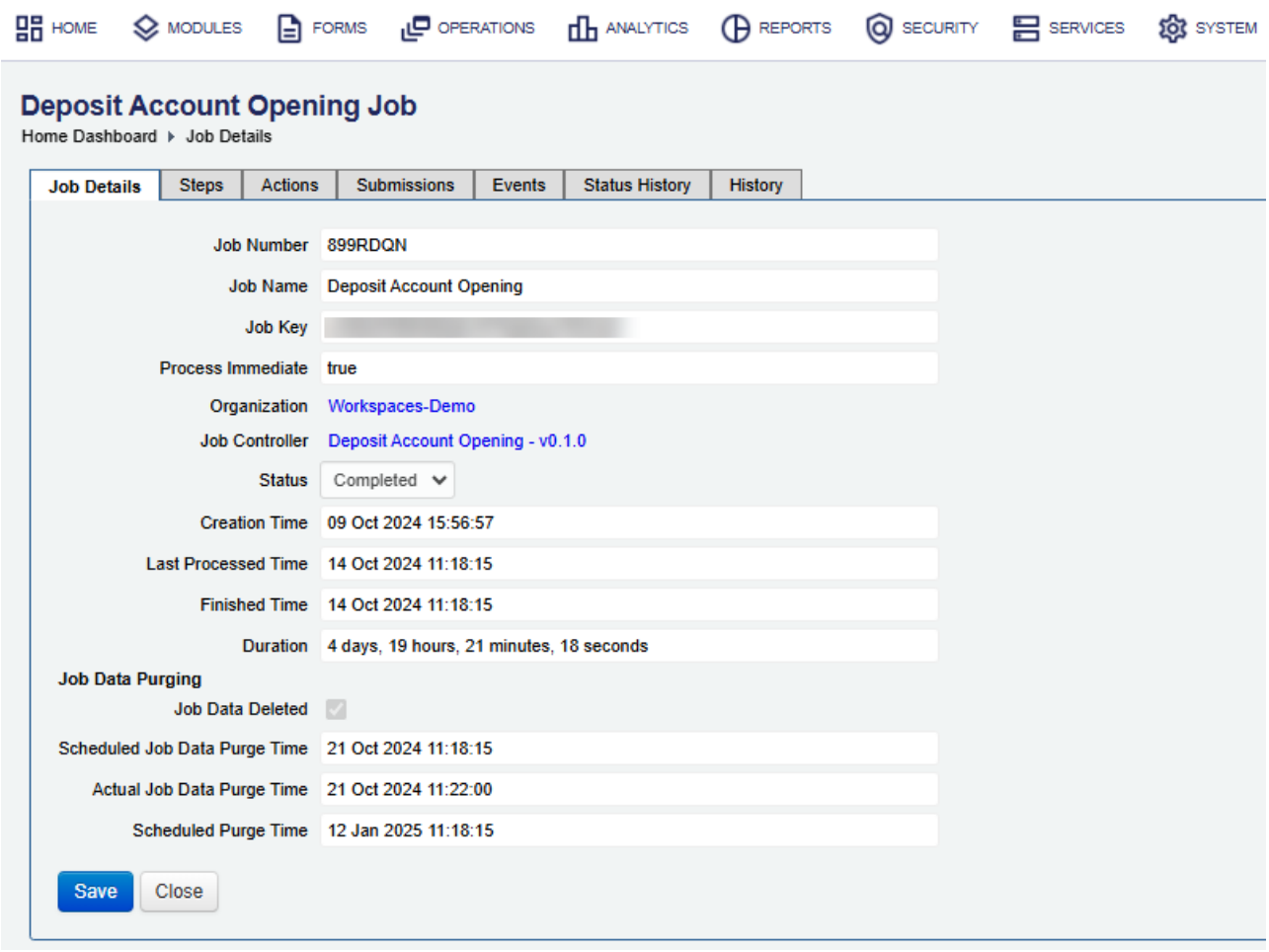
1. Select Operations > Collaboration Jobs.
2. Locate a collaboration job and click View.

   A different set of properties is shown for the same collaboration job as its status changes. This is illustrated below:

   - In Progress
   - Completed
   - Cancelled



In Progress


HOME
MODULES
FORMS
OPERATIONS
ANALYTICS
REPORTS
SECURITY
SERVICES
SYSTEM
Deposit Account Opening Job
Home Dashboard ▸ Job Details
Job Details
Steps
Actions
Submissions
Events
Status History
History
Job Number
899RDQN
Job Name
Deposit Account Opening
Job Key
Process Immediate
true
Organization
Workspaces-Demo
Job Controller
Deposit Account Opening - v0.1.0
Status
Completed
Creation Time
09 Oct 2024 15:56:57
Last Processed Time
14 Oct 2024 11:18:15
Finished Time
14 Oct 2024 11:18:15
Duration
4 days, 19 hours, 21 minutes, 18 seconds
Job Data Purging
Job Data Deleted
Scheduled Job Data Purge Time
21 Oct 2024 11:18:15
Actual Job Data Purge Time
21 Oct 2024 11:22:00
Scheduled Purge Time
12 Jan 2025 11:18:15
Save
Close

Completed

Cancelled

3. Check the collaboration job's information in theJob Details tab, which is the same as described in view collaboration jobs with few extra fields:

   - Process Immediate indicates how this collaboration job is processed by a scheduled job: `true` - the collaboration job is processed right after it is created at the end of the submission; `false` - it's queued for a scheduled job.
   - Last Processed Time shows the time the collaboration job was last processed.

     NOTE

     The Transaction Processor service uses the Last Processed Time to automatically abandon the collaboration job. For more information, see Configure Data Retention.

   - Job Data Deleted: indicates whether the PII data was purged or not.

Job Data Purging

Check the following properties:

- If the Scheduled Job Data Purge Time checkbox is selected, then the PII data was purged.
- The Scheduled Job Data Purge and Actual Transaction Data Purge times tell you when the collaboration job was purged.
- The Scheduled Purge Time time to see when the collaboration job data record is due to be deleted.

4. For collaboration jobs with the In Progress or Paused status, select a new status from the Status dropdown list, which can be one of the following:
   - In Progress
   - Paused
   - Cancelled

   WARNING

   You should understand statuses and transitions between them before changing a status of a collaboration job, as some transitions are irreversible. If you change the status from In Progress to Cancelled, you can't undo it. However, if you change the status from In Progress to Paused, you can change it back to In Progress.

5. Click Save to update the changes. It saves the updated status only.

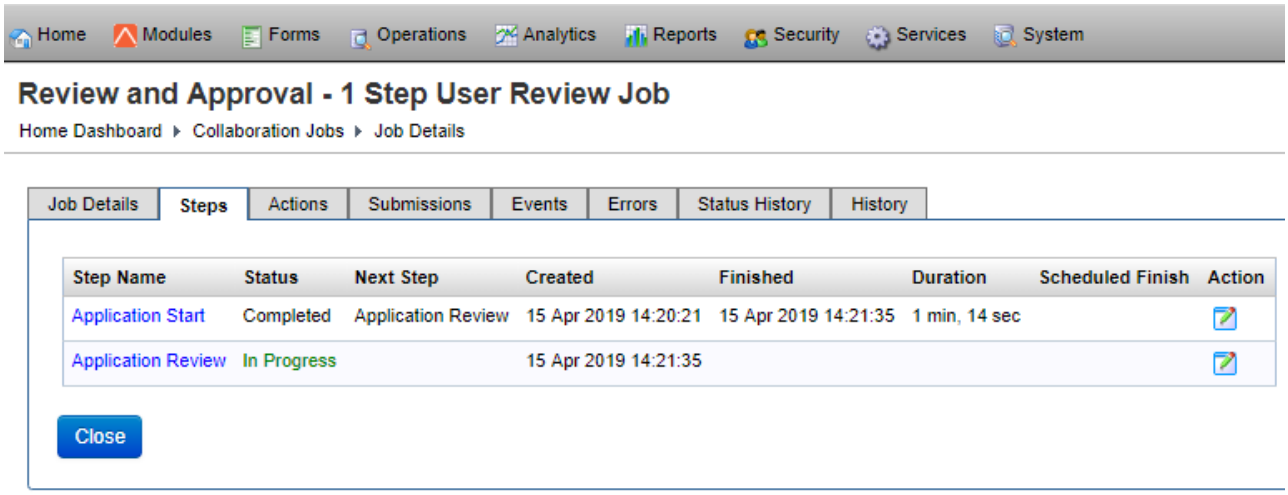
Next, learn how to view collaboration job steps.

View Collaboration Job Steps

Manager allows you to view steps comprising a collaboration job.

To view the steps of a collaboration job:

1. Select Operations > Collaboration Jobs.
2. Locate a collaboration job and click View.
3. Select the Steps tab to see the collaboration job's steps and their information, such as:

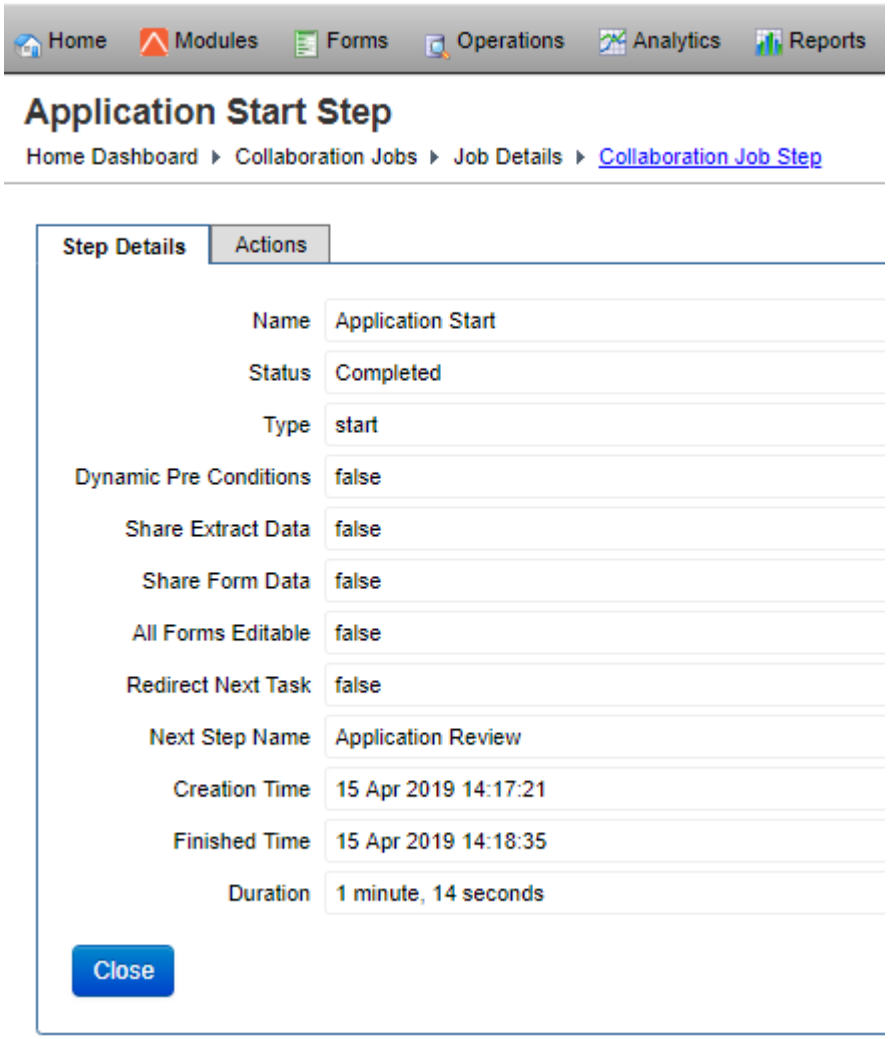


   The list displays the following details:

   - Step Name: a name of a step, which is a part of a job definition of the collaboration job.
   - Step Status: a current status of a step.

     A job step status can be one of the following:

     - In Progress- the step is in progress now.
     - Completed - the step is completed.
     - Expired - the step is expired.
     - Cancelled - the step has been manually cancelled by an administrator.
   - Next Step: a name of a next step as per the job definition.
   - Created: the day and time the step's execution started.
   - Finished: the day and time the step's execution finished.
   - Duration: the time it has taken to complete the step.
   - Scheduled Finish: the day and time the step's expected to complete.
4. Click a step name to view its extra information, such as:



The list displays the following details:

- Type: a type of the step. The type start marks the step as being the first to be executed. Otherwise, the type field is not shown.
- Dynamic Pre Conditions: whether the step has dynamic preconditions or not.
- Share Extract Data:whether the step has a share extract data or not.
- All Forms Editable: whether the step has all forms editable or not.
- Redirect Next Task: whether the step has a redirect next task or not.

5. Select the Actions tab to see actions associated for this step.
6. Click Close.

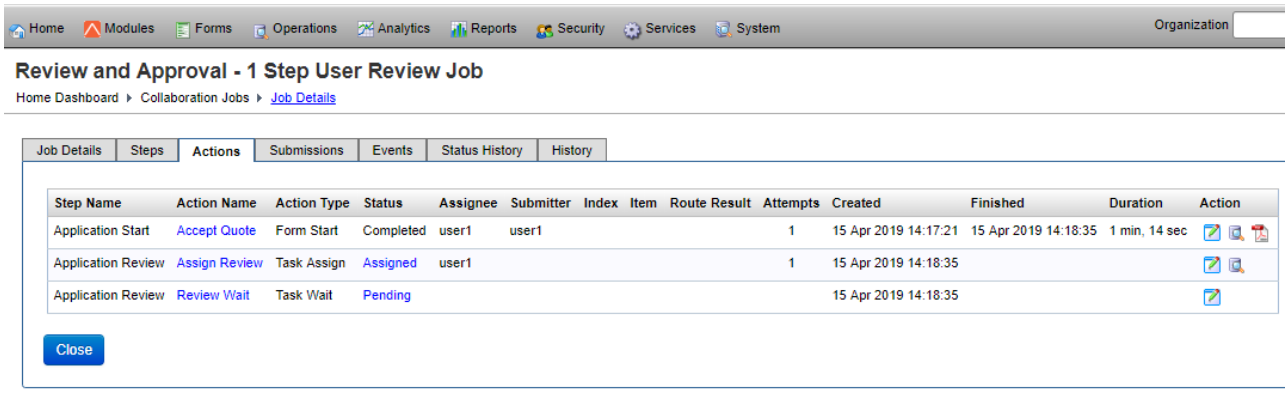
Next, learn how to view collaboration job actions.

View Collaboration Job Actions

Manager allows you to view actions of each step comprising a collaboration job, so you can see at a glance where each step is at and what action service it invokes.

To view the actions of a collaboration job:

1. Select Operations > Collaboration Jobs.
2. Locate a collaboration job and click View.
3. Select the Actions tab to see the collaboration job's actions and their information, such as:



The list displays the following details:

- Step Name: a name of a step, which is a part of a job definition of the collaboration job.
- Action Name: a name of an action, which is a part of the job definition.
- Action Type is a type of the action.
- Status: a status of the collaboration job.

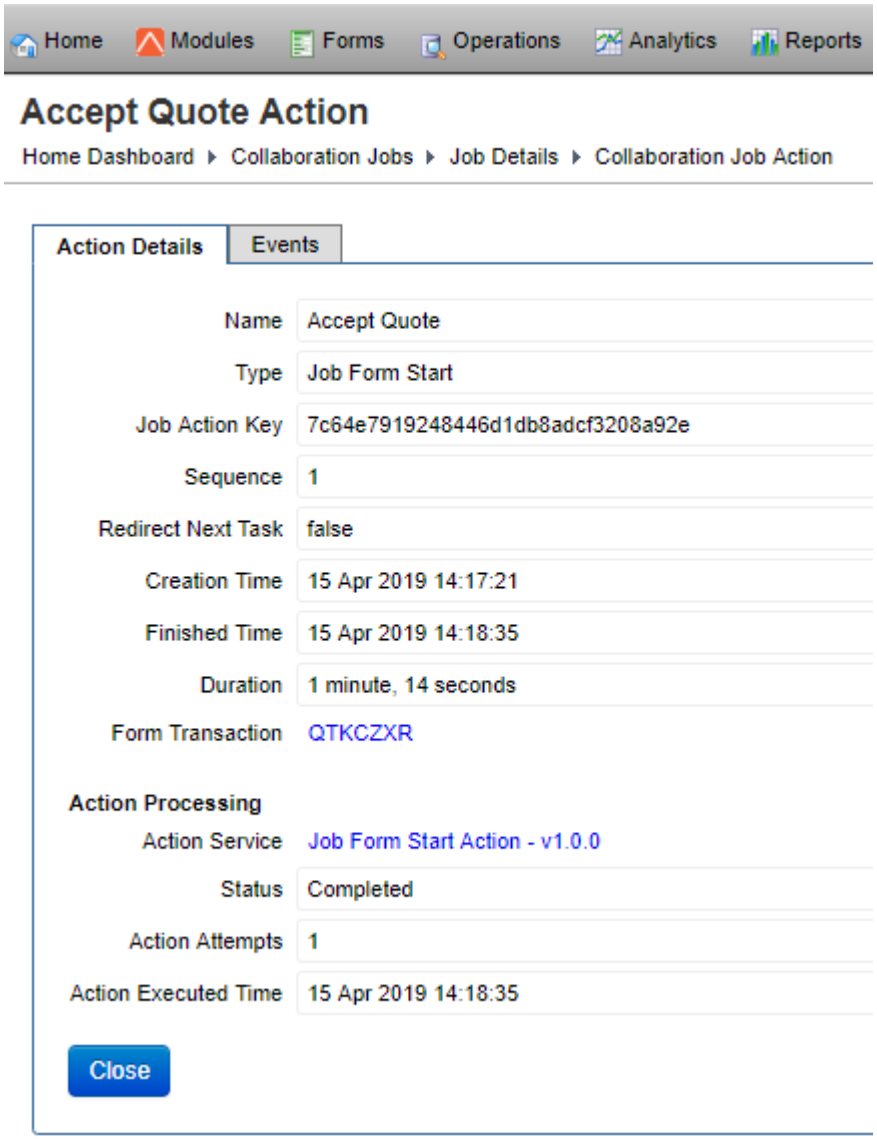
  - In Progress - the action is active and in progress. The action will be retried (looping), if needed.
  - Completed - the action has been completed.

  - Expired - the action is expired when a step is automatically expired by the system. Any actions that are In Progress (Pending, Assigned, Invoked, Ready or In Progress) will be marked as Expired. This include action associated with tasks that are in the Assigned state or job actions that are in the Invoked state awaiting a callback from a 3rd party system.
  - Cancelled - the action is manually canceled by an administrator, who cancelled a job or transaction, or a user, who clicked cancel on a form task.

    - Error - the action has a critical or unrecoverable error and requires an operator's or administrator's intervention before it can be run again.
    - Invoked - the action that is a part of an Asynchronous execution where you have a Job Action and a Job Action Wait pair. The Job Actions processing (invoking a 3rd party service) has completed successfully, so it is awaiting a callback from a 3rd party system and, therefore, it can move to the completed status.
    - Pending - or paused, the action is waiting for an external event to move it forward.
    - Assigned - the Job Task Assign Action process has successfully created and assigned a task to a user, group or email (anonymous). The task is awaiting user submission to be completed which then executes a callback on the collaboration job.
    - Ready - the action will be executed next time the scheduled job runs.

- Assignee: a group, user, or email (an anonymous user) this collaboration job is assigned to.
- Submitter: a user, or email (an anonymous user) who submitted a form.
- Index: an `AssignRepeatIndex`, if the task assign repeat action is used with tasks.
- Item: an `AssignRepeatItem`, if the task assign repeat action is used with tasks.
- Route Result: a `route` as per the collaboration job's job definition.
- Attempts: the number of attempts for the action.
- Created: the day and time the action's execution started.
- Finished: the day and time the action's execution finished.
- Duration: the time it has taken to complete the action.

4. Click an action name to view extra information, such as:



The list displays the following details:

- Type: a type of the action.
- Job Action Key: a unique action code
- Sequence: a sequence of the action as per the job definition
- Redirect Next Task: whether the action has a redirect next task or not.
- Form Transaction: the tracking code of the transaction associated with this action, which only appears for Job Task Assign or Job Form Start actions.
- Action Service: the name and the version of a action service, which is invoked during this action.
- Action Executed Time: the time it has taken to complete the action.

5. Click the Events tab to view all the events related to this action.
6. Click Edit to check or update a service, which is invoked during this job step.
7. Click View Transaction Details to check the details of the transaction associated with this action.
8. Click View PDF Receipt to view a form as PDF receipt.
9. Click Close.

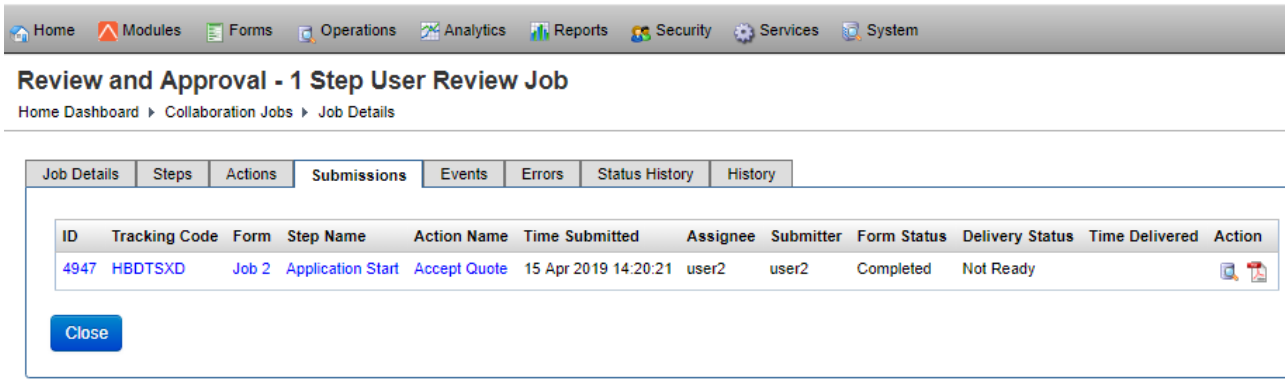
Next, learn how to view collaboration job submissions.

View Collaboration Job Submissions

Manager allows you to view transactions created for each action executed within a collaboration job.

To view the transactions within a collaboration job:

1. Select Operations > Collaboration Jobs.
2. Locate a collaboration job and click View.
3. Select the Submissions tab to see the collaboration job's transactions and their basic information such as:



The list displays the following details:

- ID: a transaction ID.
- Tracking Code: a tracking code.
- Form: a form name.
- Step Name: a step name.
- Action Name: an action name.
- Time Submitted:
- Assignee:
- Submitter:
- Form Status:
  - Abandoned: a transaction that has been abandoned by the user, or based on the data retention policy
  - Assigned: a task which is waiting to be completed by a user
  - Completed: a transaction that has been successfully submitted and contain all required data
  - Opened: a transaction that has been started by a user. When a transaction is created by a user (by starting a new form), the initial form status is set to

Opened. This status will be changed once the user saves, cancels, or submits the transaction.
    - Saved: a transaction that has been saved by the user or automatically by the system
    - Submitted: a transaction that has been submitted but may require additional information, such as attachments, payment, and so on. Once the required data is collected, the transaction status will change to Completed. This status is not common for Maestro forms as all information, such as attachments and payments, are collected prior to the user submitting the form. Most Maestro forms will skip the Submitted status and will be marked as Completed after the form is submitted by the user. This is a legacy feature.
    - Expired: a form associated with this transaction has expired and is no longer available
  - Delivery Status:
  - Time Delivered:
4. Click ID, Tracking Code or View Transaction Details to check the details of the transaction.
5. Click View PDF Receipt to view a form as PDF receipt.

Next, learn how to view collaboration jobs events.

View Collaboration Job Events

Manager allows you to view events related to each collaboration job. Events are added when the job actions are executed or processed, including:

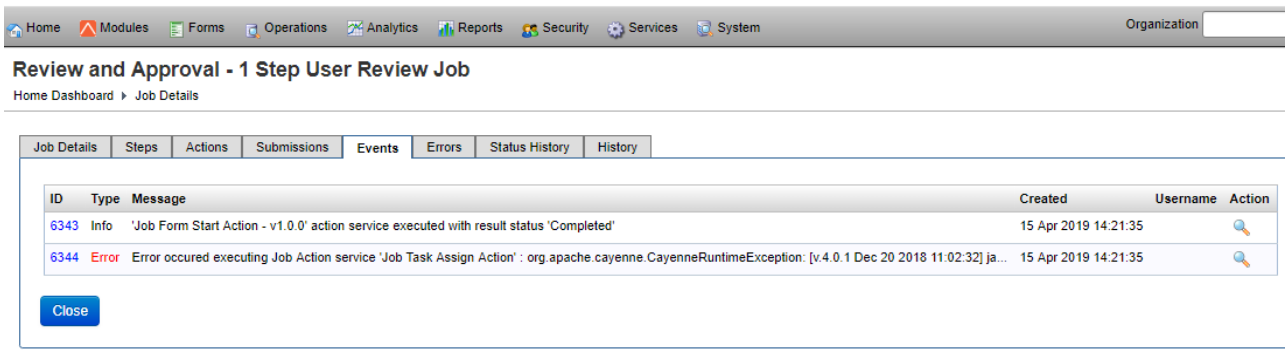
- Info events are generated when the action proceeds normally. The action status is Completed, Invoked, Assigned, Pending, and so on.
- Error events are generated where and exception is thrown by the action processing or the result of the action processing is determined to be an error. Errors will also appear in the Errors tab.
- Custom Groovy Job Actions can write either an error or info event to the JobEventLog Server.

Various services, invoked during a collaboration job execution, can write to the event log using the following Groovy code:

```
// logEvent closure
def logEvent = { msg ->
        if ( isLogEvents ) {
                // Logs to Job Action Event Tab
                jobEventLogService.logInfoEventWithAction((String) msg, jobAction)
        }
}
```

To view the events related to a collaboration job:

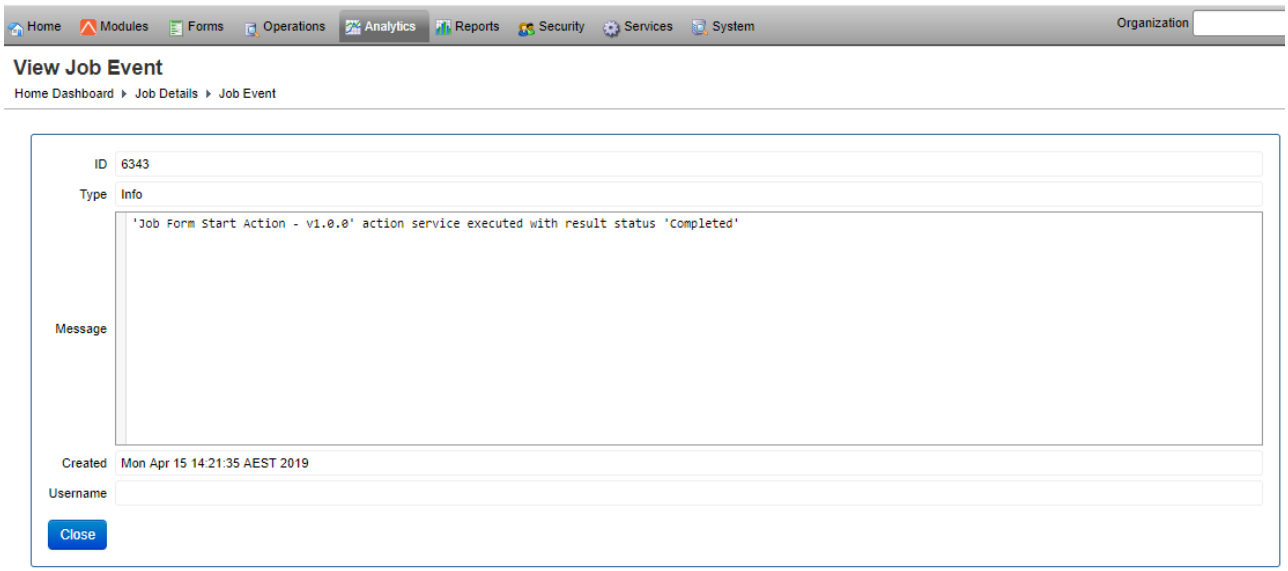
1. Select Operations > Collaboration Jobs.
2. Locate a collaboration job and click View.
3. Select the Events tab to see the collaboration job's events and their basic information such as:



The list displays the following details:

- ID: the event ID. Click ID to see the job event details.
- Type: the type of the event, which can be info or error.
- Message: a meaningful description of the event.
- Created: a times tamp of the event.
- Username: a user of the event.

4. Click View to see the job event details.



5. Click Close.

NOTE

Manager stores collaboration job events in a separate table, so you can't find them in the event log.

NOTE

You can also see errors that occurred during a collaboration job execution. Generally, errors are of greater severity than events.

Next, learn how to view collaboration job errors.

View Collaboration Job Errors

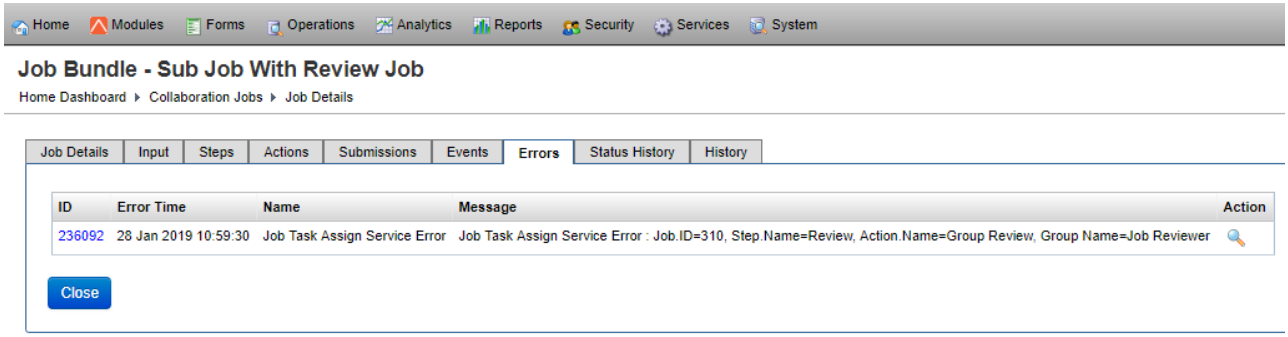
Manager allows you to view errors that occurred during execution of each collaboration job.

> NOTE
>
> If a collaboration job doesn't have errors, this tab is not shown.

To view the errors related to a collaboration job:

1. Select Operations > Collaboration Jobs.
2. Locate a collaboration job and click View.
3. Select the Errors tab to see the collaboration job's errors.

   > NOTE
   >
   > This tab is not visible if the collaboration job doesn't have any errors.

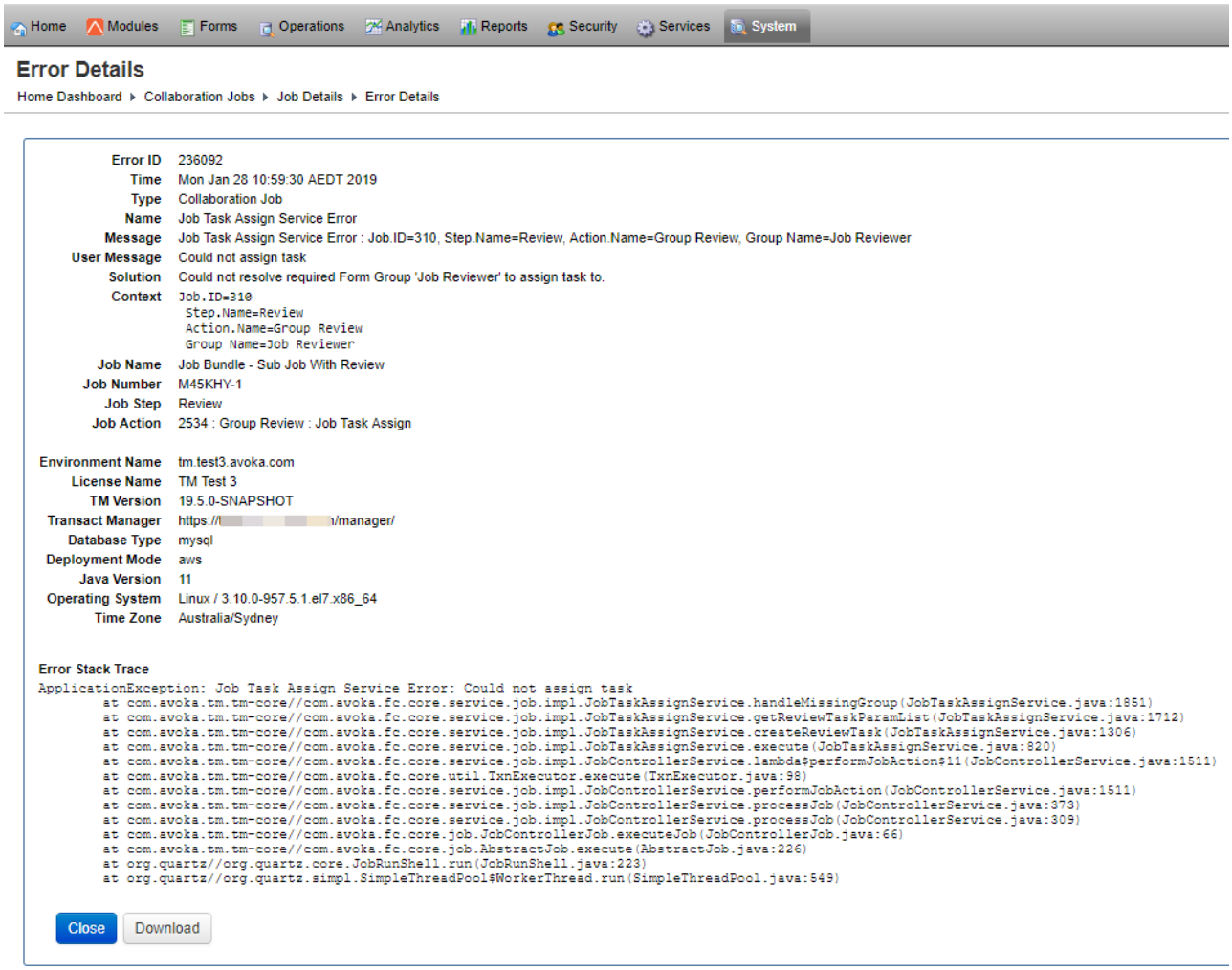


   The list displays the following details:

   - ID: the error ID. Click ID to see the job error details.
   - Name: a meaningful description of the error.
   - Message: a detailed error message.

4. Click View to see the error details including error stack trace, which is especially useful in

problem troubleshooting.



NOTE

Manager stores collaboration job errors along with all other errors, so you can find them in the error log.

NOTE

You can also see events that occurred during a collaboration job execution. Generally, errors are of greater severity than events.

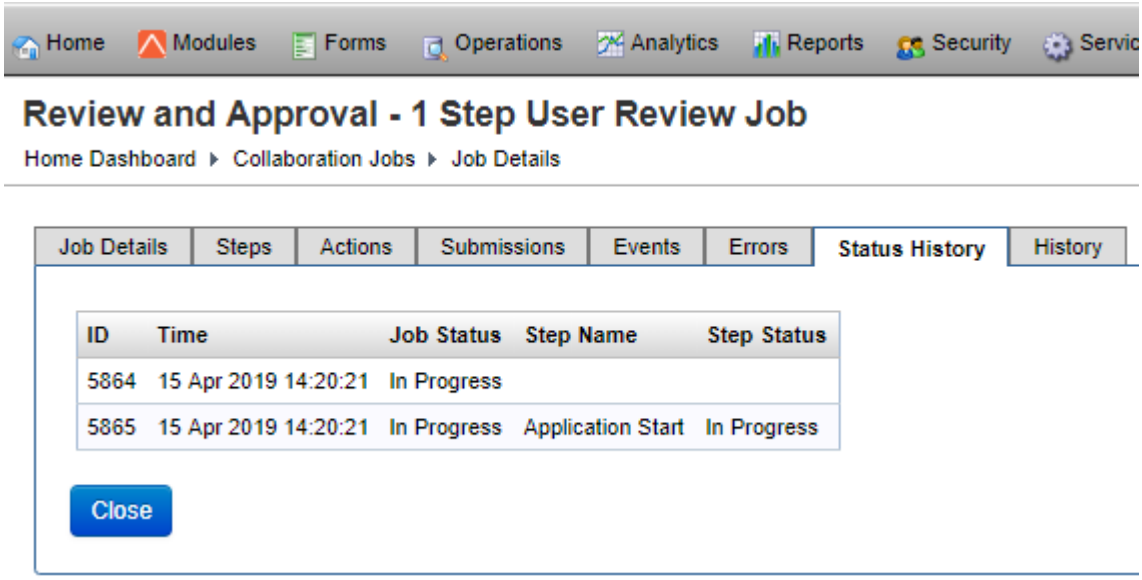
Next, learn how to view collaboration job status history.

View Collaboration Job Status History

Manager allows you to view status history of each collaboration job.

To view the status history of a collaboration job:

1. Select Operations > Collaboration Jobs.
2. Locate a collaboration job and click View.
3. Select the Status History tab to see the collaboration job's status history and its information, such as:



The list displays the following details:

- ID: a unique status ID.
- Time: the day and time the job or step status was changed.
- Job Status: a status of a collaboration job.
    - In Progress - the collaboration job is active and in progress.
    - Completed - the collaboration job has been completed. This happens when a endpoint step has completed.
    - Expired - this happens when an endpoint expiry collaboration job has been automatically expired by Journey Manager.
    - Cancelled - the collaboration job has been manually canceled by an administrator.
    - Error - the collaboration job has a critical or unrecoverable error that requires an administrator intervention before it can be run again.
    - Paused - the collaboration job has been paused temporarily.

- Step Name: a name of a step, which is a part of a job definition of the collaboration job.
- Step Status: a current status of a step.

  A job step status can be one of the following:

  - In Progress- the step is in progress now.
  - Completed - the step is completed.
  - Expired - the step is expired.
  - Cancelled - the step has been manually cancelled by an administrator.

- Click Close.

Next, learn how to view collaboration job history.

View Collaboration Job History

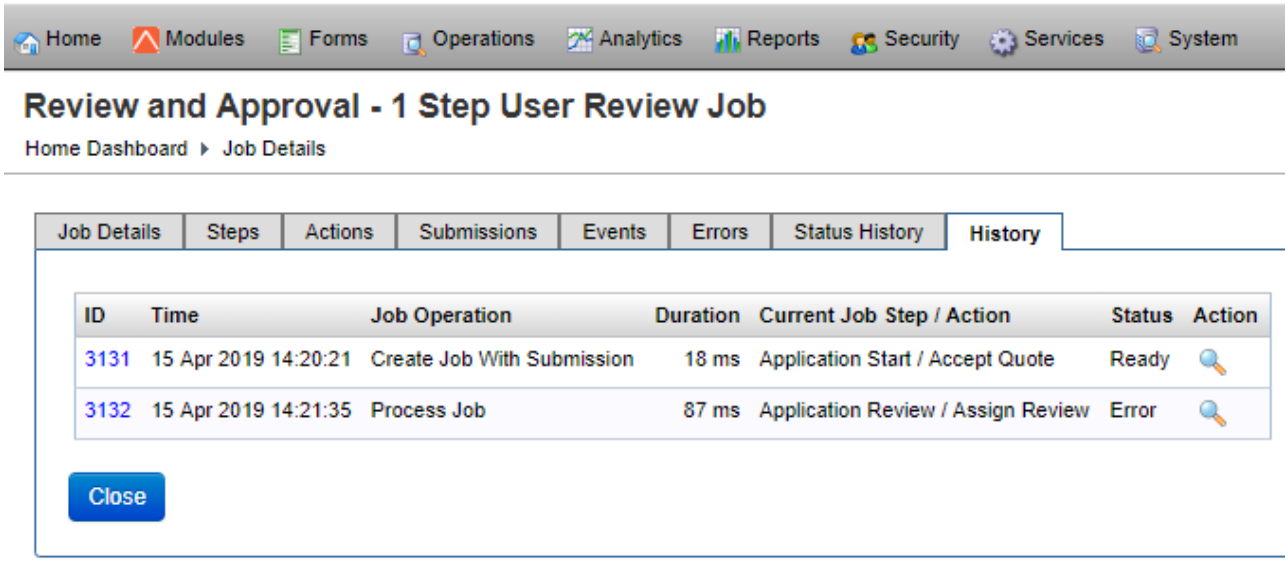
Manager allows you to view history of each collaboration job.

A collaboration job is a state machine that can be represented with a graph consisting of steps and actions. A collaboration job has life cycle events, such as action processing and submission's callbacks, that take the collaboration job from one state to another. The Action Processing may execute one or more actions in one Job Processing cycle. Similarly, a callback, such as a task submission, changes the collaboration job's action status for the task assign from Assigned to Completed and the Task Wait status to Ready.

> NOTE
>
> A collaboration job that is to be processed immediately true will have less history state changes because the callback state changes and action processing are combined.

To view the history of a collaboration job:

1. Select Operations > Collaboration Jobs.
2. Locate a collaboration job and click View.
3. Select the History tab to see the collaboration job's history:

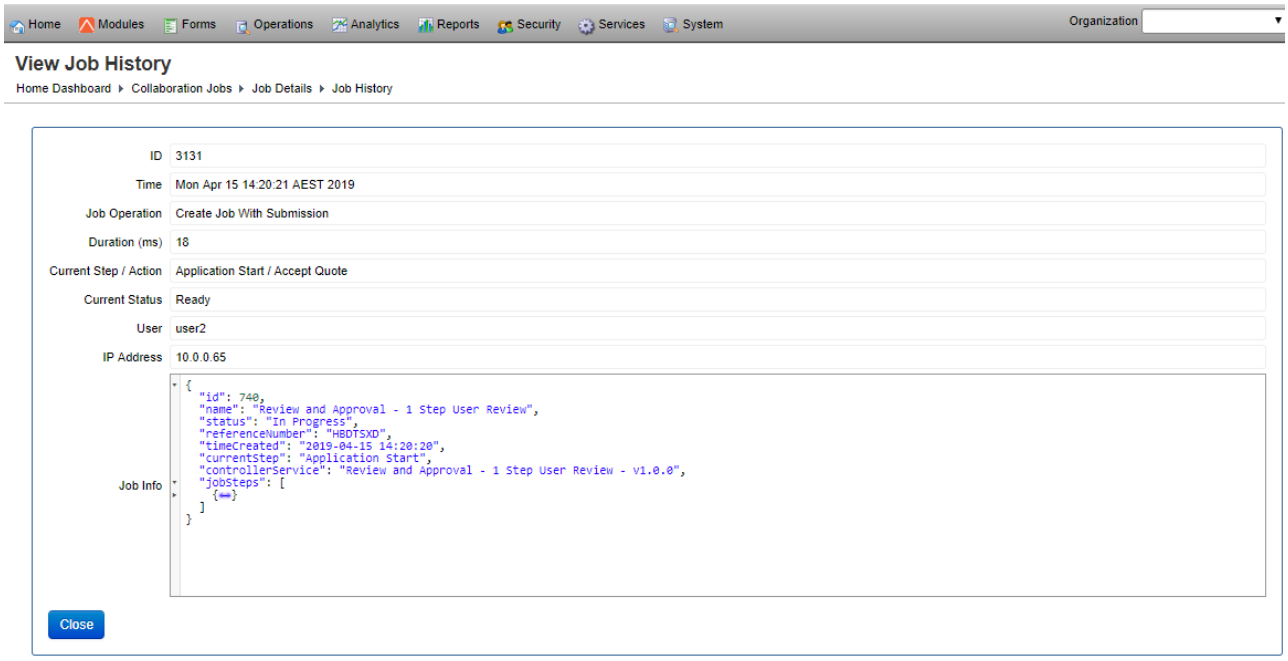


The list displays the following details:

- ID: a unique job operation ID which can be a step or an action.
- Time: the day and time the job operation was invoked.

- Job Operation: the description of the step or the action executed as a part of the collaboration job.
- Duration: the time it has taken to complete the job operation.
- Current Job Step / Action: the name of the step and the action executed as a part of the collaboration job.
- Status: the status of the job operation that can be the status of the step or the action.

4. Click View to see the detailed information about a selected job operation.



The Job Info shows the collaboration job graph as an object in JSON[1] format.For more information, see %=TransactVariables.JobDefinitionLowCase%].

```
{
        "id": 740,
        "name": "Review and Approval - 1 Step User Review",
        "status": "In Progress",
        "referenceNumber": "HBDTSXD",
        "timeCreated": "2019-04-15 14:20:20",
        "currentStep": "Application Start",
        "controllerService": "Review and Approval - 1 Step User Review - v1.0.0",
```

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

```
        "jobSteps": [
                {
                        "id": 2584,
                        "name": "Application Start",
                        "status": "In Progress",
                        "type": "start",
                        "timeCreated": "2019-04-15 14:20:20",
                        "currentAction": "Accept Quote",
                        "jobActions": [
                                {
                                        "id": 4257,
                                        "name": "Accept Quote",
                                        "status": "Ready",
                                        "type": "Job Form Start",
                                        "actionService": "Job Form Start Action - v1.0.0",
                                        "timeCreated": "2019-04-15 14:20:20",
                                        "submission": {
                                                "id": 4947,
                                                "trackingCode": "HBDTSXD",
                                                "formStatus": "Completed",
                                                "receiptStatus": "Ready",
                                                "deliveryStatus": "Not Ready",
                                                "versionNumber": "18.5-Branch",
                                                "formSpace": "Work Spaces",
                                                "timeRequest": "2019-04-15 14:20:18",
                                                "timeFormCompleted": "2019-04-15 14:20:20",
                                                "contactEmailAddress": "test@test.com",
                                                "userLoginName": "user2"
                                        }
                                }
                        ]
                }
        ]
}
```


Next, learn how to [view collaboration jobs](#).

Collaboration Job Examples

Manager comes with several collaboration job examples that you can use to create your new workflows. Every example contains forms, form spaces and services configurations available for you to download and run in Manager.

You can start with the basic examples, including:

- One step review collaboration jobs
- Multi step group review collaboration jobs
- Customer onboarding collaboration jobs - form bundle
- Maestro customer onboarding collaboration jobs - form bundle

Then, you can check the advanced examples:

- Dynamic customer onboarding collaboration jobs
- Step expiry
- Anonymous collaboration jobs
- Asynchronous step execution
- Task assign repeat action

Before you can use these examples, you must create users with specific roles as explained below.

To create the users:

1. Log on to Manager as an administrator.

2. Copy the Groovy[1] script into the Groovy console.

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

```
/**
 * Adds Users For the Collaboration Jobs Tests
 * Copy and Paste into the Groovy Console
 * Developed for Transaction Manager 4.3.0
 */

import com.avoka.fc.core.dao.ClientDao
import com.avoka.fc.core.dao.DaoFactory
import com.avoka.fc.core.dao.GroupDao
import com.avoka.fc.core.dao.PortalDao
import com.avoka.fc.core.dao.RoleDao
import com.avoka.fc.core.dao.UserAccountDao
import com.avoka.fc.core.entity.Client
import com.avoka.fc.core.entity.ClientUser
import com.avoka.fc.core.entity.Group
import com.avoka.fc.core.entity.Portal
import com.avoka.fc.core.entity.PropertyType
import com.avoka.fc.core.entity.Role
import com.avoka.fc.core.entity.UserAccount
import com.avoka.fc.core.entity.UserRole
import com.avoka.fc.core.security.ISecurityManagerService
import com.avoka.fc.core.service.ServiceFactory
import com.avoka.fc.core.service.UserService

final EMAIL_TESTTEAM = "testemailgroup@yourcompany.com"
final PASSWORD_COMMON = "replace with password"
final JOB_APPLICANT = "Job Applicants"
final JOB_REVIEWER = "Job Reviewers"
final JOB_MANAGER = "Job Managers"
final MAGUIRE = "Maguire"

PortalDao portalDao = DaoFactory.getPortalDao()
Portal portal = portalDao.getPortalByName(MAGUIRE)

UserAccountDao userAccountDao = DaoFactory.getUserAccountDao()

ISecurityManagerService securityManagerService = Ser-
viceFactory.getSecurityManagerService(portal)
GroupDao groupDao = DaoFactory.getGroupDao()
UserService userService = ServiceFactory.getUserService(portal)
```

```
def userList = [
        ["firstName": "Joe", "lastName": "Public"],
        ["firstName": "Anne", "lastName": "Applicant", "group": JOB_APPLICANT],
        ["firstName": "Roger", "lastName": "Reviewer", "group": JOB_REVIEWER],
        ["firstName": "Mike", "lastName": "Applicant", "group": JOB_MANAGER],
        ["firstName": "Cat", "lastName": "Coordinator", "group": JOB_MANAGER]
]


for (Map userMap : userList) {
    UserAccount userAccount = userAccountDao.getUserAccountForLogin((String) user-
Map.firstName)

    if ( userAccount==null) {

        Map<String, String> profileMap = new HashMap<String, String>();
        profileMap.put((String) PropertyType.USER_Property_Given_Name, (String)
userMap.firstName);
        profileMap.put((String) PropertyType.USER_Property_Family_Name, (String)
userMap.lastName);
        profileMap.put((String) PropertyType.USER_Property_Email, (String) EMAIL_
TESTTEAM);


        userAccount = securityManagerService.createUserAndProfile((String) user-
Map.firstName,
                (String) EMAIL_TESTTEAM,
                (String) PASSWORD_COMMON,
                (String) userMap.firstName,
                (String) userMap.lastName,
                "",
                profileMap,
                "",
                UserAccount.USER_TYPE_LOCAL,
                false)

        if (userMap.firstName == "Cat") {
            ClientDao clientDao = DaoFactory.getClientDao()
            Client client = clientDao.getClientByName(MAGUIRE)
```

```
            RoleDao roleDao = DaoFactory.getRoleDao()
            Role role = roleDao.getRoleForName("Maguire Staff")

            ClientUser clientUser = new ClientUser()
            clientUser.setClient(client)
            clientUser.setUser(userAccount)

            UserRole userRole = new UserRole()
            userRole.setUser(userAccount)
            userRole.setRole(role)
        }

        Group group = groupDao.getGroupForName((String) userMap.group)
        if (group != null) {
            List<String> groupIdList = new ArrayList<String>()
            groupIdList.add(""+group.id)
            userService.updateGroups(userAccount.id, groupIdList)
        }     else {
            println "Cannot link group User: " + userMap.firstName + " Group: " +
userMap.group
        }

        userService.commitChanges()
        println "Create User: " + userMap.firstName

    } else {
        println "User: " + userMap.firstName + " already exists"
    }
}
```

3. Update the script's two lines below with your test email address and password.

```
final EMAIL_TESTTEAM = "testemailgroup@yourcompany.com"
final PASSWORD_COMMON = "replace with password"
```

4. Click Execute Script to create the users.

Next, learn about one step review collaboration jobs.

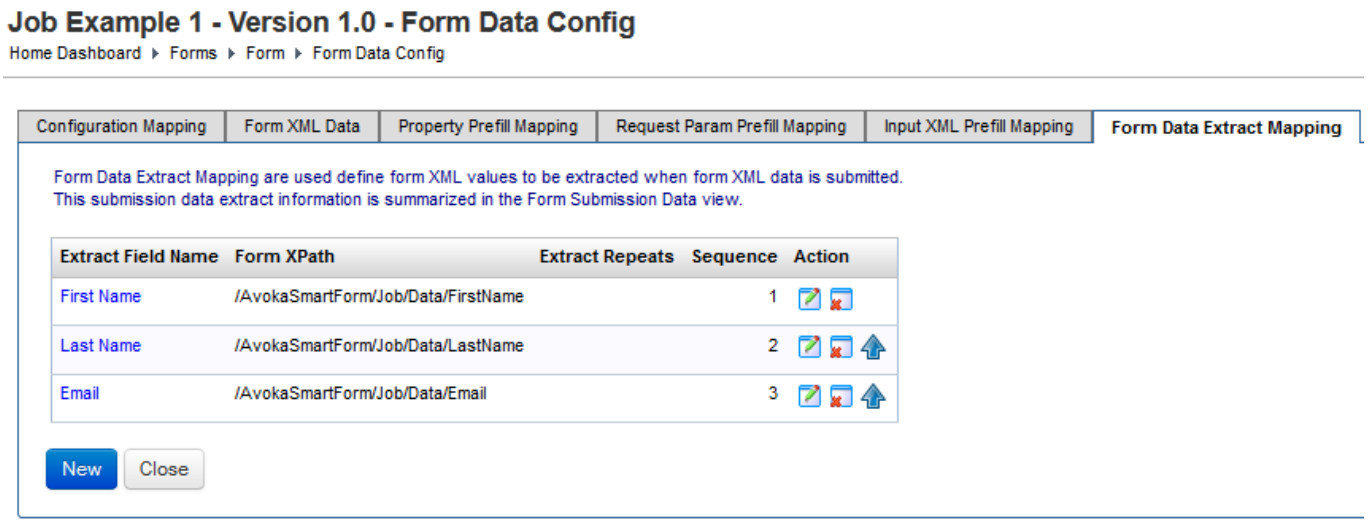
Configure Forms for Use with Collaboration Jobs

Before you can use your forms with collaboration jobs, you have to configure them in Journey Manager and Journey Maestro. The collaboration jobs use data extracts to reference data used in submitted forms. You can configure data extracts during form design time, which is the recommended approach, or you can map them manually in Manager.

> NOTE
>
> Data extracts that are configured during form design will overwrite data extracts with the same name in Manager, which happens every time the form is deployed into Manager, so the best practice is to configure them in the form.

Let's explore the data extracts configuration implemented in the Maestro form Job Example 1, shown below.

Next, you can chec whether data extracts from this form have been automatically populated and mapped in Manager. To do so, select Form Dashboard > Data Config > Form Data Extract Mapping as shown below.



The job definitions of form data extracts are shown below, including field names, how they are referenced in the action properties, and usage examples.

Extract Name Field	Extract Reference	Example
First Name	`$formDataMap.firstName`	`{ "name": "Task Subject", "value": "Job Number: <b>$-job.referenceNumber</b> $submission.form.formName From $formDataMap.firstName $formDataMap.lastName" }`
Last Name	`$formDataMap.lastName`	`{ "name": "Task Subject", "value": "Job Number: <b>$-job.referenceNumber</b> $submission.form.formName From $formDataMap.firstName $formDataMap.lastName" }`
Email	`$formDataMap.email`	`{ "name": "Task Subject", "value": "Job Number: <b>$-job.referenceNumber</b> From $formDataMap.email" }`

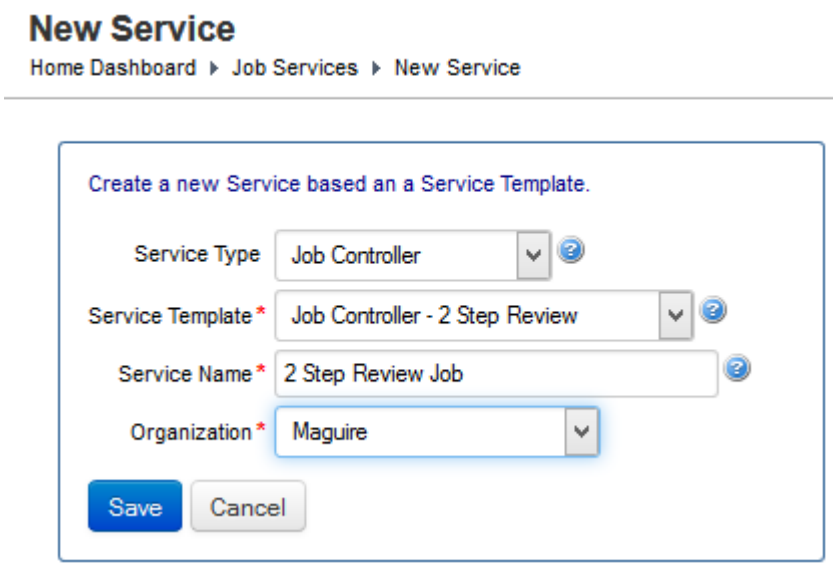
Click on each field to see the data extract mapping information. For example, the Emain Address has the /AvokaSmartForm/GettingStarted/AboutYou/contact/EmailAddress XPath. Manager passes job context information to the form via the `SFMData.SystemProfile.Job` XML node when the form is rendered.

After you have imported the form, you can configure the following in Manager:

- Form properties to reference them in the job definition for form specific settings.
- Form groups to control access to the form.
- Form delivery methods to enable the job controller to complete collaboration jobs.

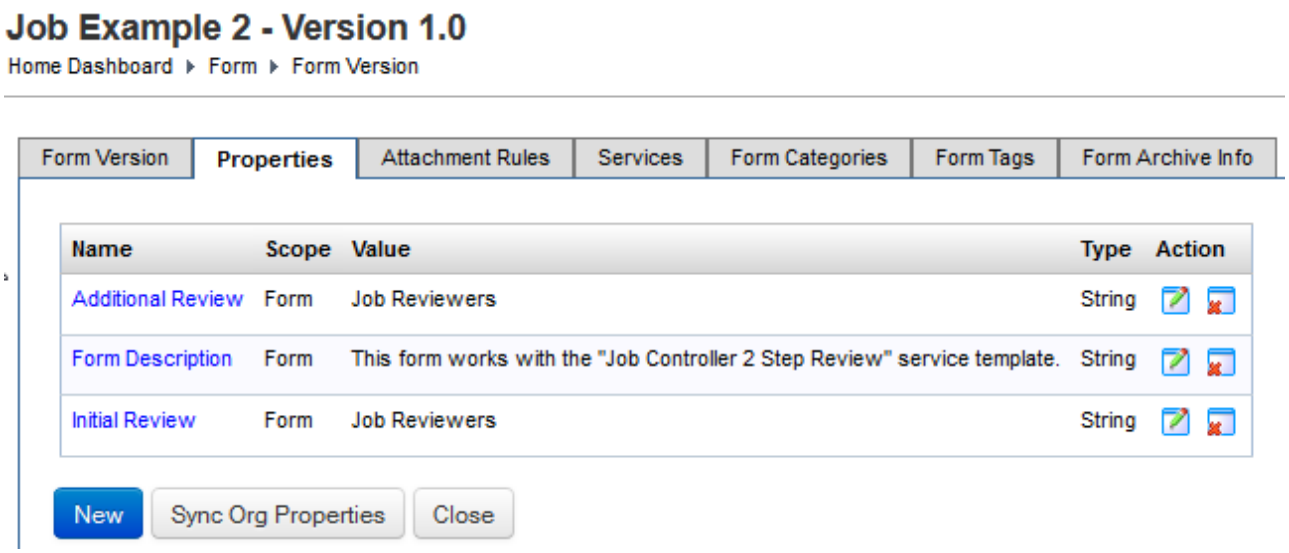
To configure a form:

1. Create a new job controller service instance from available service templates, for example, the Job Controller - 2 Step Review.

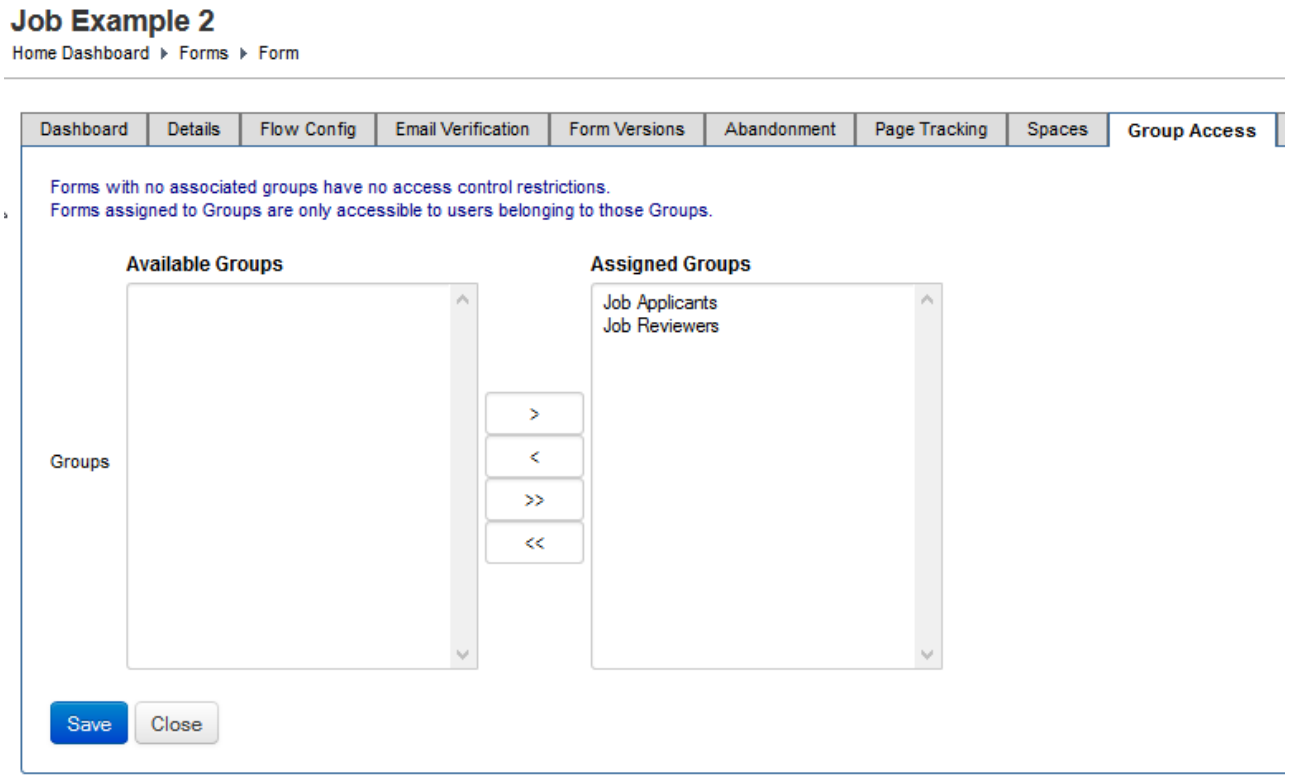


2. Import a form into Manager.
3. Open a form and click Services.
4. Select a form prefill data service from the Form Prefill Data Service dropdown list, for example, 2 Step Review Job.
5. Select the Property tab to link the reviewer for Initial Review and Additional Review Steps to be assigned to a form group.
6. Set the Additional Review and Initial Review property values.

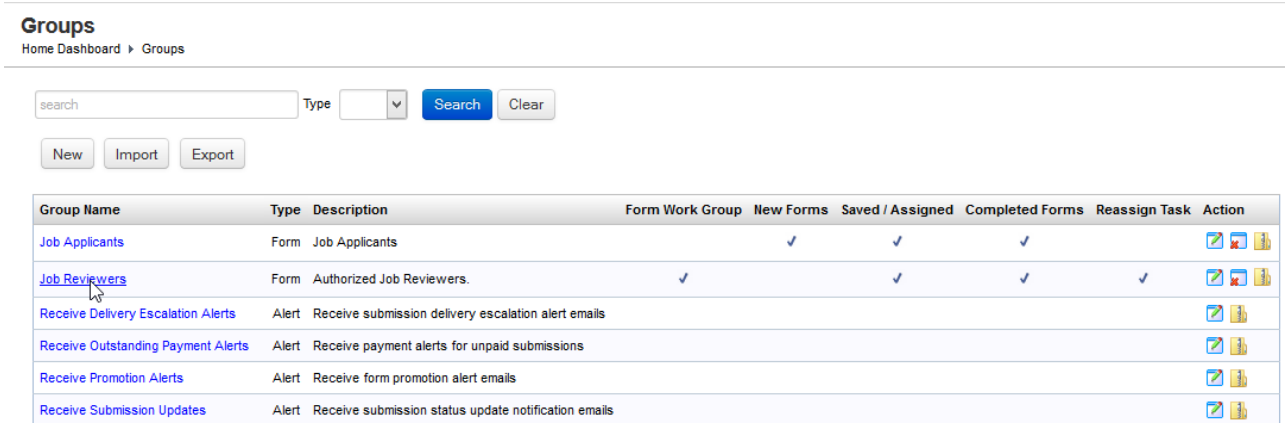
> NOTE
>
> By default, both will be mapped to the same Job Reviewers form group.



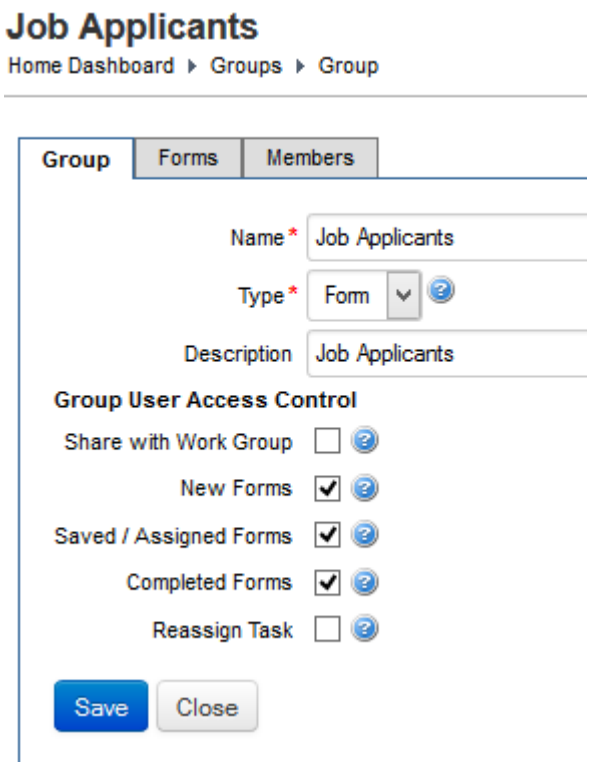
7. Open a form dashboard and select the Group Access tab to assign the form groups. Form groups are used to control access the following access to a form. By default, if a form group is not assigned to a form, it will be available to any users belonging to the organization.



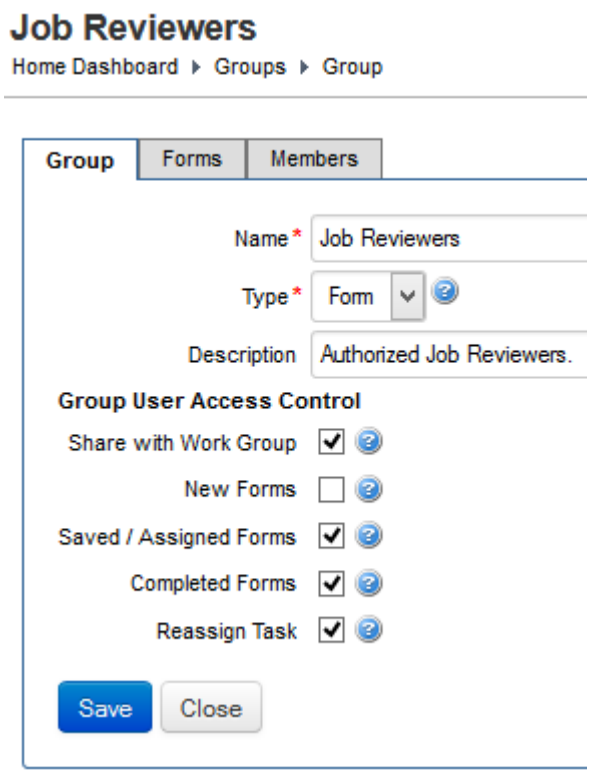
8. Assign a user that you log in with to the Job Reviewers form group to review collaboration jobs at the Initial and Additional Review steps.



9. Select the Job Applicants group to restrict users who can start the form, because the Job Reviewers can't initiate the form, but they can participate in reviewing the submitted forms.
10. Select the New Forms checkbox to allow members of this group to start new forms. Form more information, see configure a group.



11. Assign your user in the Members tab.
12. Click the Job Reviewers group and select the Share with Work Group check box to allow members of this group to participate in tasks, but not start new forms.



13. Select the Members tab and assign your user to this group.
14. Assign a job group to the applicable job coordinators' user accounts.
15. Configure a delivery channel, with a default delivery method, for your organization or your form to allow the collaboration job to run and deliver form submissions.
16. Now, your form is configured for collaboration jobs.

A form configuration can be quite extensive and it varies depending on a tool it was built with, as shown below:

- Configure Maestro forms for use with collaboration jobs
- Configure Composer forms for use with Collaboration Jobs

Next, learn how to configure a form.

Configure Maestro Forms for Use with Collaboration Jobs

Maestro forms require configuration before you can use them with collaboration jobs, including:

- How to configure data extracts.
- How to use the collaboration job components that are included in Maestro by default.
- How to set form rules, such as visibility and editability rules, based on collaboration job's steps.
- How to preview your form in Maestro and simulate the collaboration job behavior.

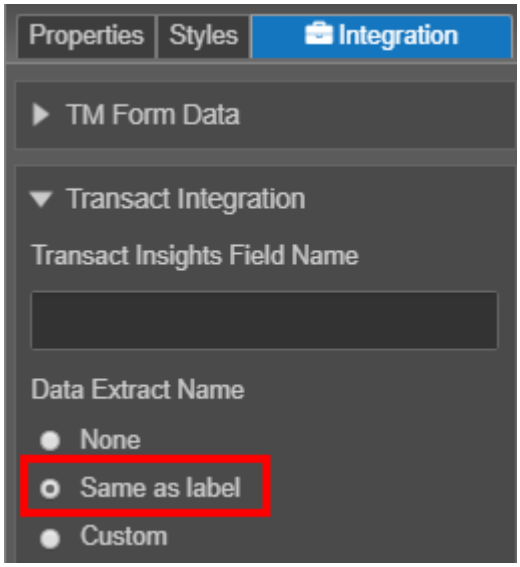
In this example, we use the Maestro form configured with the Review and Approval – Multi Step Group Review job controller service in Manager.

Data Extracts

Some form components are required to be saved as data extracts so they can be used with collaboration jobs. For example, the forms used in this example where an applicant enters a loan amount. If the reviewer approves the loan, the Loan Amount on the Request tab must be checked to validate if it exceeds the threshold of $20,000. In this case, the loan must be also approved by a manager. To achieve this, we will pass the entered loan amount from the form to the job controller service in Manager using the form of data extracts.

To include a component into the data extracts:

1. Select a component to be extracted from the Structure panel or directly from the form editor.
2. Click the Integration panel.
3. Expand the Transact Integration section.
4. Select the Same as label radio button.



5. Alternatively, if you want the data extract's name to be different from the form component's name, select the Custom radio button and enter a new name in the text field. This depends on the name of the variable used within Manager such as in the job controller service. For more information, see submission data extract.

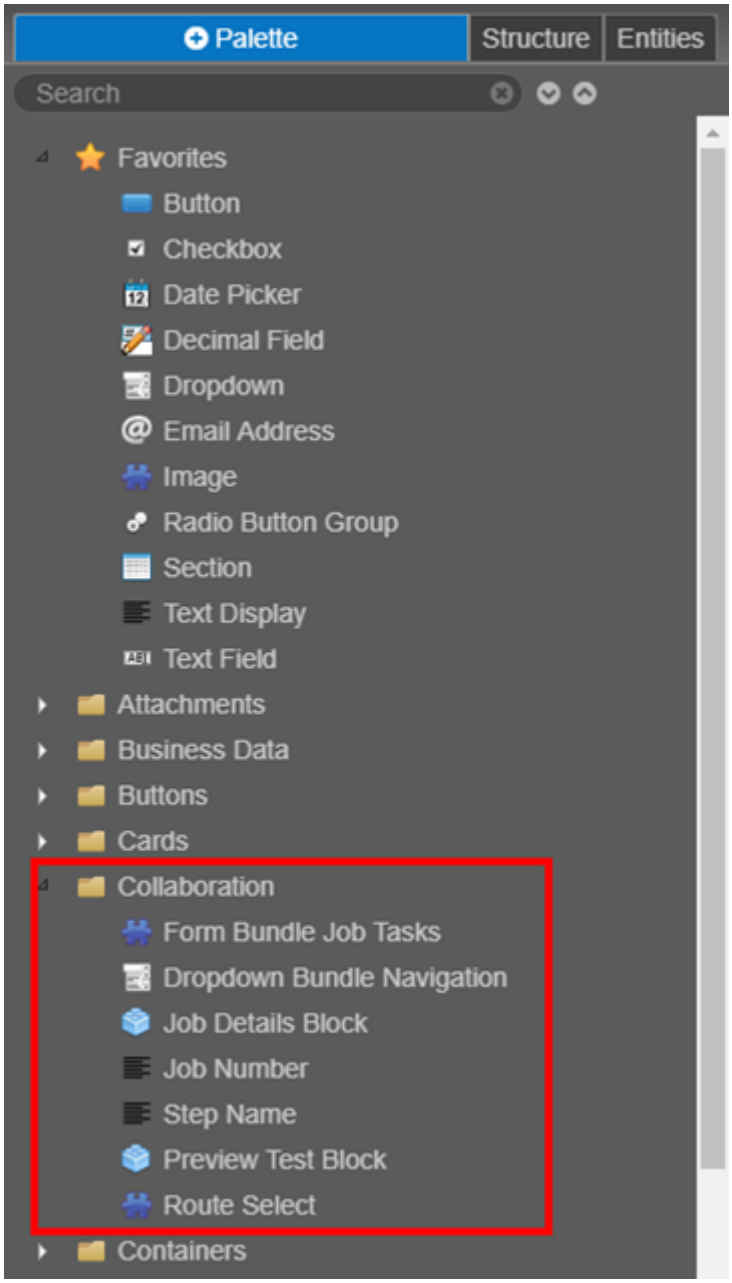
Collaboration Job Components

Collaboration job components have been created to assist the form builder in combining the collaboration jobs features into their forms.

To use a collaboration job component in Maestro:

1. Go to the Palette panel.
2. Expand the Collaboration folder.

3. Drag and drop a components you need on to your form.



The collaboration job components can be one of the following:

Form Bundle Job Tasks

This component is used for form bundles as a way of navigating and keeping track of forms that are still to be completed. It appears as a set of links for navigating to the applications for products previously selected. It has extra features such as allowing a user to click on only the applications that have not been completed and allowing to download a copy of any completed application.

Dropdown Bundle Navigation

It provides a similar feature as the form bundle job tasks, but it is presented as a simple drop-down.

Job Details Block

This component is for displaying collaboration job details on the form.

To display the collaboration job details:

1. Expand User Experience.
2. Click Form Options.
3. Selecting the Include Collaboration Job Details.
4. Use the Job Details Block to customize the location of the collaboration job details within you form or template.

Job Number

This component displays the collaboration job number.

Step Number

This component displays the step's name the collaboration job is currently at.

Preview Test Block

This provides a section in your form that can be used to test the form without having to render the form in Manager. The block is used for testing purposes and can be viewed from the preview mode in the Maestro editor. This allows the form builder to preview the form and make sure that the rules are set up correctly without having to leave Maestro. That is, without the form being first deployed to Manager and associated with a job controller. It is important to note that the Preview Test Block will never display in a built / rendered form as it only appears when using the Maestro Preview mode.

The following components may be entered:

- Job Reference will be shown in the relevant job details or job number fields. That is, it will appear in the Job Number and Job Details components where they appear on your form, or in the job details in the heading if you have ticked the Include Collaboration Job Details in the Form Options.
- Step Name is quite useful for testing the rules set up for presenting the correct form depending on which step name the form is up to. As this is a dropdown you must first configure the

options to mimic the step name values of the job controller, which is also used in the visibility and editability rules configured for this form. To do this:

1. Select the Properties panel.
2. Expand the Dropdown section.
3. Click Options.
4. Enter each of the correct step name values for each row under the Label column. Make sure that the first one is left blank which is used for testing when the applicant first submits the application.
5. Click Save.

- Available Routes is used for the Route Select dropdown component. It will plug in the correct values in the Route Select dropdown, which is the same as when passed from the job controller. The values must be entered as a pipe delimited string. You will also need to select the suitable step name from the Step Name checkbox above.

Route Select

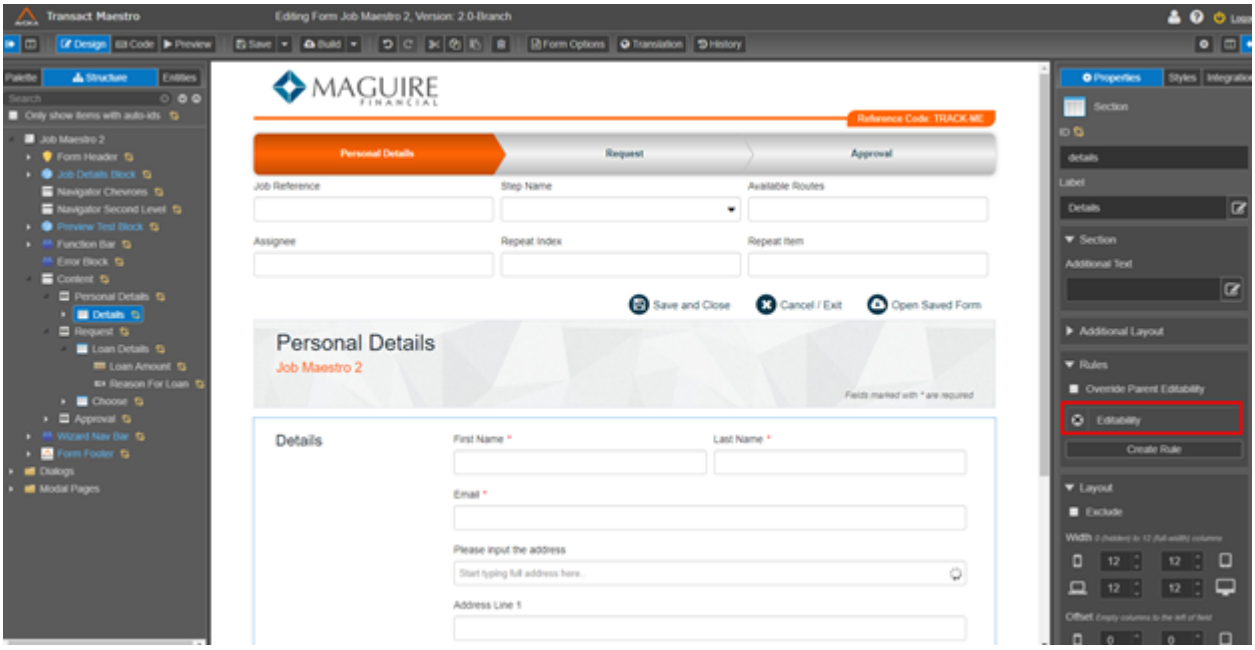
This component is a dropdown that will present to a reviewer or manager reviewing the application a list of options. A review or manager will choose an option to indicate their decision. This component will dynamically plug-in the values for the options from the job controller associated with the form. This may be useful but there has been a preference towards using radio button group component instead. For more information, see below.

Form Rules

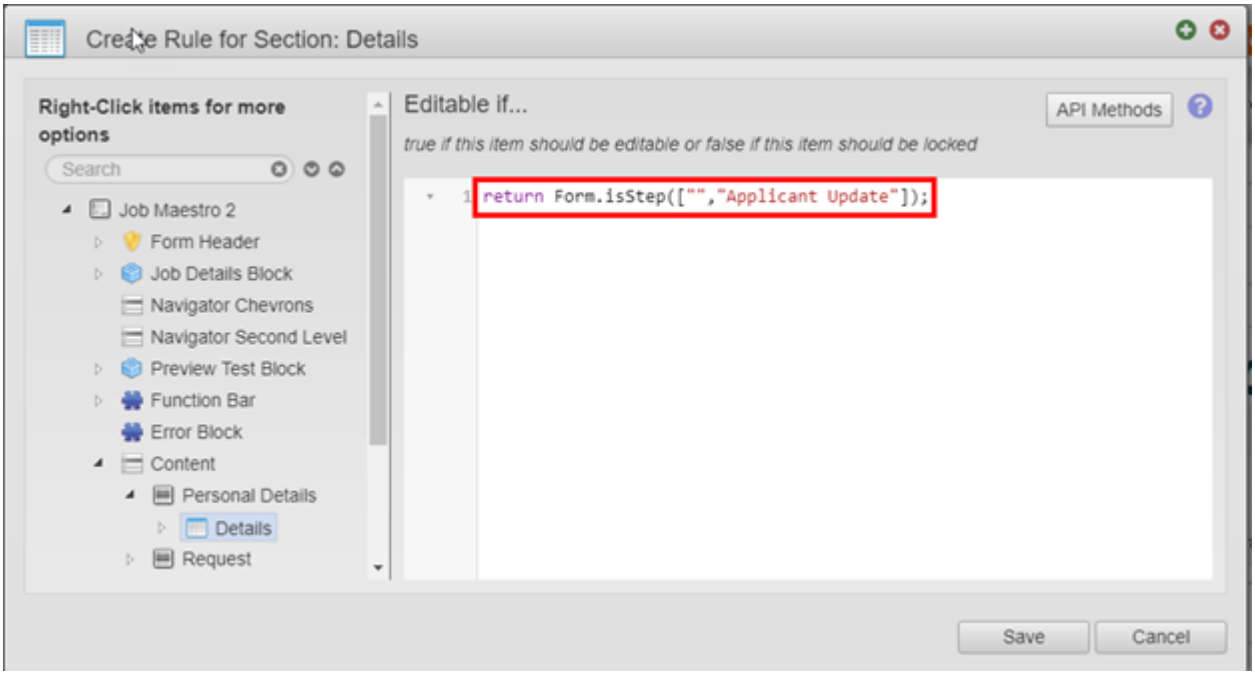
A collaboration job workflow is often required to be shared amongst the relevant parties, such as an applicant, a reviewer and a manager. For the form to be presented suitably to each user, you need to specify the form rules, such as:

- Applicants can enter their personal details but can't approve their own applications. However, they can update their initial request if it was declined.
- Reviewers can indicate their decision. They can view but not update any of the applicant's personal or form request details.
- Managers can indicate their decision which is an additional decision to the reviewer decision. Managers can view but not update any of the applicant's personal or form request details. Managers are like reviewers except they can only view but not update the reviewer's decision.

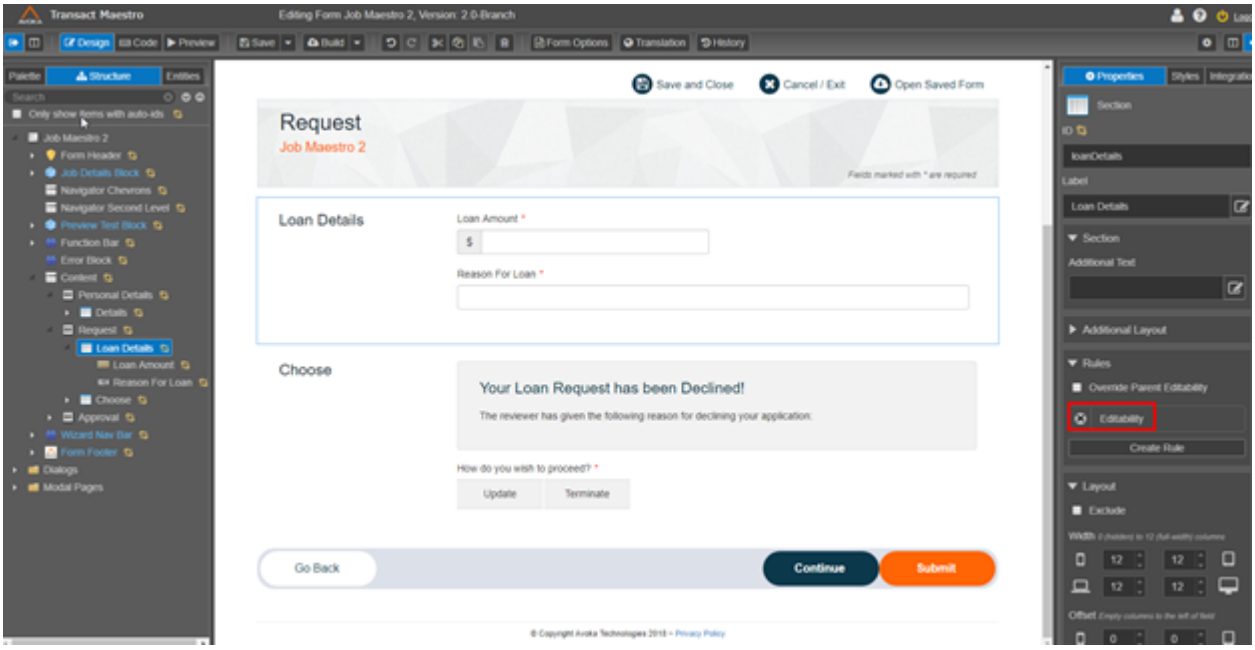
To allow only the applicant to enter and update their personal or form request details, you must create editability rules in your form as shown below.



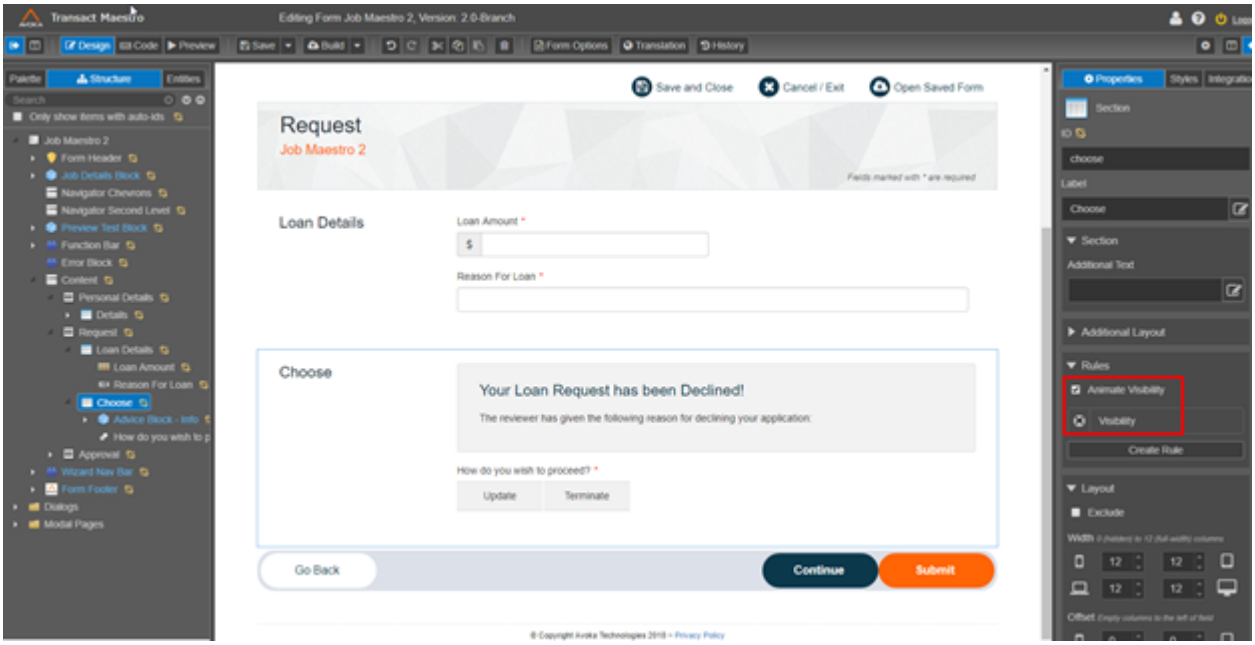
Create the editability rule with the second parameter matching the step name in your intended job controller service configured in Manager. In this example, the step name requesting the applicant to update their details is the Applicant Update step.



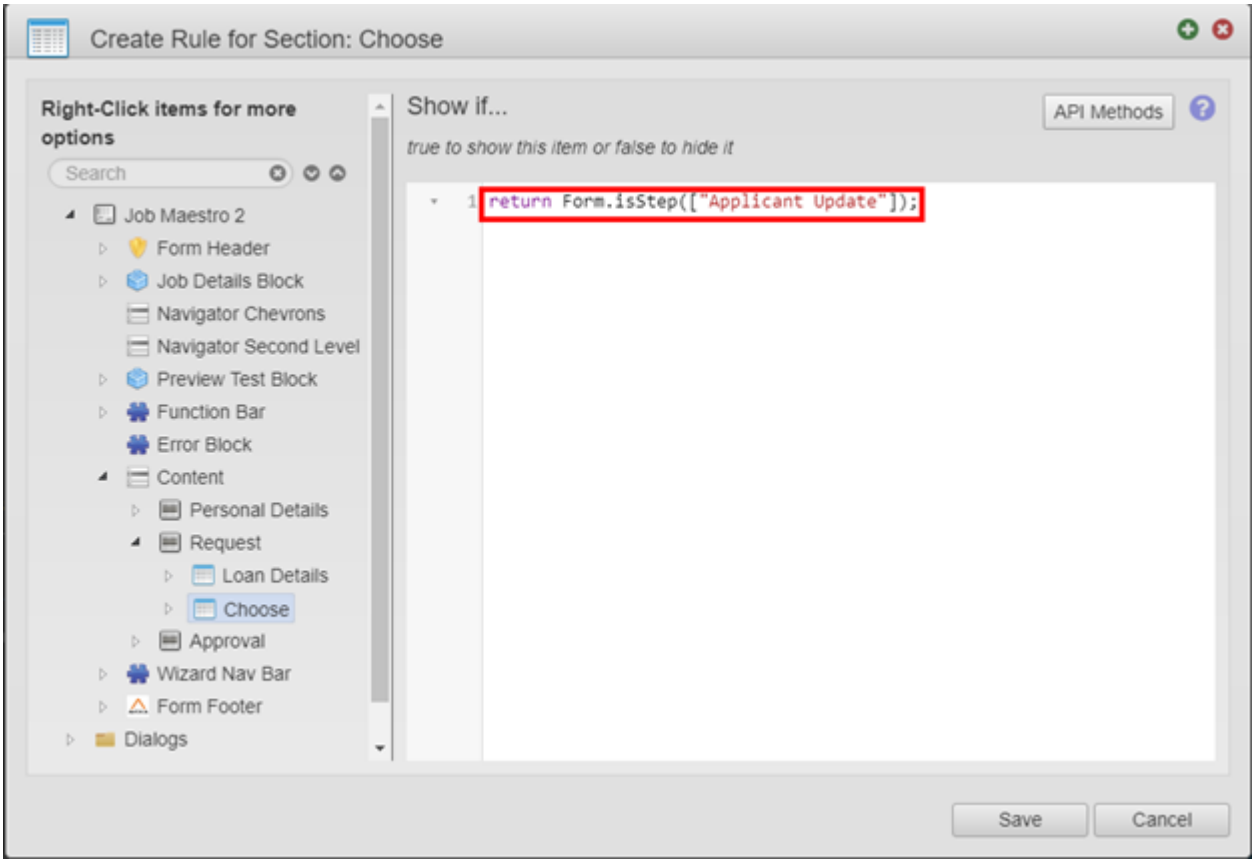
Add this same editability rule to the applicant’s request.



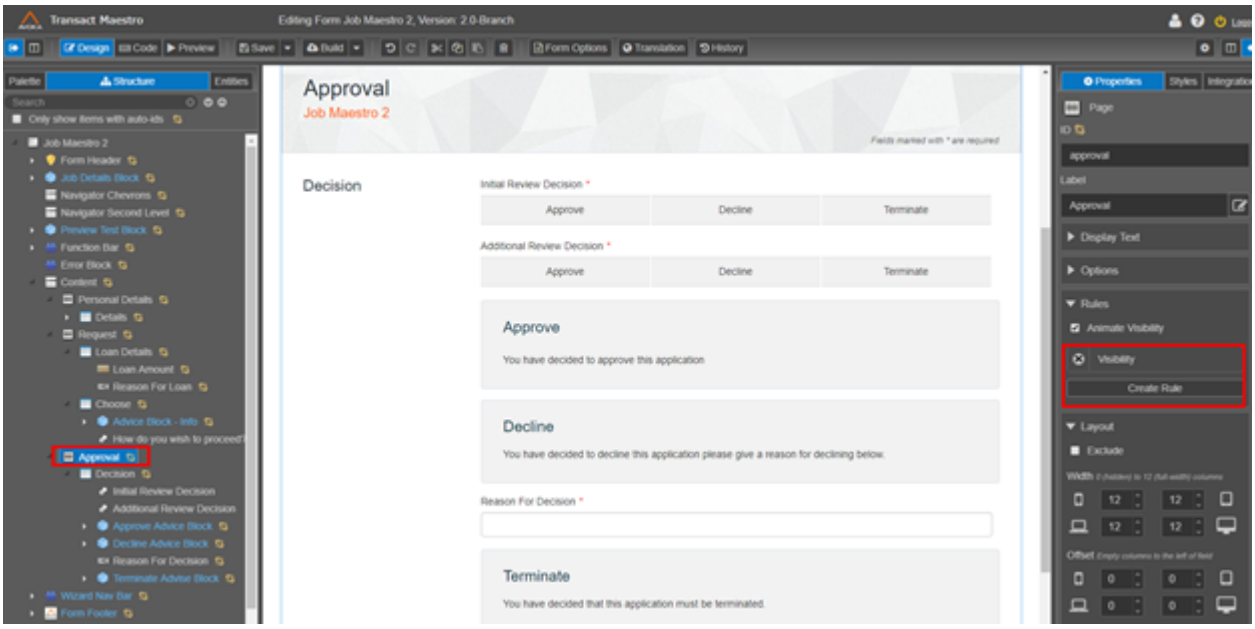
Create the following visibility rule to present the relevant information to the applicants when their application was declined. This rule allows them to choose whether to update or terminate their loan application.



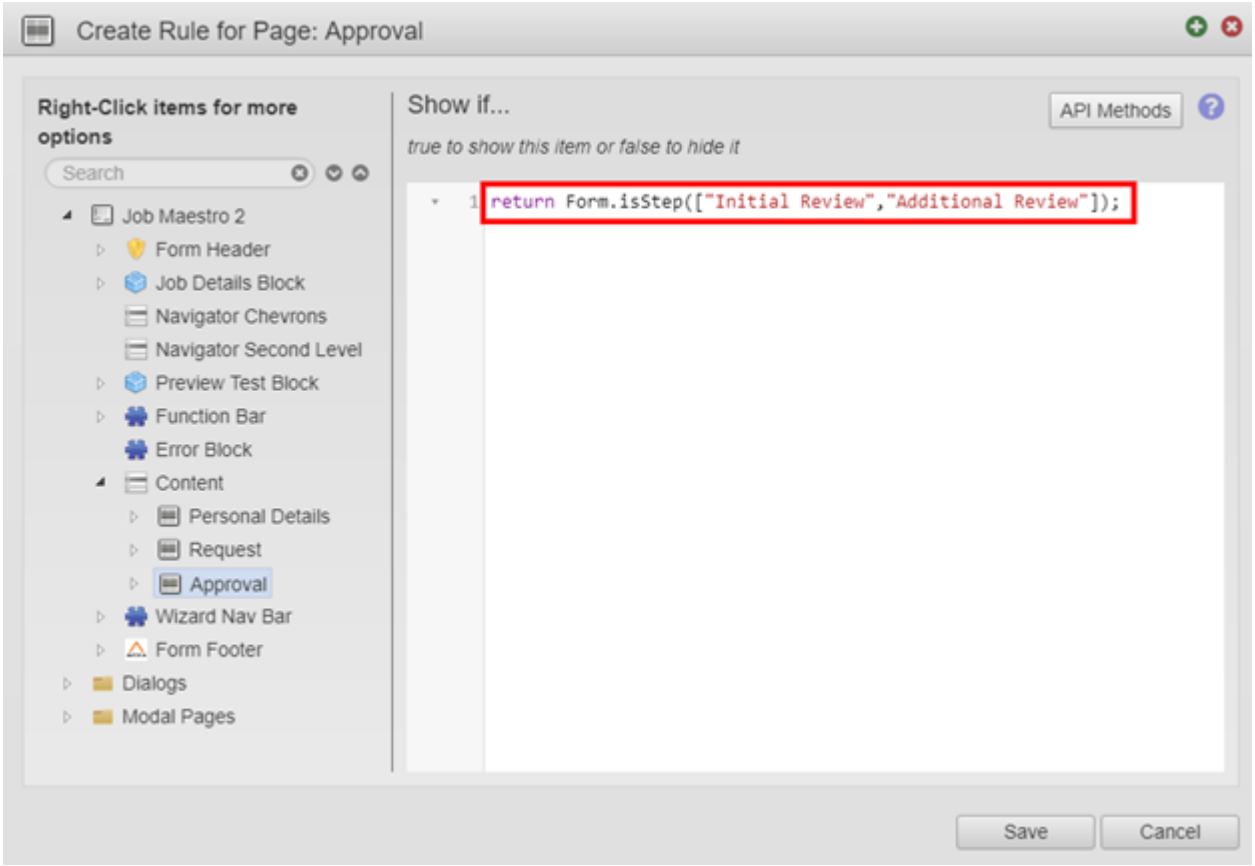
This rule is only relevant for this section to appear when the application was previously declined.



A visibility rule is created to only show this tab to both reviewers and managers.



Create a new rule as shown below.



Using a Radio Button Group for the Reviewer and Manager Decision

Both the reviewer and the manager need to be able to indicate their decision on the form. Previously, we covered the Route Select component, which dynamically loads each of the possible decisions for a reviewer and a manager, using the job controller service. However, a radio button group is much better choice because:

- It looks nicer than a dropdown list.
- Feedback and decision can be clearer.
- The form receipt indicates the decision that was made.
- It accommodates for additional instructions or feedback to the reviewer and manager.

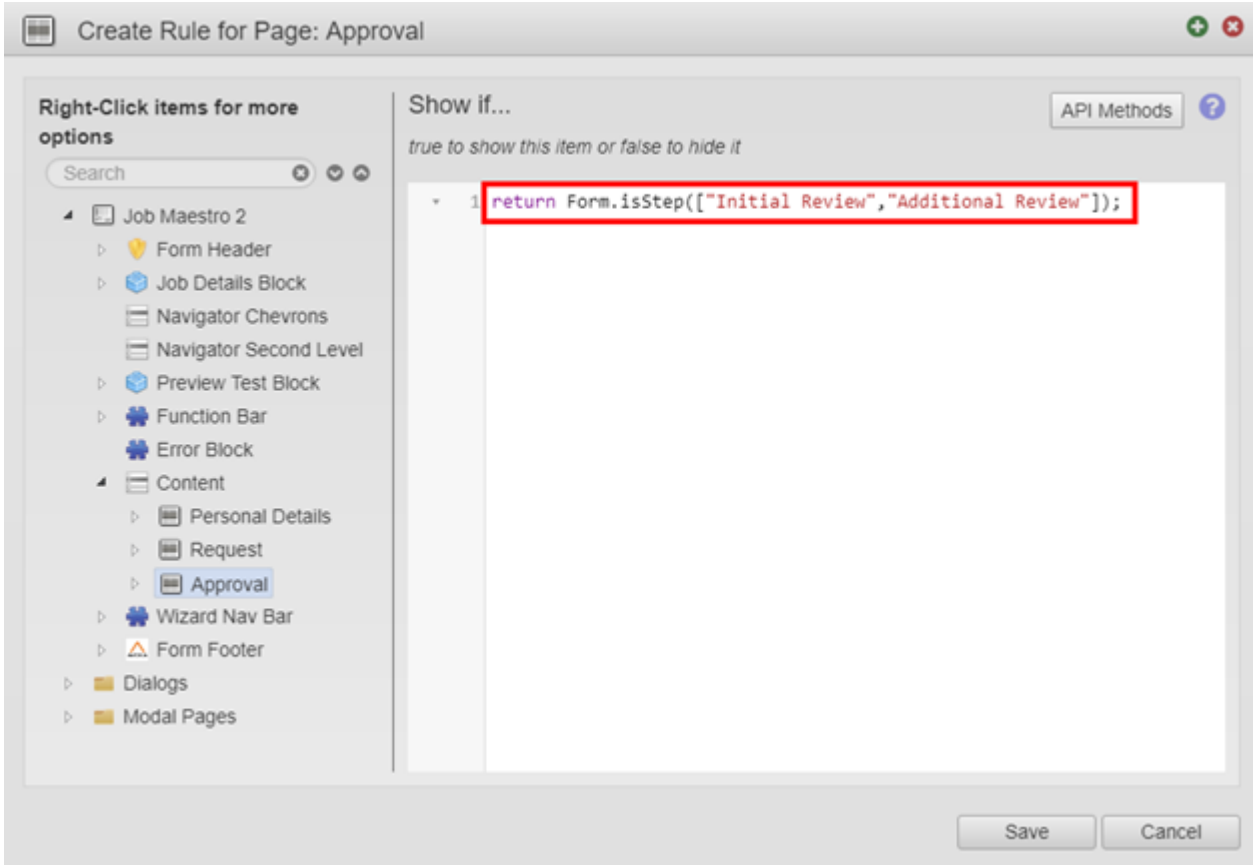
The only negative side is that the radio button group doesn't allow for the dynamic load of the radio button options based on the job definition service. You must hard code the possible decisions as radio button options on the radio buttons group.

In this example, the reviewer and the manager are both involved in the collaboration job, so we will use the Review and Approval – Multi Step Group Review job controller service.

As this decision feature is not available for the applicant, it makes sense to add a separate tab, the Approval tab, to your form, which can be shared amongst the reviewer and manager, so it is only visible to them.

To create a new visibility rule:

1. Select the Approval tab.
2. Select the Properties panel and click Create Rule.
3. Add the following rule.



4. Click Save.

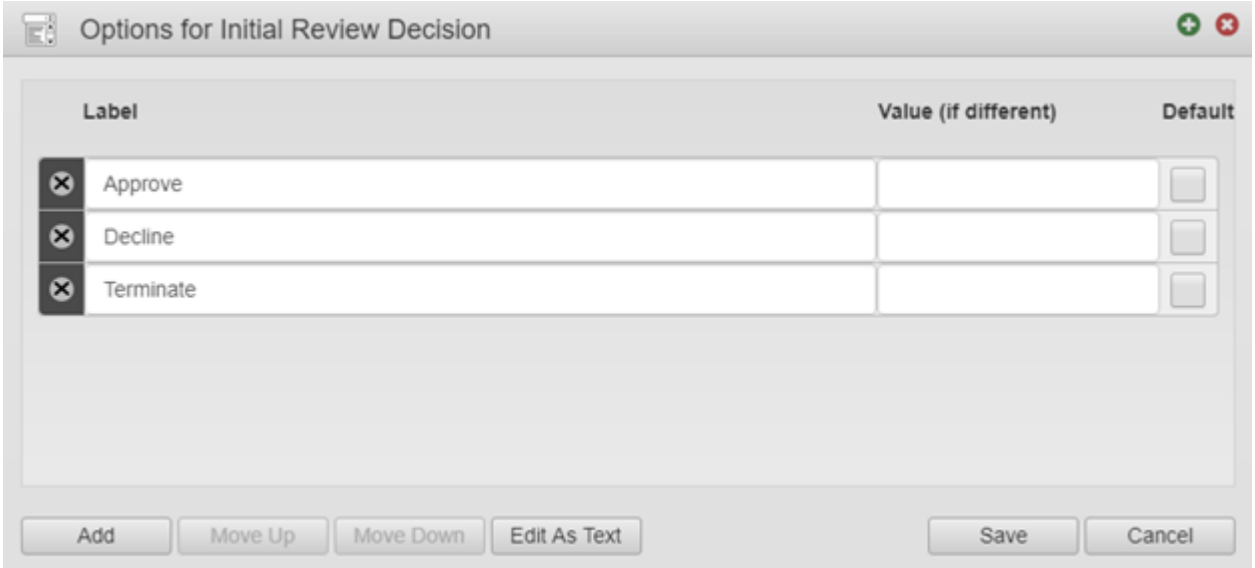
Being a shared tab, we need to manage the form interaction correctly for each of the parties involved. For example, the following rules apply:

- A reviewer can indicate their decision, but can't see the manager's decision.
- A manager can indicate their decision, which is an additional decision to the reviewer decision. A manager is like a reviewer except they can also view, but not update the reviewer's decision.

To keep it simple, we will cover some of the rules here, but you will need to determine all the rules that apply for your form.

To create a radio button group for the reviewer decision:

1. Open your form in Maestro.
2. Add a new tab, the Approval, to capture the reviewer and manager decision.
3. Drag and drop the Radio Button Group component onto your form and name it Initial Review Decision.
4. Select the Properties panel.
5. Expand the Button Group section.
6. Click Options.
7. Enter each of the decision options under the Label column, for example, Approve, Decline and Terminate.



8. Click Save.
9. Expand the Rule section and select the Mandatory checkbox to make the decision mandatory to minimize ambiguity.

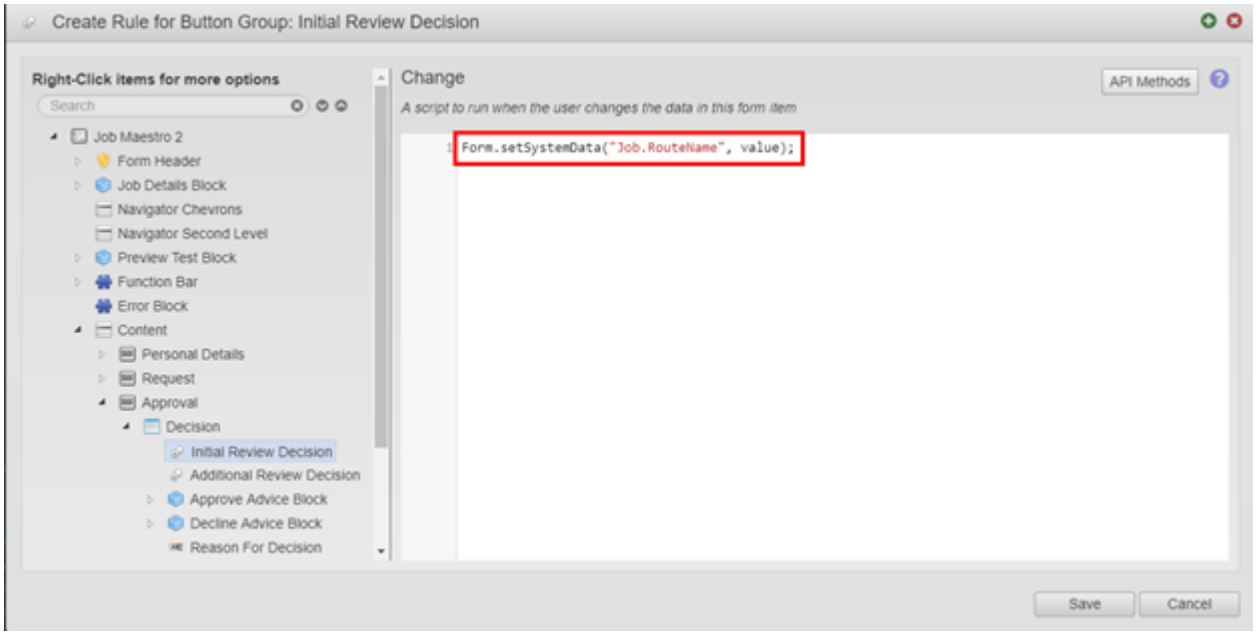
To create a radio button group for the manager decision, repeat the above steps 2 to 8 and give it a suitable name, for example Additional Review Decision.

As explained above, the two radio button groups require different rules, so for the reviewer, the Initial Review Decision radio button group, you need to create the following two rules:

Change Rule

An added complexity exists with sharing this form regarding which decision must be used for the job controller to decide what to do next. For this, you need to create a change rule which plugs the reviewer decision to the system profile route name, Job.RouteName, only when the reviewer has changed this. This means that, on change, the reviewer decision will be used by the job controller. To do this:

1. Select the Properties panel and click Create Rule.
2. Add the following rule.

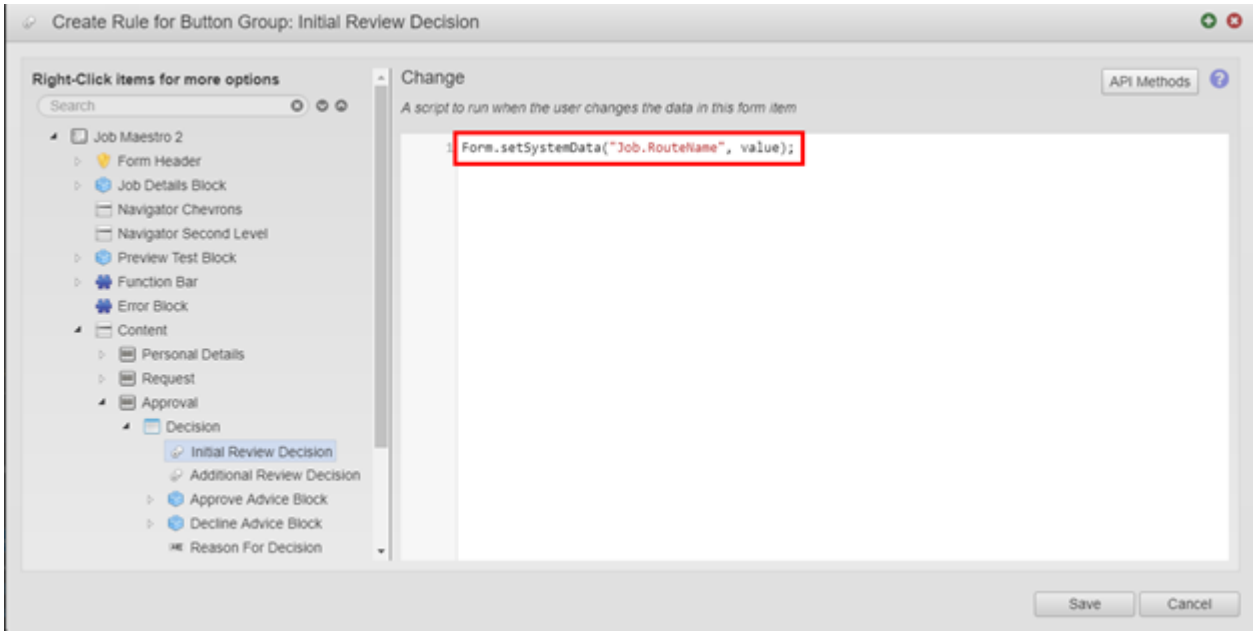


3. Click Save.

Editability Rule

As the reviewer can also be seen by the manager we need to make sure that only the reviewer can update this. To do this:

1. Select the Properties panel and click Create Rule.
2. Add the following rule.



3. Click Save.

For the manager, the Additional Review Decision radio button group, we need to create the following two rules:

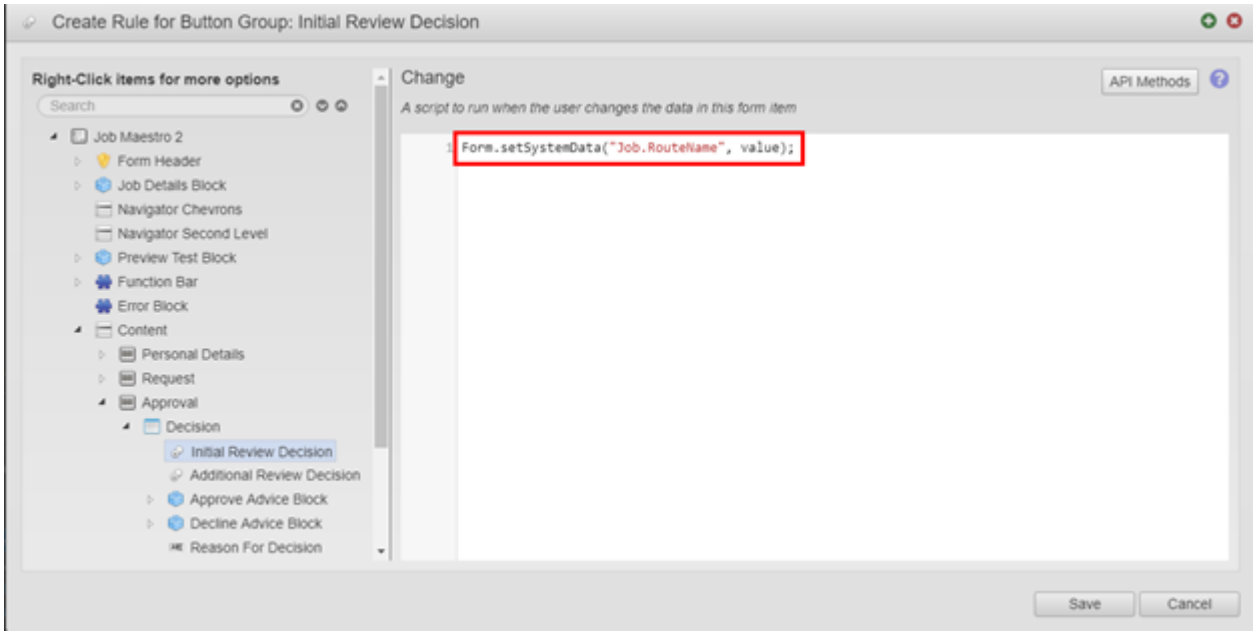
Change Rule

The same change rule as for the reviewer needs to be added here to allow the manager's decision to be used for directing what happens next when this has been changed.

Visibility Rule

This radio button group must only be visible to a manager. To do this:

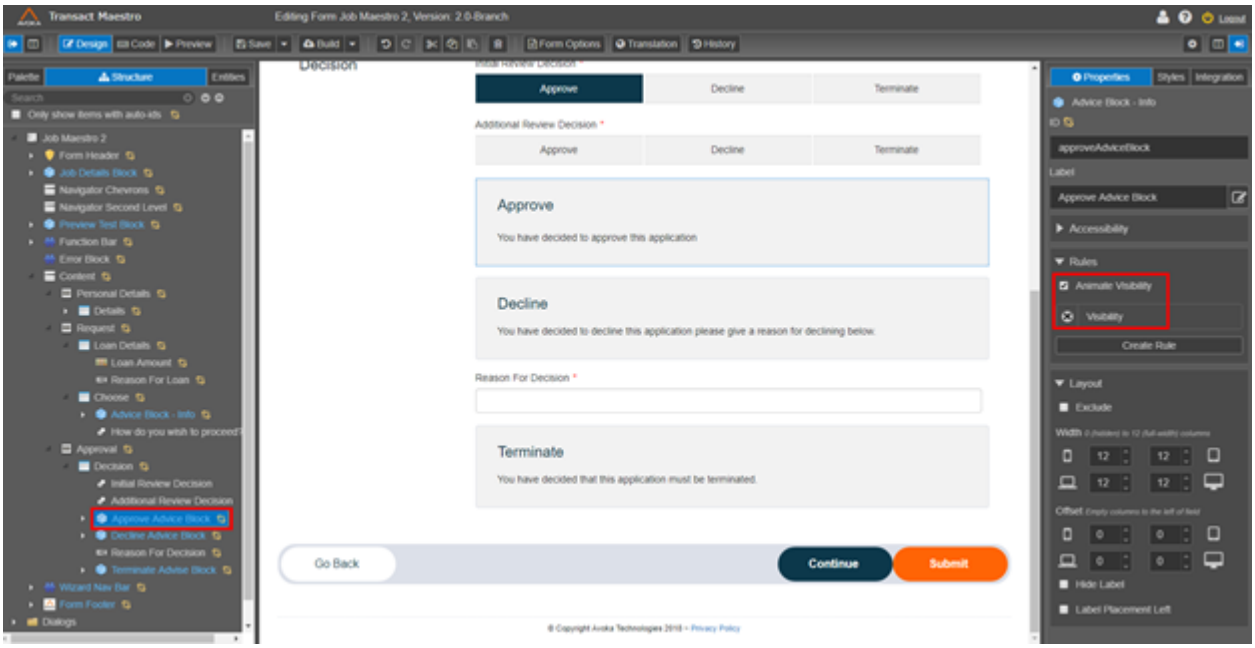
1. Select the Properties panel and click Create Rule.
2. Add the following rule.



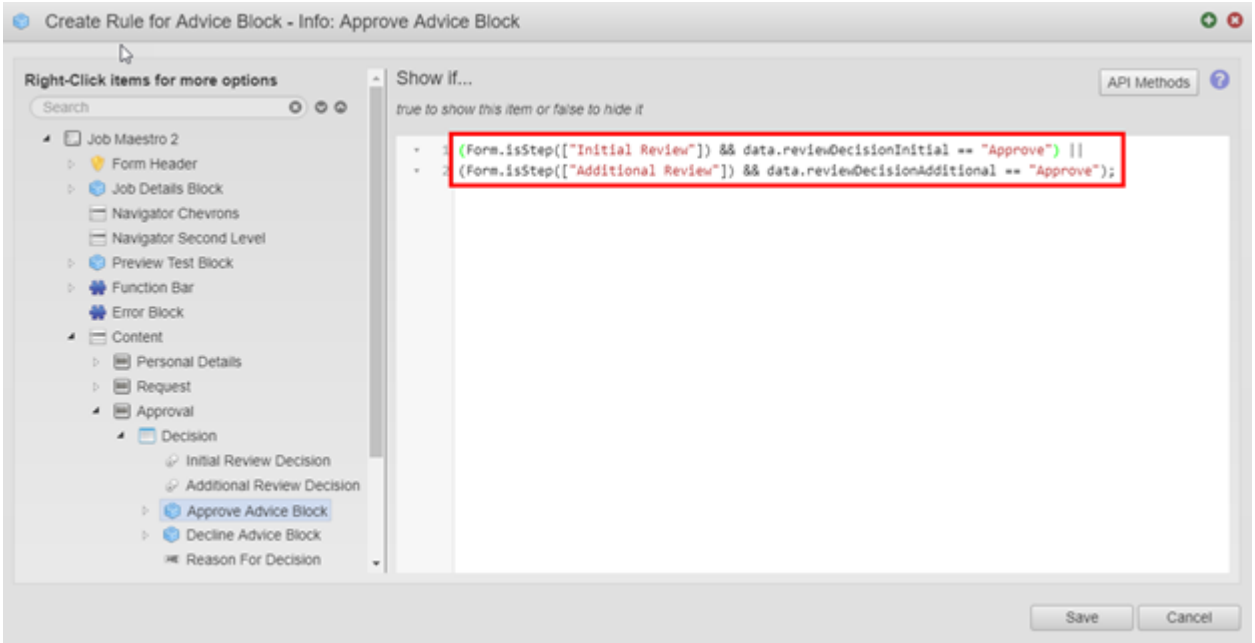
3. Click Save.

Some other rules that may be relevant are:

In this case an Approve Advice Block pops up when the reviewer or manager clicks the Approve radio button option, to highlight the decision they are about to submit.

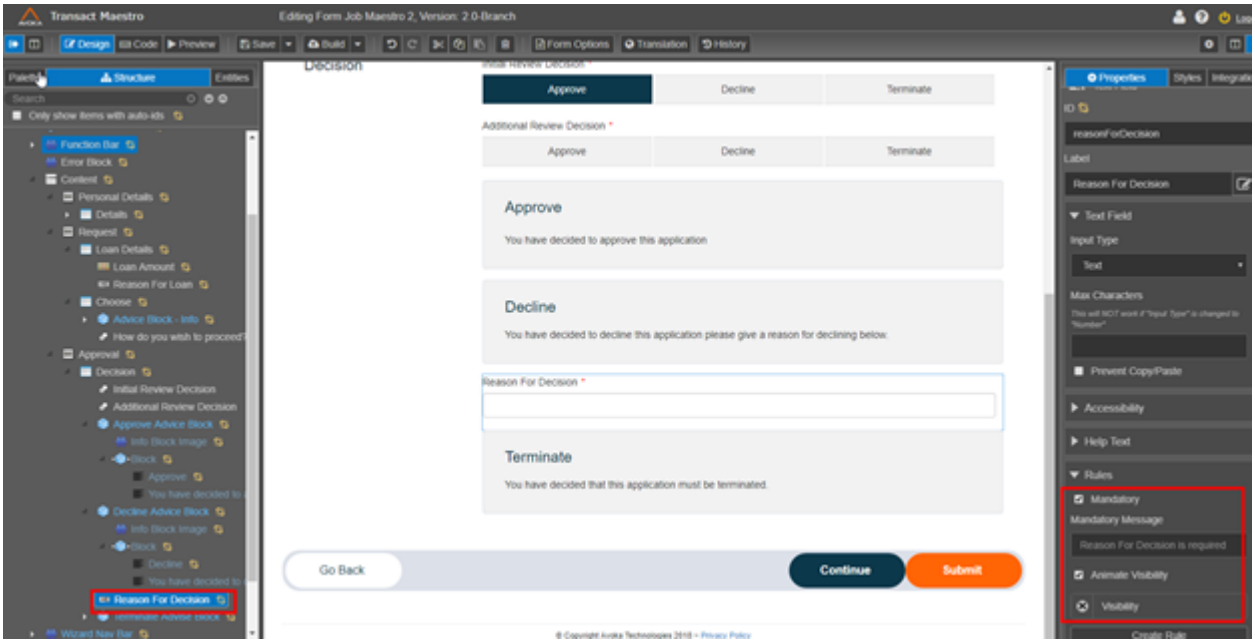


For this the visibility rule is:

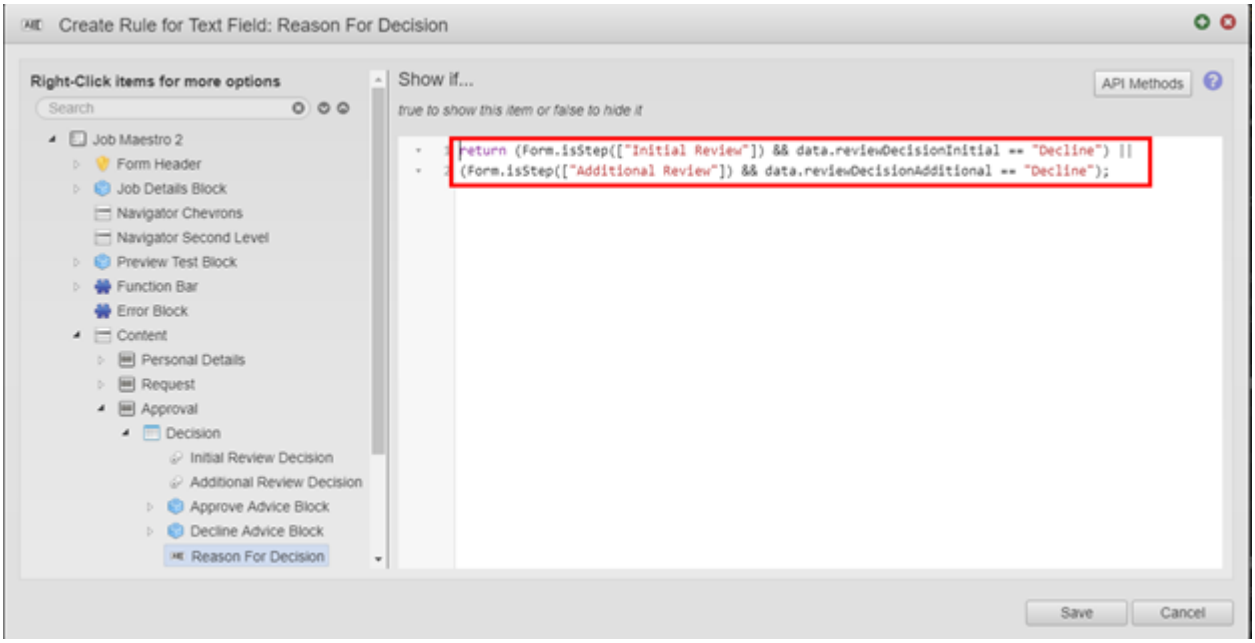


When either the reviewer or the manager declines the application, they must enter a reason for their decision.

The following visibility rule is required for the Reason for Decision field which is ticked as Mandatory.



The visibility rule is:



Now, your Maestro form is ready to be used with collaboration jobs.

Next, learn how to configure Composer forms for use with collaboration jobs.

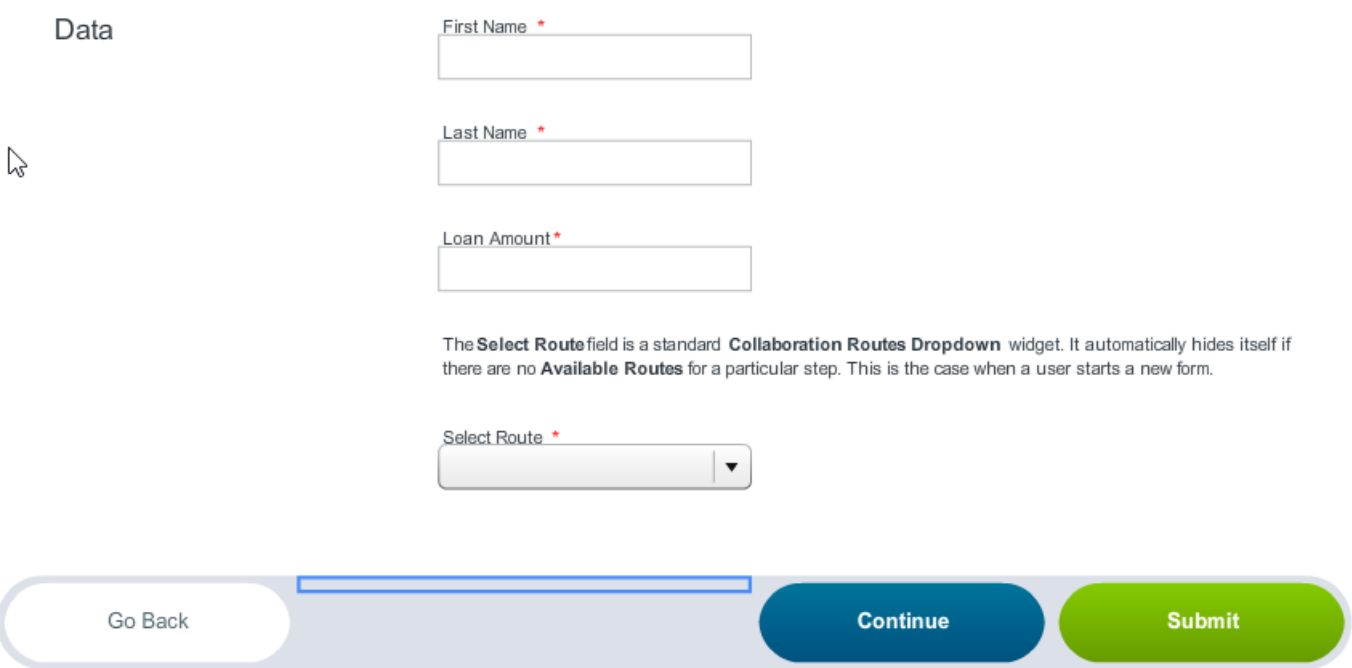
Configure Composer Forms for Use with Collaboration Jobs

Composer forms require configuration before you can use them with collaboration jobs.including:

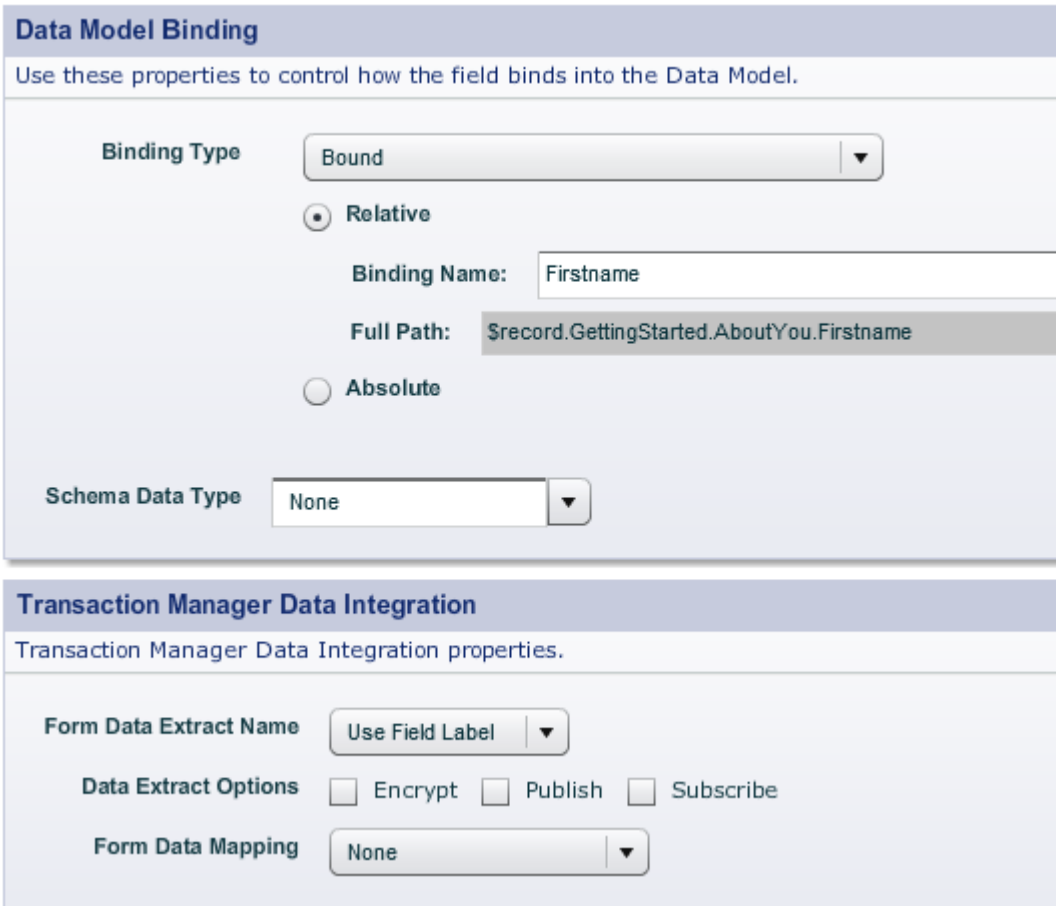
- How to configuredata extracts.
- What collaboration jobs widgets are.
- How to set rules, such as the section visibility and editabilty rules.

Data Extracts

Data extracts are used by collaboration jobs to access form data. In this example, we use the Composer form Job Example 2 Minimal.



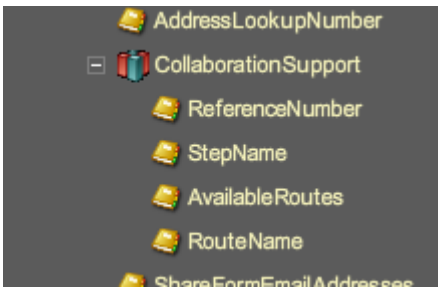
A collaboration job will put this into the `$formDataMap.routeName` values automatically. Data extracts are configured in Composer in the Data Model tab of the field's properties dialogue as shown below.



The easiest way to setup the data extract is to use the field label. However, you can use an alternative name. Composer version 4.3.0 and above has the Encrypt, Publish and Subscribe checkboxes to specify data extract options.

Form Data Mappings

The collaboration components described in the section below rely on the 4 Job data elements that are read into form variables. These can be found in Nuts & Bolts -> Transaction Manager Support -> Collaboration Support as shown below.



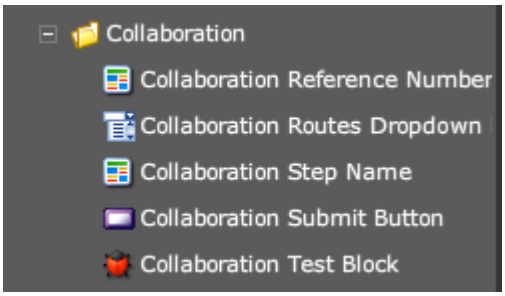
The job controller sets the following XML data elements for each task:

- ReferenceNumber, which is a Job Number
- StepName
- AvailableRoutes, which is a pipe delimited list of the current steps route names

The Composer form sets the RouteName on Task Submission via a widget or business rule.

Collaboration Widgets and Components

Composer has a number of widgets that can be used to build forms for collaboration jobs as shown below.

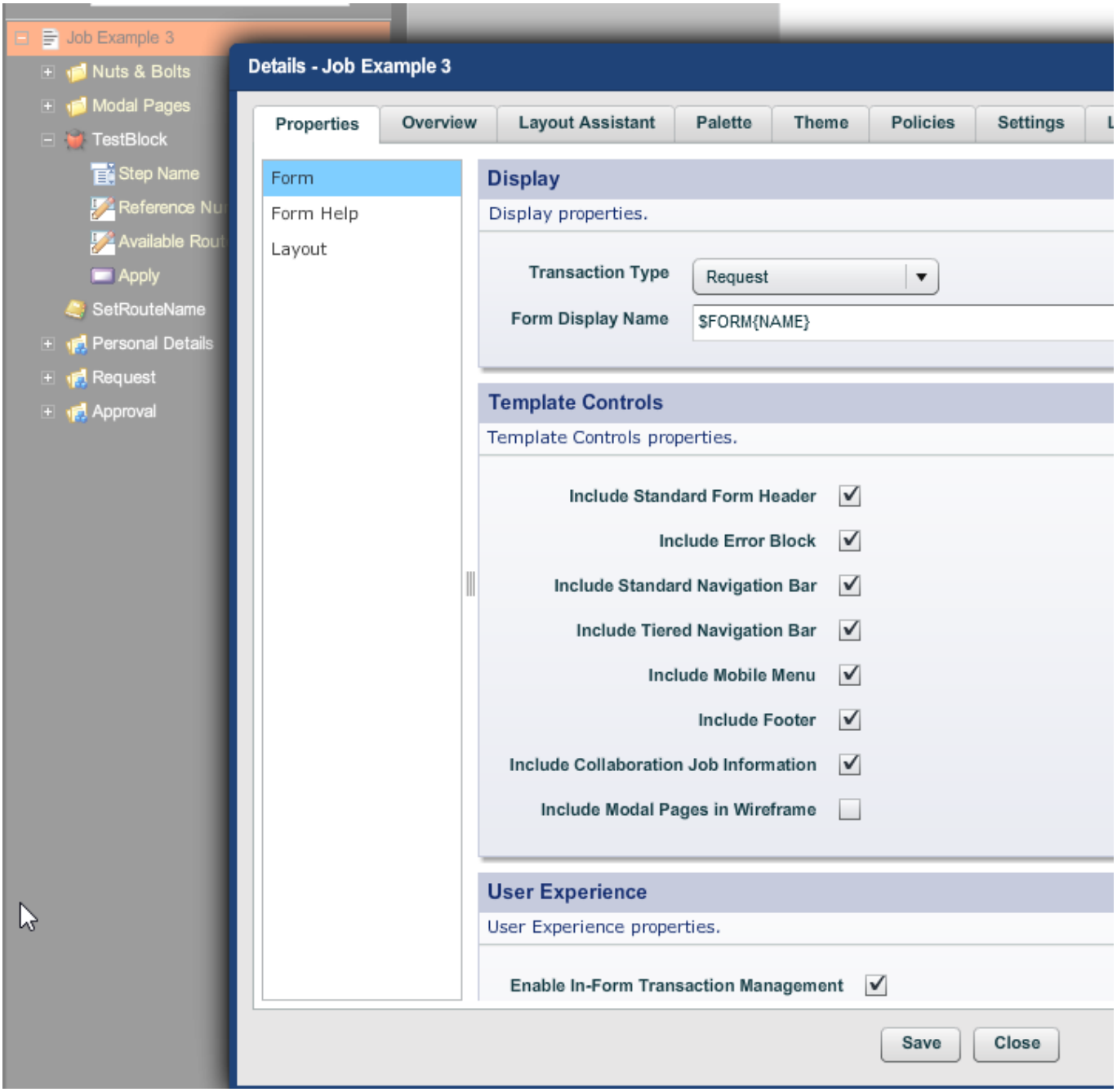


The Maguire template has additional collaboration support to display the step name and job number in the form heading.

Collaboration Reference Number and Collaboration Step Name

The Collaboration Reference Number (Job Number) and the Collaboration Step Name are Rich Text Fields to display the Job Number and Step Name on the form. Both fields can specify a prefix. The Minimal form contains Both fields above the data fields. We don't recommend that these fields be used as they are now both incorporated into standard Maguire 4.1 Template as shown below.

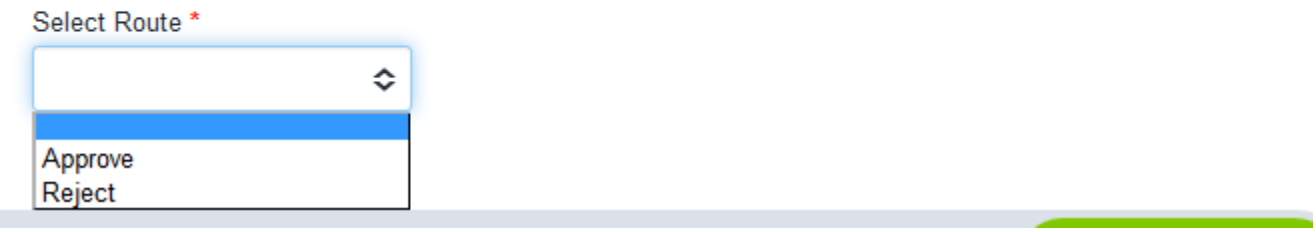
The template hides the Step - Job when they are blank such as when the applicant first fills out the form. To turn on this Maguire Feature double click on the forms root node, under the properties tab select the Include Collaboration Job Information check box.



Collaboration Routes Dropdown List

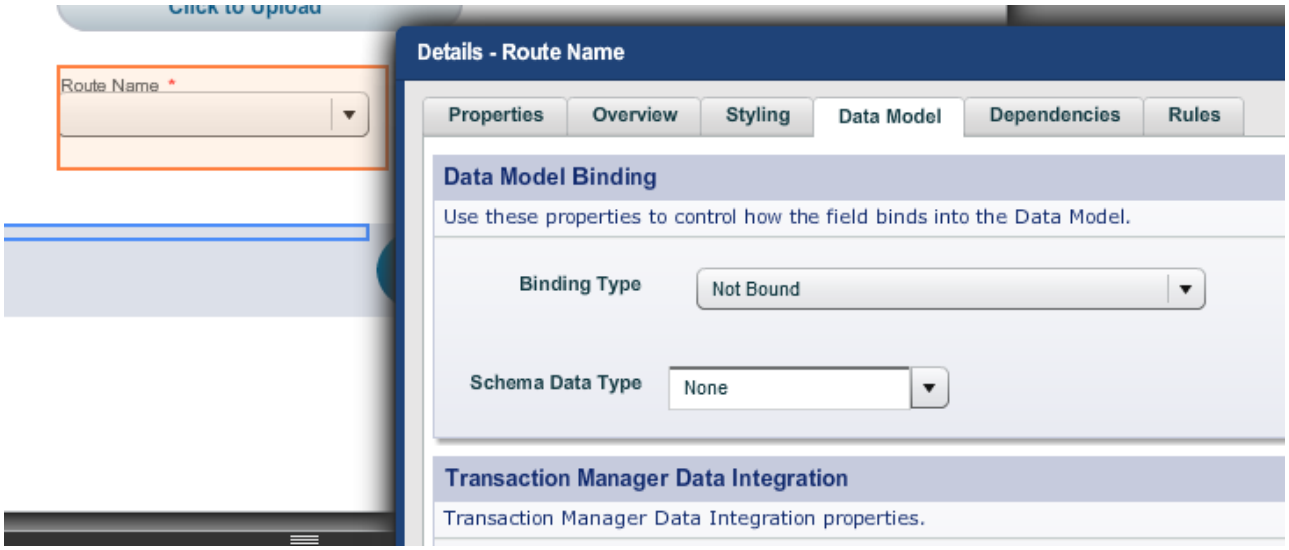
The Collaboration Routes dropdown List is the easiest way to make a user select the route to the next step.

The **Select Route** field is a standard **Collaboration Routes Dropdown** widget. It automatically hides itself if there are no **Available Routes** for a particular step. This is the case when a user starts a new form.



To ensure this item doesn't pick up the route of the previous step, its strongly recommended that the binding be changed to Not Bound. To do this:

1. Position it on the form.
2. Select Properties > Data Model > Binding Type
3. Select Not Bound.



Collaboration Submit Button

WARNING

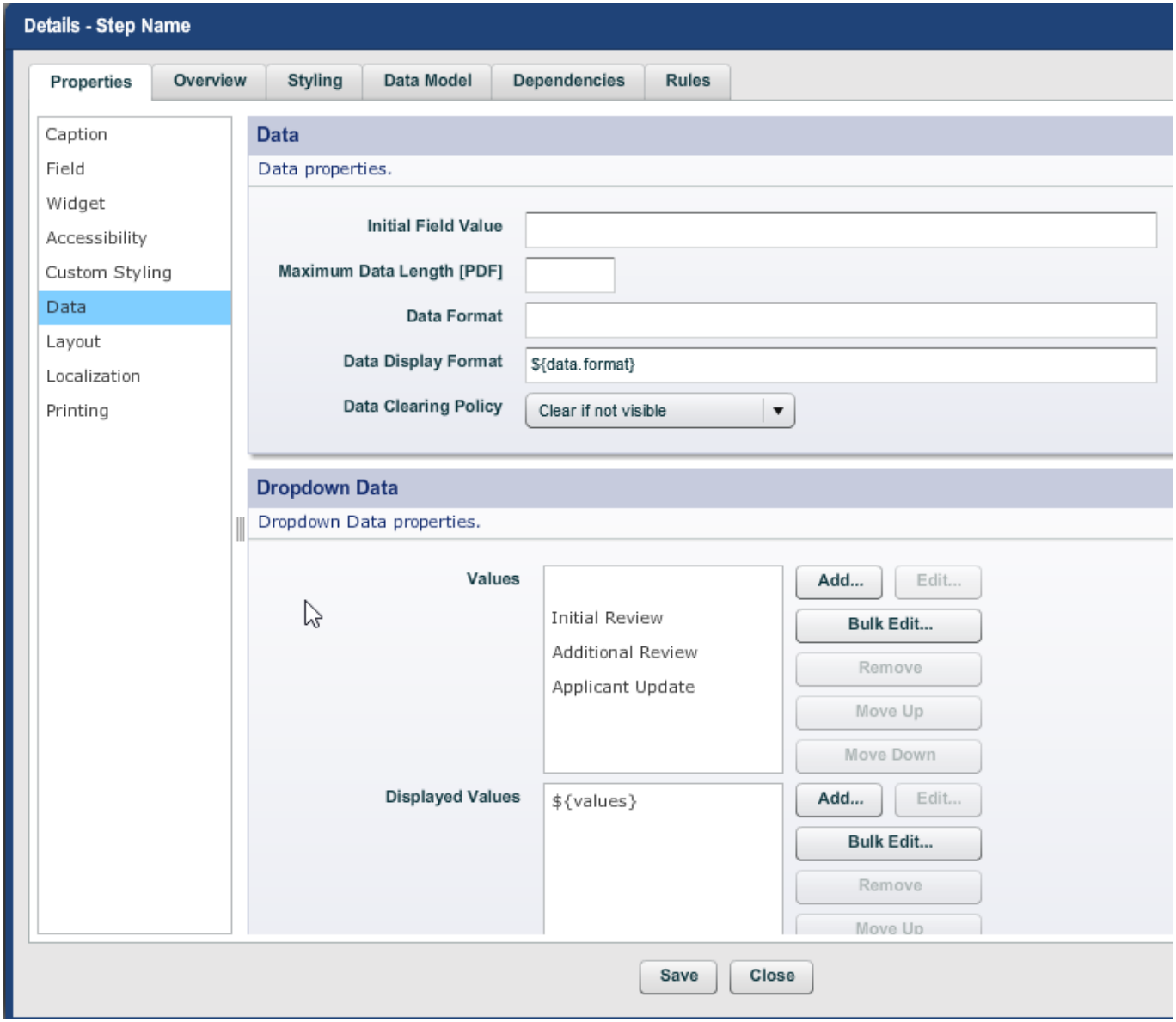
Don't use the Collaboration Submit Button widget, because the form using this button can fail to return the route. It was initially prototyped to create a submit button that sets it's label as a route.

We recommend that you either use the Collaboration Routes dropdown or the standard Radio Button Group as described below.

Collaboration Test Block

The Collaboration Test Block is used to test a form in the Composer preview. It has the Step Name, Job Number and the Available Routes. Composer version 4.2.0 and above has 3 fields that are used to test Task Assign Repeat tasks.

Select the collaboration job's step name from the Step Name dropdown list. Don't forget to include an empty row to test the field's behavior in the form before its submitted.



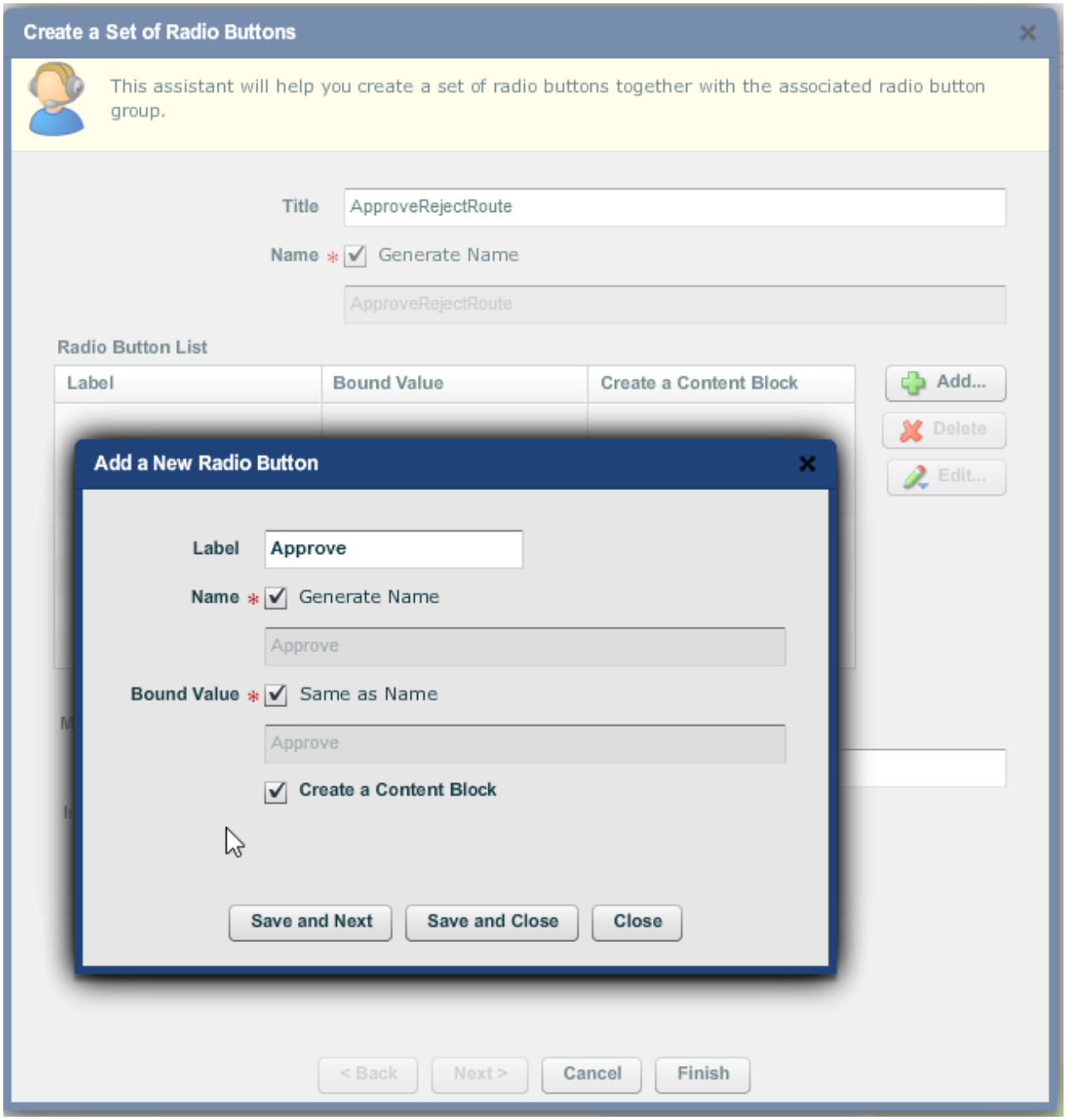
Routing Using the Standard Radio Button Group

This is the preferred method to use with the collaboration submit button.However, it doesn't dynamically create the radio buttons, so you have to hard code the values in your form. The advantages of using the radio button groups are:

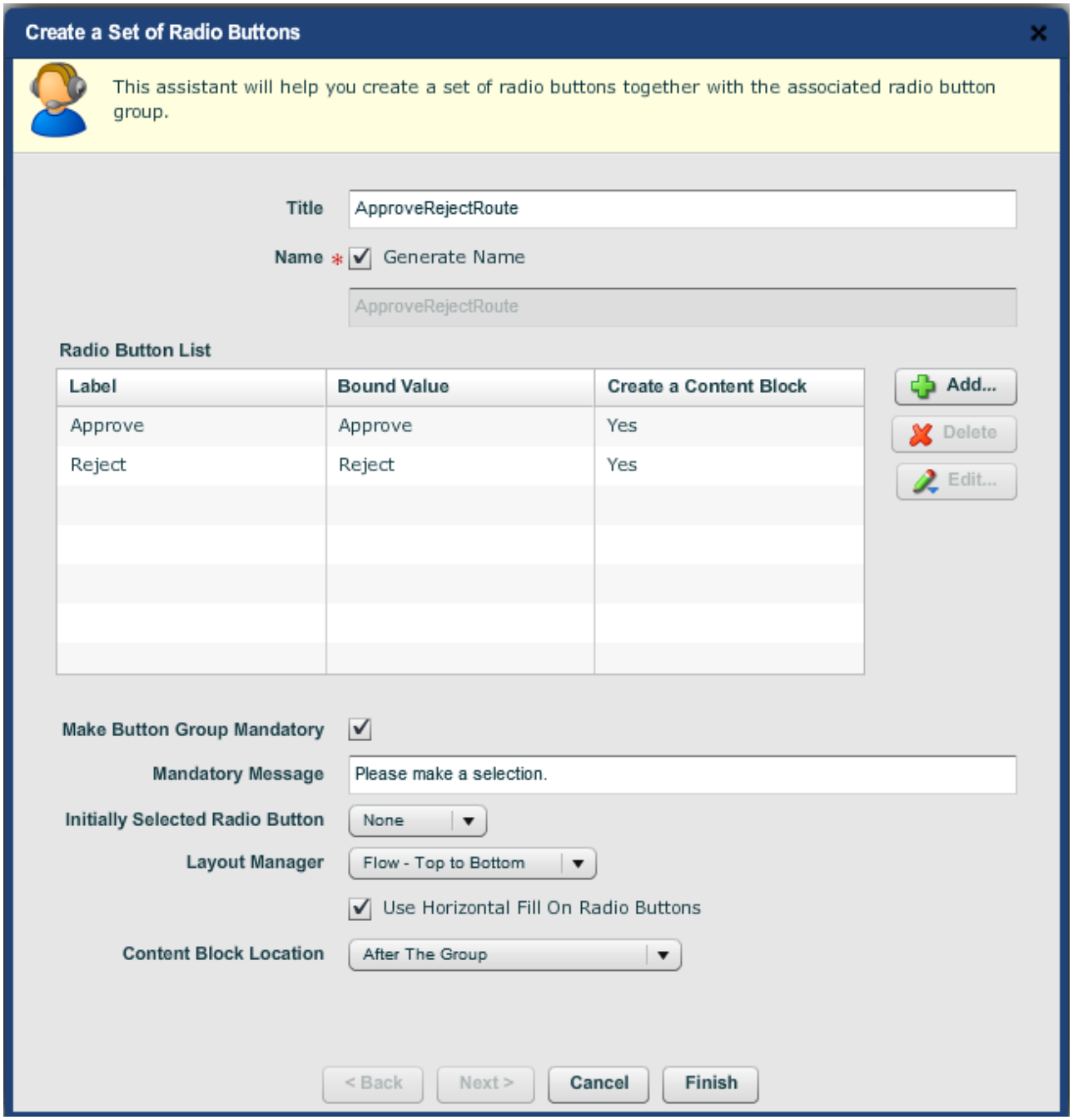
- They look nicer than the drop down.
- The give feed back to the user as to what they intend to do.
- The route choice appears on the form receipt.
- They can specify an optional content block with additional user's instructions, which is useful for new form users. The content block can also hold addition fields that give feedback to the next review step.

The following example creates a single radio button group that is used for form approval. To do this:

1. Log onto Composer.
2. Drag the Radio Button Assistant widget into you form hierarchy. This will open up the Create a Set of Radio Buttons wizard dialog.
3. Add the route names as Radio Button Labels.

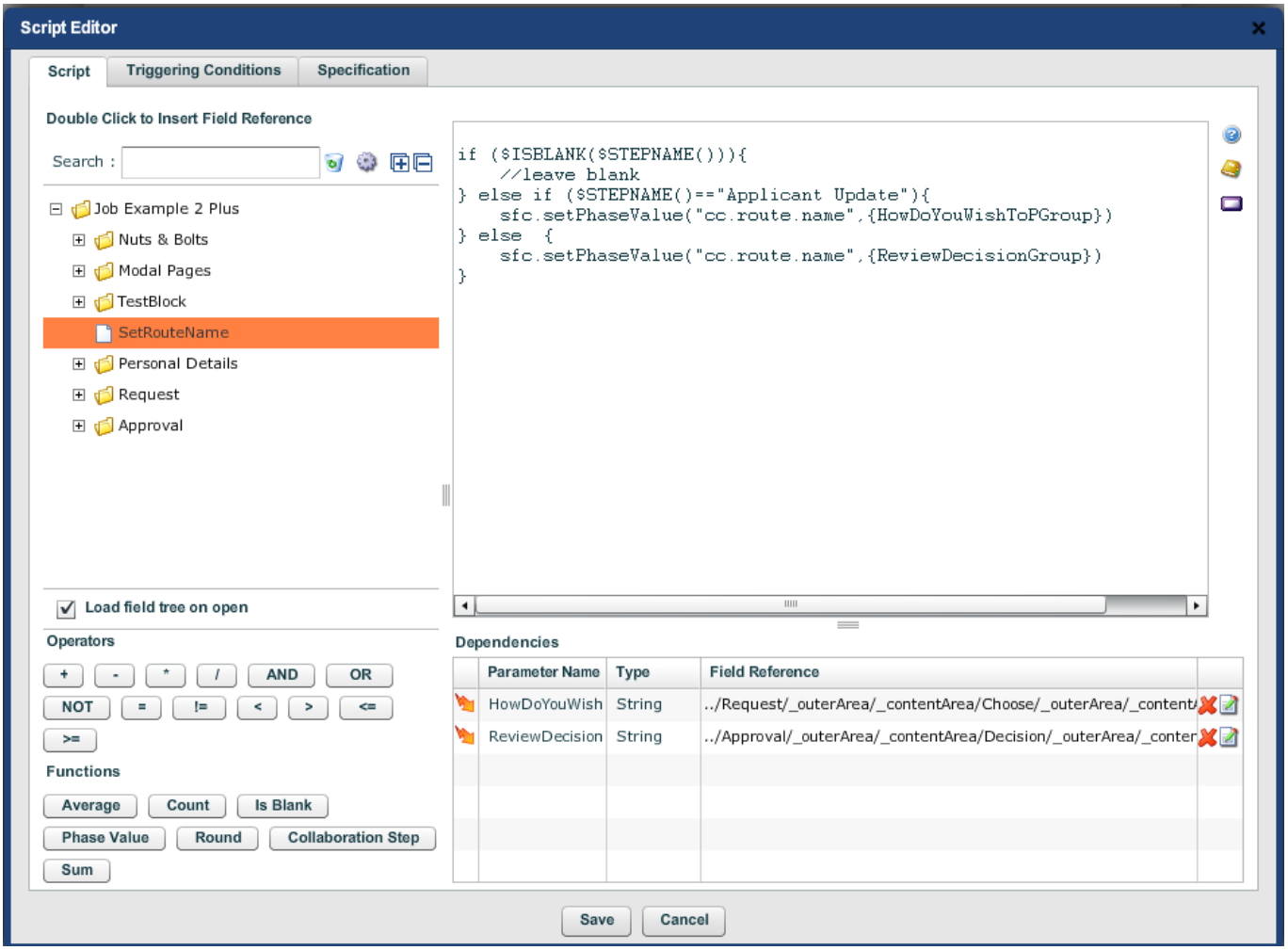


4. Make the button group mandatory.



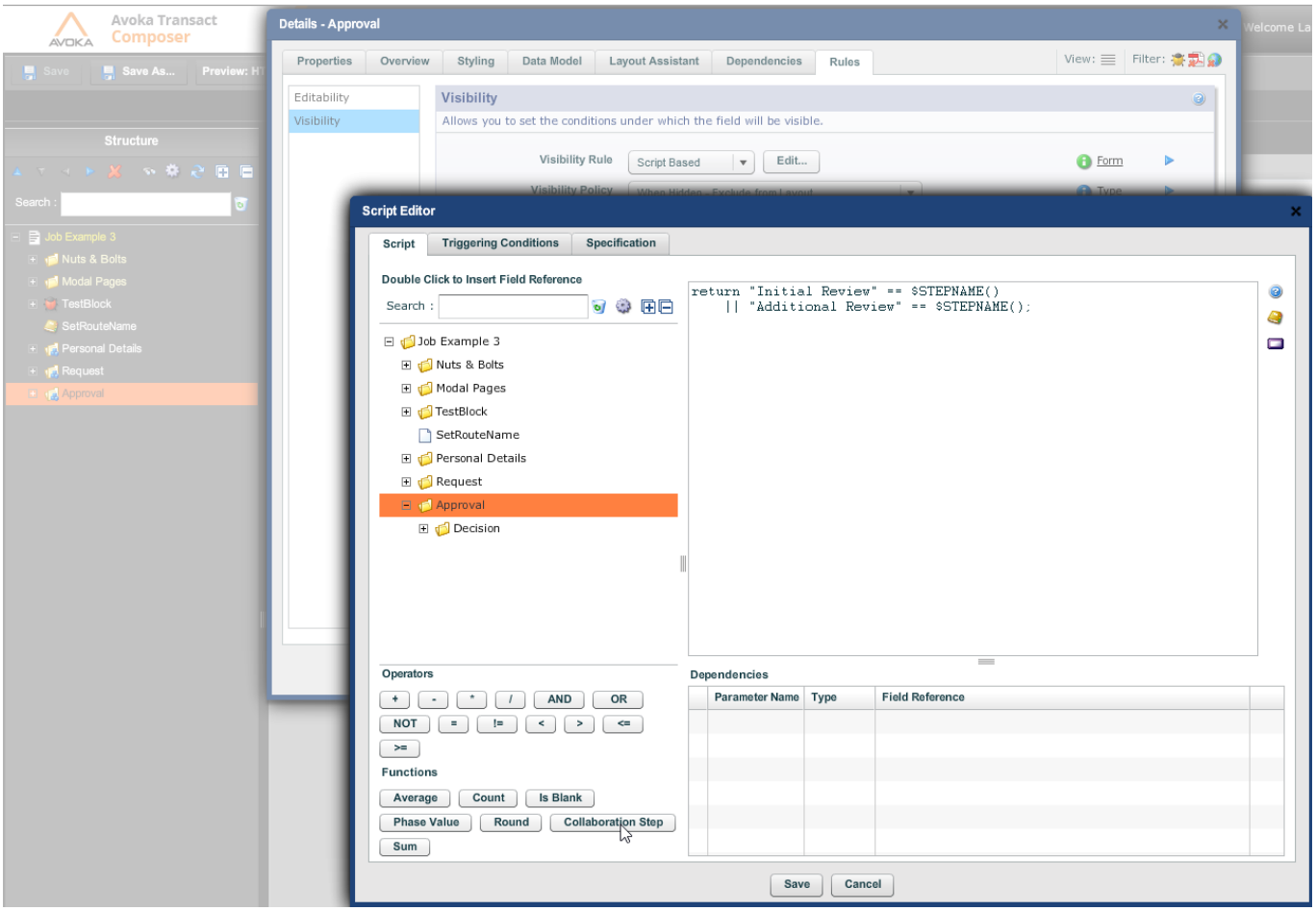
Business rules can be used to set the route. Your form has steps with different route choices like the Job Example 2 (Multi Step Review Job) with the business rule shown below.

```
if ($ISBLANK($STEPNAME())){
       //leave blank
} else if ($STEPNAME()=="Applicant Update"){
       sfc.setPhaseValue("cc.route.name",{HowDoYouWishToPGroup})
} else  {
       sfc.setPhaseValue("cc.route.name",{ReviewDecisionGroup})
}
```

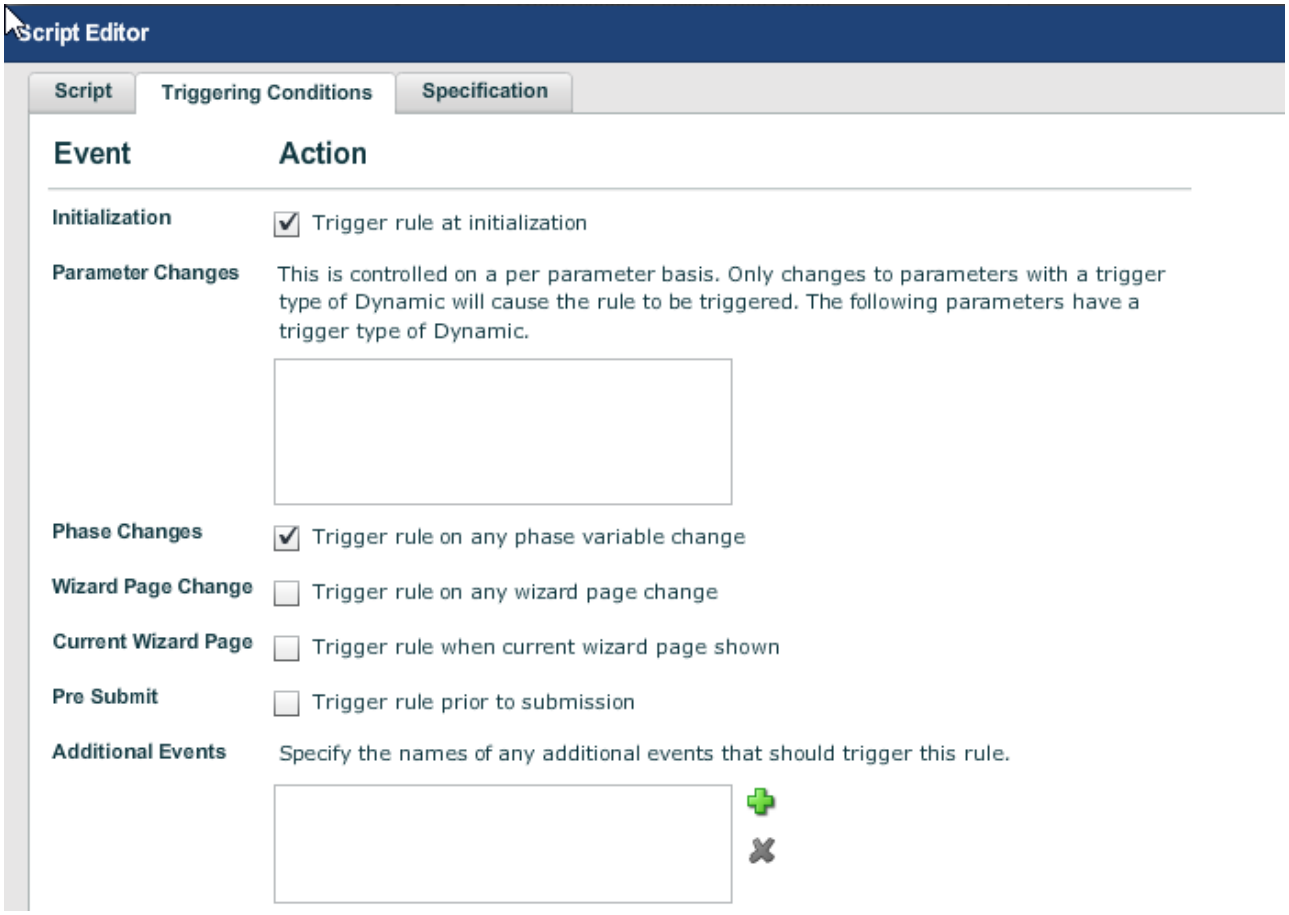


Visibility and Editability Rules

Scripts are used to setup visibility and editability rules for sections based on the Step Name. This can be used to hide the office use only section from the applicant or make the section of form that the applicant fills out read only. When configuring a script rule, use the collaboration step button that inserts a `$STEPNAME` reference to the step name into your script.



1. Select the Triggering Conditions tab.
2. Select the Phase Changes checkbox.

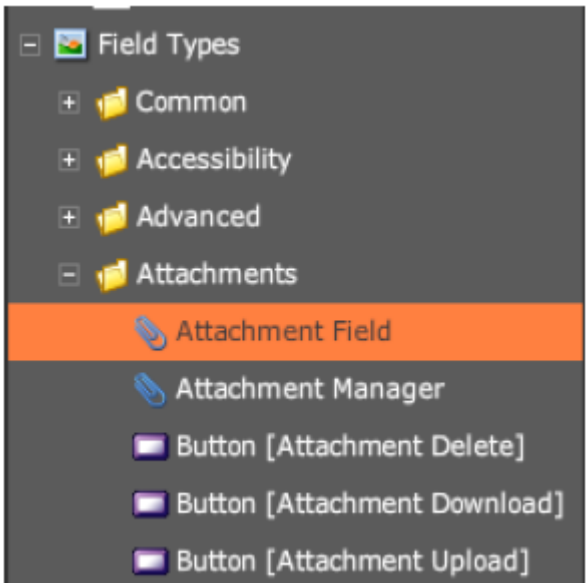


3. Click Save.

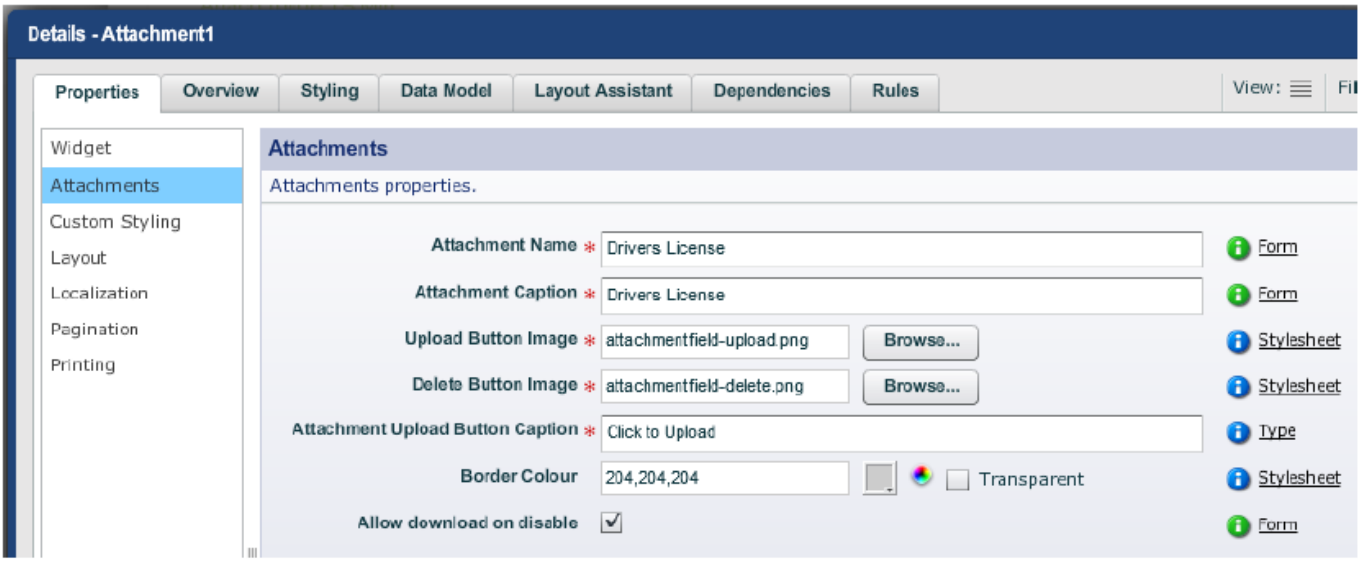
The step name is blank when this form is first submitted. The Visibility Rule that would show a section that is only visible when a user fills out the form for the first time return: `isBlank($STEPNAME())`

Attachments

The Attachment Field is available with Manager version 4.1 and above that uses the Maguire Template 4.1 as shown below.



This widget will allow a file to be uploaded. A user may want to confirm the file is correct and can download the form by clicking on the file link. They can delete the file by clicking on the X. The field works as expected when a user saves the form any attachments that have been uploaded are available when the form is next opened. This field uses the same Editability and Visibility rules as discussed above. However, if the editability rules determine that the widget is not editable at a particular step, the attached file can't be uploaded, downloaded or deleted. There is an Allow download on disabled check box as shown below. This is useful where a reviewer may want to look at the form but not delete or modify it. This is on by default.



NOTE

The widget has an internal field Attachment Manager, which must not be altered when using with Manager task assignments and the job controller.

Configure Routes with Business Rules

It is possible to set the route from within the form using a business rule, which is how the pre-built Collaboration Widgets and Components work. Using a simple script, you can set the route phase value, which allows for logical changes to the collaboration route based on form data.

To configure the route:

1. Add a business rule to your form.
2. Edit the business rule script with the following code.

```
if (condition) {
       sfc.setPhaseValue("cc.route.name", "Deliver");
} else {
       sfc.setPhaseValue("cc.route.name", "Review");
}
```

You can set a number of the values associated with the collaboration job, including the step name (`cc.step.name`), reference number (`cc.reference.number`), available route (`cc.available.routes`), and assignee (`cc.assignee`), using this method.

Next, learn how to configure Maestro forms for use with collaboration jobs.

Customer Onboarding Collaboration Jobs

A customer onboarding collaboration job implements a simple customer onboarding form bundle. The bundle consists of a number of separate forms that are filled out by a single user in the one step. We will use this as an example to illustrate the following:

- Anonymous users.
- Preconditions to determine which forms to use.
- Form bundle navigation.
- Redirect to a next form functionality or form chaining.
- Confirmation on the Signature page.
- Consolidation of form receipts.
- Dynamic preconditions.
- Shared form modal using the entire form shareFormData XML or shareExtractData for publish / subscribe only.

The Workflow

A user opens the customer onboarding triage form and fills the required details, such as the name, phone number and email address.

MAGUIRE FINANCIAL

Reference Code: 85LCKS

Have you previously started a form? Resume a saved form

Save For Later | Cancel / Exit | Share Form

Getting Started

Customer Onboarding

*Fields marked with * are required*

OK, Lets Get Started!

We make it easy to apply online and it won't take long, so **let's get going...**

About You

We just need some basic information about you (the applicant) to get started.

First Name * John

Last Name * Smith

Phone Number * 0987654321

Email Address * jsmith@gmail.com

Residential Address

Product Selection

Please select the product categories you are interested in.

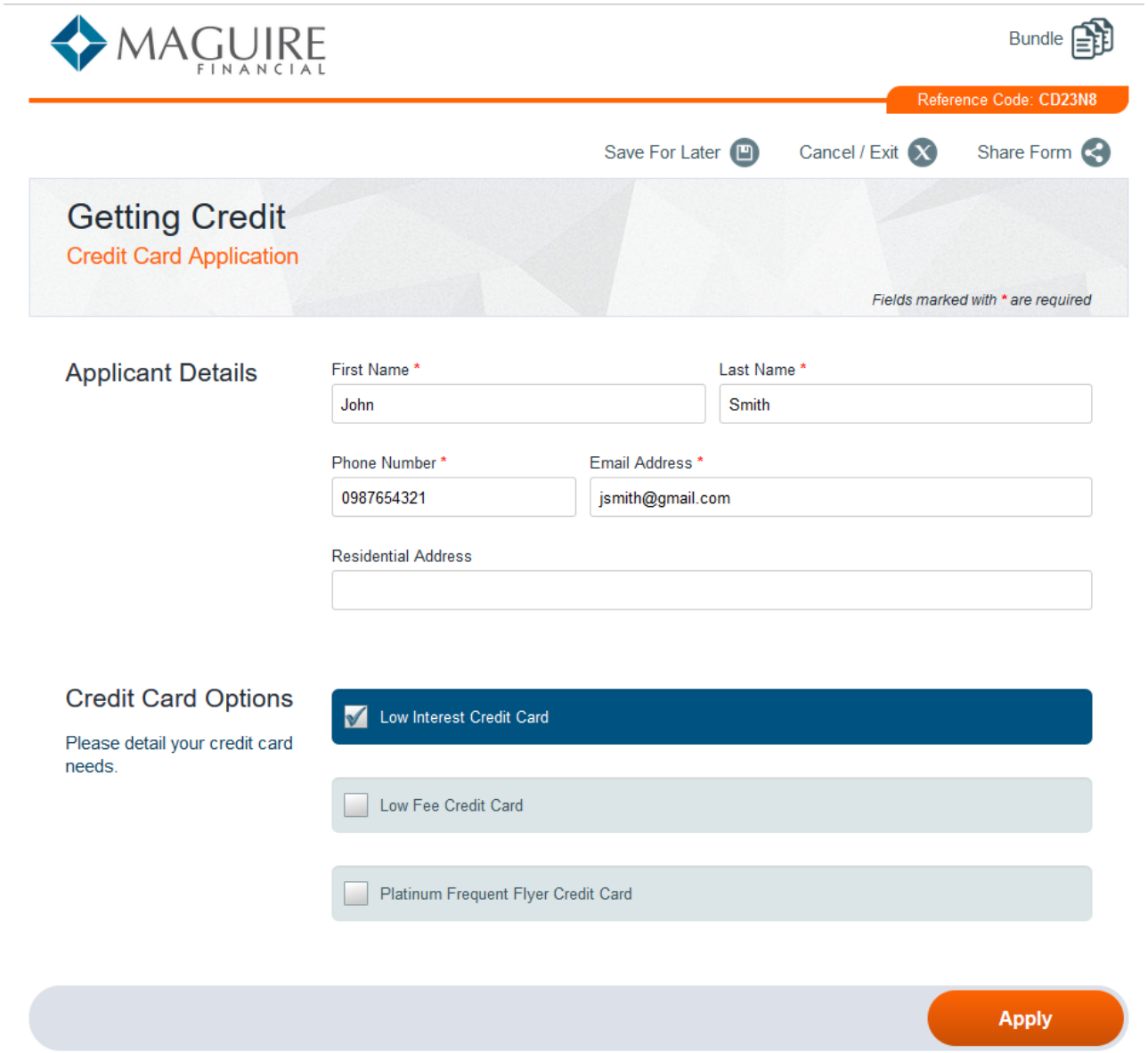
- [x] Credit Cards
- [x] Insurance

Submit

The user selects one or several available products, such as Credit Cards, Insurance or both, and clicks Submit. Manager starts a new collaboration job, which is immediately processed and goes to the Additional Products step.

Manager generates the following tasks:

1. The Customer Onboarding - Credit Card form, if the user selected the Credit Card check-box. In this case, the precondition is `"preCondition": "$formDataMap.productCreditCards == 'true'"`.



2. The Customer Onboarding - Insurance form, if use user selected the Insurance checkbox. In this case, the precondition is `"preCondition": "$formDataMap.productInsurance == 'true'"`.

MAGUIRE FINANCIAL

Bundle

Reference Code: VNQVZ6

Save For Later | Cancel / Exit | Share Form

Getting Insured

Insurance Application

*Fields marked with * are required*

Applicant Details

First Name *: John

Last Name *: Smith

Phone Number *: 0987654321

Email Address *: jsmith@gmail.com

Residential Address:

Insurance Details

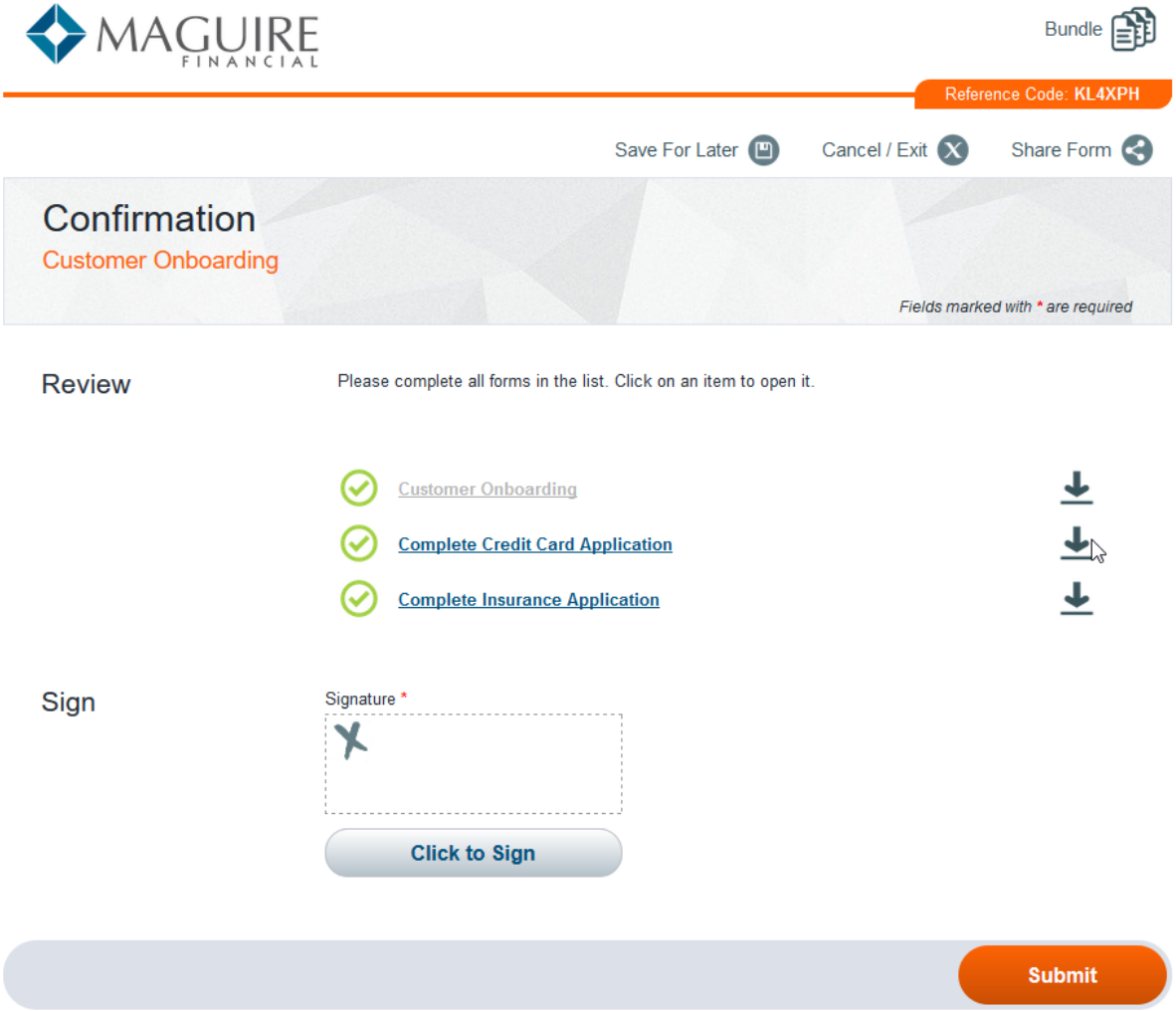
Please detail your insurance needs.

- [x] Home & Contents Insurance
- [ ] Income Insurance
- [ ] Life Insurance

Apply

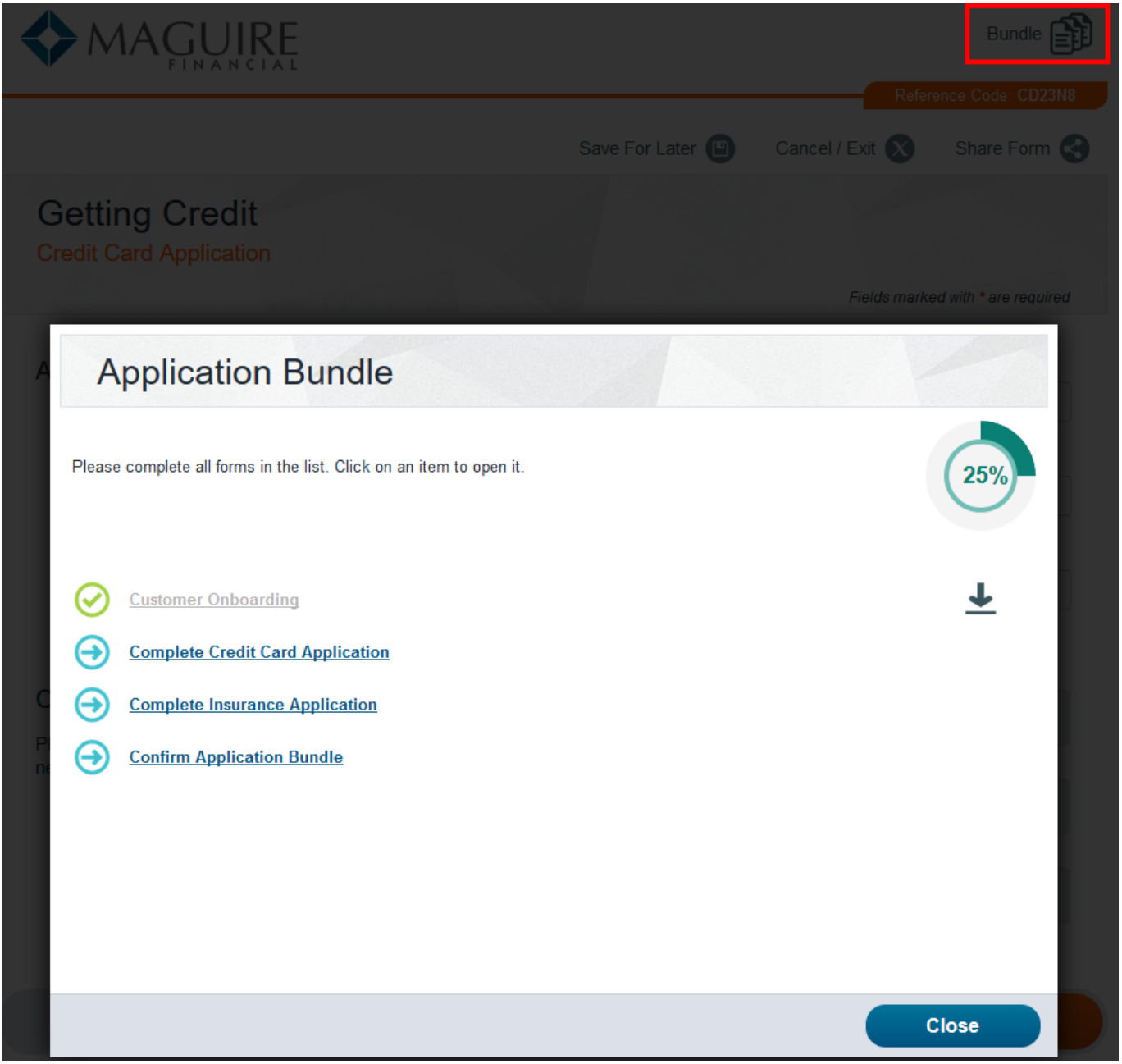
© Copyright Avoka Technologies 2015

3. The Customer Onboarding - Confirmation form to allow the user to complete and confirm the application with a signature.

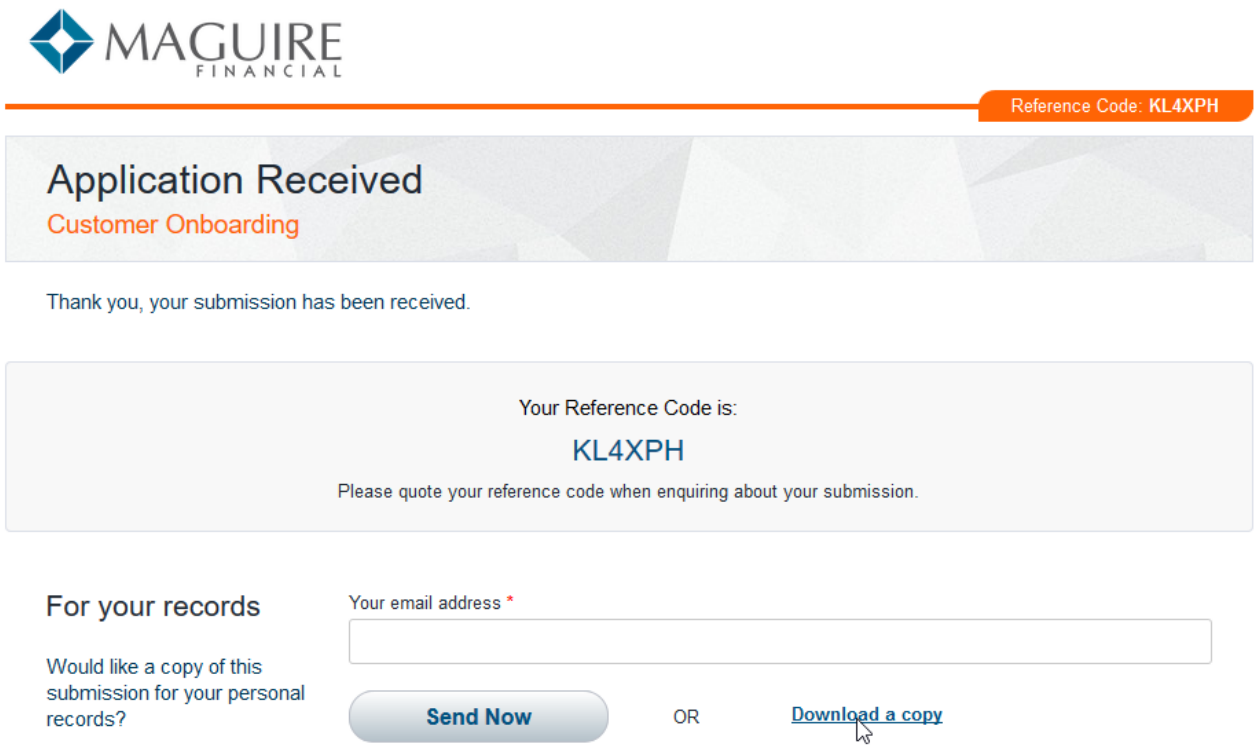


The Redirect Next functionality takes the user to the next form or task that has not be completed yet. A happy day scenario opens the credit card form, and after submitting it, the next form will be the insurance form, followed with the confirmation page.

The Maguire template has a form bundle option, the Bundle icon, which you can tun on or off. Click Bundle to open the Application Bundle navigation window to move to other forms in the bundle in a different to the Redirect Next order.



The confirmation page list the other forms (triage, credit card and insurance) so a user can download and review the receipts. At this stage they click on the link and go back and edit / modify any of the other forms in the Additional Product steps, such as the credit card and insurance. Once the user is happy with the forms they can sign and submit. The Additional products step has completed and the forms cannot be edited. The Customer Onboarding - Confirmation forms job action does not have redirectNext set. It goes to Application Received screen where they can download a consolidated receipt containing the Triage and Additional Product forms.



Click download a copy to view the PDF report.

For this example, you need to configure forms with the following form codes:

- Customer Onboarding : onboard-customer
- Customer Onboarding - Confirmation : onboard-confirmation
- Customer Onboarding - Credit Cards: onboard-credit-cards
- Customer Onboarding - Insurance : onboard-insurance

You must create a Customer Onboarding Job - v1 collaboration job and link it to the Customer Onboarding form, which is the start or triage form. The job definition is shown below:

```
{
  "jobDetails": {
    "name": "Customer Onboarding Job",
    "processSubmitImmediate": true,
    "version": "4.3.0"
  },
  "steps": [
    {
      "name": "Customer Onboarding",
      "type": "start",
      "actions": [
        {
          "name": "Customer Submission",
          "type": "Job Form Start",
          "redirectNext": true
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Additional Products" }
      ]
    },
    {
      "name": "Additional Products",
      "type": "",
      "shareExtractData": true,
      "allFormsEditable": true,
      "showPreviousForms": true,
      "redirectNext": true,
      "actions": [
        {
          "name": "Credit Cards Application",
          "type": "Job Task Assign",
          "preCondition": "$formDataMap.productCreditCards == 'true'",
          "redirectNext": true,
          "properties": [
            { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
            { "name": "Task Form Code",  "value": "onboard-credit-cards" },
            { "name": "Task Message",  "value": "Please complete the Credit Card Applic-
ation form." },
            { "name": "Task Subject",  "value": "Complete Credit Card Application" },
            { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
```

```
            { "name": "Task Type",  "value": "Anonymous" }
          ]
        },
        {
          "name": "Insurance Application",
          "type": "Job Task Assign",
          "preCondition": "$formDataMap.productInsurance == 'true'",
          "redirectNext": true,
          "properties": [
            { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
            { "name": "Task Form Code",  "value": "onboard-insurance" },
            { "name": "Task Message",  "value": "Please complete the Insurance Applic-
ation form." },
            { "name": "Task Subject",  "value": "Complete Insurance Application" },
            { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
            { "name": "Task Type",  "value": "Anonymous" }
          ]
        },
        {
          "name": "Appliction Confirmation",
          "type": "Job Task Assign",
          "preCondition": "$formDataMap.productInsurance == 'true' || $formDataMap.-
productCreditCards == 'true'",
          "properties": [
            { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
            { "name": "Task Form Code",  "value": "onboard-confirmation" },
            { "name": "Task Message",  "value": "Please sign to complete Application
bundle." },
            { "name": "Task Subject",  "value": "Confirm Application Bundle" },
            { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
            { "name": "Task Type",  "value": "Anonymous" }
          ]
        },
        {
          "name": "Review Wait",
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Delivery" }
      ]
```

```
    },
    {
      "name": "Application Delivery",
      "type": "",
      "actions": [
        {
          "name": "Application Delivery",
          "type": "Job Delivery",
          "properties": [
            { "name": "Delivery Mode",  "value": "All Submissions" }
          ]
        },
        {
          "name": "Application Delivery Wait",
          "type": "Job Delivery Wait"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Completed" }
      ]
    },
    {
      "name": "Application Completed",
      "type": "endpoint"
    }
  ]
}
```

Download

You can download the complete code example here.

Next, learn more about dynamic customer onboarding collaboration jobs.

Dynamic Customer Onboarding Collaboration Jobs

A dynamic customer onboarding collaboration job implements a simple customer onboarding dynamic form bundle. The bundle consists of a number of separate forms that are filled out by a single user in the one step. This is the enhancement of the standard customer onboarding form bundle with dynamic preconditions.

The standard form bundle allows a user to fill a triage form and submits it. On submit, form data extracts are created and used to evaluate preconditions when the bundle step is created. Dynamic bundles allow the preconditions to be re-evaluated after each form in the bundle is submitted.

Next, learn about step expiry.

Maestro Customer Onboarding Collaboration Jobs

A customer onboarding collaboration job implements the simplest customer onboarding form bundle using a Journey Maestro form. It uses the same configuration and functionality as described in customer onboarding Collaboration Jobs.

Download

You can download the complete code example here.

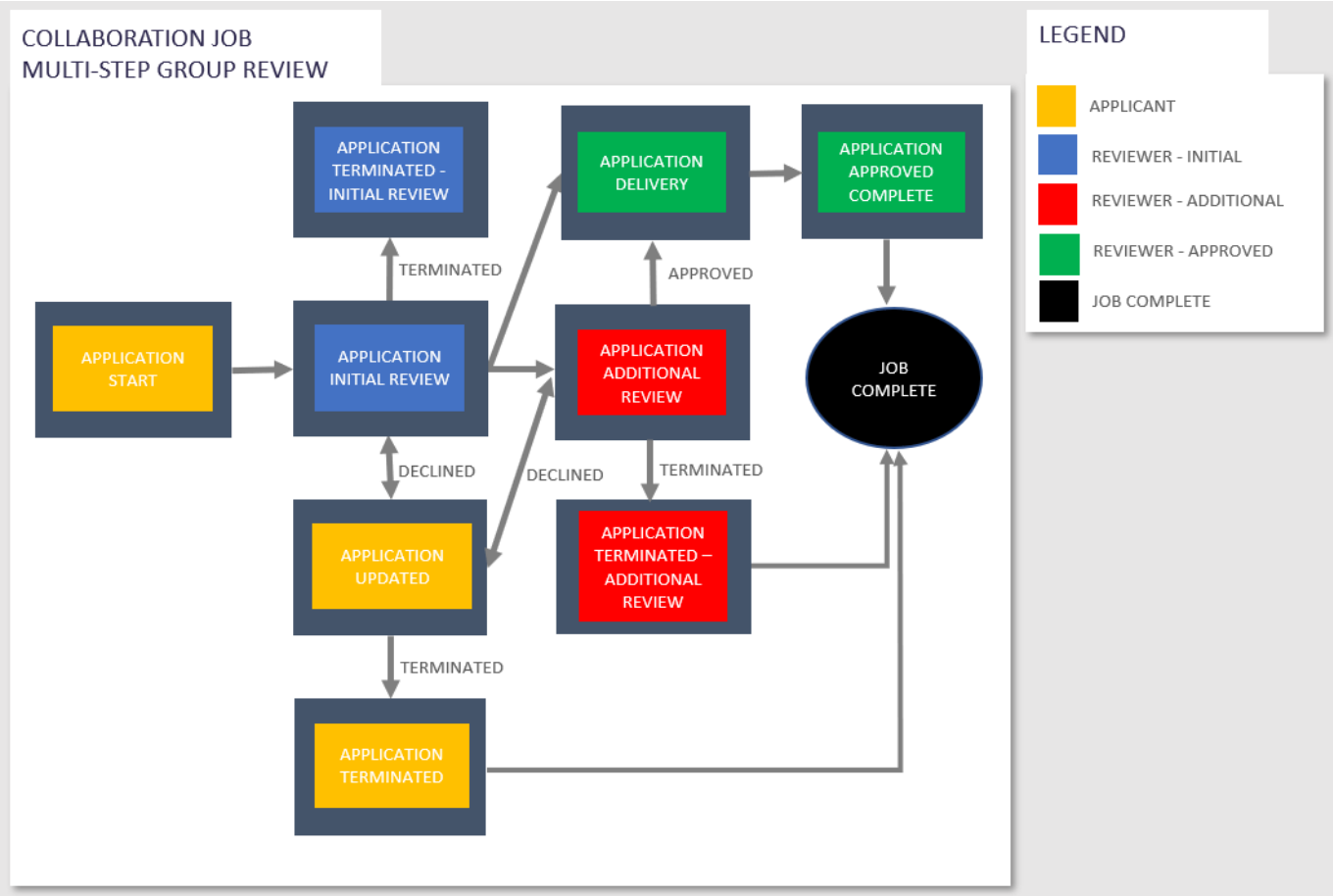
The forms require Maestro version 5.1 and higher. Forms and collaboration jobs should run on Manager version 5.0 and higher. To start the application, you can run the Maestro Onboarding - Triage form.

Next, learn more about dynamic customer onboarding Collaboration Jobs.

Multi-Step Group Review Collaboration Jobs

After a user submits an application form, it has to be reviewed and a decision has to be made whether to approve or decline the application. A review workflow can be as simple as one step review or a multi-step group review. The later is a collaboration job that implements the simplest group review process, so we will use to illustrate how a collaboration job works. TheJob Example 2 implements the review workflow using the Form - Job Example 2 form and the Review and Approval - Multi Step Group Review job controller, which is the job service template.

The review workflow is shown below.



Configure a Form

Below is the Job Example 2 form, which is built using our Composer product.

MAGUIRE FINANCIAL

Reference Code: **LGXDLR**

Personal Details | **Request**

Save For Later

Personal Details:

Job Example 2:

*Fields marked with * are required*

Details

First Name *

Anne

Last Name *

Applicant

Email *

testteam@avoka.com

Address

Address Line 1

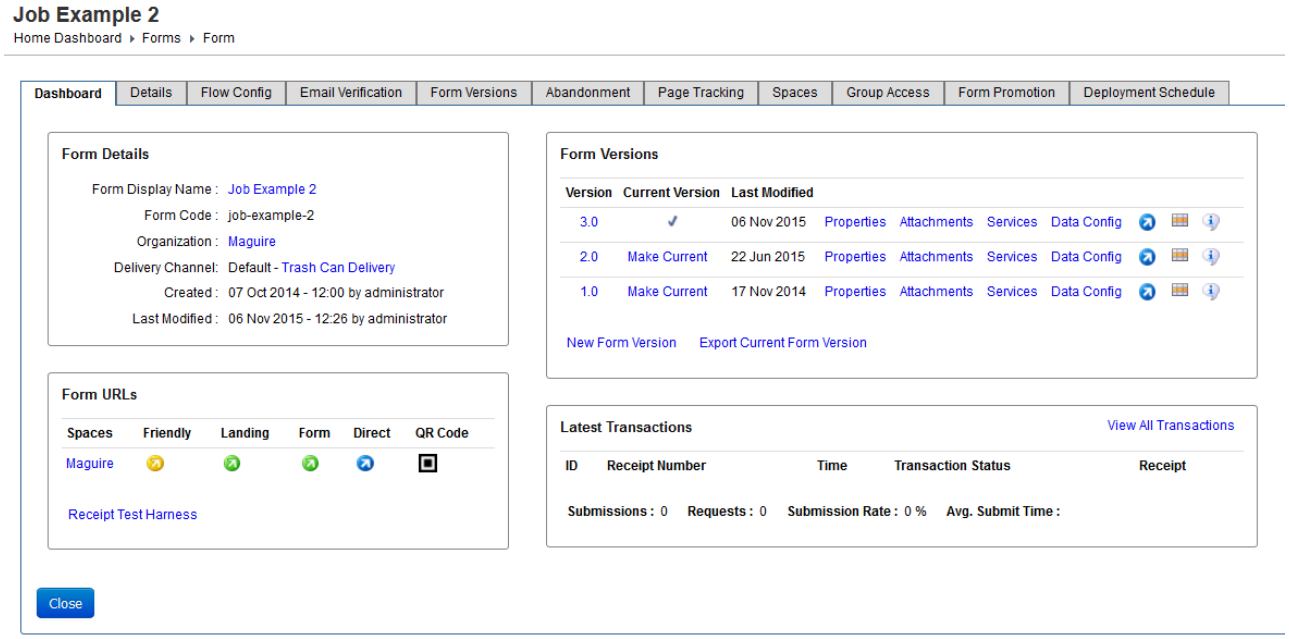
Address Line 2

City | State | Zipcode

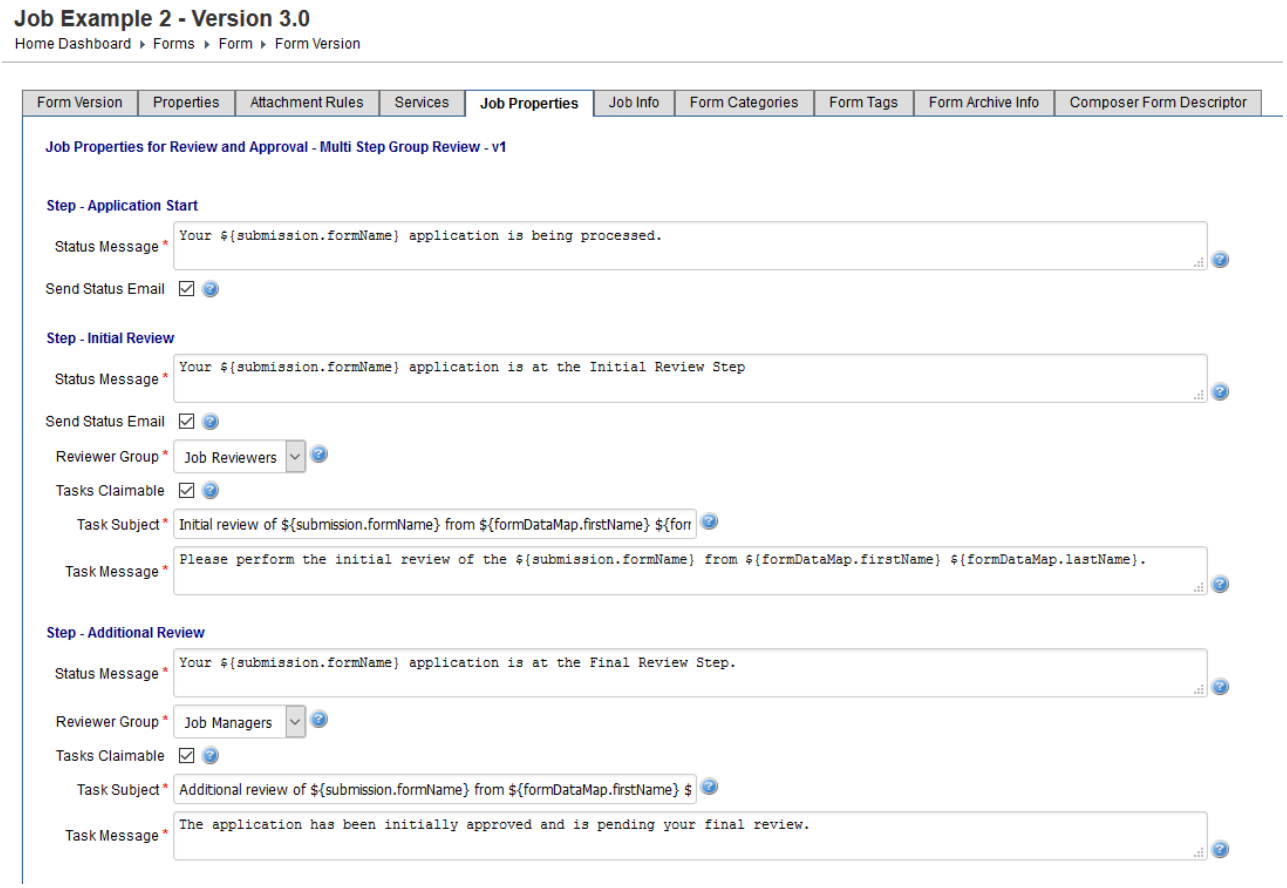
Continue

Configure Manager

1. Open the Job Example 2 form.



2. Select the form version 3 because it has the following features:
   - The visibility and editability rules based on the collaboration job step.
   - The radio buttons.
   - The job service template.
3. Click Services of the latest form version to configure its job controller service and select the Review and Approval - Multi Step Group Review - v1 from the Job Controller Service drop-down list as the template.
4. Select the Job Properties tab to configure the Reviewer Groups as shown below. Properties can use data extracts and other model elements.



5. Check the Review and Approval - Multi Step Group Review job controller's definitions, as shown below.

```
{
  "jobDetails": {
    "name": "Review and Approval - Multi Step Group Review",
    "version": "4.2.0"
  },
  "properties": [
    { "name": "Task Form Code",  "value": "$func.startFormCode()" }
  ],
  "steps": [
    {
      "name": "Start",
      "type": "start",
      "actions": [
        {
          "name": "Handle Submission",
          "type": "Job Form Start",
```

```
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty
('Start Send Email')" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Start
Process Message')" }
          ]
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Initial Review" }
      ]
    },
    {
      "name": "Initial Review",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty
('Initial Review Send Email')" },
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Ini-
tial Review Process Message')" },
            { "name": "Task Assign Group",  "value": "$func.formProperty('Initial
Review Group')" },
            { "name": "Task Enable Claiming",  "value": "$func.formProperty('Ini-
tial Review Enable Claiming')" },
            { "name": "Task Message",  "value": "$func.formProperty('Initial Review
Task Message')" },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "$func.formProperty('Initial Review
Task Subject')" },
            { "name": "Task Type",  "value": "Review" }
          ]
        },
        {
          "name": "Handle Submission",
          "type": "Job Task Wait",
```

```
          "properties": [
            { "name": "Conditional Route Name",  "value": "#if (
$formDataMap.routeName == 'Approve' && $formDataMap.loanAmount >= 20000 ) Exceeds
Threshold #end" }
          ]
        }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "dis-
play": "false" },
        { "name": "Terminate",  "nextStep": "Terminated Initial" }
      ]
    },
    {
      "name": "Additional Review",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Addi-
tional Review Process Message')" },
            { "name": "Task Assign Group",  "value": "$func.formProperty('Addi-
tional Review Group')" },
            { "name": "Task Enable Claiming",  "value": "$func.formProperty('Addi-
tional Review Enable Claiming')" },
            { "name": "Task Message",  "value": "$func.formProperty('Additional
Review Task Message')" },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "$func.formProperty('Additional
Review Task Subject')" },
            { "name": "Task Type",  "value": "Review" }
          ]
        },
        {
          "name": "Handle Submission",
```

```
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Terminate",  "nextStep": "Terminated Additional" }
      ]
    },
    {
      "name": "Applicant Update",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Submission",  "value": "$fun-
c.startSubmission()" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Applic-
ant Update Process Message')" },
            { "name": "Task Assign User",  "value": "$func.startUser()" },
            { "name": "Task Message",  "value": "$func.formProperty('Applicant
Jpdate Task Message')" },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "$func.formProperty('Applicant
Jpdate Task Subject')" },
            { "name": "Task Type",  "value": "Review" }
          ]
        },
        {
          "name": "Handle Submission",
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Update",  "nextStep": "##PREVIOUS_STEP" },
        { "name": "Terminate",  "nextStep": "Terminated Applicant" }
      ]
    },
```

```
    {
      "name": "Application Delivery",
      "type": "",
      "actions": [
        {
          "name": "Application Delivery",
          "type": "Job Delivery"
        },
        {
          "name": "Application Delivery Wait",
          "type": "Job Delivery Wait"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Approved Complete" }
      ]
    },
    {
      "name": "Approved Complete",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty
('Approved Complete Send Email')" },
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty
('Approved Complete Process Message')" }
          ]
        }
      ]
    },
    {
      "name": "Terminated Initial",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
```

```
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty
('Terminated Initial Send Email')" },
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Ter-
minated Initial Process Message')" }
          ]
        }
      ]
    },
    {
      "name": "Terminated Additional",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty
('Terminated Additional Send Email')" },
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Ter-
minated Additional Process Message')" }
          ]
        }
      ]
    },
    {
      "name": "Terminated Applicant",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty
('Terminated Applicant Send Email')" },
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty
```

```
('Terminated Applicant Process Message')" }
          ]
        }
      ]
    }
  ]
}
```

Configure Users and Groups

This example uses the following users, which are the actors in the job definition above.

Group	Login	Name	Description
Job Applicants	anne	Anne Applicant	Anne is the applicant.
Job Reviewers	roger	Roger Reviewer	Roger looks after the Initial Review step.
Job Managers	mike	Mike Manager	Mike looks performs the Additional Review.
Job Coordinators	cat	Cat Coordinator	Cat looks after the collaboration job as a whole and needs to track where collaboration jobs are up to using the Reviews page.

Route Options

The review workflow has the following route options at each user's step.

Step	Group	Login	Description
Start	Job Applicants	anne	Anne starts the collaboration job (Start step) by the initial submission.
Initial Review	Job Reviewers	roger	Roger can either: Approve: A Conditional Route Rule checks if the loan amount is: • Below or equal to threshold the route stays as Approved and is routed to Application Delivery then to the Approved Complete

			endpoint. • Above the route is changes to Exceeds Threshold and is routed to the Additional Review.
Additional Review	Job Managers	mike	Decline: Routed to the Applicant Update Step Terminate: Routed to the Terminated Initial endpoint Mike can either Approve: Routed to Application Delivery step then to the Approved Complete endpoint, Decline: The reviewer enters feedback and the job is routed to the Applicant Update Step Terminate: Routed to the Terminated Additional endpoint
Applicant Update	Job Applicants	anne	Anne can read the feedback from the reviewer, plus she can: Update: Anne can update the form and the route it back to the review step that declined the application. Terminate: Routed to the Terminated Applicant endpoint.

You can see that the Roger's Approve action, the Conditional Route Rule, is the Conditional Route Name. The following action property belongs to the Task Wait Service, which defines a rule that is applied with the collaboration job on the Manager server side. The rule is shown below:

```
{ "name": "Conditional Route Name",  "value": "#if ( $formDataMap.routeName == 'Approve'
&& $formDataMap.loanAmount >= 20000 ) Exceeds Threshold #end" }
```

The following example shows the listed routes where the Exceeds Threshold has the display attribute set to false, so this route is not put in the Available Routes, as shown below:

```
"routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display":
"false" },
        { "name": "Terminate",  "nextStep": "Terminated Initial" }
      ]
```

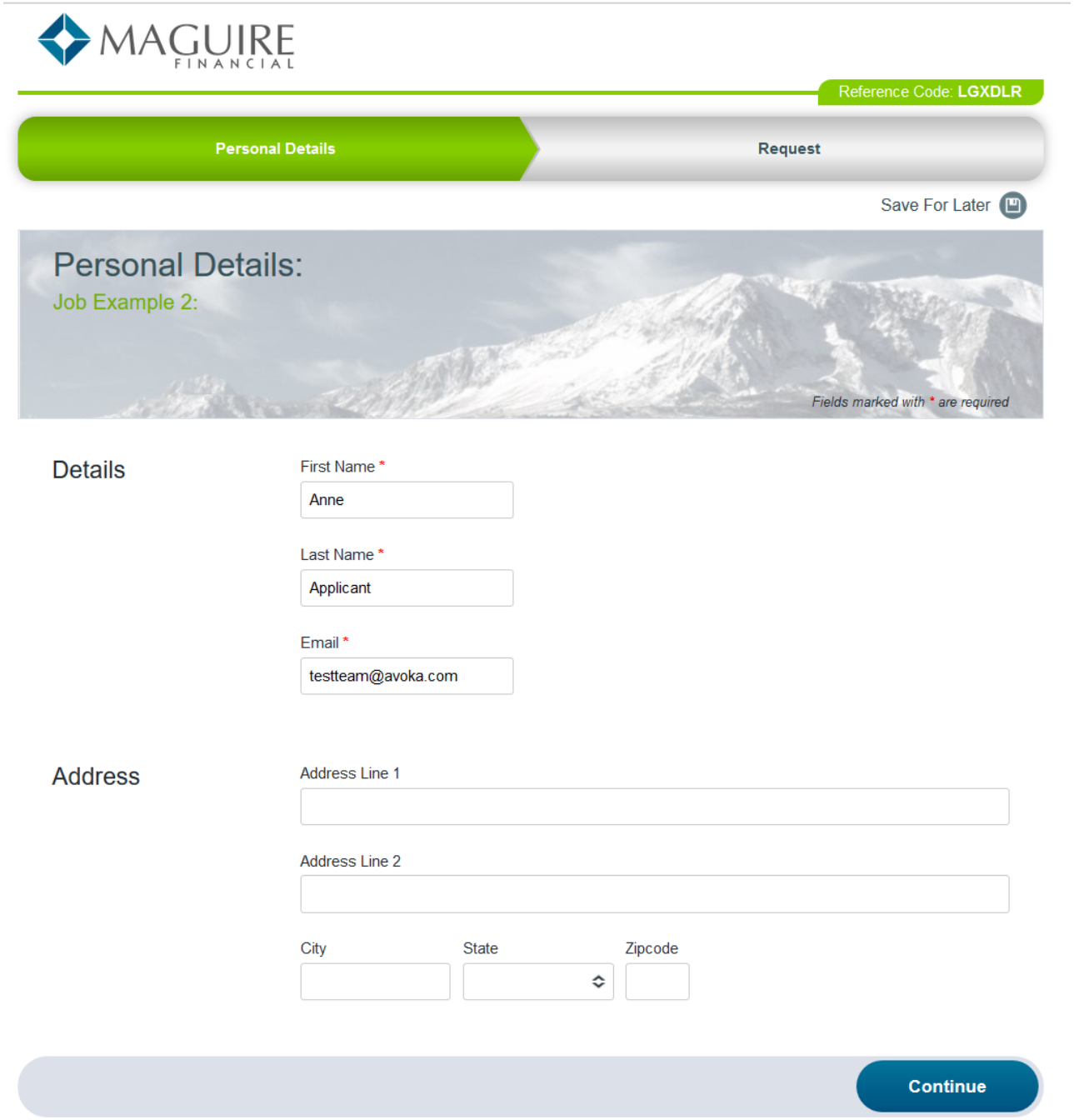
The Anne's Update action is a previous step. The following shows the routes for the Applicant Update step. When the Initial Review declines it will be sent back to the Initial Review, and same with the Additional Review.

```
"routes": [
        { "name": "Update",  "nextStep": "##PREVIOUS_STEP" },
        { "name": "Terminate",  "nextStep": "Terminated Applicant" }
      ]
```

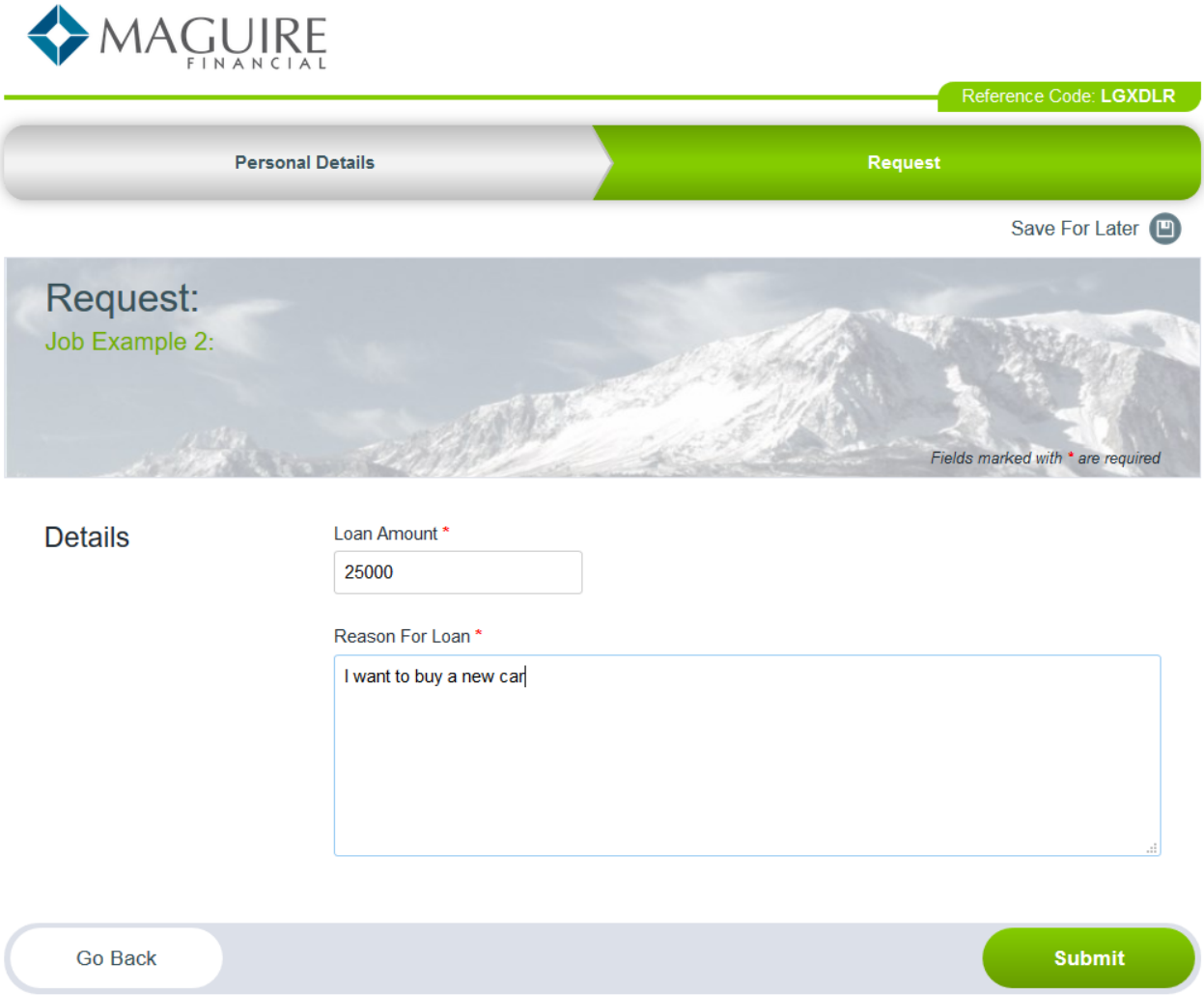
Now, you can run the example and see how collaboration job works.

Form - Anne

1. Log on to Journey Space as Anne.
2. Fills in the personal details.



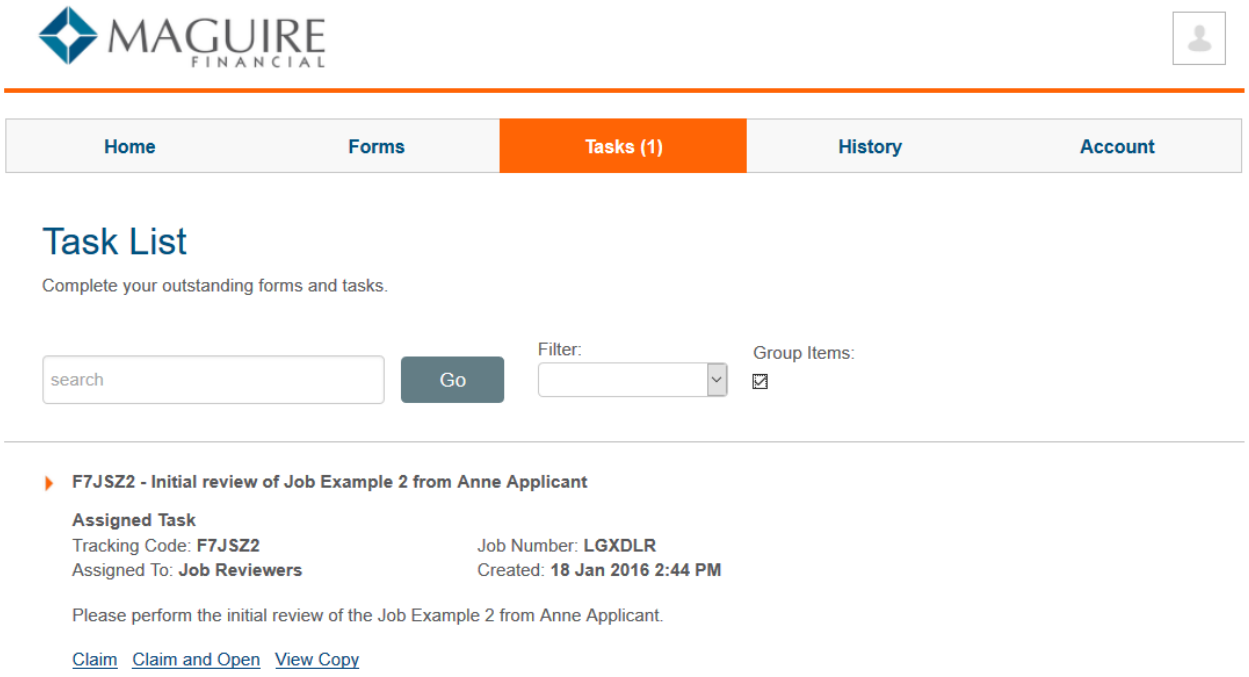
3. Enter the desired loan amount and the reason for the loan.



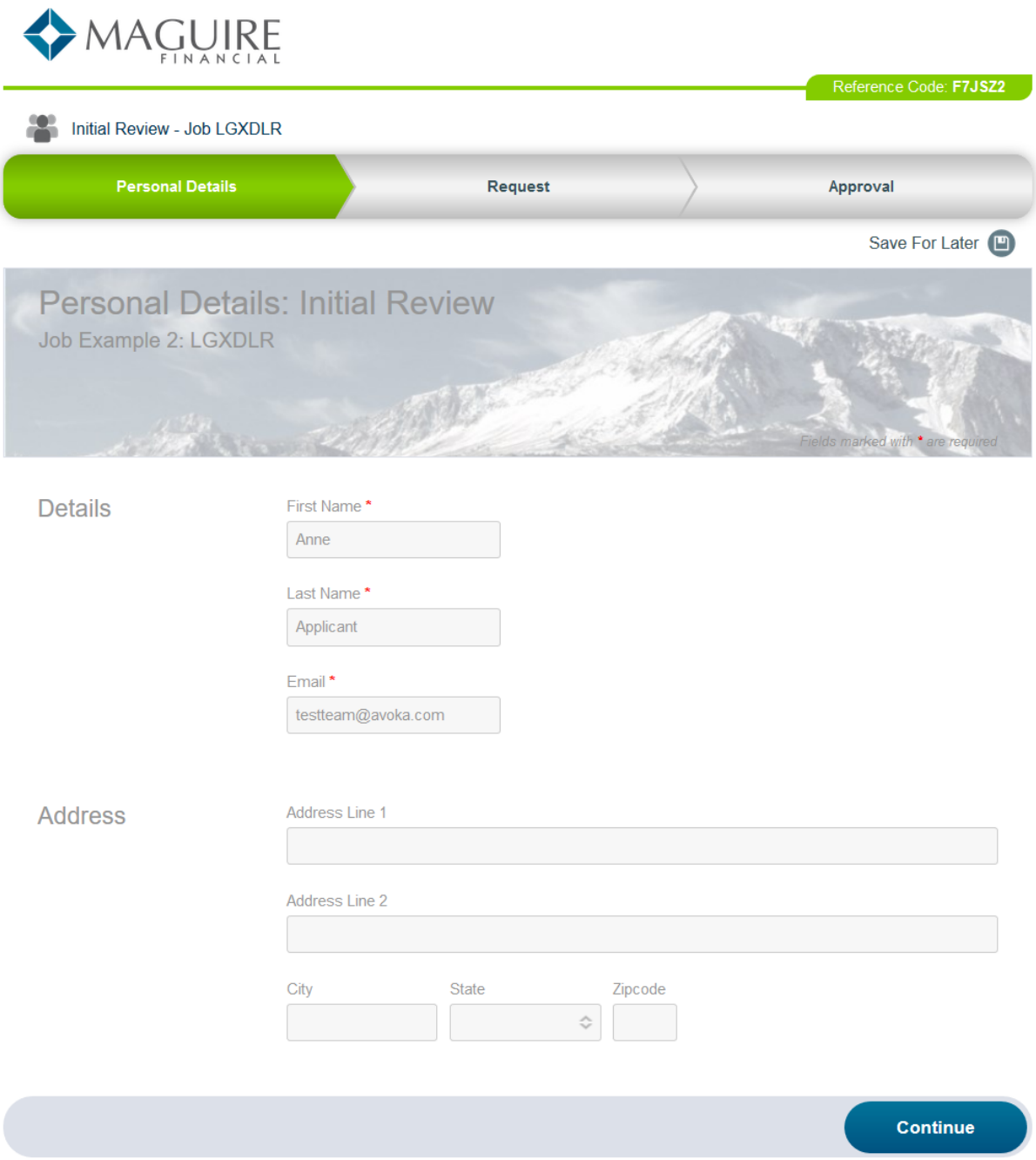
4. Click Submits.

Initial Review - Roger

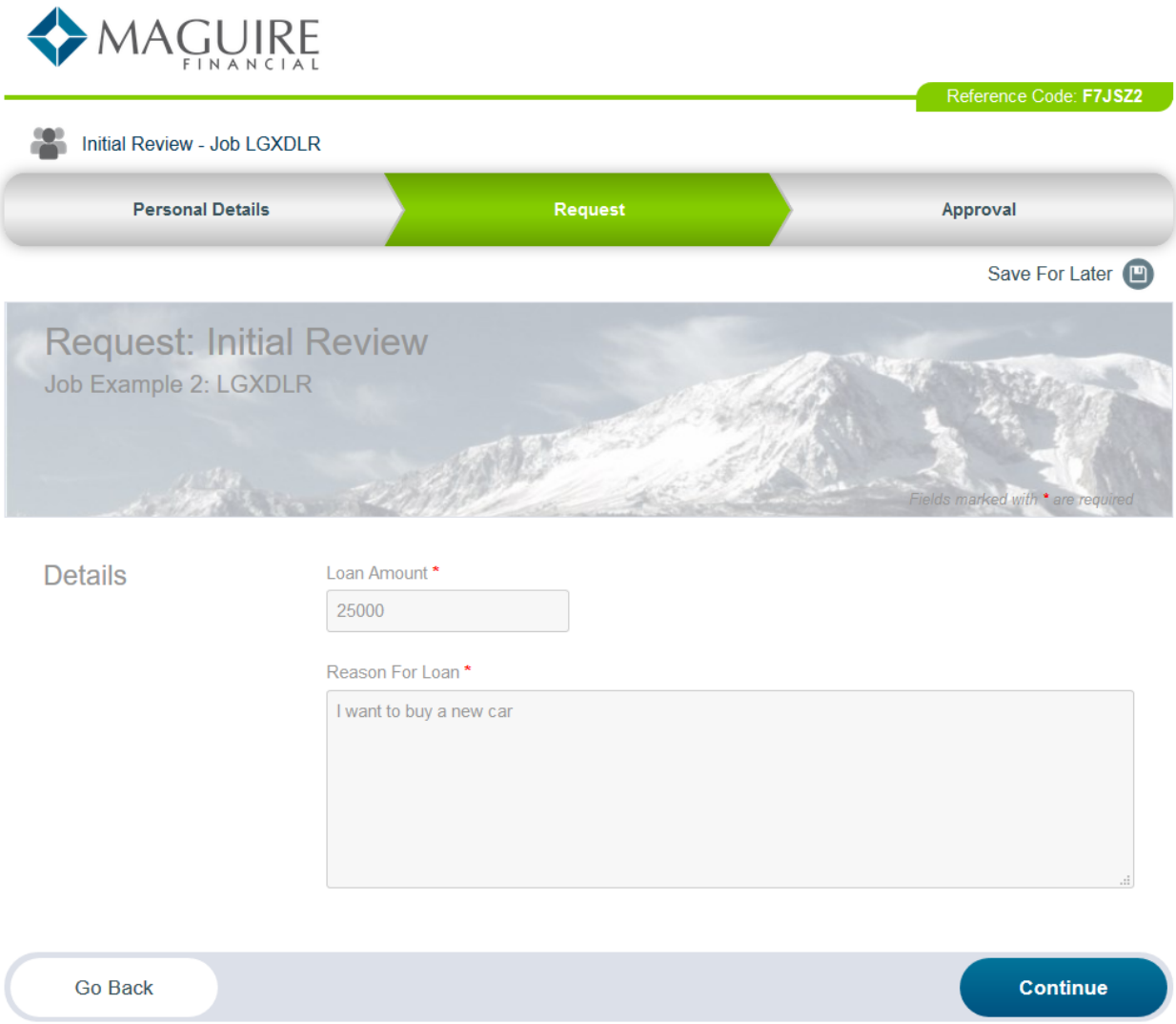
1. Log on to Journey Space as Roger.
2. Check the group task, claim it and open it for the review.



3. Review the applicant details' section.



4. Review the application loan amount and the reason for the loan.



5. The application doesn't have sufficient information to justify the loan, so Roger decides to decline the application. He selects the Decline radio button and adds a comment for Anne "What car do you want?".

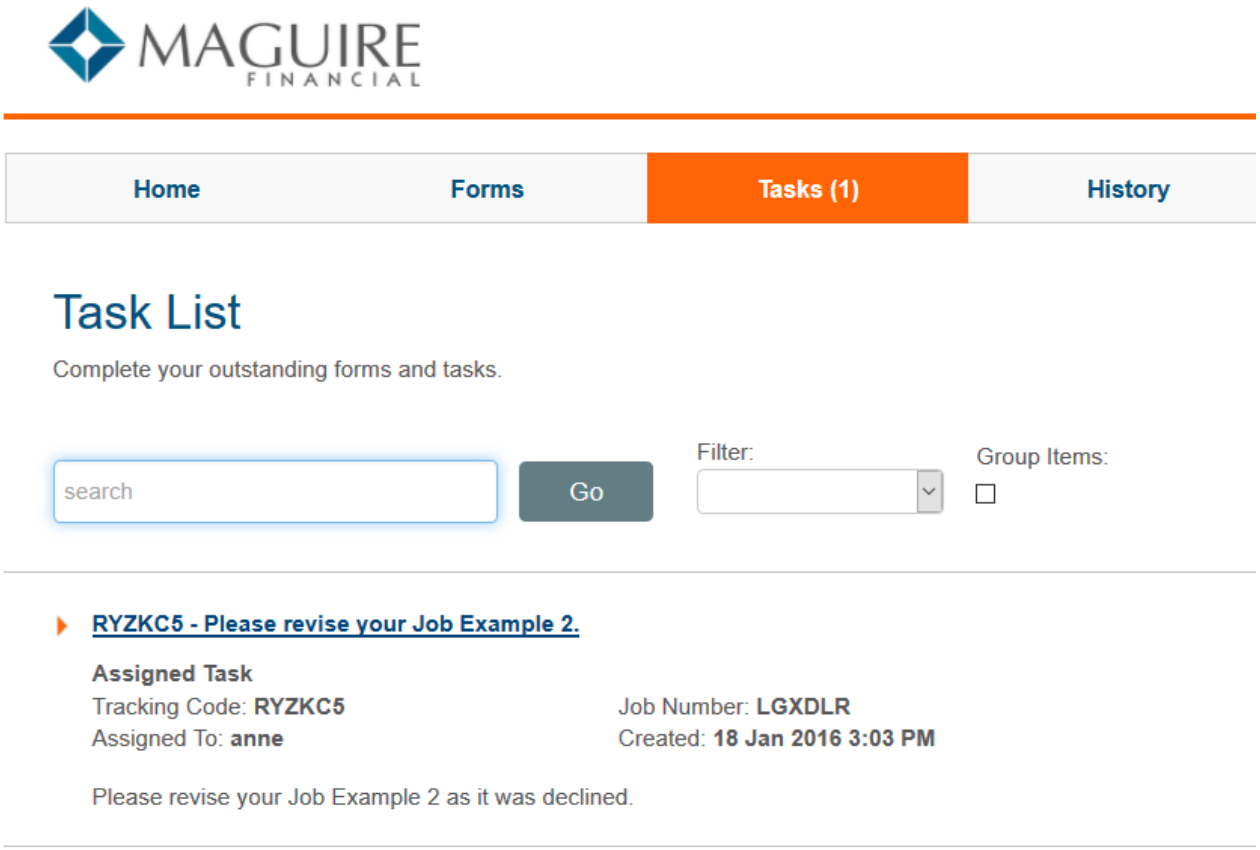
6. The application goes back to Anne, so she can update or terminate (cancel) it.

NOTE

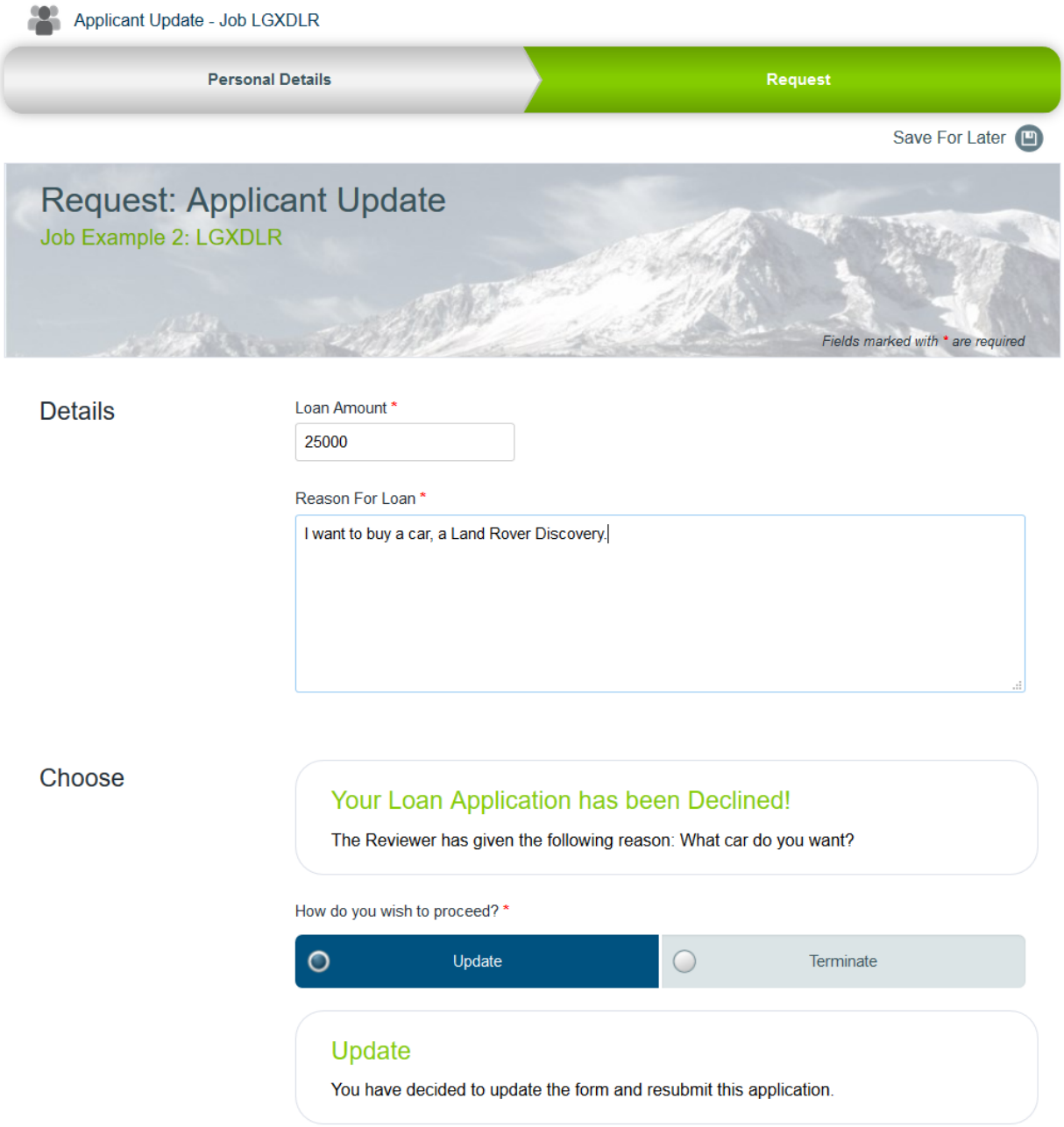
The routes in the form use radio buttons. If you use more that one radio button groups to control the Route Name, see collaboration job configuration in Composer.

Applicant Update - Anne

1. Log on to Journey Space as Anne.
2. Check the tasks assigned to her.



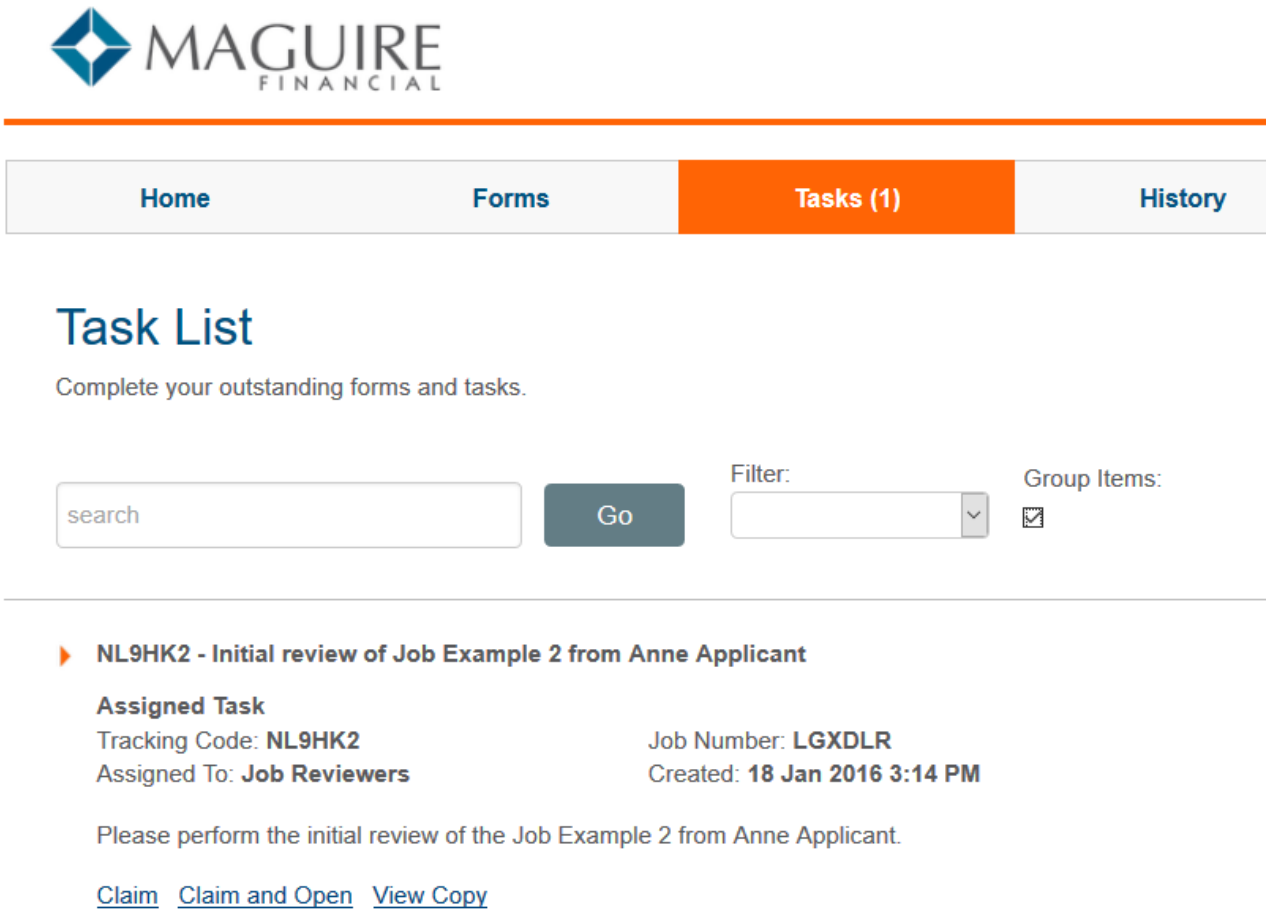
3. Open the form and select the Request page that now has the Choose section so Anne can either update or terminate (cancel) the application.



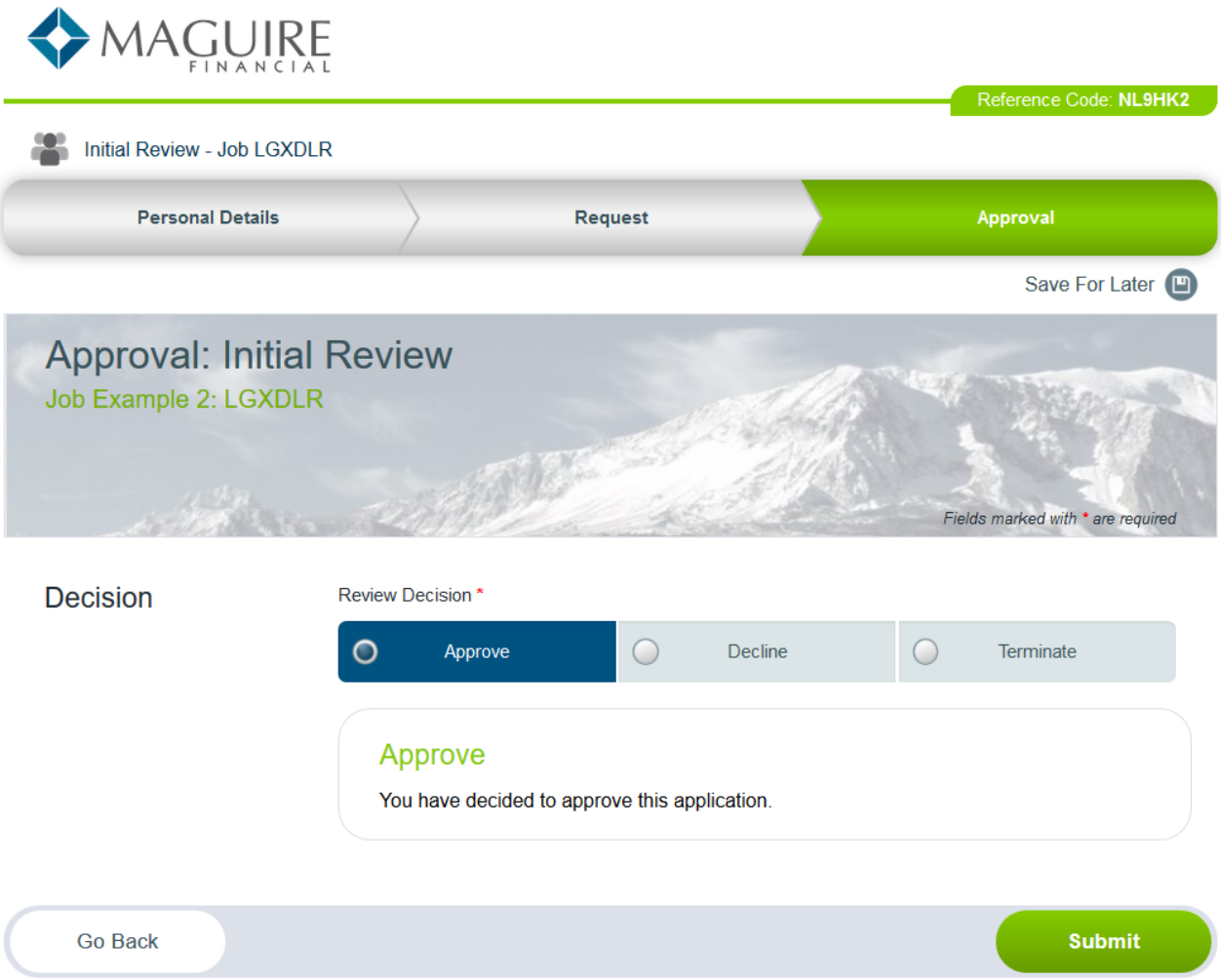
4. Select the Update radio button to provide the required additional information, such as the new car make in the Reason For Loan field.
5. Clicks Submit.

Initial Review - Roger

1. Log on to Journey Space as Roger.
2. View the updated task.



3. Click Claim and Open to view the Request - Reason For Loan section. The updated comment is sufficient, so the application can be approved.

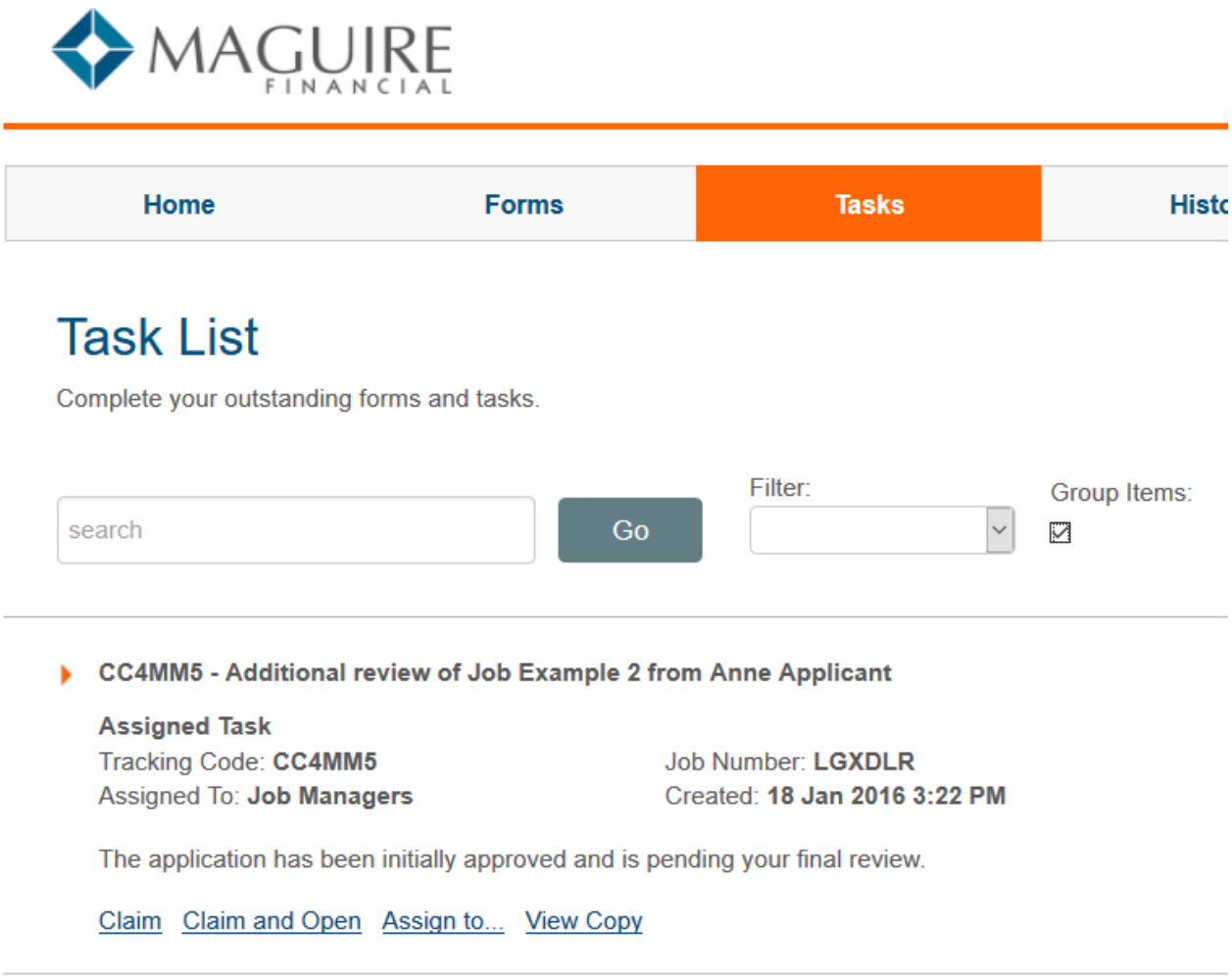


4. Select the Approve radio button to approve the application.

Additional Review - Mike

Because the loan amount was greater that $20,000, the collaboration job is routed to Mike.

1. Log on to Journey Space as Mike.
2. View the task.



3. Click Claim and Open and review the Personal Details and Request sections.
4. Navigate to the Approval section and select the Approve radio button.
5. The application is approved now so the applicant loan will be served.

Below is the summary of what is visible and editable in each application form section.

Section	Start	Initial Review	Additional Review	Application Update
User Details	Editable	Read Only	Read Only	Editable
Request	Editable	Read Only	Read Only	Editable
Request - Choose	Hidden	Hidden	Hidden	Visible
Approve	Hidden	Visible	Visible	Hidden

This concludes the example.

Download

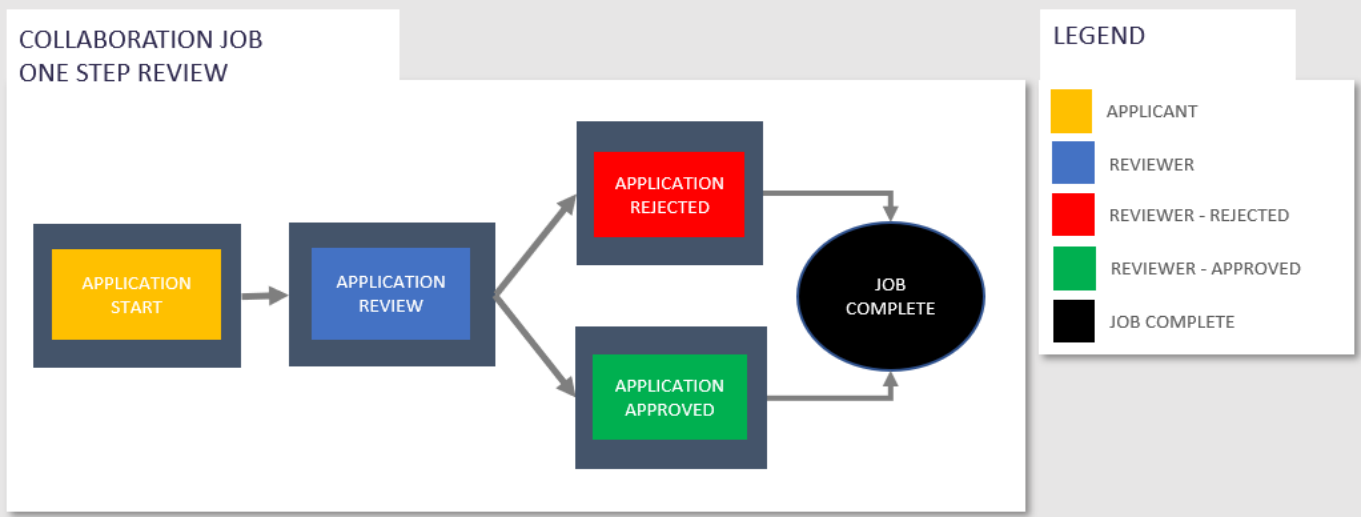
You can download the complete code example here.

Next, learn about customer onboarding collaboration jobs.

One Step Review Collaboration Job

After a user submits an application form, it has to be reviewed and a decision has to be made whether to approve or decline the application. A one step review collaboration job implements the simplest review process, so we will use to illustrate how a collaboration job works. TheJob Example 1 implements the review workflow using the Form - Job Example 1 form, which contains only the fields that are used in the collaboration job. It also explains how attachments are reviewed using the In Form Attachment widget.

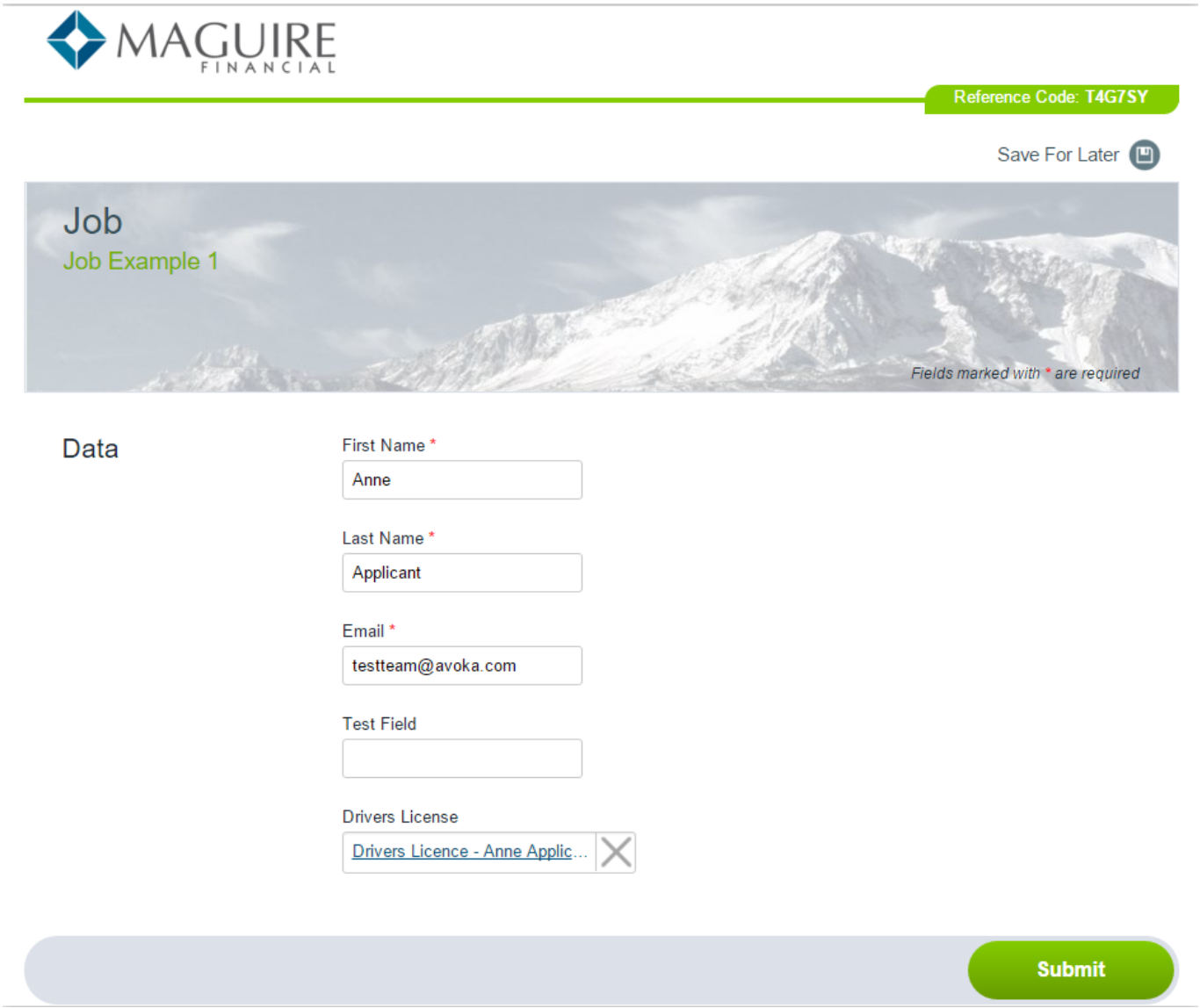
The review workflow is shown below.



1. A user fills in an application form and submits it.
2. The form goes to the Application Review step where it is reviewed by a member of the Job Reviewer group.
3. The reviewer may Accept or Reject the application:
   - If it is rejected, the collaboration job moves to the Application Rejected endpoint.
   - If it is accepted, the collaboration job progresses to the Application Approved endpoint and the form is delivered (this step is not shown) .

Configure a Form

Below is the Job Example 1 form, which is built using our Composer product.



The form has the following mandatory fields, which are data extracts, which are populated when an authenticated user is logged into the form space. Forms can use the same Job Controller as long as they have the same data extract fields.:

- First Name - the Text Field
- Surname - the Text Field
- Email - the Email Field

The form also has the optional fields:

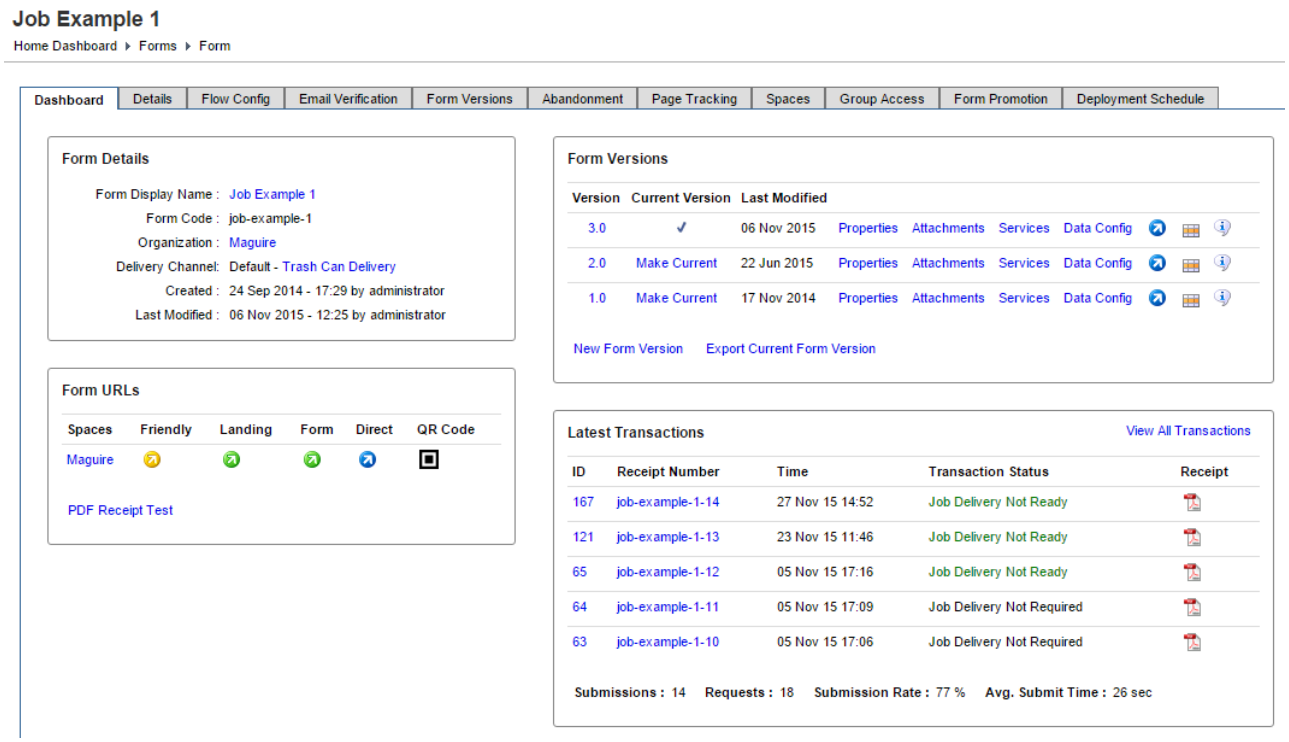
- Test Field - the Text Field.
- Drivers License - the Inform Attachment Field, which is configured as read-only in the Application Review step.
- Route Name - the Collaboration Routes Dropdown list.

NOTE

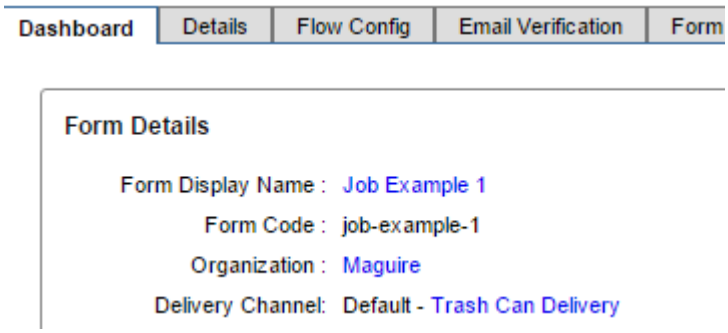
When the form is opened by the Applicant the Route Name Dropdown is hidden.This is because the Job hasn't started until this form has been submitted. As the form is not a Task that part of a Job and the job context isn't populated in the form data and hence the Available Routes Names are empty. The Route Name Dropdown field will appear when the form is a Task at the Application Review step Available Routed Names Approve and Reject associated with the step are listed in the form data.

Configure Manager

1. Open the Job Example 1 form.

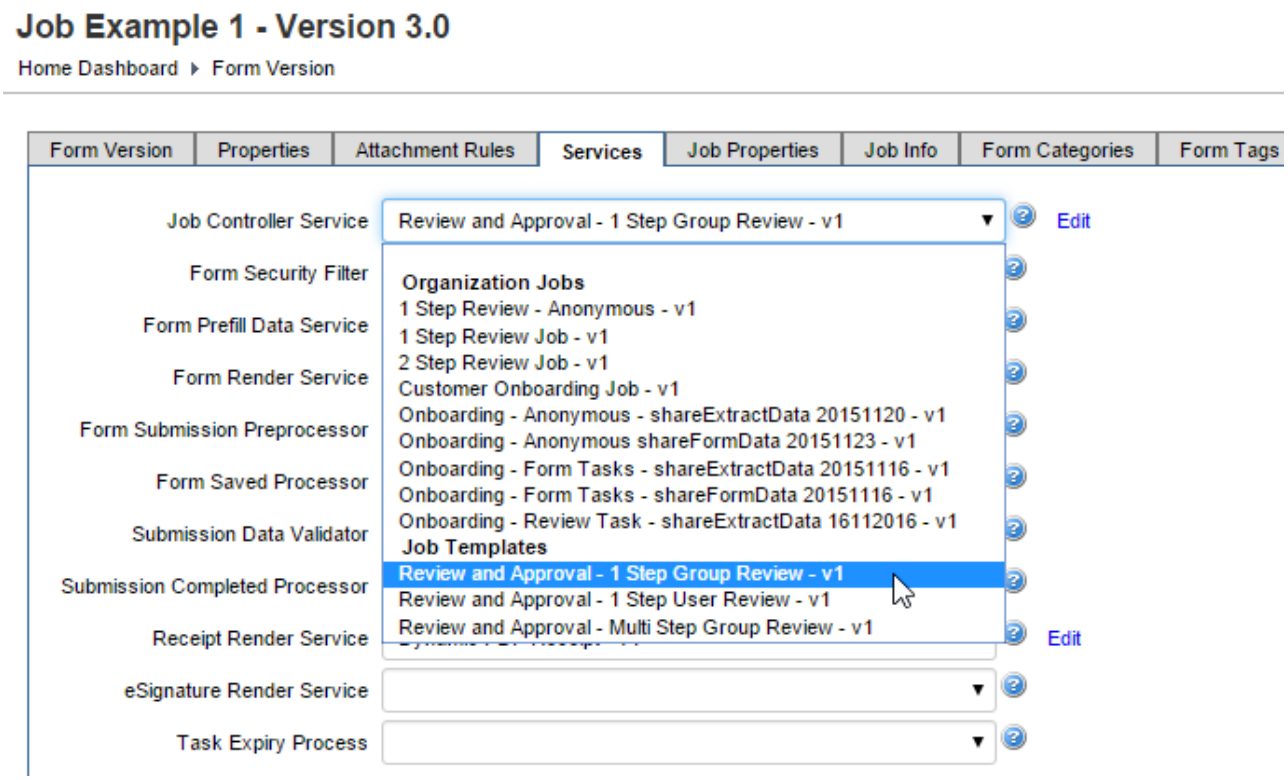


2. Confirm that the delivery channel in the Form Details Delivery Channel is set to the Trash Can Delivery.



3. Click Services of the latest form version to configure its job controller service.

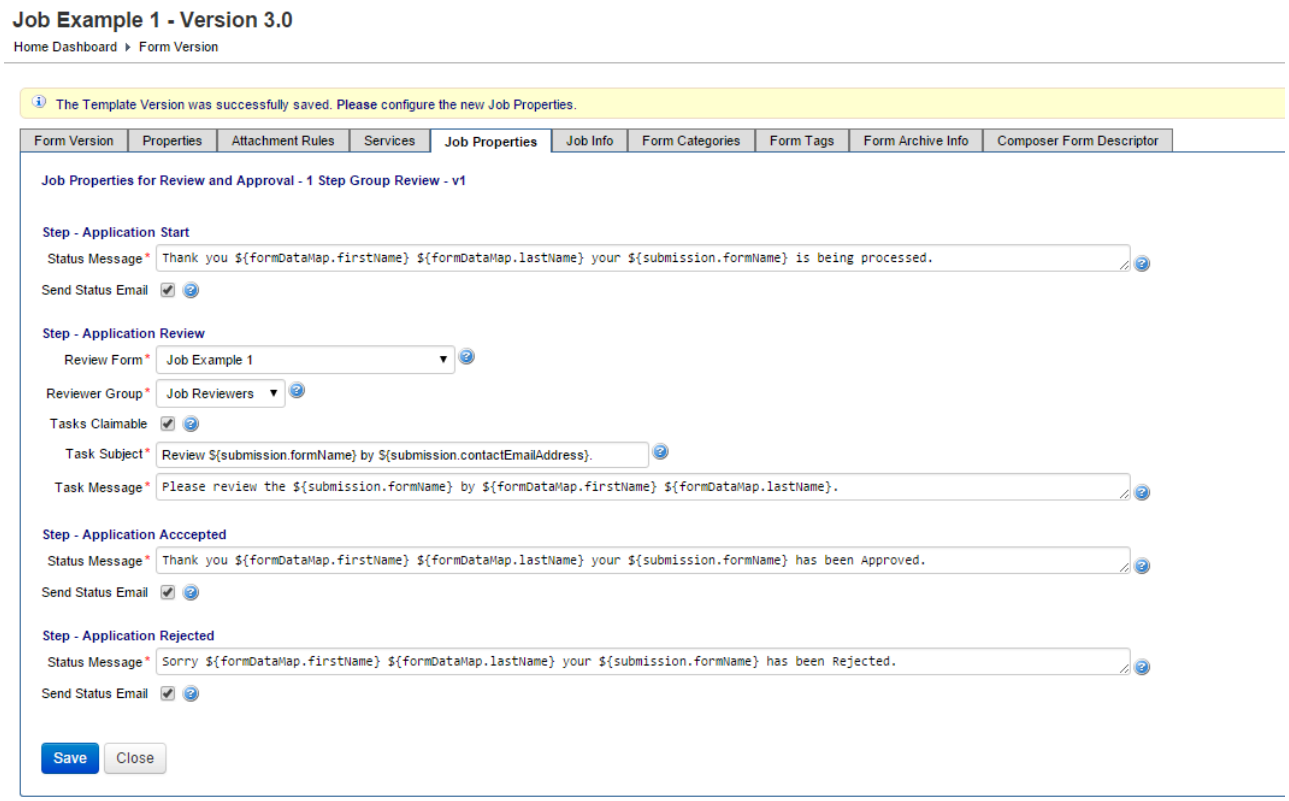
4. Select the Review and Approval - 1 Step Group Review service as a job template from the Job Controller Service dropdown list and click Save.



NOTE

The Job Example 1 form also works with the Review and Approval - 1 Step User Review job controller.

5. Select the Job Properties tab, which is available for job templates, to customize the job controller under the Step - Application Review section. Properties can use data extracts and other model elements.



6. Select the Job Example 1 from the Review Form dropdown list.
7. Select the Job Reviewers from the Reviewer Group dropdown list.
8. Click Save to update the changes.
9. Check the Review and Approval - 1 Step Group Review job controller's definitions.

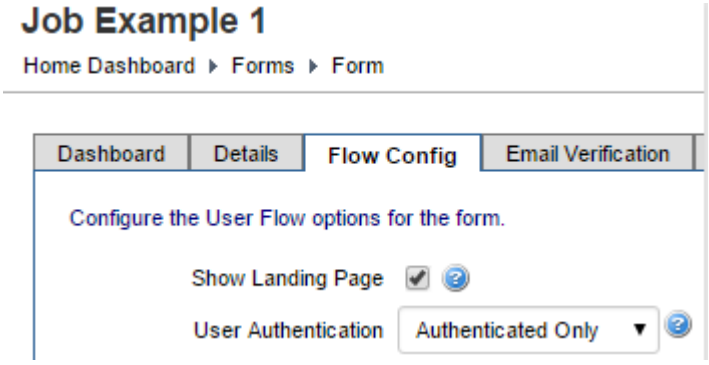
Configure Users and Groups

This example uses the following users, which are the actors in the job definition. The users have access to the Maguire form space and are in the Maguire organization.

Group	Login	Name	Description
Job Applic-ants	anne	Anne Applicant	Anne is the applicant who submits the application.
Job Reviewers	roger	Roger Reviewer	Roger is the reviewer who reviews the application and belongs to the Job Reviewers group.

However, you can substitute existing users for Anne and Roger. The form can be submitted as an public anonymous user or an authenticated user, but the reviewer has to be an authenticate user.

The Job Example 1 form pre-populates data for an authentication user only. You can customize your form flow by selecting the Authenticated Only from the User Authentication dropdown list, as shown below.

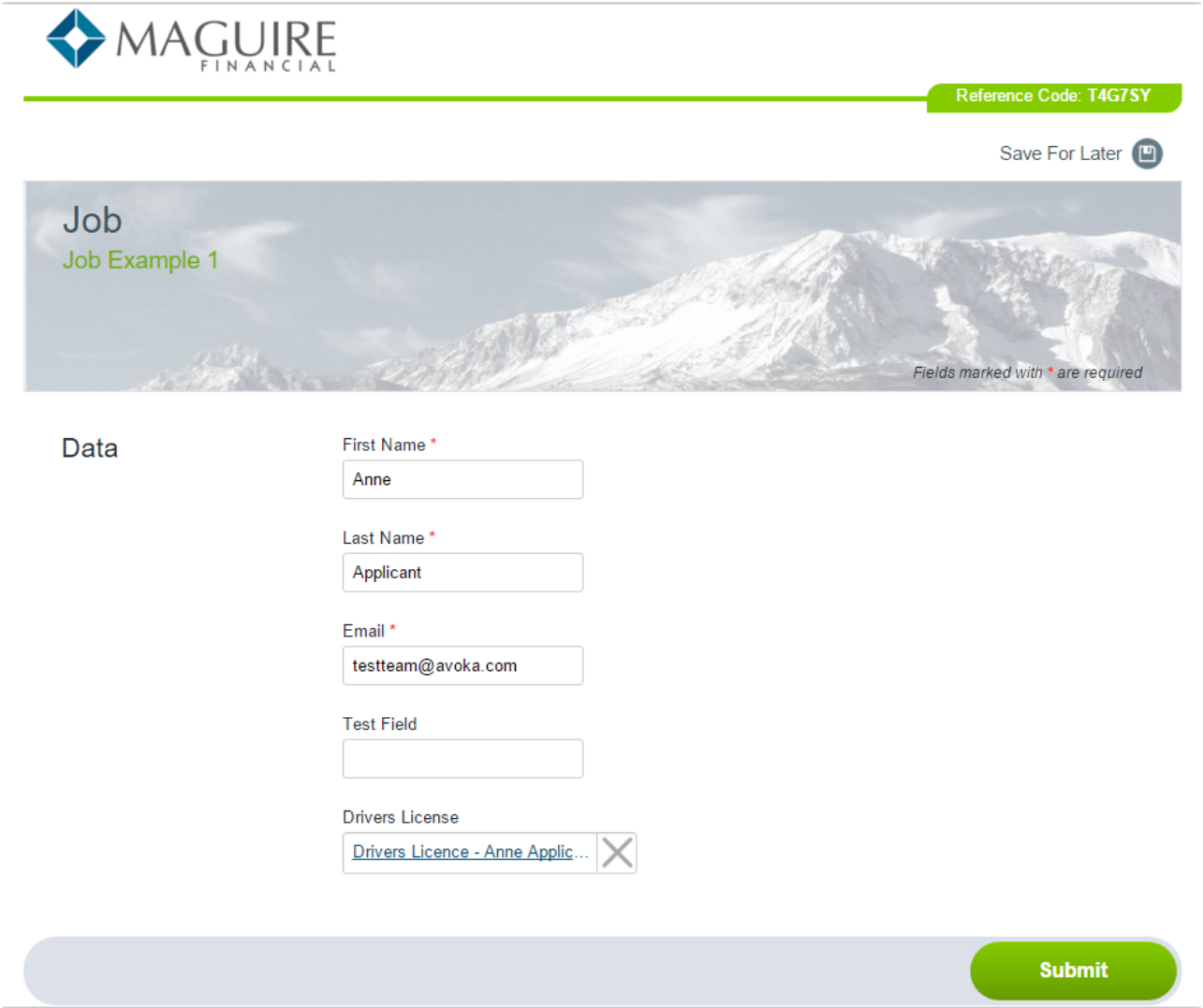


You should also configure the Job Reviewers group by adding Roger to the Group Members in the Members tab.

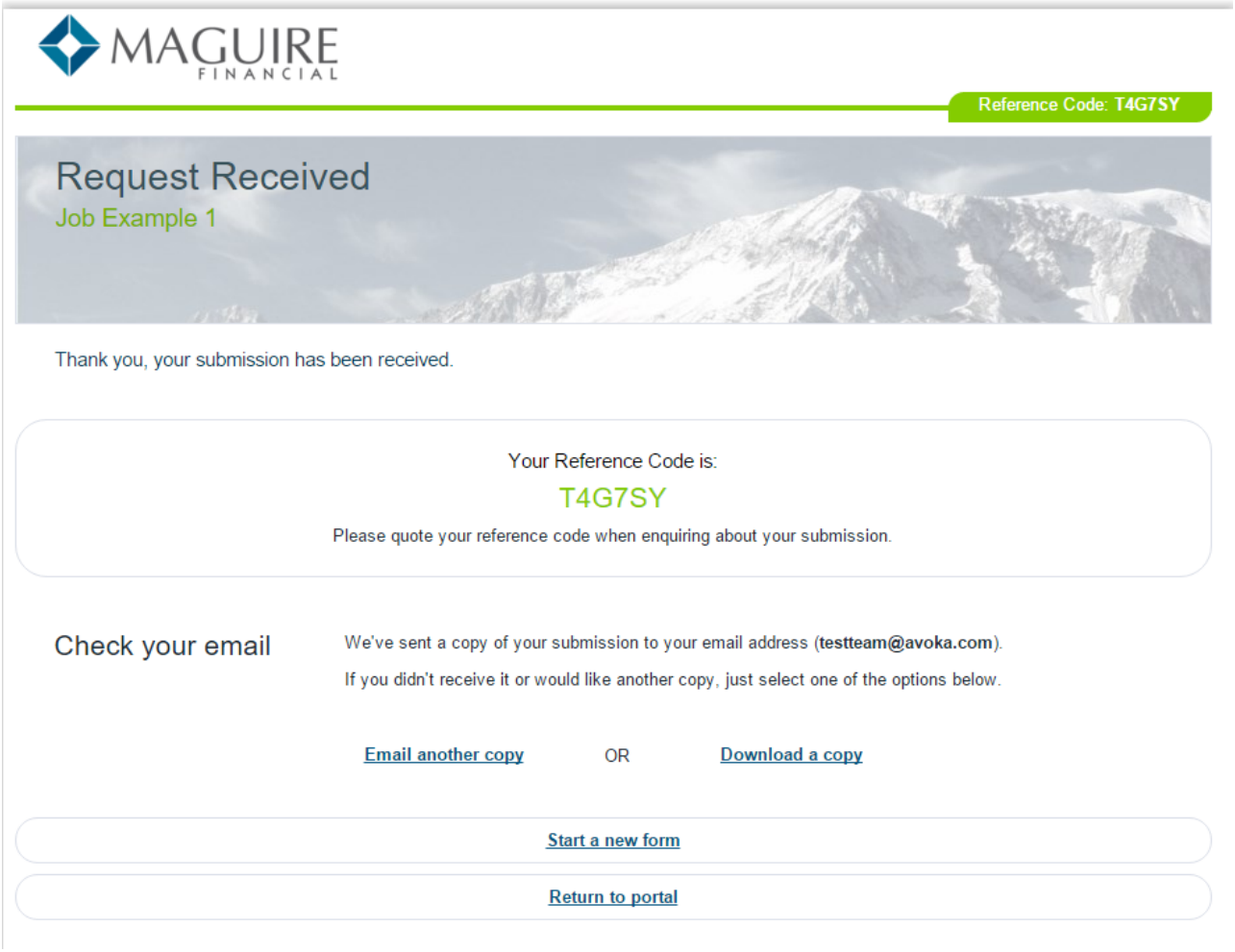
Now, you can run the example and see how collaboration job works.

Form - Anne

1. Log on to Manager as Anne.
2. Open the Maguire form space and select the Forms tab.
3. Locate the Job Example 1 form and click Direct to display the form in a form space in a new browser tab. If the form requires an authenticated user, It will first redirect to the portal login page. After logging in with the correct credentials, it will redirect you to the form page itself.



4. Fill in the First Name, Last Name and Email fields.
5. Upload a copy of a driver license as an image file using the Drivers License field browser and click Submit.



6. Take a note of the Reference Code of the form submission, T4G7SY in this case, so you can later use as the Job Number.
7. Click Return to Portal to return to your form space.

Manager - Administrator

This submission creates a collaboration job, which you can check in Manager:

1. Log on to Manager as an administrator.
2. Locate the created collaboration job.

ID	Job Number	Job Name	Org.	Created	Job Status	Current Step	Current Action	Action Type	Action Status	Action
37	T4G7SY	Review and Approval - 1 Step Gro...	maguire	14 Jan 15:32	In Progress	Application Start	Accept Quote	Form Start	Ready	

3. Check the collaboration job state, which should be:
   - Current Step - Application Start
   - Current Action - Accept Quote
   - Action Status - Ready.

This means the collaboration job is available and is waiting to be processed. Form more information, see collaboration job modes.

4. Click Process Job to execute this collaboration job, so you can test it right away. Manager processes the collaboration job and moves it to the Application Review step, as shown below.

ID	Job Number	Job Name	Org.	Created	Job Status	Current Step	Current Action	Action Type	Action Status	Action
37	T4G7SY	Review and Approval - 1 Step Gro...	maguire	14 Jan 15:32	In Progress	Application Review	Review Wait	Task Wait	Pending	

5. Click View and select the Actions tab for this collaboration job instance to see each action and its status.

Review and Approval - 1 Step Group Review Job

Home Dashboard ▸ Collaboration Jobs ▸ Job Details

Job Details | Steps | Actions | Events | Submissions

Step Name	Action Name	Action Type	Status	Assignee	Submitter	Index	Item	Route Result	Attempts	Created	Finished	Duration	Action
Application Start	Accept Quote	Form Start	Completed	anne	anne				1	14 Jan 16 15:32	14 Jan 16 15:33	41 sec	
Application Review	Assign Review	Task Assign	Assigned	Job Reviewers					1	14 Jan 16 15:33			
Application Review	Review Wait	Task Wait	Pending							14 Jan 16 15:33			

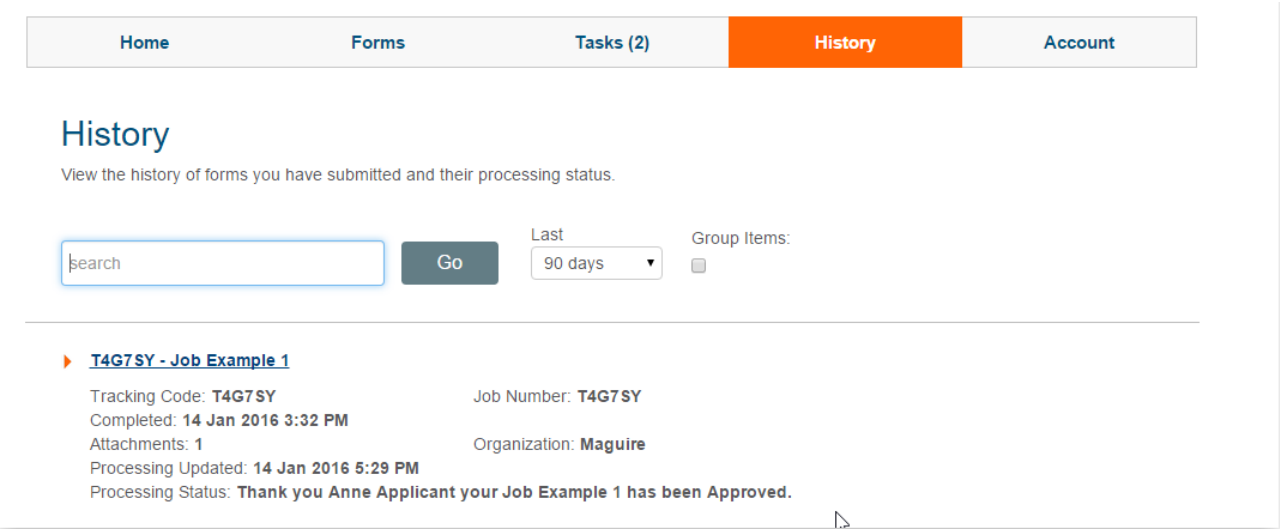
> NOTE
>
> The Application Review - Assign Review entry is assigned to Job Reviewers.

6. Click Close.

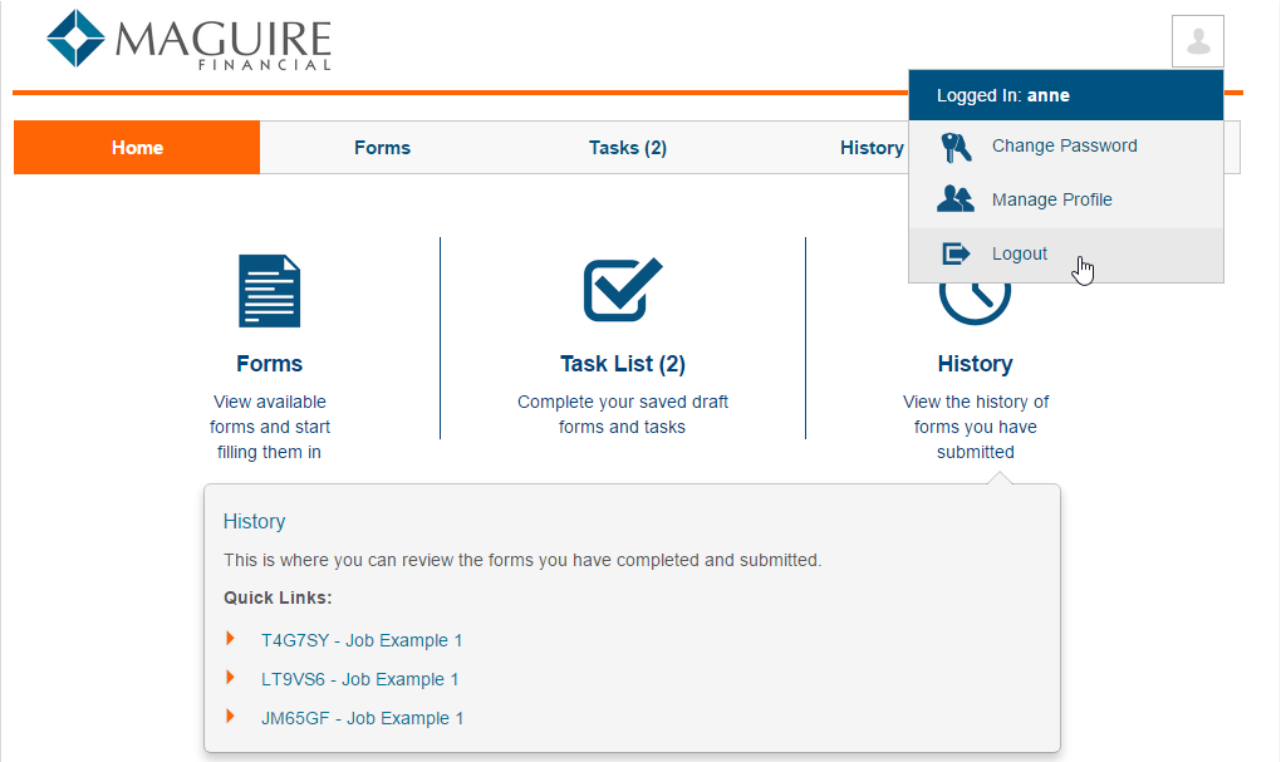
Form Space - Anne

You can see the collaboration job in your Journey Space:

1. Log on to Journey Space as Anne.
2. Click History tab to see the Processing Status of the collaboration job.



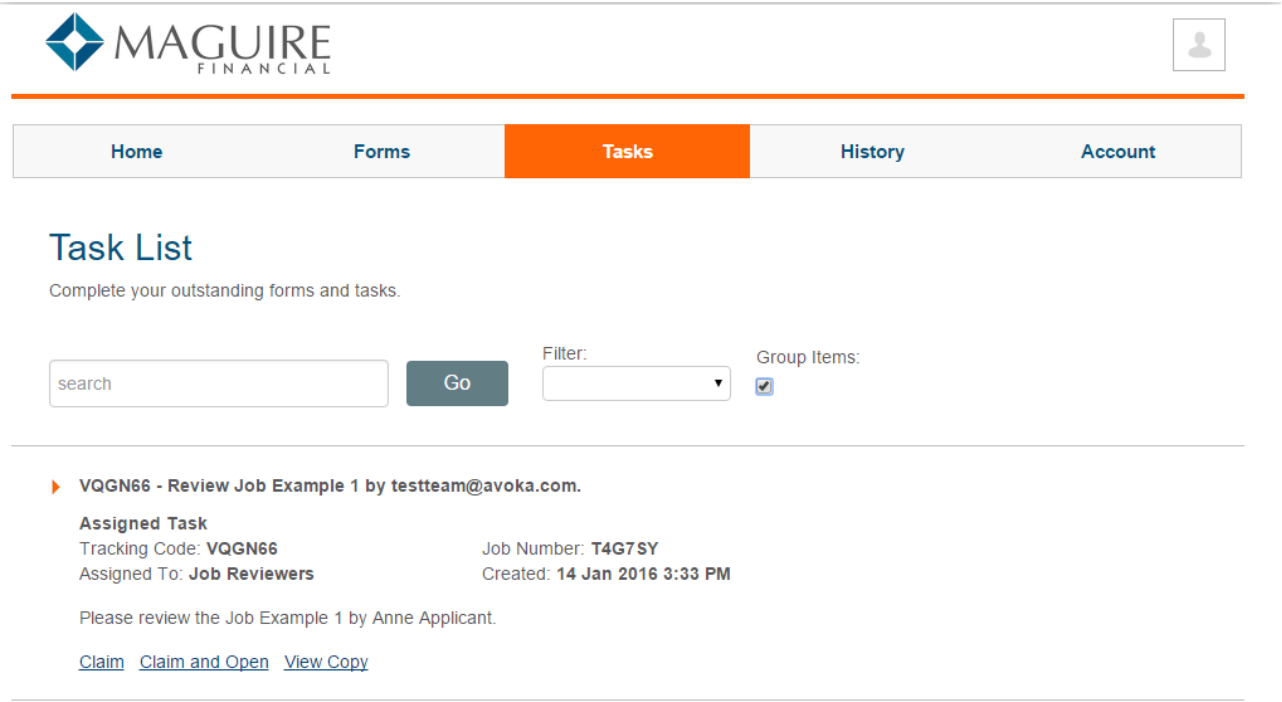
3. Log out as Anne



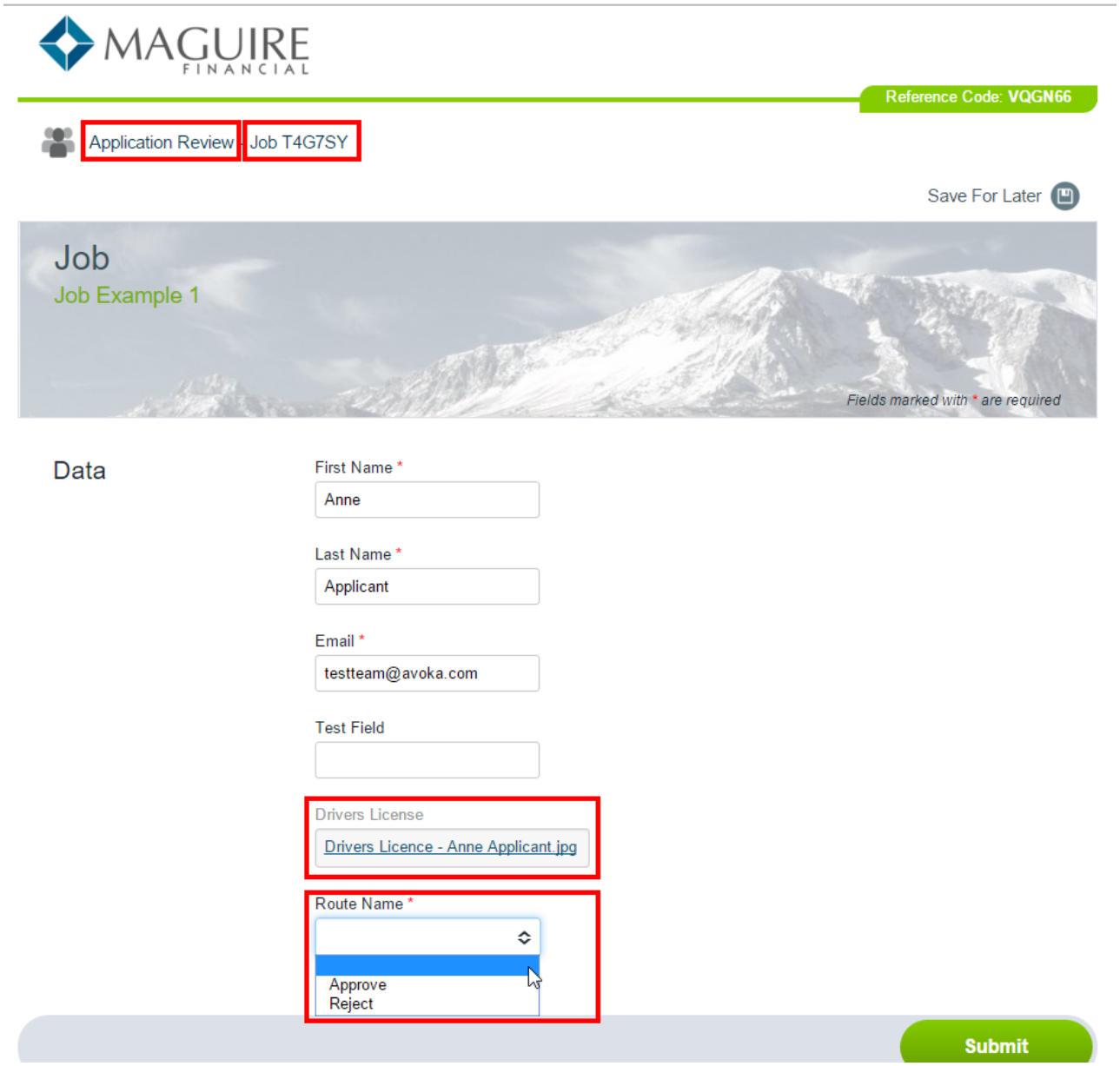
Form Space - Roger

You can review the collaboration job and decide whether to approve or reject it.

1. Log on to the Maguire form space as Roger.
2. Click the Tasks tab and select the Group Items checkbox to see the Group Task details as shown below.



3. Click Claim and Open to open the form and check the Step Name and Job Number. The Drivers License field, which is the inform attachment, is read only. The Route Name drop-down is now populated with the following choices: Approve or Reject.



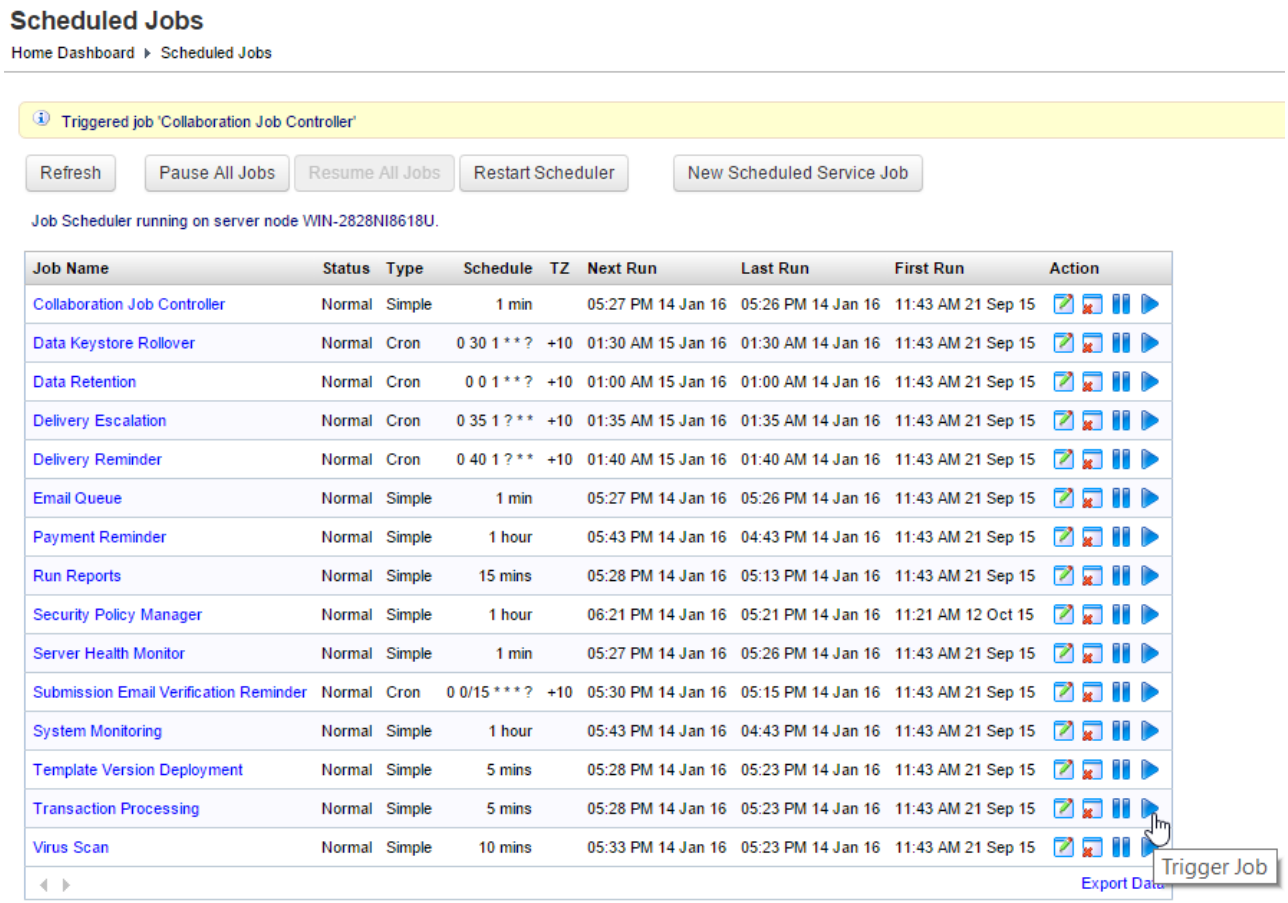
4. Select Approve from the Route Name dropdown list and click Submit.
5. Click Return to Portal on the confirmation page and click Logout.

Manager - Administrator

1. Locate the collaboration job and click Press Process Job, if its still at the Application Review step.
2. Check that the collaboration job moves to the Application Delivery step where it waits for the delivery to be completed.

ID	Job Number	Job Name	Org.	Created	Job Status	Current Step	Current Action	Action Type	Action Status	Action
37	T4G7SY	Review and Approval - 1 Step Gro...	maguire	14 Jan 15:32	In Progress	Application Delivery	Application Delivery	Delivery	In Progress	

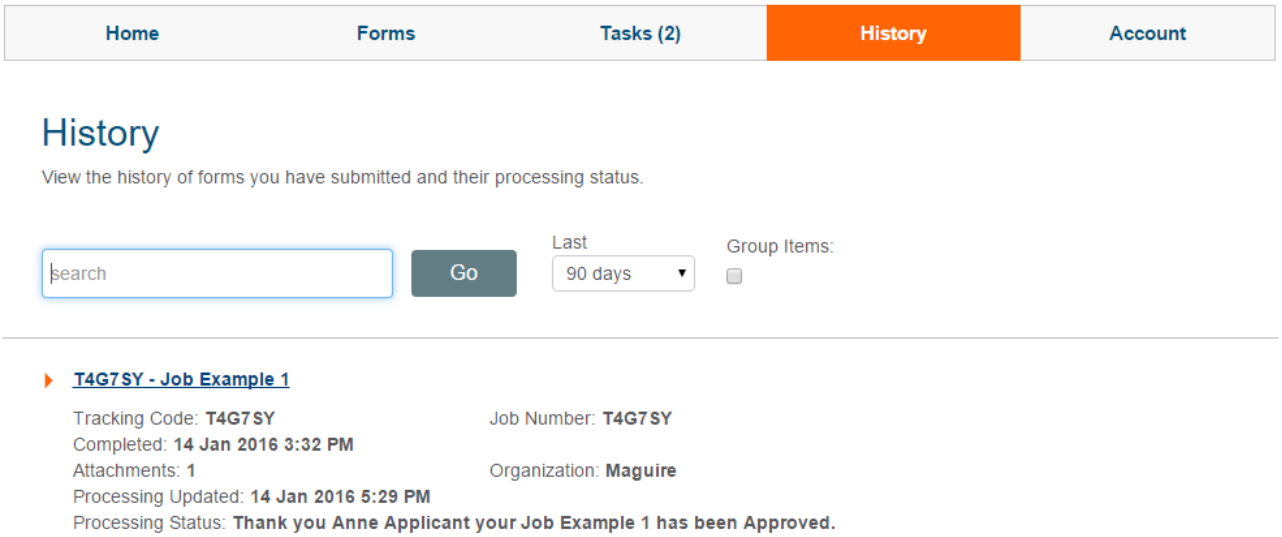
3. A scheduled job performs the delivery, which runs every 5 minutes by default. However, you can click Trigger Job to execute it right away, as shown below.



Job Name	Status	Type	Schedule	TZ	Next Run	Last Run	First Run	Action
Collaboration Job Controller	Normal	Simple	1 min		05:27 PM 14 Jan 16	05:26 PM 14 Jan 16	11:43 AM 21 Sep 15	
Data Keystore Rollover	Normal	Cron	0 30 1 * * ?	+10	01:30 AM 15 Jan 16	01:30 AM 14 Jan 16	11:43 AM 21 Sep 15	
Data Retention	Normal	Cron	0 0 1 * * ?	+10	01:00 AM 15 Jan 16	01:00 AM 14 Jan 16	11:43 AM 21 Sep 15	
Delivery Escalation	Normal	Cron	0 35 1 ? * *	+10	01:35 AM 15 Jan 16	01:35 AM 14 Jan 16	11:43 AM 21 Sep 15	
Delivery Reminder	Normal	Cron	0 40 1 ? * *	+10	01:40 AM 15 Jan 16	01:40 AM 14 Jan 16	11:43 AM 21 Sep 15	
Email Queue	Normal	Simple	1 min		05:27 PM 14 Jan 16	05:26 PM 14 Jan 16	11:43 AM 21 Sep 15	
Payment Reminder	Normal	Simple	1 hour		05:43 PM 14 Jan 16	04:43 PM 14 Jan 16	11:43 AM 21 Sep 15	
Run Reports	Normal	Simple	15 mins		05:28 PM 14 Jan 16	05:13 PM 14 Jan 16	11:43 AM 21 Sep 15	
Security Policy Manager	Normal	Simple	1 hour		06:21 PM 14 Jan 16	05:21 PM 14 Jan 16	11:21 AM 12 Oct 15	
Server Health Monitor	Normal	Simple	1 min		05:27 PM 14 Jan 16	05:26 PM 14 Jan 16	11:43 AM 21 Sep 15	
Submission Email Verification Reminder	Normal	Cron	0 0/15 * * * ?	+10	05:30 PM 14 Jan 16	05:15 PM 14 Jan 16	11:43 AM 21 Sep 15	
System Monitoring	Normal	Simple	1 hour		05:43 PM 14 Jan 16	04:43 PM 14 Jan 16	11:43 AM 21 Sep 15	
Template Version Deployment	Normal	Simple	5 mins		05:28 PM 14 Jan 16	05:23 PM 14 Jan 16	11:43 AM 21 Sep 15	
Transaction Processing	Normal	Simple	5 mins		05:28 PM 14 Jan 16	05:23 PM 14 Jan 16	11:43 AM 21 Sep 15	
Virus Scan	Normal	Simple	10 mins		05:33 PM 14 Jan 16	05:23 PM 14 Jan 16	11:43 AM 21 Sep 15	

Form Space - Anne

1. Log on to the Maguire form space as Anne.
2. Select the History tab to see that the application has now been approved now.



3. Close the form space.

This concludes the example.

Download

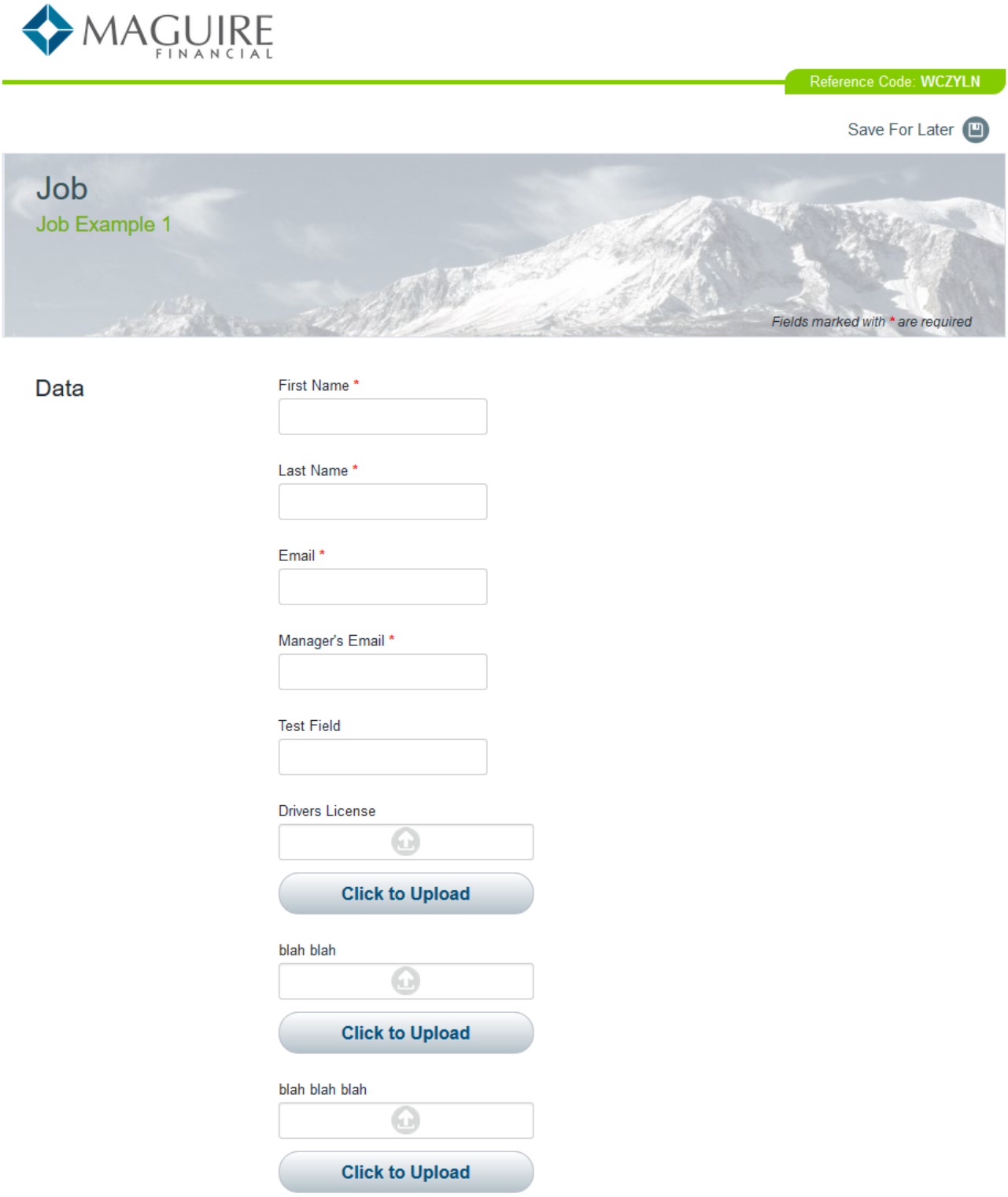
You can download the complete code example here.

Next, learn about multi-step group review collaboration jobs.

One Step Anonymous Review Collaboration Jobs

A one step anonymous review collaboration job is similar to a one step review collaboration job but with anonymous tasks.

The application form is shown below.



The form has the extra field, Managers Email, which is who the Job Step Application Review is assigned to. The first inform attachment field drivers license is read only at the review step. The other two inform attachment fields have no rules applied.

The job definition of the 1 Step Review - Anonymous is shown below.

```
{
  "jobDetails": {
    "name": "Anonymous 1 Step",
    "processSubmitImmediate": "false",
    "version": "SNAPSHOT"
  },
  "steps": [
    {
      "name": "Application Start",
      "type": "start",
      "actions": [
        {
          "name": "Accept Quote",
          "type": "Job Form Start",
          "properties": [
            { "name": "Process Message Send Email",  "value": "true" },
            { "name": "Process Message Text",  "value": "Thank you
${formDataMap.firstName} ${formDataMap.lastName} your ${submission.formName} is being pro-
cessed." }
          ]
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Review" }
      ]
    },
    {
      "name": "Application Review",
      "type": "",
      "actions": [
        {
          "name": "Assign Review",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Task Assign Email",  "value": "${formDataMap.managersEmail}" },
            { "name": "Task Form Code",  "value": "$func.startFormCode()" },
            { "name": "Task Message",  "value": "Please review the ${submission.formName}
by ${formDataMap.firstName} ${formDataMap.lastName}." },
            { "name": "Task Subject",  "value": "Review ${submission.formName} by ${sub-
mission.contactEmailAddress}." },
            { "name": "Task Assign Portal",  "value": "Maguire" },
```

```
            { "name": "Task Form XML Data",  "value": "$func.startSubmissionXml()" },
            { "name": "Task Attachments Submission Step",  "value": "Application Start"
},
            { "name": "Task Send Email",  "value": "true" },
            { "name": "Task Email Message",  "value": "$func.formProperty('Anonymous Task
Email Message')" },
            { "name": "Task Email Subject",  "value": "$func.formProperty('Anonymous Task
Email Subject')" },
            { "name": "Task Type",  "value": "Anonymous" }
          ]
        },
        {
          "name": "Review Wait",
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Reject",  "nextStep": "Application Rejected" }
      ]
    },
    {
      "name": "Application Delivery",
      "type": "",
      "actions": [
        {
          "name": "Application Delivery",
          "type": "Job Delivery"
        },
        {
          "name": "Application Delivery Wait",
          "type": "Job Delivery Wait"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Completed" }
      ]
    },
    {
      "name": "Application Completed",
      "type": "endpoint",
```

```
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "true" },
            { "name": "Process Message Submission Step",  "value": "Application Start" },
            { "name": "Process Message Text",  "value": "Thank you
${formDataMap.firstName} ${formDataMap.lastName} your ${submission.formName} has been
Approved." }
          ]
        }
      ]
    },
    {
      "name": "Application Rejected",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "true" },
            { "name": "Process Message Submission Step",  "value": "Application Start" },
            { "name": "Process Message Text",  "value": "Sorry ${formDataMap.firstName}
${formDataMap.lastName} your ${submission.formName} has been declined." }
          ]
        }
      ]
    }
  ]
```

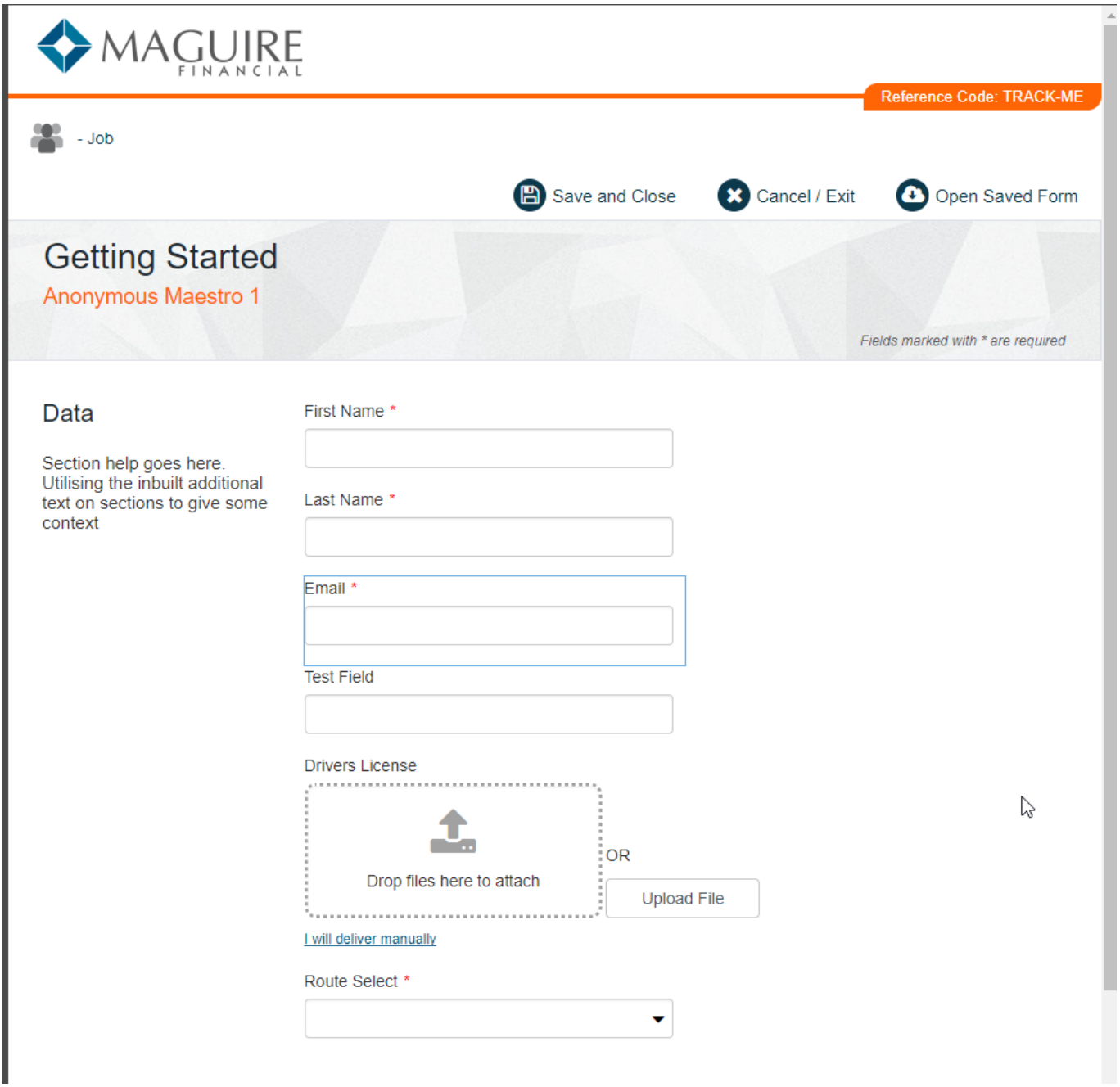
Download

You can download the complete code example here.

Cancel an Anonymous Task in Maestro Forms

You can cancel an authenticated task in the same way as per the one step review described above.

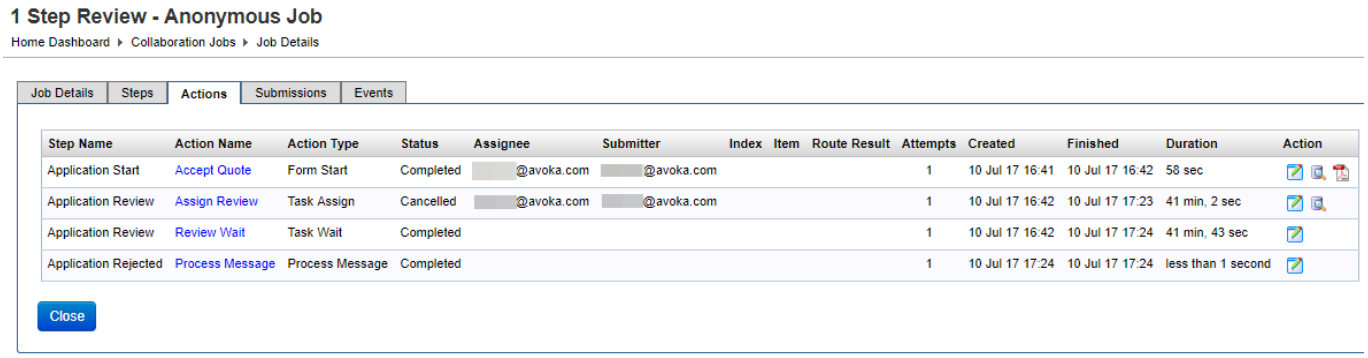
.



When we initially submit the form we enter our details. The task that is created will get assigned back to the address in the email field. You will get notified of a saved email with a link to open the form.

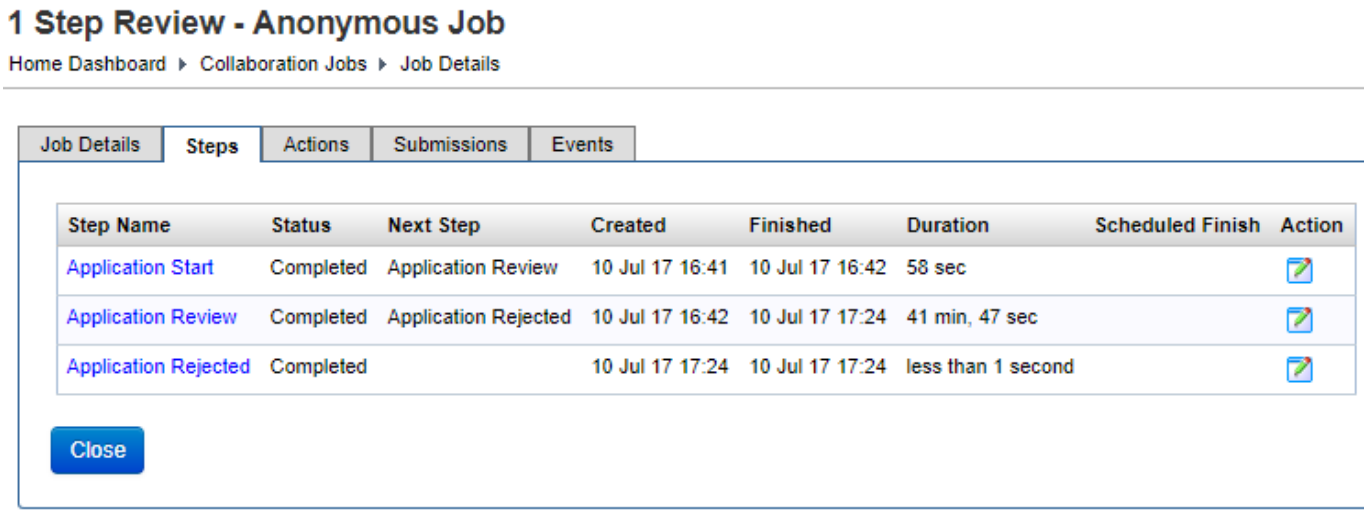
You have opened the form associated with the Application Review - Assign Review action. Then click on the Cancel Exit button. Cancelling doesn't return any form.xml and hence the route is empty. The empty route is picked up by the new Default route which send to the next step Application rejected.

Here we can see the actions that have been executed. We can see the status of the Application Review has been changed to cancelled and the route result is blank.



Step Name	Action Name	Action Type	Status	Assignee	Submitter	Index	Item	Route Result	Attempts	Created	Finished	Duration	Action
Application Start	Accept Quote	Form Start	Completed	@avoka.com	@avoka.com				1	10 Jul 17 16:41	10 Jul 17 16:42	58 sec	
Application Review	Assign Review	Task Assign	Cancelled	@avoka.com	@avoka.com				1	10 Jul 17 16:42	10 Jul 17 17:23	41 min, 2 sec	
Application Review	Review Wait	Task Wait	Completed						1	10 Jul 17 16:42	10 Jul 17 17:24	41 min, 43 sec	
Application Rejected	Process Message	Process Message	Completed						1	10 Jul 17 17:24	10 Jul 17 17:24	less than 1 second	

Here we can see that when the default route is routing to the next step Application Rejected.



Step Name	Status	Next Step	Created	Finished	Duration	Scheduled Finish	Action
Application Start	Completed	Application Review	10 Jul 17 16:41	10 Jul 17 16:42	58 sec		
Application Review	Completed	Application Rejected	10 Jul 17 16:42	10 Jul 17 17:24	41 min, 47 sec		
Application Rejected	Completed		10 Jul 17 17:24	10 Jul 17 17:24	less than 1 second		

The cancel doesn't submit any form data resulting in the empty `routeName`, so you needed to add the Default route to the application Review step of a job definition of the collaboration job as shown below.

```
{
  "name": "Application Review",
  "type": "",
  "actions": [
    {
      "name": "Assign Review",
      "type": "Job Task Assign",
      "properties": [
        { "name": "Task Assign Email",  "value": "${formDataMap.email}" },
        { "name": "Task Form Code",  "value": "$func.startFormCode()" },
        { "name": "Task Message",  "value": "Please review the ${submission.formName} by
${formDataMap.firstName} ${formDataMap.lastName}." },
        { "name": "Task Subject",  "value": "Review ${submission.formName} by ${sub-
mission.contactEmailAddress}." },
        { "name": "Task Assign Portal",  "value": "Web Plug-in" },
        { "name": "Task Form XML Data",  "value": "$func.startSubmissionXml()" },
        { "name": "Task Attachments Submission Step",  "value": "Application Start" },
        { "name": "Task Send Email",  "value": "true" },
        { "name": "Task Email Message",  "value": "$func.formProperty('Anonymous Task
Email Message')" },
        { "name": "Task Email Subject",  "value": "$func.formProperty('Anonymous Task
Email Subject')" },
        { "name": "Task Type",  "value": "Anonymous" }
      ]
    },
    {
      "name": "Review Wait",
      "type": "Job Task Wait"
    }
  ],
  "routes": [
    { "name": "Approve",  "nextStep": "Application Delivery" },
    { "name": "Reject",  "nextStep": "Application Rejected" },
    { "name": "Default",  "nextStep": "Application Rejected" }
  ]
},
```

This concludes the example.

Download

You can download the complete code example here.

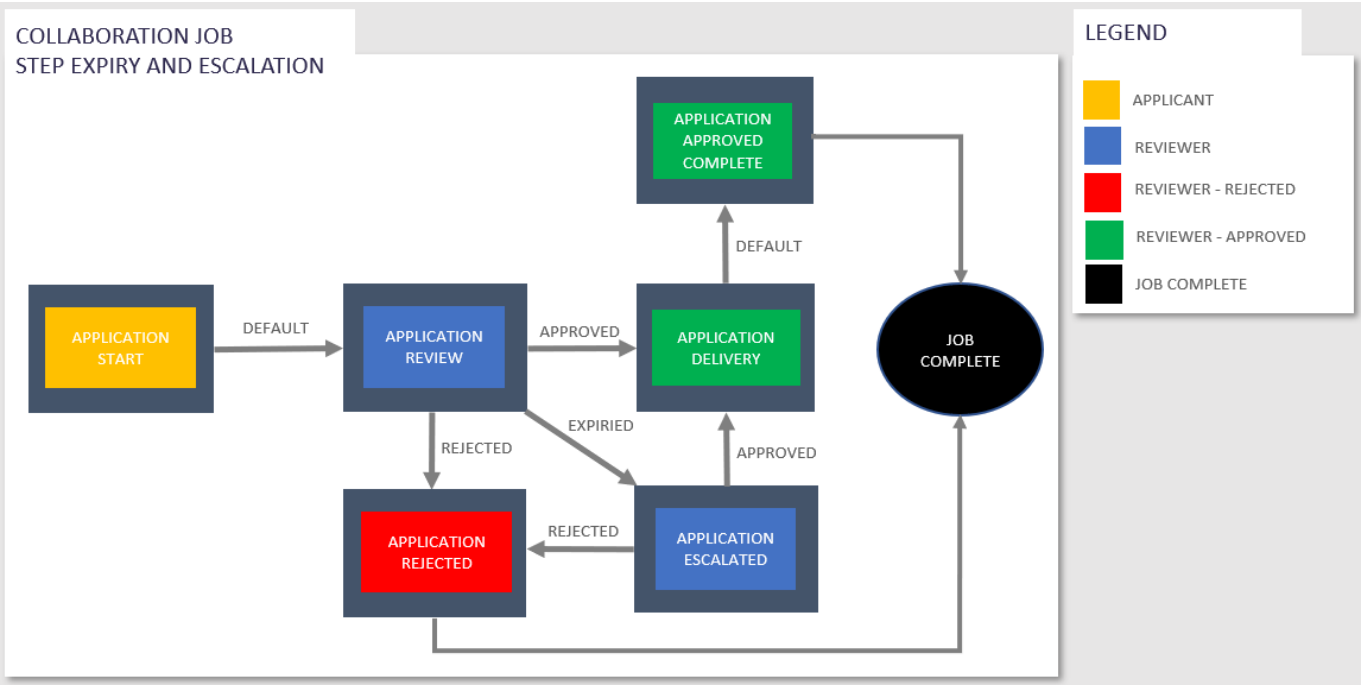
Next, learn about asynchronous step execution.

Last updated on : 22 October 2019

Step Expiry

You can configure steps of a collaboration job to expire depending on customer requirements, such as a Service Level Agreement[1]. For example, you can implement a Review step to assign a task to a group, such as Reviewers group, so they have two days to action the task. Otherwise, it expires and will be routed to an Escalation step for a manager to review it.

The step expiry workflow is shown below.



You can implement step expiration using one of the following mechanisms:

1. The step's `expiryRule` attribute to specify the period of time, from its creation date, when the step will expire.
2. Any action in a step can control the step expiry by returning an action result that sets an action expiry or returns the Expired status.

You can also configure the Expiry Action, see Job Expiry Service, within a step. This allows for the clean-up code to be run. Steps that expire must define an Expiry Route; that is, a route with the Route Name Expiry. When the step expires, it is routed to the Next Step defined in the Expiry Route.

---

[1]A service-level agreement (SLA) is a commitment between a service provider and a client. Particular aspects of the service – quality, availability, responsibilities – are agreed between the service provider and the service user.

Download

You can also download the application package with this example here.

Configure a Form

This example uses the AVT-3547 Test Expiry form, which is a copy of Job Example 1 form, initiates a job controller. You can download the example below.

Configure a Job Controller

You can update an existing job controller by adding an Application Escalated step, so when the Application Review step expires, it will be routed to the Application Escalated step.

The AVT-3547 Test Expiry B job controller is a modified copy of the 1 Step Review job controller, where the version B introduces a new Group Task Step.

The Application Review step is modified to:

- Expire the step after 1 minute. | v4.3 & Higher This feature is related to v4.3 and higher.
- Call the job expiry service.
- Call the `expiryServiceName` service.
- Expiry routes to the net step Application Escalated.
- Expiry route added with display set to false.

The Application Review and Application Escalated steps definition is shown below.

```
{
  "name": "Application Review",
  "type": "",
  "expiryRule": "+1m",
  "expiryServiceName": "Job Expiry - Expire Step Tasks",
  "actions": [
    {
      "name": "Assign Review",
      "type": "Job Task Assign",
      "properties": [
        { "name": "Task Assign Groups",  "value": "Job Reviewers" },
        { "name": "Task Form Code",  "value": "$func.startFormCode()" },
        { "name": "Task Message",  "value": "Please review the ${submission.formName} by
${formDataMap.firstName} ${formDataMap.lastName}." },
        { "name": "Task Review Previous Step",  "value": "true" },
        { "name": "Task Subject",  "value": "Review ${submission.formName} by by
${formDataMap.firstName} ${formDataMap.lastName}" },
        { "name": "Task Type",  "value": "Review" }
      ]
    },
    {
      "name": "Review Wait",
      "type": "Job Task Wait"
    }
  ],
  "routes": [
    { "name": "Approve",  "nextStep": "Application Delivery" },
    { "name": "Reject",  "nextStep": "Application Rejected" },
    { "name": "Expiry",  "nextStep": "Application Escalated",  "display": "false" }
  ]
},
{
  "name": "Application Escalated",
  "type": "",
  "actions": [
    {
      "name": "Assign Review",
      "type": "Job Task Assign",
      "properties": [
        { "name": "Task Assign Groups",  "value": "Job Managers" },
        { "name": "Task Form Code",  "value": "$func.startFormCode()" },
```

```
        { "name": "Task Message",  "value": "The ${submission.formName} by
${formDataMap.firstName} ${formDataMap.lastName} has been escalated." },
        { "name": "Task Review Previous Step",  "value": "true" },
        { "name": "Task Subject",  "value": "Review Escalated ${submission.formName} by
${formDataMap.firstName} ${formDataMap.lastName}." },
        { "name": "Task Type",  "value": "Review" }
      ]
    },
    {
      "name": "Review Wait",
      "type": "Job Task Wait"
    }
  ]
```

Configure a Job Expiry Service

This service cleans up an task that hasn't been submitted and returns the Expiry route name. If you are expiring a step that has tasks that have not been completed, then you will need to run a Job Expiry Action that cleans these up, as shown in the code below.

```
/* Groovy Job Expiry action service is a special type of job action service that is
executed when a step expires.
   It is used to clean up items such as tasks created by the step runs its actions.
   Script parameters include:
       actionContext : com.avoka.fc.core.service.job.ActionContext
       actionStepProperties :  com.avoka.fc.core.service.job.ActionStepProperties
       serviceDefinition : com.avoka.fc.core.entity.ServiceDefinition
       serviceParameters : Map<String, String>
       job : com.avoka.fc.core.entity.Job
       submission : com.avoka.fc.core.entity.Submission
       formDataMap : Map<String, String>
   Script return:
       actionResult : com.avoka.fc.core.service.job.ActionResult
*/

import com.avoka.fc.core.dao.DaoFactory
import com.avoka.fc.core.entity.Job
import com.avoka.fc.core.entity.JobAction
import com.avoka.fc.core.entity.JobStep
import com.avoka.fc.core.entity.Submission
import com.avoka.fc.core.service.AbandonmentService
import com.avoka.fc.core.service.job.ActionResult
import com.avoka.fc.core.service.job.ActionResult.Status
import com.avoka.core.groovy.GroovyLogger as logger
final EXPIRY_ROUTE_NAME = "Expiry"
Date now = new Date()

JobAction expiryJobAction = actionContext.getJobAction()
JobStep step = expiryJobAction.jobStep
List<JobAction> jobActions = step.getOrderedJobActions();
jobActions.remove(expiryJobAction)

// the msg can't be longer than 2000 characters.
StringBuilder msg = new StringBuilder();
msg.append("The job step associated with this task has expired. ref Job: ")
        .append(job.referenceNumber)
        .append(" Step: ")
        .append(step.name)

for (JobAction jobAction in jobActions) {
    // Fix Job Actions that have not completed
```

```
    if (jobAction.isStatusInProgress()
            || jobAction.isStatusAssigned()
            || jobAction.isStatusInvoked()
            || jobAction.isStatusPending()
            || jobAction.isStatusReady()) {
        jobAction.setStatus(JobAction.STATUS_EXPIRED);
        jobAction.setTimeFinished(now);
    }

    // Fix Tasks / Submissions in progress
    Submission submission = jobAction.getSubmission()
    if (submission != null) {
        // Set For all Tasks in this step as the step has Expired we don't want to
deliver the
        submission.setDeliveryStatus(Submission.STATUS_NotRequired);
        if (submission.isFormOpened()
                || submission.isFormAssigned()
                || submission.isFormSaved()
                || submission.isFormSubmitted()) {
            submission.setFormStatus(Submission.STATUS_Abandoned)
            submission.setAbandonmentType(AbandonmentService.ABANDONMENT_TYPE_SYSTEM)
            submission.setAbandonmentTimestamp(now)
            submission.setAbandonmentReason(msg.toString())

        } else {
            // For Tasks that were submitted
            submission.setDeliveryTime(now);
            submission.setAbandonmentReason(msg.toString())
        }

        if (submission.isDeliveryNotRequired()) {
            if (submission.getPurgeTimeDataDelivered() == null) {
                DaoFactory.submissionDao.setPurgeDatesForDeliveredAbandonedForms(sub-
mission);
            }
        }
    }
}
ActionResult actionResult = new ActionResult(Status.COMPLETED)
actionResult.setRoute(EXPIRY_ROUTE_NAME)
return actionResult
```

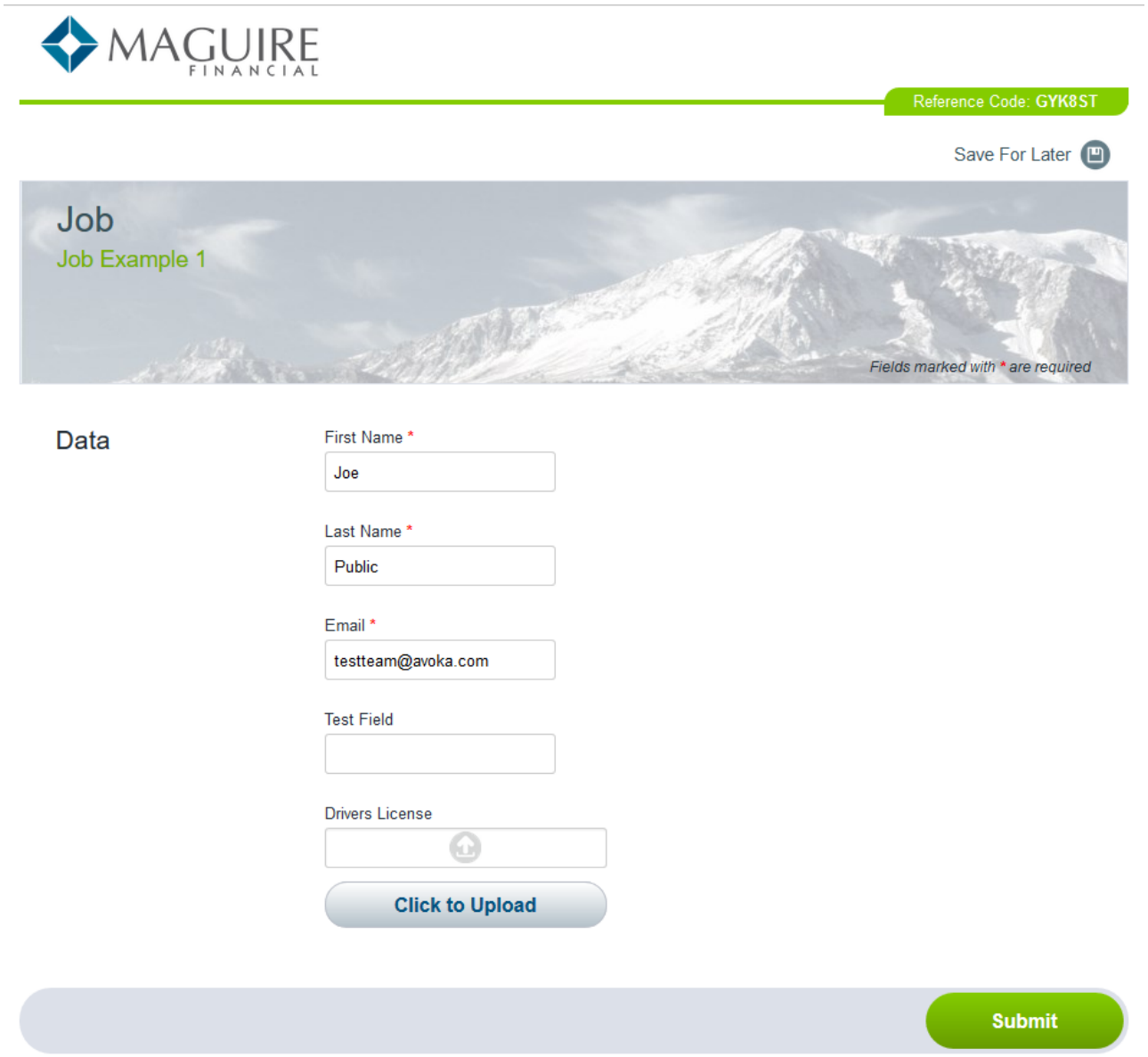
This example uses the Core Transact API, but we recommend using the Fluent API instead whenever possible. A simple example is shown below:

```
import com.avoka.tm.job.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import groovy.transform.TypeChecked
import java.util.*
import javax.servlet.http.*

@TypeChecked
class FluentJobExpiryActionService {

    /*
     * Perform a job expiry action service invocation
     *
     * return: the job action result
     */
    ActionResult invoke(SvcDef svcDef, Job job, JobAction jobAction, Map actionProp-
erties, HttpServletRequest request, User user) {

        // TODO: perform your logic

        String xml = Jobs.getStartTxnXml(job)
        Path path = new Path(xml)
        String email = path.val('//Contact/Email')

        return new ActionResultBuilder()
            .setStatus('Completed')
            .setMessage('Action completed for: ' + email)
            .build()
    }
}
```

Now, you can run the example and see how collaboration job's step expiry works.

Form - Start

1. Joe click on the AVT-3547 Step Expiry Form which opens.

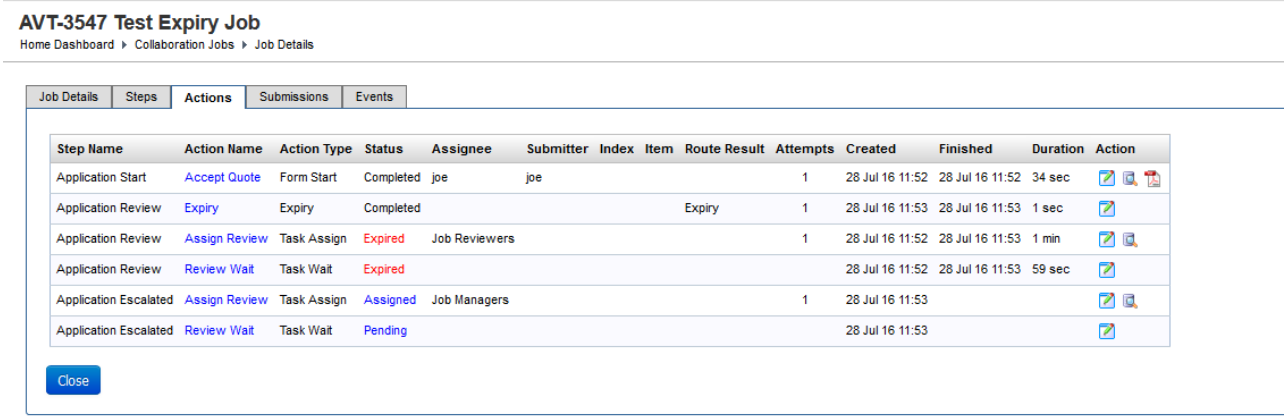


2. Joe's detail are prefilled so he clicks Submit.

Manager - Action Review Expiry

1. The collaboration job creates a Application Review step with an Review Task Assign action link to a submission assigned to the Job Reviewers and Review. There is also a task Review Task Wait Actions.

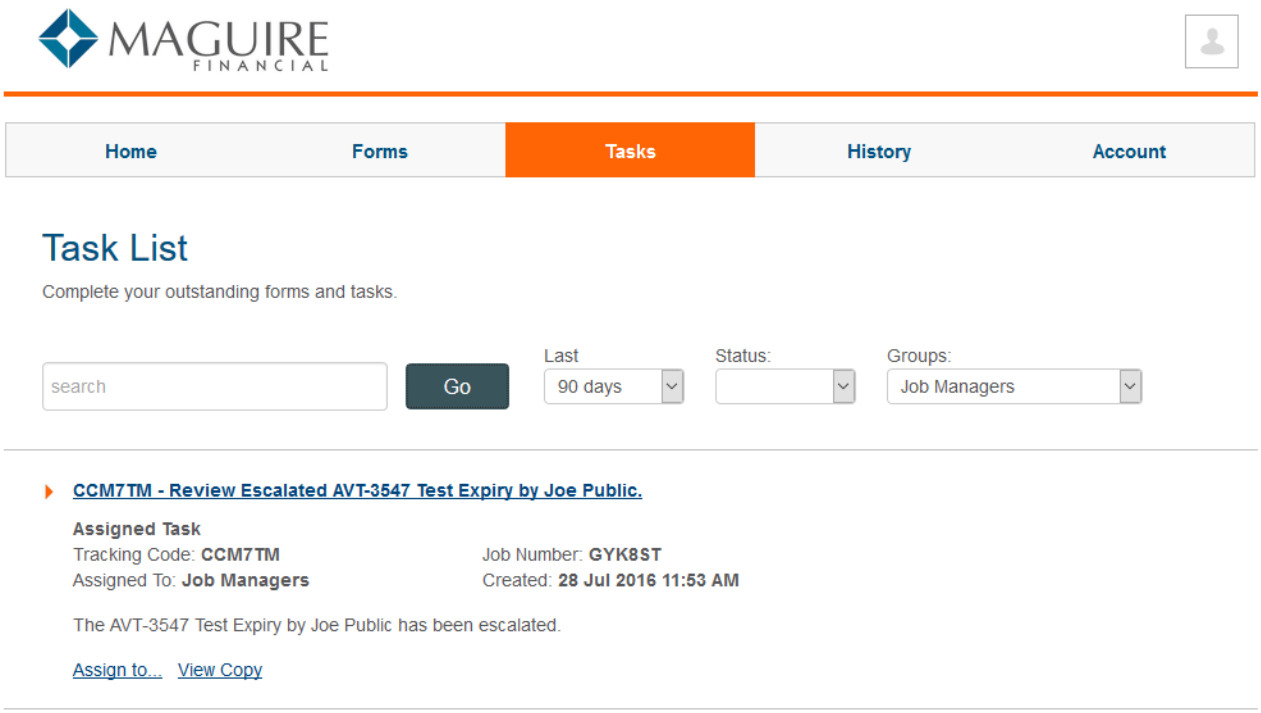
2. When the step expires these 2 actions are expired. The Application Review - Expiry action (Custom Groovy Service) runs to clean up the submissions for this step. For more details see the Job Expiry Service below.



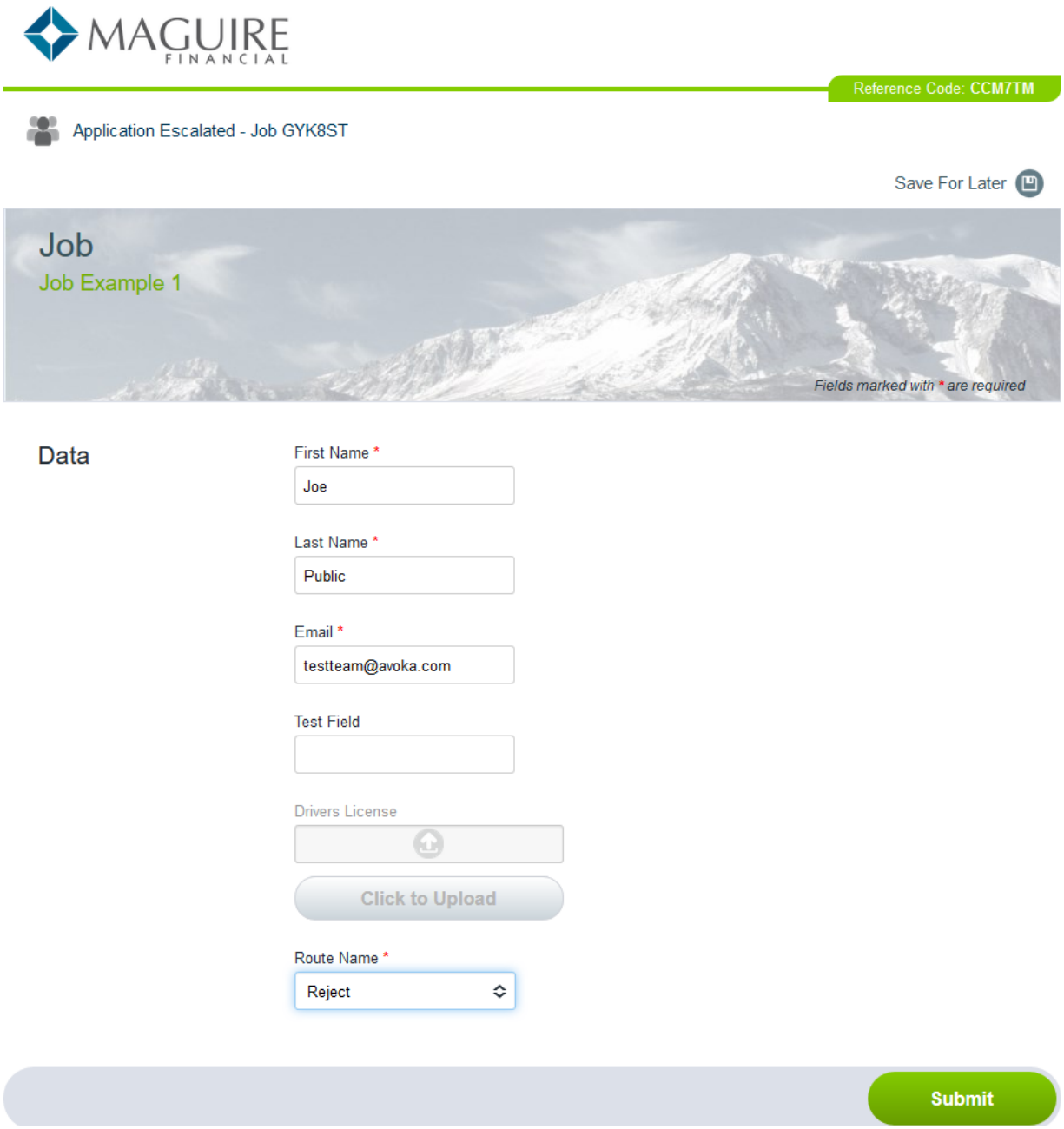
Step Name	Action Name	Action Type	Status	Assignee	Submitter	Index	Item	Route Result	Attempts	Created	Finished	Duration	Action
Application Start	Accept Quote	Form Start	Completed	joe	joe				1	28 Jul 16 11:52	28 Jul 16 11:52	34 sec	
Application Review	Expiry	Expiry	Completed					Expiry	1	28 Jul 16 11:53	28 Jul 16 11:53	1 sec	
Application Review	Assign Review	Task Assign	Expired	Job Reviewers					1	28 Jul 16 11:52	28 Jul 16 11:53	1 min	
Application Review	Review Wait	Task Wait	Expired							28 Jul 16 11:52	28 Jul 16 11:53	59 sec	
Application Escalated	Assign Review	Task Assign	Assigned	Job Managers					1	28 Jul 16 11:53			
Application Escalated	Review Wait	Task Wait	Pending							28 Jul 16 11:53			

Form Space - Application Escalated

1. Log in as Mike, who is a Job Manager.
2. Select the Tasks tab.
3. Select the Job Managers from the Groups dropdown list and click Go to view the following task list.



4. Click the CCM7TM task.



5. Select the Approve or Reject from the Route Name dropdown list and click Submit.

This concludes the example.

Download

You can also download the application package with this example here.

Next, learn about one step anonymous review collaboration jobs.

Task Assign Repeat Action

A collaboration job can have a task assign repeat action that allows you to do the following in a single step:

- Create a form bundle, with multiple tasks assigned to the same user, from a single form.
- Assign the Reviews and Approval workflow with multiple tasks to different users so an application can be reviewed in parallel.

The collaboration job must provide context back to the form to differentiate tasks. The context is represented with the following properties (node names):

Assignee

It's used by Parallel tasks, in a form as a read-only parameter. The repeat processing iterates over a pipe delimited list of assignees. This shows the current assignee which can be a User-name, Group or Email Address (Anonymous Tasks).

The example Job Definition - Task Assign - Job Action.

```
{
 "name": "Parallel Tasks",
 "type": "",
 "actions": [
 {
  "name": "Create Repeat Tasks",
  "type": "Job Task Assign",
  "properties": [
   { "name": "Task Assign Repeating", "value": "true" },
   { "name": "Task Assign Repeat Item", "value": "$func.invoke('FCT-3097 Get Repeat
Item', ${formDataMap.itemDelimited}, ${assignRepeatIndex})" },
   { "name": "Task Assign User",  "value": "$formDataMap.assignDelimited" },
   { "name": "Task Form Code",  "value": "fct-3097-composer" },
   { "name": "Task Message",  "value": "Please approve or reject" },
   { "name": "Task Subject",  "value": "${assignRepeatItem} Complete Credit Card Applic-
ation." },
   { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
   { "name": "Task Type",  "value": "Form" }
  ]
 },]
}
```

To specify users, you must apply the Task Assign User action property with a pipe delimited list of usernames, for example, `Task Assign User = $formDataMap.assignDelimited`. This data extract can be evaluated as `"peter|paul|mary"`. The first task is created for peter with the Assignee as peter. The second task is created for paul with the assignee as paul, and so on.

To specify groups, you must apply the Task Assign Groups action property with a pipe delimited list of groups.

To specify anonymous users, you must apply the action or Email Address anonymous tasks that an individual item is assigned to.

AssignRepeatIndex

It's used by Parallel tasks. It's used in a form as a read-only parameter. The repeat processing creates a is the index (Starting at 1) of the task that was created. A Forms can use this to look up a repeat section and hide show values on the form. In the example above 3 tasks will be created:

- (Assignee: peter, AssignRepeatIndex: 1)
- (Assignee: paul, AssignRepeatIndex: 2)
- (Assignee: mary, AssignRepeatIndex: 3)

AssignRepeatItem

It's used by Parallel tasks. It's used in a form as a read-only parameter. A repeating section in the form may be broken down by something other than a user or group. For example, a student fills out what sports they want to play: cricket, basketball or rugby

Sport	Coach	Season
Basketball	Al	All Year
Cricket	John	Summer
Rugby	John	Winter

A student Ginger Meggs wants to do Cricket, Basketball and Rugby, the issue is that the coach John is the same for Cricket and Rugby. How do we present the unique item value? It can be done in repeat form XML, as shown below.

```
<Sports>
  <Sport><Name>Cricket</Name><Selected>True<Selected><Coach>john</Coach></Sport>
  <Sport><Name>Basketball</Name><Selected>True<Selected><Coach>al</Coach></Sport>
  <Sport><Name>Rugby</Name><Selected>True<Selected><Coach>john</Coach></Sport>
</Sports>
```

The Job Task Assign properties are like this

```
{ "name": "Task Assign Repeating", "value": "true" },
{ "name": "Task Assign Repeat Item", "value": "$func.invoke('Get Repeat Item From Sport
Name', ${assignRepeatIndex})" },
{ "name": "Task Assign User", "value": "$formDataMap.assignDelimited" },{ "name": "Task
Subject",  "value": "${assignRepeatItem} application from ${assignee}" }

$formDataMap.assignDelimited = "john|al|john"
```

In this example 3 tasks are created:

- (Assignee: john, AssignRepeatIndex: 1, AssignRepeatItem: Cricket) and the Task Subject will be "Cricket application by Ginger Meggs"
- (Assignee: al, AssignRepeatIndex: 2, AssignRepeatItem: Basketball) and the Task Subject will be "Basketball application by Ginger Meggs"
- (Assignee: john, AssignRepeatIndex: 3, AssignRepeatItem: Rugby) and the Task Subject will be "Rugby application by Ginger Meggs"

Next, learn about task assign actions.

Collaboration Jobs Reuse

Configuring collaboration jobs from scratch can be a tedious task, so Manager lets you re-use[1] the following components to make the configuration easier and increase your productivity:

- Existing collaboration jobs.
- Action services, which can be reused between collaboration jobs and within the same collaboration job by providing different attributes and action properties.
- Common services when you can balance isolation of services between organizations.
- Common collaboration job actions.
- Forms when a collaboration job can be associated with many forms.
- Form code, when it uses the task, for example:`{"name": "Task Form Code", "value": "$func.startFormCode()" }`
- Form properties, when it stores the name of the form, for example: `{"name": "Task Assign Group", "value": "$func.formProperty('Initial Review')" }`

Next, learn how to view job services.

[1]Code reuse aims to save time and resources and reduce redundancy by taking advantage of assets that have already been created in some form within the software product development process. The key idea in reuse is that parts of a computer program written at one time can be or should be used in the construction of other programs written at a later time.

Collaboration Job Developers Documentation

Journey Manager comes with rich API that developers can use to build complex collaboration jobs. You can find Collaboration Job API documentation Services guide and Core API documentation.

Services guide

The services guide shows you how to create custom Groovy job actions and services that are referenced by collaboration jobs. For more information, see the version-specific documentation; for example, Services Guide 22.04.

> NOTE
>
> To access the developers documentation, check our online Services Guide or select Services > Transact Services Guide, if you are using Manager prior to version 19.11.

Core API

The Core API contains collaboration job API that developers can use to create custom build solutions.

> NOTE
>
> To access the developers documentation, download Core API Javadoc from our community web site or select Services > Core API Javadoc, if you are using Manager version prior to 19.11.

The collaboration job API comprises the following packages and classes:

Package: com.avoka.fc.core.service.job
: Provides collaboration jobs service classes.

Package: com.avoka.fc.core.service.job.config
: Provides collaboration job process map configuration classes. JSON attributes definitions used by the JobDef, StepDef and ActionDef classes.

Package: com.avoka.fc.core.service.job.impl
: Provides implementation of collaboration jobs service classes.

Package: com.avoka.fc.core.entity

Provides the SmartForm Manager entity classes. Details fields and methods in the Job, JobStep, JobAction and Submission Entities, which are used in the Velocity template substitutions.

Next, learn how to view collaboration jobs.

Form Bundles Overview

Journey Manager comes with form bundles that allow you to implement customer on-boarding solutions where a user can select several products. A form bundle contains two or more forms grouped together in a one step, so they can be used instead of building a single monolithic form. Breaking a large form into more manageable chunks makes it easier for you to develop and maintain. Form bundles also enable you to develop parts of a form concurrently.

Manager has the following form bundle features:

- Authenticated form bundles
- Anonymous form bundles
- Share form data
- Form bundle context
- Dynamic preconditions
- Form chaining
- Share extract data

A sample job definition of a collaboration job for a form bundle is shown below:

```
{
  "jobDetails": {
    "name": "Customer Onboarding Job",
    "processSubmitImmediate": true,
    "version": "4.3.0"
  },
  "steps": [
    {
      "name": "Customer Onboarding",
      "type": "start",
      "actions": [
        {
          "name": "Customer Submission",
          "type": "Job Form Start",
          "redirectNext": true
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Additional Products" }
      ]
    },
    {
      "name": "Additional Products",
      "type": "",
      "shareExtractData": true,
      "allFormsEditable": true,
      "showPreviousForms": true,
      "redirectNext": true,
      "actions": [
        {
          "name": "Credit Cards Application",
          "type": "Job Task Assign",
          "preCondition": "$formDataMap.productCreditCards == 'true'",
          "redirectNext": true,
          "properties": [
            { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
            { "name": "Task Form Code",  "value": "onboard-credit-cards" },
            { "name": "Task Message",  "value": "Please complete the Credit Card Applic-
ation form." },
            { "name": "Task Subject",  "value": "Complete Credit Card Application" },
            { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
```

```
            { "name": "Task Type",  "value": "Anonymous" }
          ]
        },
        {
          "name": "Insurance Application",
          "type": "Job Task Assign",
          "preCondition": "$formDataMap.productInsurance == 'true'",
          "redirectNext": true,
          "properties": [
            { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
            { "name": "Task Form Code",  "value": "onboard-insurance" },
            { "name": "Task Message",  "value": "Please complete the Insurance Applic-
ation form." },
            { "name": "Task Subject",  "value": "Complete Insurance Application" },
            { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
            { "name": "Task Type",  "value": "Anonymous" }
          ]
        },
        {
          "name": "Appliction Confirmation",
          "type": "Job Task Assign",
          "preCondition": "$formDataMap.productInsurance == 'true' || $formDataMap.-
productCreditCards == 'true'",
          "properties": [
            { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
            { "name": "Task Form Code",  "value": "onboard-confirmation" },
            { "name": "Task Message",  "value": "Please sign to complete Application
bundle." },
            { "name": "Task Subject",  "value": "Confirm Application Bundle" },
            { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
            { "name": "Task Type",  "value": "Anonymous" }
          ]
        },
        {
          "name": "Review Wait",
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Delivery" }
      ]
```

```
    },
    {
      "name": "Application Delivery",
      "type": "",
      "actions": [
        {
          "name": "Application Delivery",
          "type": "Job Delivery",
          "properties": [
            { "name": "Delivery Mode",  "value": "All Submissions" }
          ]
        },
        {
          "name": "Application Delivery Wait",
          "type": "Job Delivery Wait"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Completed" }
      ]
    },
    {
      "name": "Application Completed",
      "type": "endpoint"
    }
  ]
}
```

A typical form bundle asks users to provide their basic details, such as a name and address, and select a product or products they wish to apply for. The initial form in the form bundle is used to determine the overall flow of the collaboration job based on the products the customers selected, so it is basically triaging initial users. Therefore, this form is referred to as a triage form or a product selector form.

When a user submits the triage form, Manager performs the following:

1. Executes the next section of a collaboration job, which is the bundle step.
2. Associates a task with a job action.
3. Assigns a task to the user who submitted the triage form.
4. Redirects a browser to load the next available Form Task.

This makes the whole user experience looks like they have just moved to the next page of a bigger form.

A form bundle provides the following additional features:

- Sharing Data Between Tasks
- Show Previous Forms
- Forms in the Same Step Editable
- Preconditions
- Dynamic Preconditions
- Dynamic Precondition and Action Order
- User Authentication

Let's look at each of these features in more detail.

Sharing Data Between Tasks

There are three valid options that can be taken to share data between forms in a form bundle:

No Sharing

This sets the step attribute `shareFormData` is false, and `shareExtractData` is false, so no data is copied between the forms.

Shared XML Data Model

This sets the step attribute `shareFormData` is true, and `shareExtractData` is false. When a Form Task is submitted the data is automatically written to the other forms. To use this , all forms must use the same data model. This is the most common way to share data between forms.

Share Extract Data

This sets the step attribute `shareFormData` is false, and `shareExtractData` is true. This can be used where the forms don't have a common data model. This type of sharing is also called Publish - Subscribe sharing of data extracts.

Let's demonstrate it with a simple example where a form A sets a field as a data extract Demo Type and a form B subscribes to the same data extract Demo Type. The form A is opened, the Demo Type data extract value is changed to a new value and then the form A is submitted. The Demo Type data extract is published on the save action. Because the form B has subscribed to the value, it picks its data extract and form XML is updated with the new value.

Show Previous Forms

The form bundle context is populated in the task when the step attribute `showPreviousForms` is true. You can see the content of the data that is returned to the form and how the form and widget can use it to display the previous forms in the form bundle.

Forms in the Same Step Editable

Forms in a bundle step are editable when the step attribute `allFormsEditable` is true and when the step has still not completed. This happens even if the form tasks in the step have previously been submitted.

You can use the Collaboration Job Bundle Navigation widgets to list form links to previous forms. Click on a link to open the selected form as you will be redirected to the form. You can also click on a link to download a receipt for a previous submitted form.

Preconditions

All the Job Task Assign actions in form bundle steps can have optional preconditions, such as `"preCondition": "$formDataMap.productCreditCards == 'true'"`. The precondition is triggered, if the velocity template on the right evaluates to true, so the action instance and associated task are created.

By default, the preconditions are evaluated against a start submission. If the previous submission isn't the start submission, its possible that the data extract fields are populated in the previous submission, but are missing from the start submission. In this case, the precondition may not fire.

To fix this, you must specify the Precondition Submission step to be used for evaluating preconditions. You should replace the "Triage Step Name" with the actual step name that contains triage form submission.

```
{
  "name": "A Form Bundle",
  "type": "",
  "dynamicPreConditions": true,
  "shareFormData": true,
  "showPreviousForms": true,
  "redirectNext": true,
  "actions": [ ... ],
  "properties": [
    {"name": "Pre Condition Submission Step", "value" : "Triage Step Name"}
  ],
  "routes": [ ... ]
}
```

Dynamic Preconditions

Normal or static preconditions are evaluated once when the step is first run using the data extracts for the precondition submission. Dynamic preconditions fire after the submission of any task in a form bundle. The data extracts are taken from the form bundle and are submitted and evaluated against the actions whose preconditions have not already fired.

Actions that have already been created, as a result of the previous preconditions being true, are not affected, if their preconditions are evaluated to false now. To configure `dynamicPreconditions` in a form bundlestep, add the `"dynamicPreConditions": true` step's attribute:

Dynamic Precondition and Action Order

The order that the forms are made available in a form bundle is taken from the order they are listed in a job definition of a collaboration job. With dynamic preconditions the next form loaded is the highest one on the list that hasn't been submitted yet.

Let's use the Additional Products step of a collaboration job as an example. A user can select the insurance product option only in the triage form. The result is the insurance application and the application confirmation are created.

The next available form is the Insurance Application so it's opened next, so the user can now select the Credit Card product. The dynamic precondition is triggered and the Credit Card Application form is created.

The next available form is the Credit Card Application, because it is at the top of the list. The user submits this form.

As the Insurance Application was already submitted, the form chaining opens the next available form that is the Application Confirmation.

The Application Confirmation lists all the forms in order that they appear in the collaboration job, as shown below:

```
{
  "name": "Additional Products",
  "type": "",
  "dynamicPreConditions": true,
  "shareFormData": true,
  "showPreviousForms": true,
  "redirectNext": true,
  "actions": [
    {
      "name": "Credit Cards Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productCreditCards == 'true'",
      ...
    },
    {
      "name": "Insurance Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productInsurance == 'true'",
      ...
    },
    {
      "name": "Appliction Confirmation",
      "type": "Job Task Assign",
      ...
    },
    ...
  ],
  "routes": [ ... ]
}
```

User Authentication

Form bundles support both anonymous and authenticated flows, which is based on the Form or Review task. For more information on what action properties are available to the task, see action properties.

NOTE

Mixing of task types, such as Anonymous, Form and Review, in the same form bundle step is not supported.

Next, learn about a form bundle context.

Form Bundles Context

A collaboration job provides additional context information to form bundles created with Maestro, Composer, or the Maguire template. This information is automatically used by forms and widgets to provide a list of all the other forms in the form bundle, the completion status and a way to navigate to the chosen form as well as print a receipt. A form bundle content includes `allFormsEditable` option that enables users to open any previously completed forms of their application bundle in a step.

The form XML shown below is the part of the customer onboarding collaboration job where the data is created in the `//SystemProfile/Job/StepTasks` XML node, which is a list of the previous and current step tasks for the form bundle.

```
<SystemProfile>
    <DisplayMode>Entry</DisplayMode>
    <HostContext>Page</HostContext>
    <ReceiptNumber></ReceiptNumber>
    <SubmitDateString></SubmitDateString>
    <SubmissionMessage></SubmissionMessage>
    <FormDataSer-
viceURL>ht-
tps://tm.maestro.avoka.com/maguire/servlet/FormDynamicDataServlet</FormDataServiceURL>
    <RequestLogKey>ab45894a127f2e26a3e516ba4291080a</RequestLogKey>
    <TrackingCode>PT3F6P</TrackingCode>
    <Job>
        <ReferenceNumber>QF3FJK</ReferenceNumber>
        <StepName>Additional Products</StepName>
        <AvailableRoutes>Default</AvailableRoutes>
        <RouteName></RouteName>
        <Assignee>jsmith@gmail.com</Assignee>
        <AssignRepeatIndex></AssignRepeatIndex>
        <AssignRepeatItem></AssignRepeatItem>
        <Description></Description>
        <StepTasks>
            <Task>
                <StepName>Customer Onboarding</StepName>
                <MenuLabel>Customer Onboarding</MenuLabel>
                <MenuCategory>Customer Onboarding</MenuCategory>
                <TaskSubject>Customer Onboarding</TaskSubject>
                <FormCompleted>true</FormCompleted>
                <FormEditable>false</FormEditable>
                <FormName>Customer Onboarding</FormName>
                <ReceiptURL>ht-
tps://t-
m.maes-
tro.a-
voka.-
com/maguire/ser-
vlet/FormRe-
ceipt.pdf?submitKey=dfe15aab5345ee23505021f889ac0ff1&amp;renderMode=file</ReceiptURL>
                <CurrentForm>false</CurrentForm>
                <TaskMessage></TaskMessage>
                <ScheduledTime></ScheduledTime>
                <ExpiryTime></ExpiryTime>
```

```
                <FormURL></FormURL>
            </Task>
            <Task>
                <StepName>Additional Products</StepName>
                <MenuLabel>Complete Credit Card Application</MenuLabel>
                <MenuCategory>Additional Products</MenuCategory>
                <TaskSubject>Complete Credit Card Application</TaskSubject>
                <TaskMessage>Please complete the Credit Card Application form.</TaskMes-
sage>
                <FormCompleted>false</FormCompleted>
                <FormEditable>true</FormEditable>
                <FormName>Customer Onboarding - Credit Cards</FormName>
                <FormURL>ht-
tps://t-
m.maestro.avoka.com/maguire/form.htm?submitKey=2ef6da250d9d1dbee74749172f64e3e4</FormURL>
                <CurrentForm>true</CurrentForm>
                <ScheduledTime></ScheduledTime>
                <ExpiryTime></ExpiryTime>
                <ReceiptURL></ReceiptURL>
            </Task>
            <Task>
                <StepName>Additional Products</StepName>
                <MenuLabel>Complete Insurance Application</MenuLabel>
                <MenuCategory>Additional Products</MenuCategory>
                <TaskSubject>Complete Insurance Application</TaskSubject>
                <TaskMessage>Please complete the Insurance Application form.</TaskMes-
sage>
                <FormCompleted>false</FormCompleted>
                <FormEditable>true</FormEditable>
                <FormName>Customer Onboarding - Insurance</FormName>
                <FormURL>ht-
tps://t-
m.maestro.avoka.com/maguire/form.htm?submitKey=ff13980c03dccd2963ba8c6be469e248</FormURL>
                <CurrentForm>false</CurrentForm>
                <ScheduledTime></ScheduledTime>
                <ExpiryTime></ExpiryTime>
                <ReceiptURL></ReceiptURL>
            </Task>
            <Task>
                <StepName>Additional Products</StepName>
                <MenuLabel>Confirm Application Bundle</MenuLabel>
```

```
                <MenuCategory>Additional Products</MenuCategory>
                <TaskSubject>Confirm Application Bundle</TaskSubject>
                <TaskMessage>Please sign to complete Application bundle.</TaskMessage>
                <FormCompleted>false</FormCompleted>
                <FormEditable>true</FormEditable>
                <FormName>Customer Onboarding - Confirmation</FormName>
                <FormURL>ht-
tps://t-
n.maestro.avoka.com/maguire/form.htm?submitKey=8e6b6352eb284b03b10ea36fa0b454f2</FormURL>
                <CurrentForm>false</CurrentForm>
                <ScheduledTime></ScheduledTime>
                <ExpiryTime></ExpiryTime>
                <ReceiptURL></ReceiptURL>
            </Task>
        </StepTasks>
    </Job>
```


Next, learn about Job Definition elements for a Form Bundle configuration.

Configure Form Bundles

Manager allows you to configure form bundles using the following pre-defined set of job definition elements that are part of step and action attributes or properties.

Step Attributes For Form Bundles with More Than Two Tasks

> NOTE
>
> The following are ignored if there is only one task in a step.

`dynamicPreConditions` -re-evaluates action preconditions for forms with the multiple step, if set to true. Otherwise, they are only evaluated on the triage form.

`shareFormData` - copies the entire form XML data between form tasks within a form bundle step.

`shareExtractData` - copies data between form tasks within a form bundle step using a publish subscribe method.

`allFormsEditable` - if set to true, the form tasks within a form bundle are mutable, until the step is completed.

Step Attributes For Form Bundles with One or More Tasks

`showPreviousForms` - shows the previous forms.

`redirectNext` - allows form chaining along with the action attribute redirect next.

Step Properties

This is used to specify the triage step and is only required if triage form is not the start form.

```
"properties": [
    {"name": "Pre Condition Submission Step", "value" : "Triage Step Name"}
  ],
```

Action Properties

`preCondition` - allows you to use the Apache velocity templates[1].

`redirectNext` - allows form chaining along with the step attribute redirect next.

Next, learn about anonymous Form Bundles.

Last updated on : 21 October 2019

[1]Apache Velocity Templates are used extensively throughout Temenos Journey Manager platformto define html page layout, email body and subject. A template is combined with model containing a number or objects to produce a String. For more information see http://velocity.apache.org/engine/devel/user-guide.html

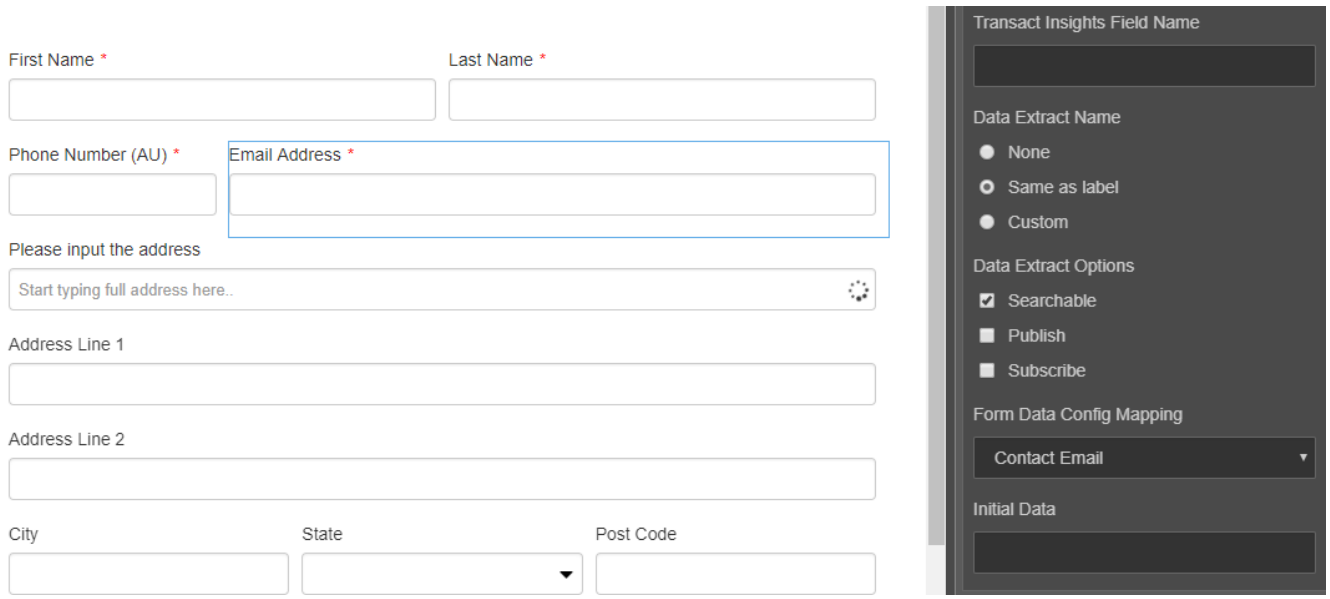
Anonymous Form Bundles

Anonymous bundles are form bundles that require the use of an email address to allow form users to partially fill out the forms and then complete them later. For example, a triage form can ask a user to enter an email address, so after the form is filled out and submitted, the user can be sent an email with a link to open and continue editing this form bundle. You can use a hard coded contact email in the form, but you must decide how much information the user should re-enter, in case they are not able to complete the form in one go. Regardless of the email address, an anonymous form bundle must be always protected with a save challenge.

We will demonstrate anonymous form bundles with the Maestro Onboarding Job and four Maestro forms examples.

Configure a Form

Below is the Maestro Onboarding - Triage form including the Email Address field. This form starts the anonymous collaboration job.



1. Select the Same as label checkbox under the Data Extract Name, so you can reference it in the collaboration job as `$formDataMap.emailAddress`.
2. Select the Contact Email Address from the Form Data Config Mapping dropdown list.

> NOTE
>
> This email address can't be blank, so the field must be mandatory where the email address is to be entered by the form user.

You can hard code an email address, such as onboarding@company.com, in the form by doing one of the following:

- Setting the initial data value to onboarding@company.com.
- Creating a Maestro form rule to hide the email address or to make it read-only.

However, we don't recommend using a hard-coded address.

Job Definition - Additional Products

The job definitions for additional products shows the configuration bundle step below:

```
{
        "name": "Additional Products",
        "type": "",
        "dynanicPreConditions": true,
        "shareExtractData": false,
        "shareFormData": true,
        "allFormsEditable": true,
        "showPreviousForms": true,
        "redirectNext": true,
        "actions": [
                {
                "name": "Credit Cards Application",
                "type": "Job Task Assign",
                "preCondition": "$formDataMap.productCreditCards == 'true'",
                "redirectNext": true,
                "properties":[
                        {"name": "Task Assign Email", "value": "$formDataMap.enailAddress"},
                        {"name": "Task Form Code", "value": maestroonboardingcre},
                        {"name": "Task Message", "value": "Please complete the Credit Card Application
form."},
                        {"name": "Task Subject", "value": "Complete Credit Card Application"},
                        {"name": "Task Input XML Prefil", "value": "$func.startSubmissionXml()"},
                        {"name": "Task Type", "value": "Anonymous"},
                        {"name": "Task Save Challenge", "value": "$formDataMap.phoneNumber"}
                ]
                },
                {
                "name": "Insurance Application",
                "type": "Job Task Assign",
                "preCondition": "$formDataMap.productInsurance == 'true'",
                "redirectNext": true,
                "properties": [
                {"name": "Task Assign Email", "value": "$formDataMap.enailAddress"},
                        {"name": "Task Form Code", "value": maestroonboardingins},
                        {"name": "Task Message", "value": "Please complete the Insurance Application for
                        {"name": "Task Subject", "value": "Complete Insurance Application"},
                        {"name": "Task Input XML Prefil", "value": "$func.startSubmissionXML()"},
                        {"name": "Task Type", "value": "Anonymous"},
                        {"name": "Task Save Challenge", "value": "$formDataMap.phoneNumber"}
                ]
                },
```

```
...    ]
}
```

The following action properties are typically used for an anonymous form bundle:

- `{ "name": "Task Type", "value": "Anonymous" }`
- `{ "name": "Task Assign Email", "value": "$formDataMap.emailAddress" }`. Anonymous form bundles require an email to be set. If the value is blank, the Job Task Assign service throws an error. The email address can be set to a value or it can be read from a data extract of the triage form.
- `{ "name": "Task Input XML Prefill", "value": "$func.startSubmissionXml()" }`. The Anonymous form task gets the XML data from the previous submission. You can also use `$func.stepOrStartSubmissionXml()` instead. This is required if the step uses the `dynamicPreConditions`. For more information, see job functions.
- `{ "name": "Task Save Challenge", "value": "${formDataMap.phoneNumber}" }`. The save challenge data is set for the form.

NOTE

It is possible for anonymous tasks to copy attachments from a previous task, but sharing of attachments is not supported in form bundles. For more information, see action properties.

Example 1

The Job Controller below has been extracted from the mOnboarding - Anonymous - shareExtractData job definition JSON above with the following step level attributes: `dynamicPreConditions`, `shareExtractData`, as well as the standard ones: `allFormsEditable`, `showPreviousForms` and `redirectNext`.

```
"dynanicPreConditions": true,
"shareExtractData": true,
"shareFormData": false,
"allFormsEditable": true,
"showPreviousForms": true,
"redirectNext": true,
```

Example 2

In this example, the job controller is extracted from the mOnboarding - Anonymous - shareFormData job definition above with the following step attributes:

- `dynamicPreConditions`, `ShareFormData`
- `allFormsEditable`,
- `showPreviousForms`
- `redirectNext`.

```
"dynanicPreConditions": true,
"shareExtractData": true,
"shareFormData": false,
"allFormsEditable": true,
"showPreviousForms": true,
"redirectNext": true,
```

Next, learn about authenticated form bundles.

Last updated on : 21 October 2019

Authenticated Form Bundles

You can enforce an authenticated access to form bundles by using authenticated tasks that are assigned to a group. Authenticated form bundles can be built based on a form and Review Tasks types. However, we recommend you use the Form Task types.

Authenticated tasks can be one of the following:

User Task

The current user is assigned to a new task:

```
{"name": "Task Assign User", "value": "$func.startUser()" }
```

Alternatively

```
{"name": "Task Assign User", "value": "$func.previousUser()" }
```

Non Claimable Tasks

This task is assigned to a group the current user belongs to, for example, the accounts group:

```
{"name": "Task Assign Group", "value": "accounts" }
```

It can be assigned to two groups, if there is a supervisor group for the task:

```
{"name": "Task Assign Groups", "value": "accounts,supervisor" }
```

Claimable Group Tasks

This is also required to be claimed by the current user, which is required for the `redirectNext` to work:

```
{"name": "Task Assign User", "value": "$func.startUser()" }
```

```
{"name": "Task Assign Group", "value": "accounts" }
```

If there is a supervisor group for the task:

```
{"name": "Task Assign User", "value": "$func.previousUser()" }
```

```
{"name": "Task Assign Groups", "value": "accounts,supervisor" }
```

> NOTE
>
> A hard-coded address is not recommended.

Job Definition - Additional Products

The job definition for additional products shows the form bundle configuration below:

```
{
        "name": "Additional Products",
        "type": "",
        "dynanicPreConditions": true,
        "shareExtractData": true,
        "shareFormData": false,
        "allFormsEditable": true,
        "showPreviousForms": true,
        "redirectNext": true,
        "actions": [
                {
                        "name": "Credit Cards Application",
                        "type": "Job Task Assign",
                        "preCondition": "$formDataMap.productCreditCards == 'true'",
                        "redirectNext": true,
                        "properties":[
                                {"name": "Task Assign User", "value": "$formDataMap.startUser()"},
                                {"name": "Task Form Code", "value": maestroonboardingcred2},
                                {"name": "Task Message", "value": "Please complete the Credit Card Appli
form."},
                                {"name": "Task Subject", "value": "Complete Credit Card Application"},
                                {"name": "Task Input XML Prefil", "value": "$func.stepOrStartSubmissionX
                                {"name": "Task Type", "value": "Form"},
                        ]
                },
                {
                        "name": "Insurance Application",
                        "type": "Job Task Assign",
                        "preCondition": "$formDataMap.productInsurance == 'true'",
                        "redirectNext": true,
                        "properties": [
                                {"name": "Task Assign User", "value": "$formDataMap.startUser()"},
                                {"name": "Task Form Code", "value": maestroonboardinginsu2},
                                {"name": "Task Message", "value": "Please complete the Insurance Applica
                                {"name": "Task Subject", "value": "Complete Insurance Application"},
                                {"name": "Task Input XML Prefil", "value": "$func.stepOrStartSubmissionX
                                {"name": "Task Type", "value": "Form"},
                        ]
                },
...     ]
}
```

Example 1

The job controller below has been extracted from the mOnboarding - Form Tasks - shareExtractData job definition. Its features setup in the with step level attribute flags `dynamicPreConditions`, `shareExtractData`, as well as the standard `allFormsEditable`, `showPreviousForms` and `redirectNext`.

All form tasks in the form bundle require action properties, such as:

- `{ "name": "Task Type", "value": "Form" }`
- `{ "name": "Task Input XML Prefill", "value": "$func.stepOrStartSubmissionXml()" }` or `{ "name": "Task Input XML Prefill", "value": "$func.stepOrPreviousSubmissionXml()" }`

> NOTE
>
> `$func.stepOrStartSubmission` or `$func.stepOrPreviousSubmissionXml()` are required for dynamic preconditions to work. They also work for the static bundles.

Example 2

This example uses a form task bundle configured with the `shareFormData` attributes:

```
"shareExtractData": true,
"shareFormData": false
```

Review Task

Review tasks are primarily used for review and approval processes that use the same form to review as was submitted by the user. They are configured by specifying a previous step or submission. The review task will copy the XML data, attachments from the triage form. You need to specify a form code so the review task doesn't use the triage form. Review tasks should be only configured with the `shareExtractData` option.

Example 3

This example is extracted from mOnboarding - Review Task - shareExtractData. The attributes are the same as in the job definition of the Example 1.

The Task Type should be Review:

`{ "name": "Task Type", "value": "Review" }`

The following code snippet retrieves the review data from a previous step:

`{ "name": "Task Review Previous Step", "value": "true" }`

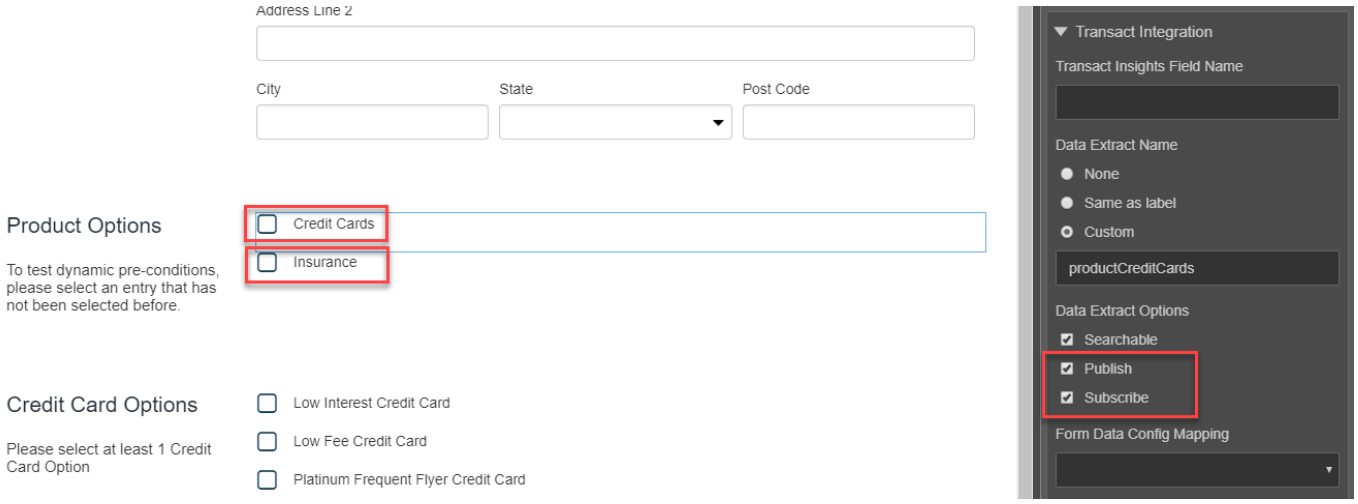
```
  "dynanicPreConditions": true,
  "shareExtractData": true,
  "shareFormData": false,
  "allFormsEditable": true,
  "showPreviousForms": true,
  "redirectNext": true
```

Next, learn about anonymous form bundles.

Form Bundle Flow Example

The following example illustrates the simple usage of a form bundle as an anonymous form bundle flow that uses nearly all `redirectNext`, `sharedExtractData` (from a users perspective the `shareFormData` looks the same), `dynamicPreConditions`, `preconditions`, `allFormsEditable`.

The test cases extend the Maestro onboarding example where the Credit Card Application and Insurance Application forms have been replaced with forms that include the Product Options to allow you to test the `dynamicPreCondition` attribute. The Product options - Credit Card and insurance check boxes are set to both Publish and Subscribe. This allows you to test the `shareExtractData` step attribute. The Maestro build form is shown below.



The Flow

A form user fills out the Triage form and submits it.

MAGUIRE FINANCIAL

Reference Code: 2VKK4JT

Save and Close | Cancel / Exit | Open Saved Form

Getting Started

Maestro Onboarding - Triage

*Fields marked with * are required*

OK, Lets Get Started!

We make it easy to apply online and it won't take long, so **let's get going...**

About You

Please enter you details

First Name *: John

Last Name *: Smith

Phone Number (AU) *: 0299764500

Email Address *: jsmith@company.com

Please input the address

Start typing full address here..

Product Options

Please select at least 1 Product Option

- [x] Credit Cards
- [ ] Insurance

Submit

© Copyright Avoka Technologies 2018 ~ Privacy Policy

Manager executes the Additional Package step. Preconditions match for Credit Cards and the Confirmation Page and tasks are created.

The browser is redirected to the next available task, which is the Credit Card.

The user selects the first credit card application and clicks Submit.

MAGUIRE FINANCIAL

Reference Code: GMQTHZX

Save and Close | Cancel / Exit | Open Saved Form

Form Bundle

MTest Onboarding - Credit Cards

*Fields marked with * are required*

About You

Please enter the mandatory fields.

First Name *: John

Last Name *: Smith

Phone Number (AU) *: 0299764500

Email Address *: jsmith@company.com

Please input the address

Start typing full address here..

Product Options

To test dynamic pre-conditions, please select an entry that has not been selected before.

- [x] Credit Cards
- [ ] Insurance

Credit Card Options

Please select at least 1 Credit Card Option

- [x] Low Interest Credit Card
- [ ] Low Fee Credit Card
- [ ] Platinum Frequent Flyer Credit Card

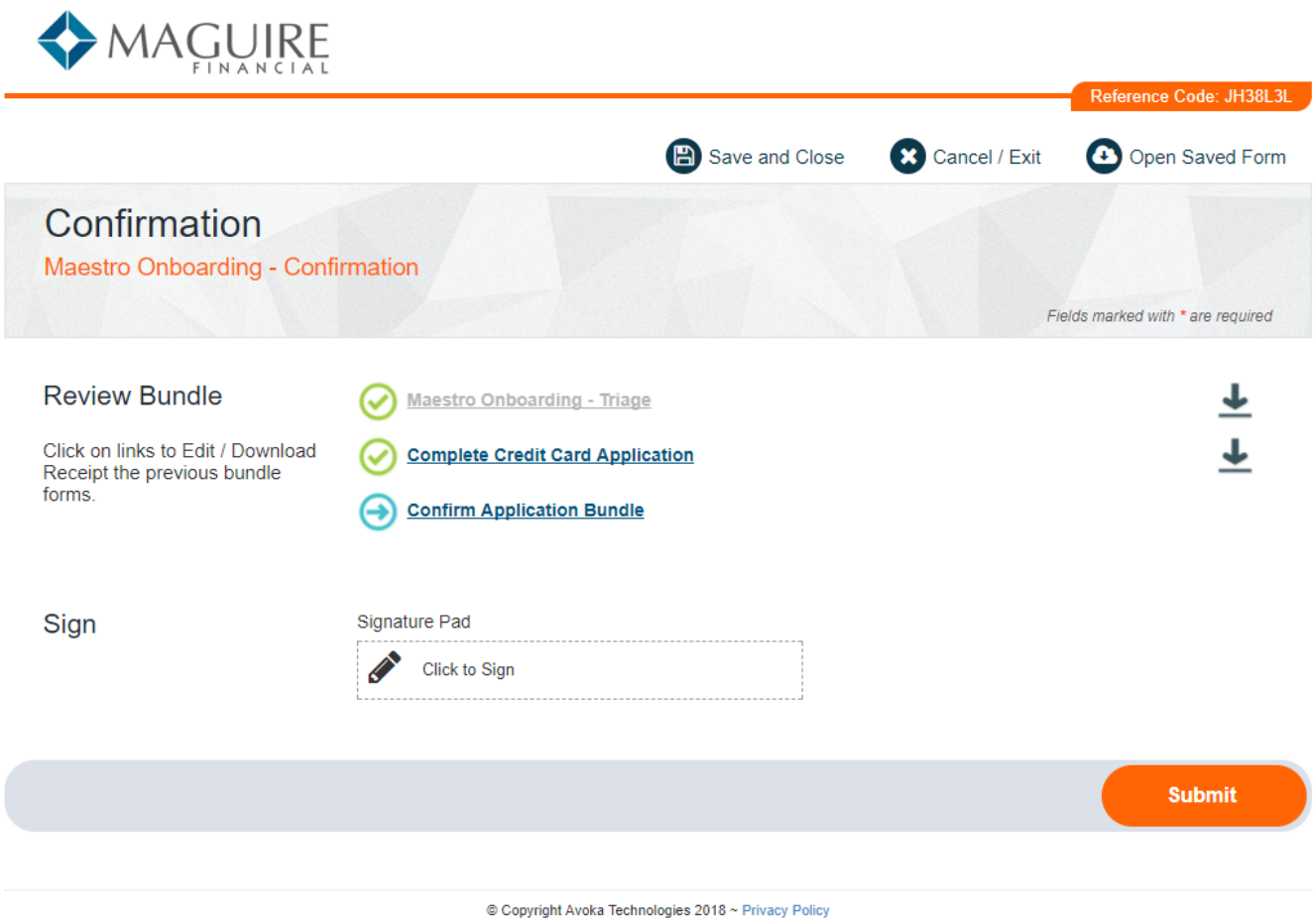
Submit

© Copyright Avoka Technologies 2018 ~ Privacy Policy

The browser is redirected to the Application Confirmation page, so you can see the Triage Form (Grayed out - it is outside the step and is read only, for example, print receipt)

You can see the Credit card for has been completed.

Click on the link to edit it. The `allFormsEditable` is set to true in the Additional Products step.



The browser redirects back to the Credit Card Page. This shows the behavior of `dynamicPreConditions`.

Select Product Options: Insurance and click Submit.

The data extracts for the Product Options insurance are set to true and the precondition for the insurance task is evaluated and an Insurance Task created.

MAGUIRE FINANCIAL

Reference Code: GMQTHZX

Save and Close | Cancel / Exit | Open Saved Form

Form Bundle

MTest Onboarding - Credit Cards

*Fields marked with * are required*

About You

Please enter the mandatory fields.

First Name *: John

Last Name *: Smith

Phone Number (AU) *: 0299764500

Email Address *: jsmith@company.com

Please input the address

Start typing full address here..

Product Options

To test dynamic pre-conditions, please select an entry that has not been selected before.

- [x] Credit Cards
- [x] Insurance

Credit Card Options

Please select at least 1 Credit Card Option

- [x] Low Interest Credit Card
- [ ] Low Fee Credit Card
- [ ] Platinum Frequent Flyer Credit Card

Submit

© Copyright Avoka Technologies 2018 ~ Privacy Policy

The browser redirects next to the Insurance Page. This is because the order of the Insurance is above the confirmation in the Job Action List

Clear the Product Options - Credit Card, Check the Home Contents and Insurance then submit

MAGUIRE FINANCIAL

Reference Code: 84CPMCC

Save and Close | Cancel / Exit | Open Saved Form

Getting Started

Maestro Onboarding - Insurance

*Fields marked with * are required*

About You

Please enter your details

First Name *: John

Last Name *: Smith

Phone Number (AU) *: 0299764500

Email Address *: jsmith@company.com

Please input the address

Start typing full address here..

Product Options

To test dynamic pre-conditions, please select an entry that has not been selected before.

- [ ] Credit Cards
- [x] Insurance

Insurance Options

Please select at least 1 Insurance Options

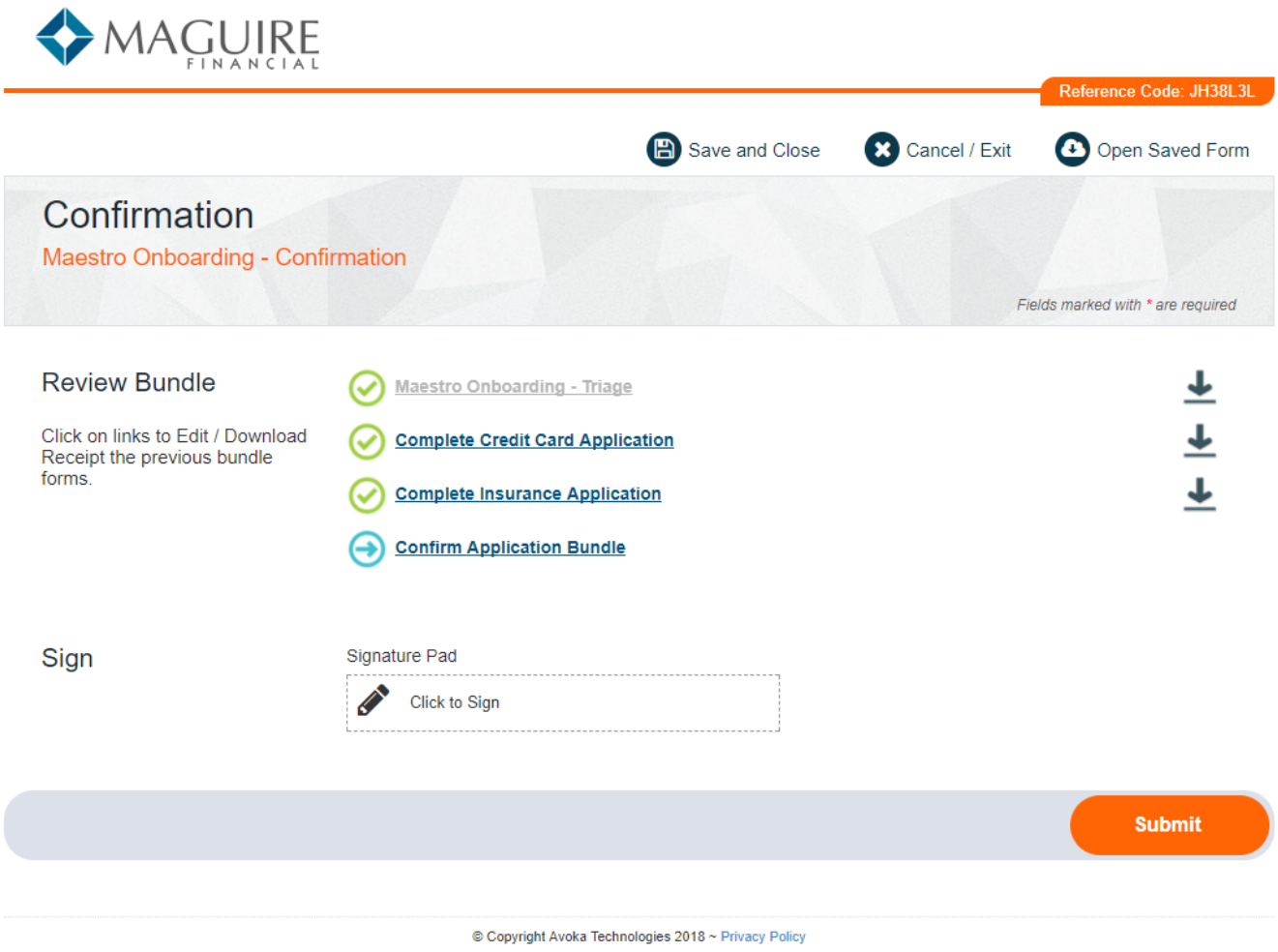
- [x] Home & Contents Insurance
- [ ] Income Insurance
- [ ] Life Insurance

Submit

© Copyright Avoka Technologies 2018 ~ Privacy Policy

The browser is redirected to the confirmation page, which shows the list of the forms and the insurances.

Click Complete Credit Card Application.



The browser is redirected to the Credit Card form. The Credit Card product option is cleared. The is done by the `shareFormData` or shareExtractData.

The credit card form isn't removed. The precondition is only evaluated once. A form user can click Cancel to remove them.

Click Submit

MAGUIRE FINANCIAL

Reference Code: GMQTHZX

Save and Close | Cancel / Exit | Open Saved Form

Form Bundle

MTest Onboarding - Credit Cards

*Fields marked with * are required*

About You

Please enter the mandatory fields.

First Name *: John

Last Name *: Smith

Phone Number (AU) *: 0299764500

Email Address *: jsmith@company.com

Please input the address

Start typing full address here..

Product Options

To test dynamic pre-conditions, please select an entry that has not been selected before.

- [ ] Credit Cards
- [x] Insurance

Credit Card Options

Please select at least 1 Credit Card Option

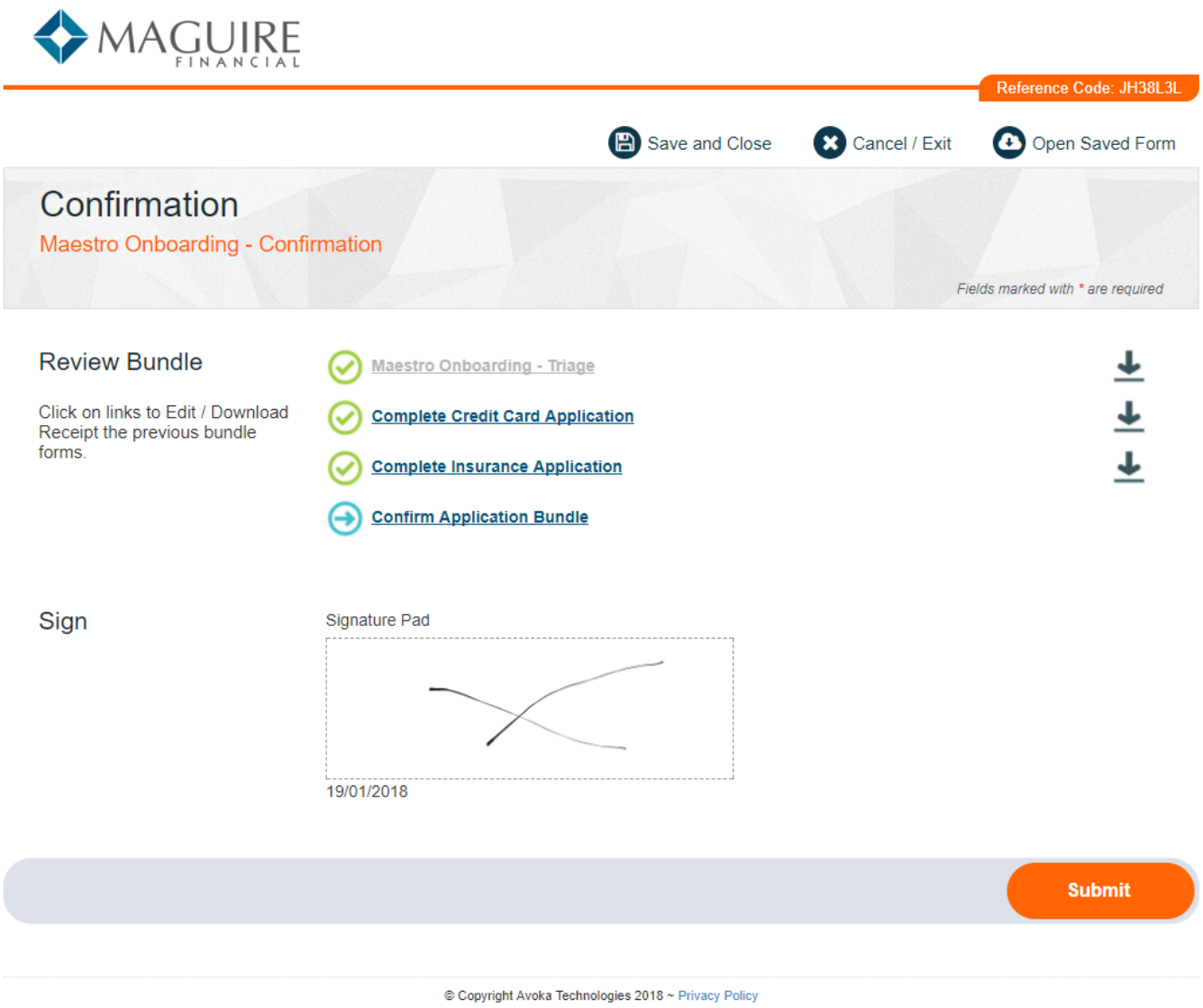
- [x] Low Interest Credit Card
- [ ] Low Fee Credit Card
- [ ] Platinum Frequent Flyer Credit Card

Submit

© Copyright Avoka Technologies 2018 ~ Privacy Policy

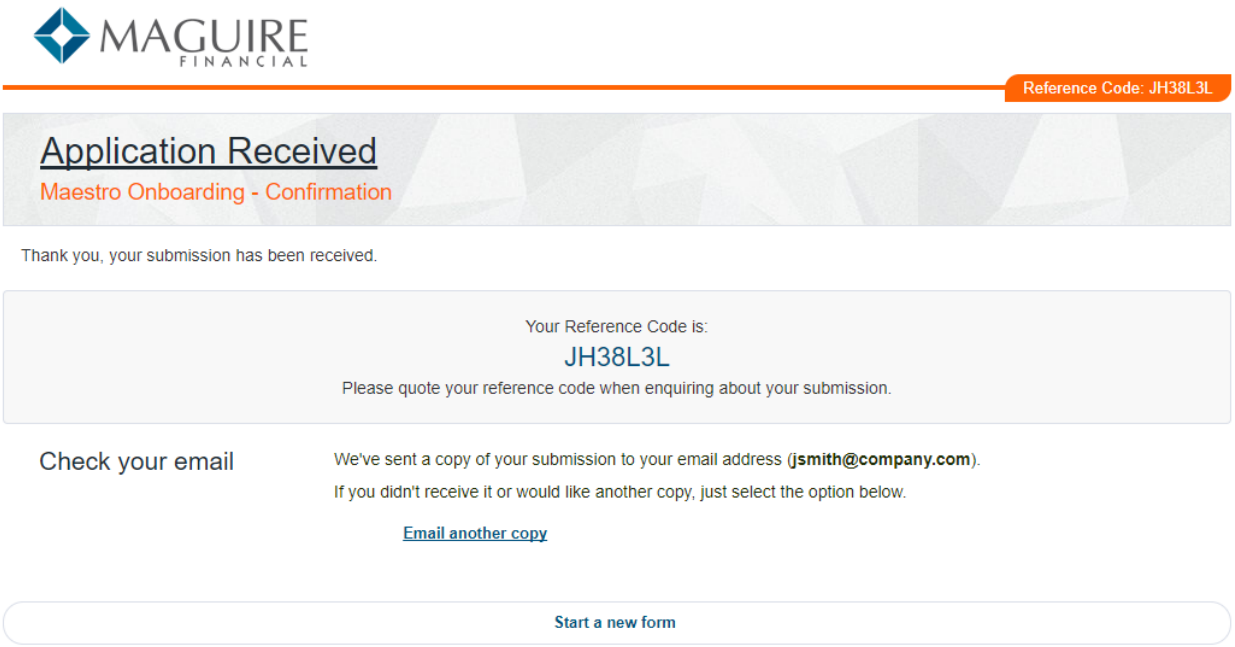
The browser is redirected back to the Confirmation page

Sign the form and click Submit.



The form goes to the Application Received page, which informs the user that a receipt will be sent via email.

This is standard form processing where form chaining was turned off because the `redirectNext` attribute is not set on the Maestro Onboarding - confirmation page.



The form flow above can be bundled up to make it look like it's one form by applying the following minor changes:

- Change the submit button label to Next on all but the final Application Confirmation form.
- Add a back button to go to the previous form.
- Use the Custom Navigator Maestro Widgets that allow a user to easily jump to any part of the form.

This example uses the following forms, which are part of the standard Maestro Onboarding example:

- Maestro Onboarding - Confirmation
- Maestro Onboarding - Credit Cards
- Maestro Onboarding - Insurance
- Maestro Onboarding - Triage

These are two forms that substitute the equivalent standard form:

- MTest Onboarding - Credit Cards
- MTest Onboarding - Insurance

These are used, instead of the standard triage form, to start authenticated bundle tests of type Form and Review:

- MTest Onboarding - Triage Form
- MTest Onboarding - Triage Review

The example uses the following services:

- Maestro Onboarding - v17.10.0 - The standard job controller, Anonymous with shareFormData, Not Dynamic
- Maestro Onboard-Anony-SharedExtractData - v17.10.0 - Anonymous with shareExtractData, Not Dynamic
- mOnboarding - Anonymous - Dynamic - shareExtractData - v17.10 - Anonymous with shareExtractData, Dynamic
- mOnboarding - Anonymous - Dynamic - shareFormData - v17.10.0 - Anonymous with shareFormData, Not Dynamic
- mOnboarding - Anonymous - No Dynamic - v17.10.0 - Anonymous No sharing, Not Dynamic
- mOnboarding - Form Tasks - shareExtractData - v17.10.0 - Authenticated Form Task, shareExtractData, Dynamic
- mOnboarding - Form Tasks - shareFormData - v17.10.0 - Authenticated Form Task, shareFormData, Dynamic
- mOnboarding - Review Task - shareExtractData - v17.10.0 - Authenticated Review Task, shareExtractData, Dynamic

Download

You can download the complete code example here.

Next, learn about authenticated form bundles

Form Chaining Overview

The type of UI flow mentioned in form bundles is often referred to as form chaining. Its possible to configure a collaboration job to have one task created per one step and still implement chaining between the steps. The `redirectNext` is both a step and an action attribute. If you want to the form chaining to be used within steps, set the step attribute `redirectNext` to `true`. To allow the form chaining from one action to another, set the action attribute `redirectNext` to `true`.

Let's look at the customer onboarding collaboration job. The job definition's additional products step configuration is shown below.

```
{
  "name": "Additional Products",
  "type": "",
  "shareExtractData": true,
  "allFormsEditable": true,
  "showPreviousForms": true,
  "redirectNext": true,
  "actions": [
    {
      "name": "Credit Cards Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productCreditCards == 'true'",
      "redirectNext": true,
      "properties": [
        { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
        { "name": "Task Form Code",  "value": "onboard-credit-cards" },
        { "name": "Task Message",  "value": "Please complete the Credit Card Application
form." },
        { "name": "Task Subject",  "value": "Complete Credit Card Application" },
        { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
        { "name": "Task Type",  "value": "Anonymous" }
      ]
    },
    {
      "name": "Insurance Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productInsurance == 'true'",
      "redirectNext": true,
      "properties": [
        { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
        { "name": "Task Form Code",  "value": "onboard-insurance" },
        { "name": "Task Message",  "value": "Please complete the Insurance Application
form." },
        { "name": "Task Subject",  "value": "Complete Insurance Application" },
        { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
        { "name": "Task Type",  "value": "Anonymous" }
      ]
    },
    {
      "name": "Appliction Confirmation",
      "type": "Job Task Assign",
```

```
      "preCondition": "$formDataMap.productInsurance == 'true' || $formDataMap.-
productCreditCards == 'true'",
      "properties": [
        { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
        { "name": "Task Form Code",  "value": "onboard-confirmation" },
        { "name": "Task Message",  "value": "Please sign to complete Application bundle."
},
        { "name": "Task Subject",  "value": "Confirm Application Bundle" },
        { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
        { "name": "Task Type",  "value": "Anonymous" }
      ]
    },
    {
      "name": "Review Wait",
      "type": "Job Task Wait"
    }
  ],
  "routes": [
    { "name": "Default",  "nextStep": "Application Delivery" }
  ]
},
```

Here, you can see the steps has "redirectNext": true. The first 2 Task Actions,Credit Card Application and Insurance Application Forms, have redirect next "redirectNext": true. The last Action Application Confirmation does not have a redirectNext attribute or "redirectNext": false. The Triage form is filled in, depending upon the user selection in the triage form and the either or both Credit Card and Insurance preCondition will evaluate to true.

Lets assume that the both products selected. The user submits the triage form it chains to the Credit Card. The order for the next available form is determined by the order of the Action in the Step that is Triage Form > Credit Card > Insurance > Confirmation. The form chaining continues to the confirmation page. However at the confirmation page it breaks the chaining flow and goes to the Application Confirmation form's - Submission Confirmation page. At this page the user can download a combined PDF receipt.

Change of User

Form chaining can happen only when the current user has access to newly created tasks:

1. Anonymous Forms the email address needs to stay the same.
2. Authenticated task either:
    - the current user is assigned to the new task
    - the task is assigned to a group that the current user belongs to
    - group tasks are claimable are also required to be claimed by the current user.

When there is a change of assignment that does not meet the conditions above, additional tasks cannont be openned by the current user. The current form stops chaining and goes to the forms submission confirmation page.

Legacy Mode

There is an legacy UI flow that was in earlier revisions of form bundles 4.2 did not load the next available form, that is did not perform form chaining.

When a user submits the submission the user is then presented with a list of forms on the forms submission confirmation page, one for each form that they were to completed.

> NOTE
>
> We recommend to use the form chaining because it provides a better user experience.

Next, learn about a form bundle.

Transact Functions Overview

Journey Manager allows you to create services as Transact functions, which can be called from Maestro forms.

You should consider calling Transact functions from a Maestro form to:

- Use the rich Fluent API, as opposed to JavaScript[1] calls.
- Implement complex business logic that requires extensive coding, so Maestro forms can only use simple JavaScript calls to leverage on them.
- Make code more secure because it resides on the server side, not on the client side, which is a Maestro form.
- Provide better integration with back end, such as database, because it resides on the server side.

Manager defines the following Transact functions:

- Fluent Function, also know as Form Function
- Delivery Function

> NOTE
>
> Both Fluent and Delivery Transact functions are Fluent Functions as they implement the `com.avoka.fc.core.service.fluent.FluentFunctionService` Fluent Service.

Let's look at each of these functions in more detail.

Fluent Function

A Fluent Function, also known as a Form Function, provides an improved programming model for client applications to call server-side business functions. The Form Function is designed to make calls from form applications to other systems to execute a remote data look-up or some secure business logic on a remote server.

A Fluent Function is a replacement for Form Dynamic Data Services with an improved programming and security model. It provides server-side function event handlers for a variety of form event's triggers. The Fluent Function programming interface is the same across a

---

[1]Javascript (JS) is a scripting languages, primarily used on the Web. It is used to enhance HTML pages and is commonly found embedded in HTML code. JavaScript is an interpreted language. Thus, it doesn't need to be compiled.

variety of function triggers enabling you to re-use function's code in a variety of contexts. For example, you could have a single function for doing both data pre-fill when a form is first opened and when a saved form is resumed.

Delivery Function

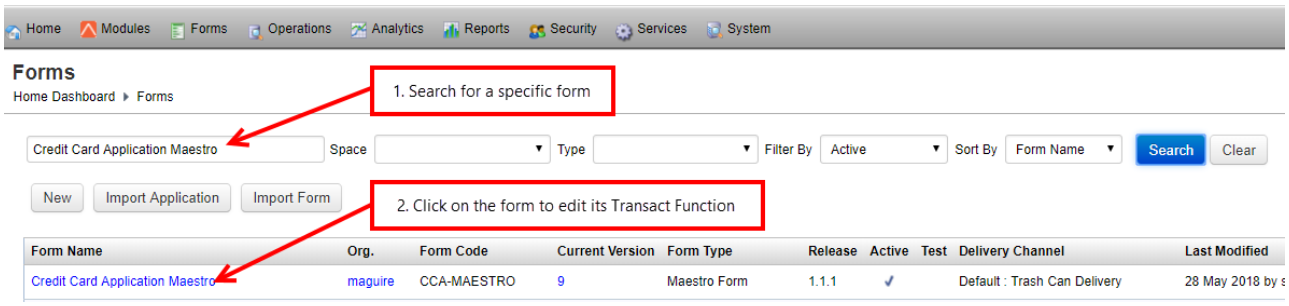
A Delivery Function is used to deliver a form data entered by a user when a form is submitted. The form data includes a submitted form XML, a PDF receipt document and many also include user uploaded file attachments.

How It Works

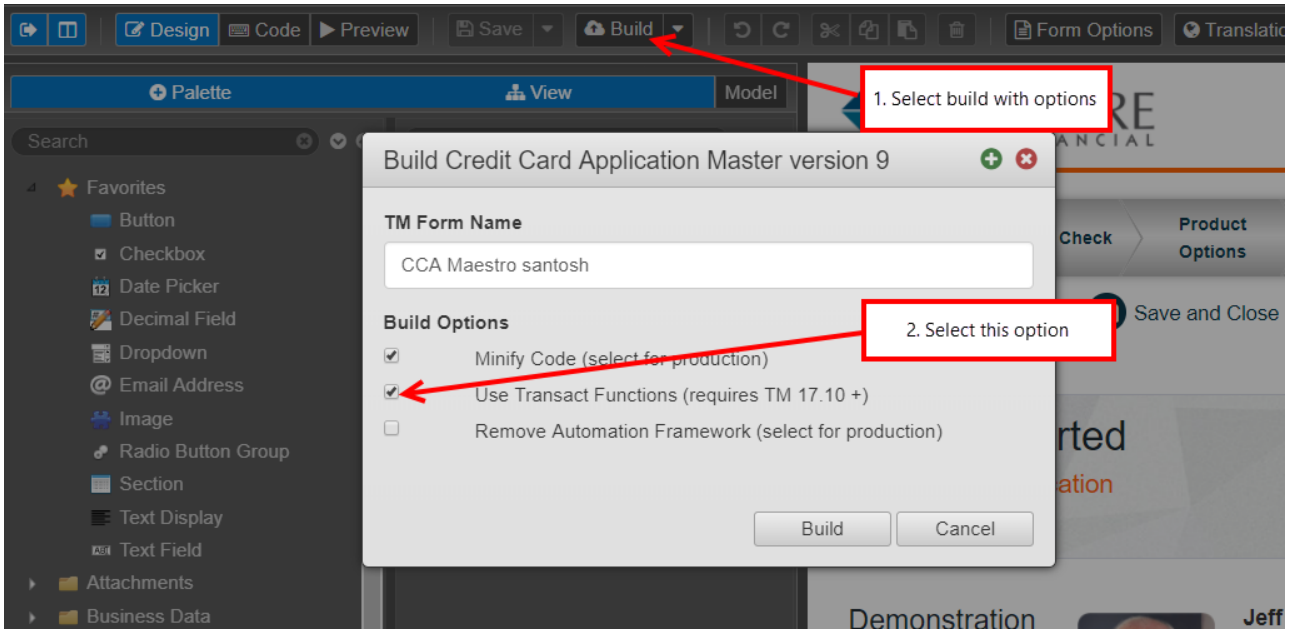
Transact functions are configured in Manager and they can be used from Maestro forms.

To update a Transact function:

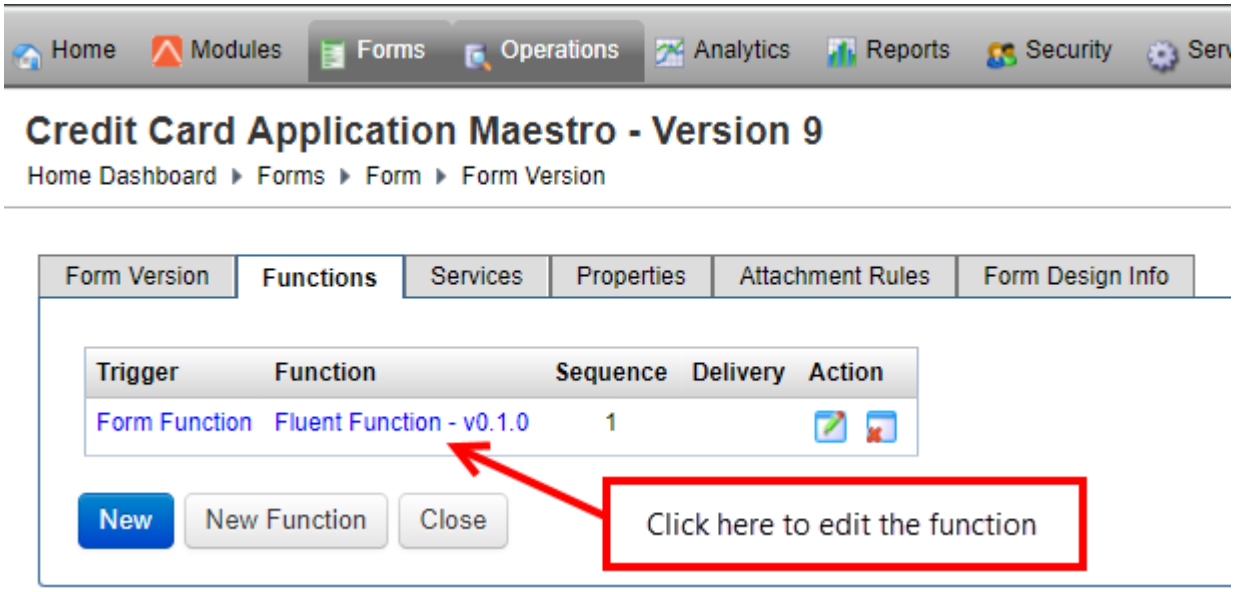
1. Select Forms > Forms and search for a form.



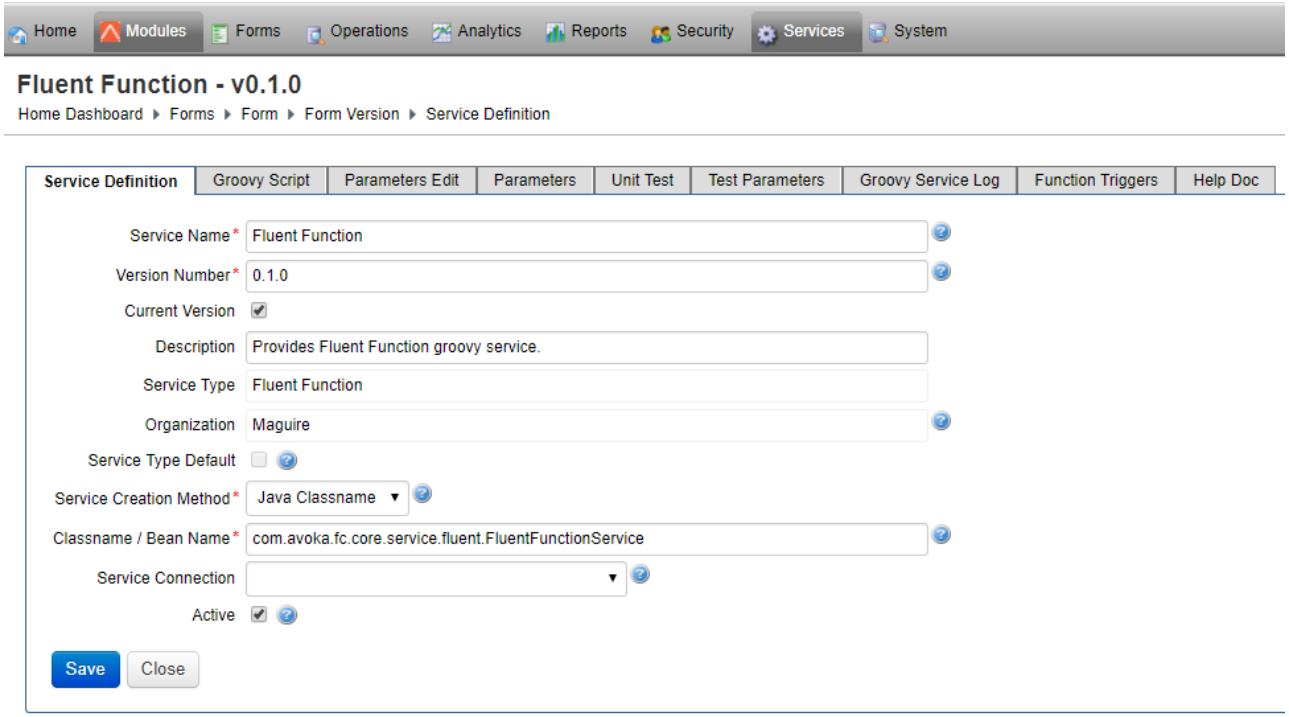
2. Click the Functions link to configure a Transact function.
3. If the form does not have a function associated to it, you need to re-build it in Journey Maestro with the Use Transact Functions option.



4. Refresh your form view in Manager. You should see the Functions link available now.
5. Click the link to continue the configuration. You will be presented with the Form Versions tab from where you can configure the Transact function.



6. Complete the Transact function configuration.



As Transact functions may have complex implementations, it can be difficult and time consuming for you to troubleshoot any errors associated with forms that use these Transact functions. Manager provides you with a quick way to access service logging information, using Groovy Service Log, where you can find some key Transact function stats generated during its execution.

Manager allows you to view, create, copy, configure, and remove Transact functions. You can use this functionality to manage the Transact functions life cycle.

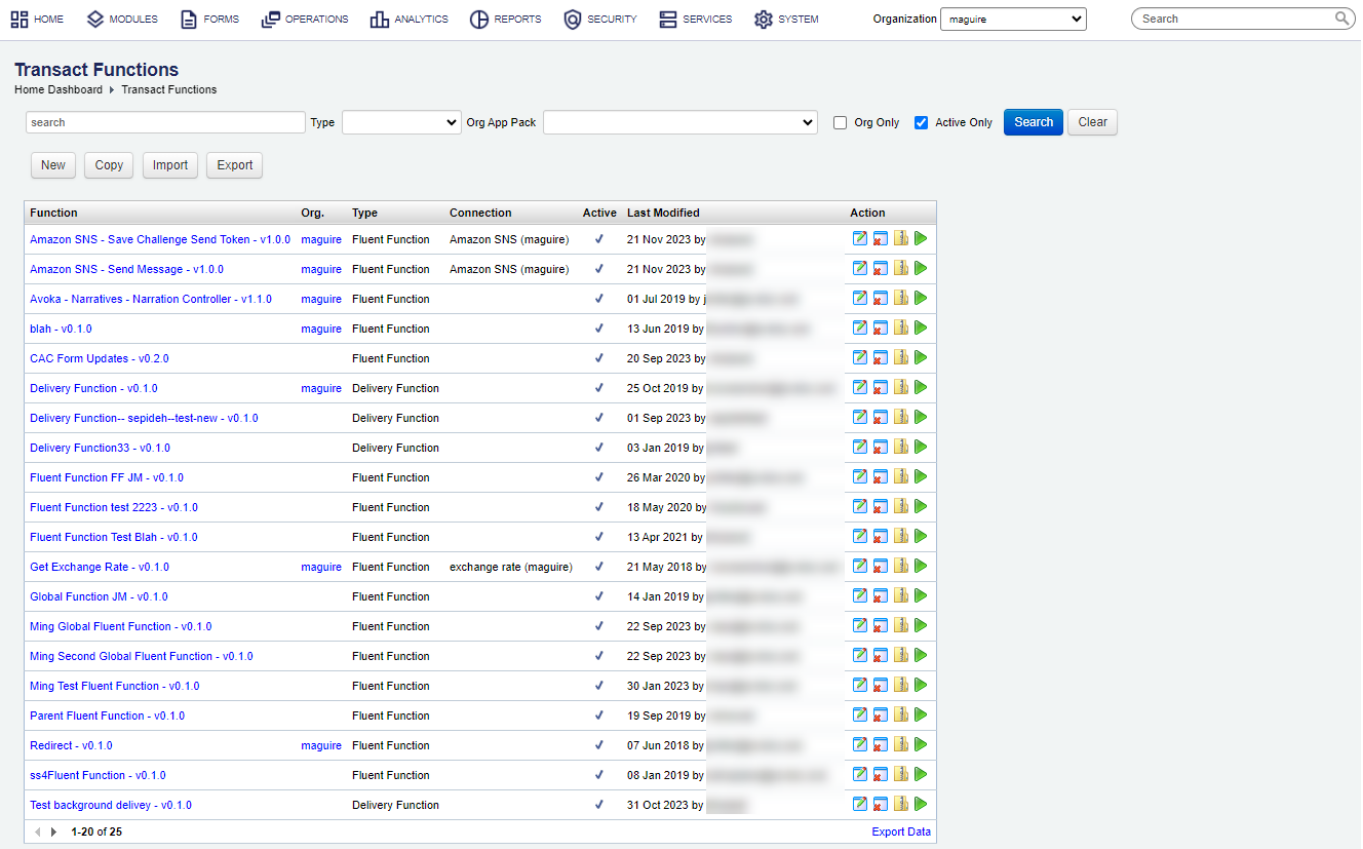
Next, learn how to view Transact functions.

View Transact Functions

Manager allows you to view Transact Functions, so you can manage them and their life cycle.

To view all Transact functions, select Services > Transact Functions.

Transact functions are shown page by page, with the maximum number of records per page configured in the users preferences.



The Transact Function list displays the following details:

- Function - a name of the Transact function.
- Org. - an organization this Transact Function belongs to.
- Type - a type of a Transact Function, which can be one of the following:
    - Fluent Function - A Fluent Function, also known as a Form Function, provides an improved programming model for client applications to call server-side business functions. The Form Function is designed to make calls from form applications to other systems to execute a remote data look-up or some secure business logic on a remote server.

    - Delivery Function - A Delivery Function is used to deliver a form data entered by a user when a form is submitted. The form data includes a submitted form XML, a PDF receipt document and many also include user uploaded file attachments.
- Connection - a service connection that a Transact Function uses.
- Active - whether a Transact function is active or not.
- Last Modified - a date when a Transact Function was last modified and a user account who modified it.

Searching and filtering of Transact functions become important when there are many functions created in Manager.

To filter or search Transact functions by one or more criteria, specify the following settings and click Search:

- Name or part of the name of the Transact function.
- Type of a Transact function.
- Org App Pack to filter functions by organization application pack.
- Select the Org Only. checkbox to list Transact functions that have their organizations configured. Otherwise, everything is listed.
- Select the Active Only checkbox to list Transact functions that are active.

> NOTE
>
> We don't recommend removing a Transact function but making it inactive instead. You can also use this setting in the Transact function rollback process.

Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

You can also import and export Transact functions.

> NOTE
>
> You can list all Transact Function by running a Groovy script in the Groovy console.

Next, learn how to create a Transact function.

Create a Transact Function

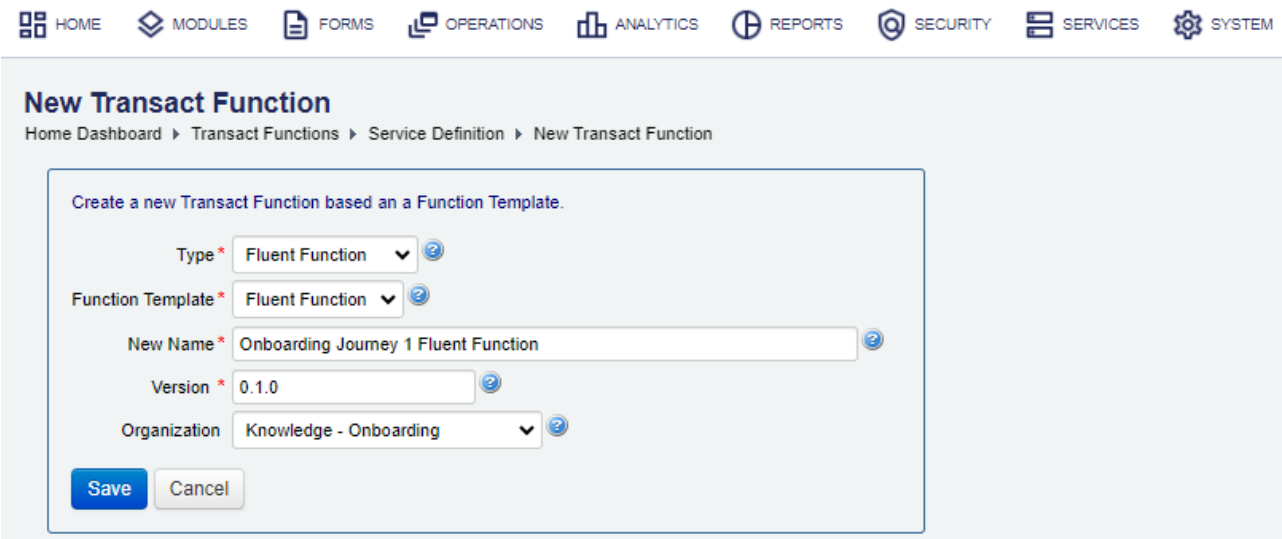
Manager allows you to create Transact functions, which are called within a Journey Manager server instance or from a Maestro form running in a browser on a client side.

Behind the scenes, Manager generates a `service-def.json` file as well as a Groovy service source file and unit test source file, based on a chosen template. You then implement required business logic in the Groovy script file by extending the `com.avoka.fc.core.service.fluent.FluentFunctionService` class using Fluent API.

> NOTE
>
> You can't create a Transact function via Fluent API, but you can do it by calling the `svc-scaffold` scaffolding task via Journey SDK. For more information, see svc-scaffold.

To create a Transact function:

1. Select Services > Transact Functions and click New.



2. Select a type of the Transact function from the Type dropdown list. A type allows you to filter a suitable function template that you wish to use to create a Transact function from. A type can be one of the following:
   - Fluent Function - A Fluent Function, also known as a Form Function, provides an improved programming model for client applications to call server-side business functions. The Form Function is designed to make calls from form applications to other systems to execute a remote data look-up or some secure business logic on a remote server.

- Delivery Function - A Delivery Function is used to deliver a form data entered by a user when a form is submitted. The form data includes a submitted form XML, a PDF receipt document and many also include user uploaded file attachments.

3. Select a Transact function template, which is used to create a Groovy service source file and unit test source file, from the Function Template dropdown list. A template can be one of the following:
   - Fluent Function
   - Delivery Function
4. Provide a unique function name in the New Name field.
5. Specify the function version in the Version field.
6. Select an organization from the Organization dropdown list so the Transact function is associated only with forms and other entities belonging to that organization. If you do not select an organization, the Transact function will be globally available.

   > NOTE
   >
   > You must have global access to create global Transact functions, both Fluent Functions and Delivery Functions. | 23.10 This feature was introduced in 23.10

7. Click Save to update the changes.

Next, learn how to configure a Transact function.

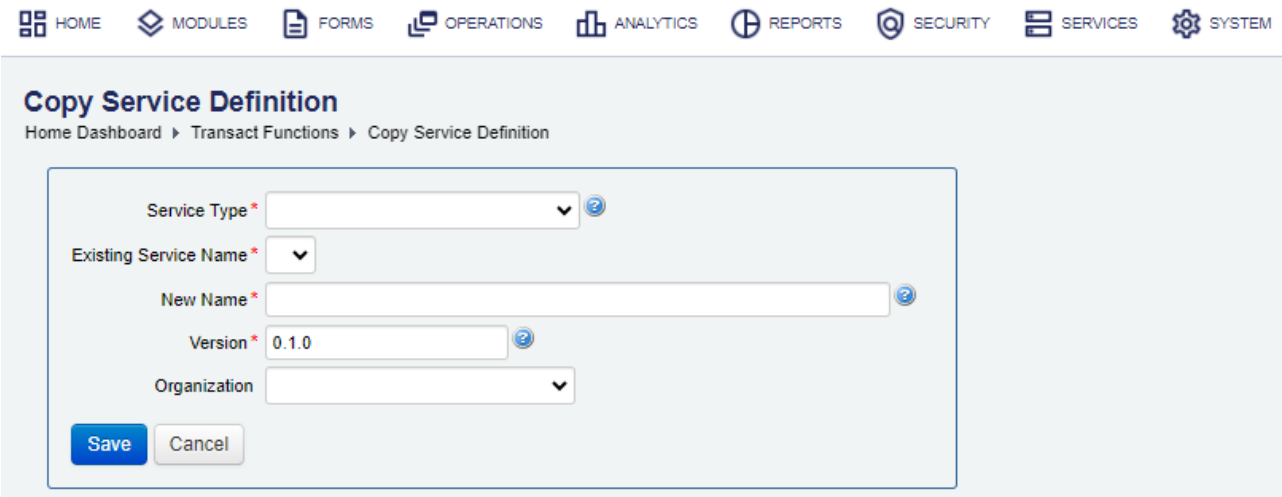
Copy a Transact Function

Manager allows you to copy a Transact function. You can use this functionality to:

- Create a new Transact function as a copy of another Transact function, which implements a different business logic.
- Create a new version of the existing Transact function.

To copy a Transact function:

1. Select Services > Transact Functions and click Copy.



2. Select a Transact function from the Service Type dropdown list:
   - Fluent Function
   - Delivery Function
3. Select a function name you want to copy from the Existing Service Name dropdown list, for example Func1 - v0.1.0.
4. Provide a unique function name in the New Name field.
5. Specify the function version in the Version field.
6. Select the organization from the Organization dropdown menu or leave it blank.
7. Click Save to update the changes.

Next, learn how to configure a Transact function.

Import a Transact Function

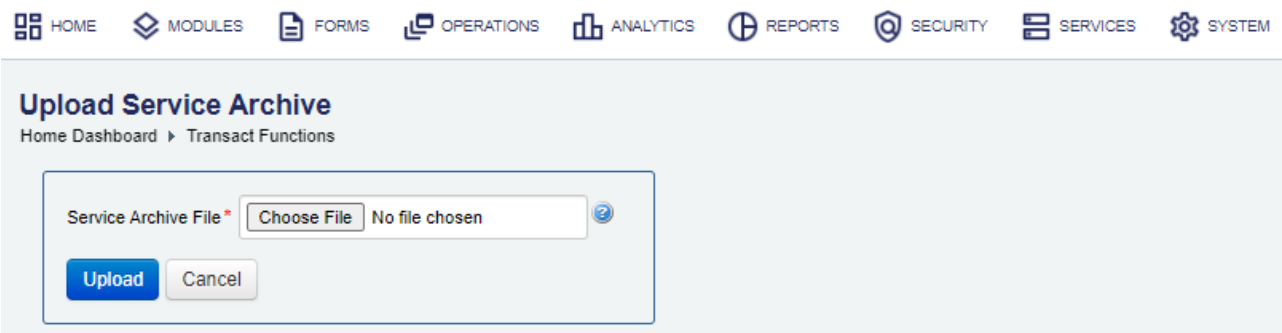
Manager allows you to import Transact functions. Use this functionality to:

- Rollback Transact functions.
- Re-create or promote Transact functions in another environment.

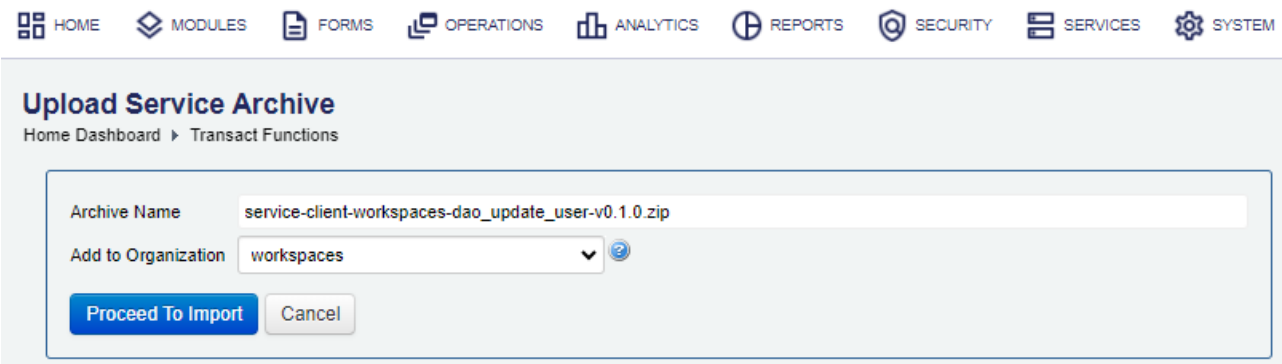
Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

To import a Transact function:

1. Select Services > Transact Functions and click Import.

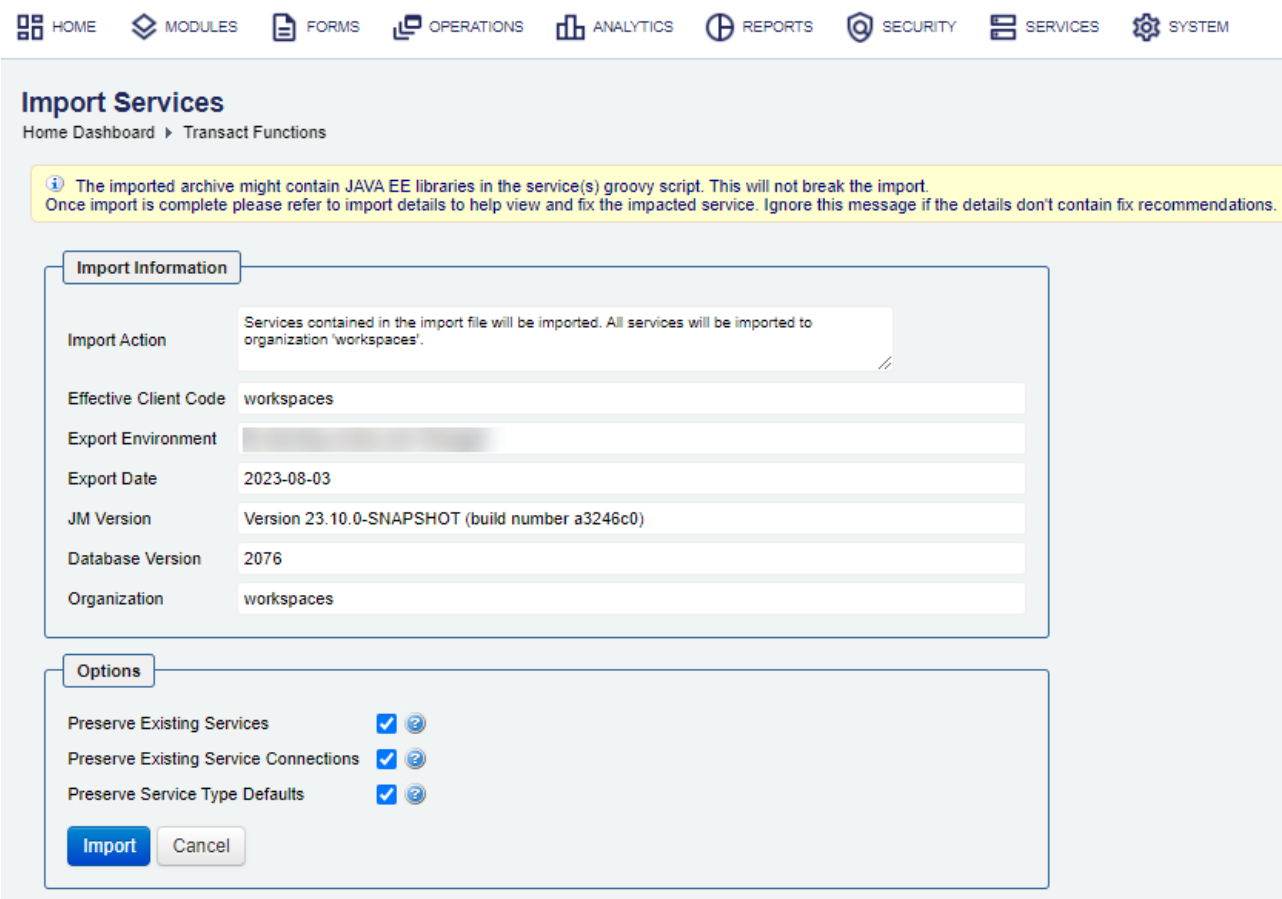


2. Click Choose File and choose a service archive ZIP file, which has been exported, for example, service-archive-all-2018-06-07.zip.
3. Click Upload.



4. Select an organization that you want to add the services to or use the organization already configured in the import archive file.

5. Click Proceed To Import to review the Import Options screen, as shown below.



6. Select the Preserve Existing Services checkbox to preserver any services that already exist on the target system, so they will not be modified in any way. Only new services will be imported. Note that services are resolved by name. (Default.)
7. Clear the Preserve Existing Services option on import if you are importing it as a rollback.
8. Select the Preserve Existing Service Connections checkbox to preserve any service connections that already exist on the target system, so they will not be modified in any way. New service connections will still be created. Note that service connections are resolved by name. This option is selected by default.
9. Select the Preserve Service Type Defaults checkbox to preserve types of the imported services, so they will be promoted to be the default for their service types, regardless of the default flags on the source system. If not selected, the service type default flag will be imported, and the existing services may be demoted if needed. This isn't selected by default.

10. Click Import to complete the import. The expected result page is shown below.

11. Select the Detail Messages tab to see more information about the import. This tab pops up open if the import operation has any issues you may need to check and resolve.
12. Click Close and check the list of Transact functions to verify the import was successful.

NOTE

To help you avoid enabling the Groovy Debug Logging option for a Groovy service or a function by mistake in a production environment, during the import, Manager checks if the Production Mode option is selected and then sets the Groovy Debug Logging option to false even if it was set to true in an archive file. | 23.04 This feature was introduced in 23.04

NOTE

You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

Next, learn how to create a Transact function.

Export a Transact Function

Manager allows you to export Transact functions. Use this functionality to:

- Backup or archive Transact functions.
- Migrate or promote Transact functions in another environment.

To export one or several Transact functions:

1. Select Services > Transact Functions.
2. Click Export Archive to export one Transact function or click Export to export all Transact functions.

Manager exports them into a service archive ZIP file, which will be downloaded to your PC. The ZIP file contains service-config.xml file with all Transact functions' definitions.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to import Transact functions.

Backup a Transact Function

It's recommended to back up Transact functions before you update them in Manager. This allows you to:

- Restore a Manager environment after a configuration change was rolled back.
- Re-created Transact functions in another environment.
- Promote or migrate Transact functions into a new environment.

Next, learn how to import Transact functions.

Rollback a Transact Function

The purpose of a rollback is to restore the target environment to its former state and should only be considered in a situation where the verification tests have determined that the promotion has encountered a significant failure of some kind.

Note that a rollback is not a decision that should be taken lightly, as it can require a lengthy and detailed process to restore components back to their previous state. Often it is preferable to adopt a fix-forward and rectify issues in situ if possible.

If a Transact function has been changed and it does not work as expected anymore, you can try to roll it back to the previous working version. This is only possible if:

- There are backups of previous Transact functions.
- There are records of changes made to the Transact function, known as the changelog[1].

To rollback a Transact function:

1. Locate the previous Transact function backup, which is an exported ZIP archive file.
2. Import it into Manager.
3. Verify that it works as expected.
4. Update your changelog.

Next, learn how to backup Transact functions.

[1]is a record of changes to a software project or other technology product. This type of record has its own format for various types of projects, but is generally useful in showing when and why changes were made, and who made them.

Remove a Transact Function

Manager allows you to remove a Transact function. You may want to do this if:

- It is incorrectly configured
- It is not required any more

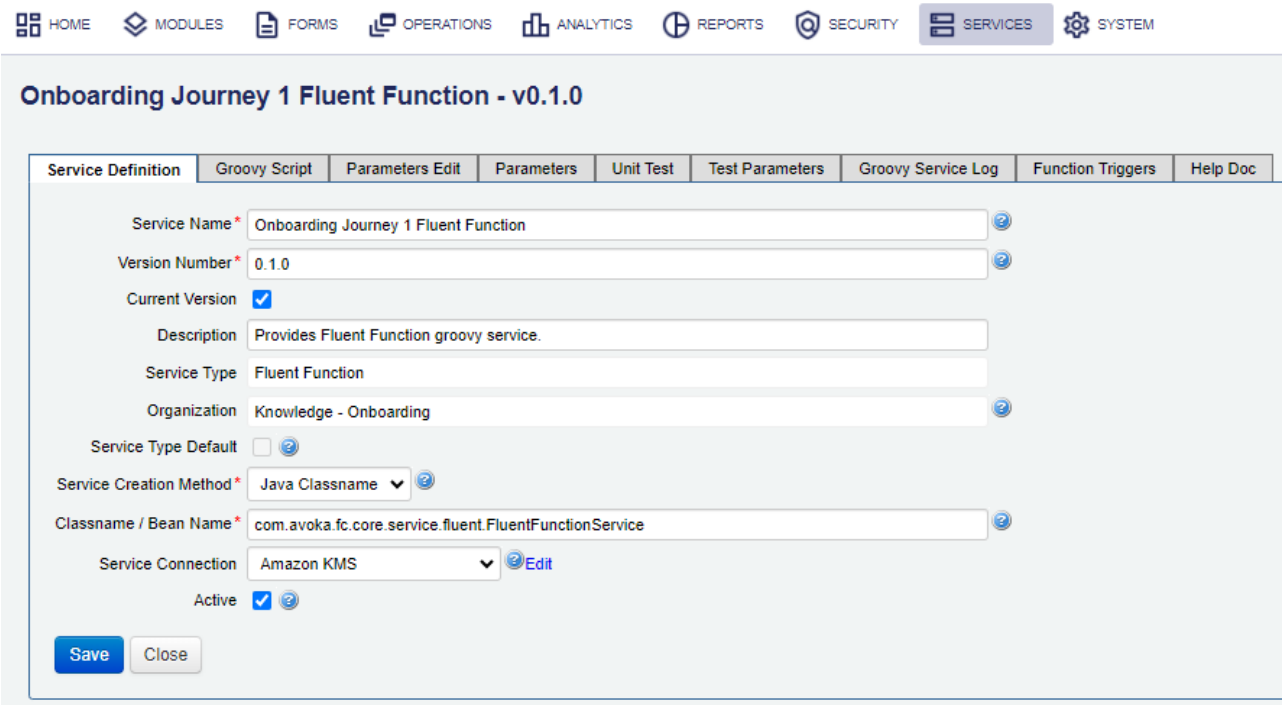
> NOTE
>
> We recommend deactivating a Transact Function rather than removing it. Also, you can deactivate a Transact Function when you need to rollback its configuration change.

To remove a Transact function:

1. Select Services > Transact Functions.
2. Locate a function and click Remove.
3. Click OK to accept it.

To deactivate a Transact function:

1. Select Services > Transact Functions.
2. Locate a function and click Edit.



3. Clear the Active checkbox.
4. Click Save to update the changes.

Next, learn how to create a Transact function.

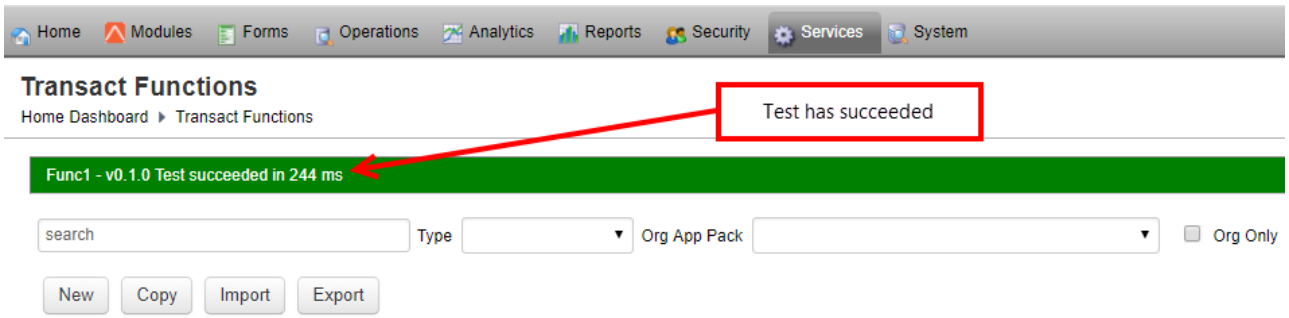
Run a Transact Function Unit Test

Manager provides the convenient way to test a Transact function by executing a Unit test[1] suite, which you must configure for each Transact function.

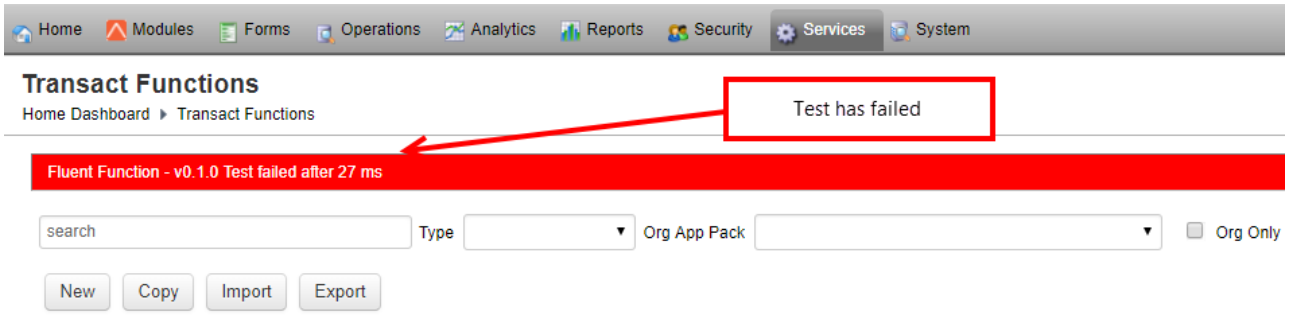
You can run a Unit test from the list of Transact functions or from the Transact function Unit Test tab.

To run a Unit test from the list of Transact functions:

1. Select Services > Transact Functions.
2. Locate a Transact function and click Run Unit Test.
3. Check the test result displayed at the top of the page. The successful test is shown below.



4. If the test has an error, as in the example below, re-run it from the Unit Test tab, see below.

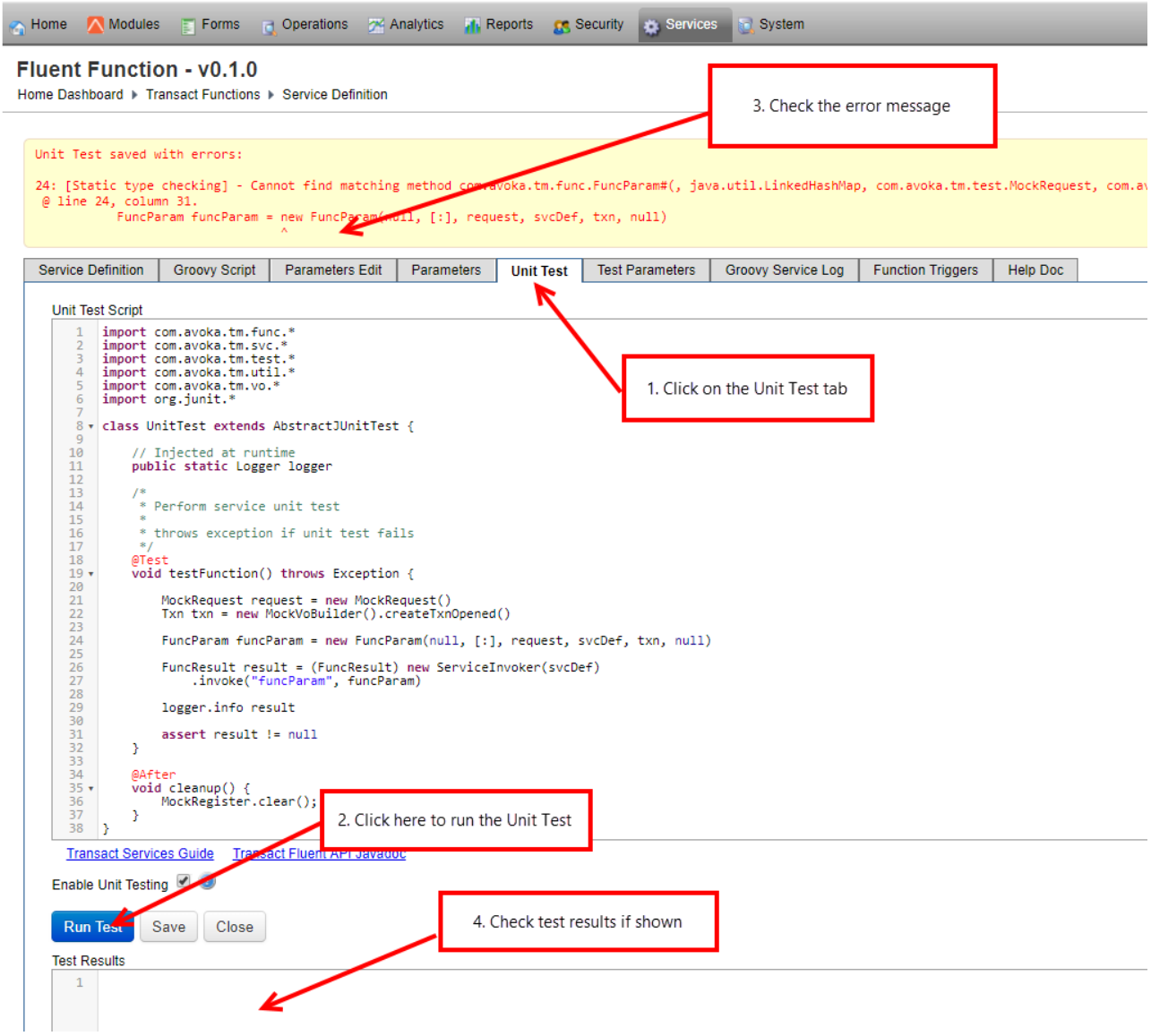


To run a unit test from the Unit Test tab:

---

[1]is a software development life cycle (SDLC) component in which a comprehensive testing procedure is individually applied to the smallest parts of a software program for fitness or desired operation. Unit Testing is a software testing method by which individual units of source code are tested to determine whether they are fit for use. For more information, see http://docs.groovy-lang.org/next/html/documentation/core-testing-guide.html.

1. Select Services > Transact Functions.
2. Locate a Transact function and click Edit.
3. Select the Unit Test tab and click Run Test.



4. Review the test results or error messages. If there are errors, correct the Transact function Groovy script or Unit test Groovy script and re-run the Unit test to see if it works. The successful test message is shown below.

Home | Modules | Forms | Operations | Analytics | Reports | Security | Services | System

Func1 - v0.1.0

Home Dashboard ▸ Transact Functions ▸ Service Definition

Test succeeded in 225 ms

Service Definition | Groovy Script | Parameters Edit | Parameters | Unit Test | Test Parameters | Groovy Service Log | Function Triggers | Help Doc

Unit Test Script

```
1  package com.avoka.func1;
2
3  import com.avoka.tm.func.*
4  import static com.avoka.tm.func.FuncParam.*
5  import com.avoka.tm.svc.*
6  import com.avoka.tm.test.*
7  import com.avoka.tm.util.*
8  import com.avoka.tm.vo.*
9  import org.junit.*
10
11 public class Func1Test extends TransactSDKUnitTest {
12
13     /*
14      * Perform service unit test
15      *
16      * throws exception if unit test fails
17      */
18     @Test
19     void testFunction() throws Exception {
20
21         String xmlData = testParams['Test XML Data']
22         XmlDoc xmlDoc = new XmlDoc(xmlData)
23
24         MockRequest request = new MockRequest()
25         Txn txn = new MockVoBuilder().createTxnSavedWithXml(xmlData)
26
27         FuncParam funcParam = new FuncParam(TRIGGER_FORM_UPDATE, xmlDoc.document, [:], request, svcDef, txn, null)
28
29         FormFuncResult result = (FormFuncResult) new ServiceInvoker(svcDef)
30             .setLogger(logger)
31             .invoke("funcParam", funcParam)
32
33         logger.info result
34         logger.info svcDef.svcConn
35         logger.info svcDef.svcConn.endpoint
36
37         assert result != null
38         assert svcDef.svcConn != null
39         assert svcDef.svcConn.endpoint != null
40     }
41 }
```

Transact Services Guide Transact Fluent API Javadoc

Enable Unit Testing ☑

Run Test | Save | Close

Check test results

Test Results

```
1  14:03:47,300 INFO  com.avoka.tm.func.FormFuncResult@1a1b3be0[
2    errors=[]
3    includeFormData=false
4    continueProcessing=true
5    data={}
6  ]
7  14:03:47,300 INFO  com.avoka.tm.vo.SvcConn@70368715[
8    id=1225
```

5. Check the event log to see the script's execution output.

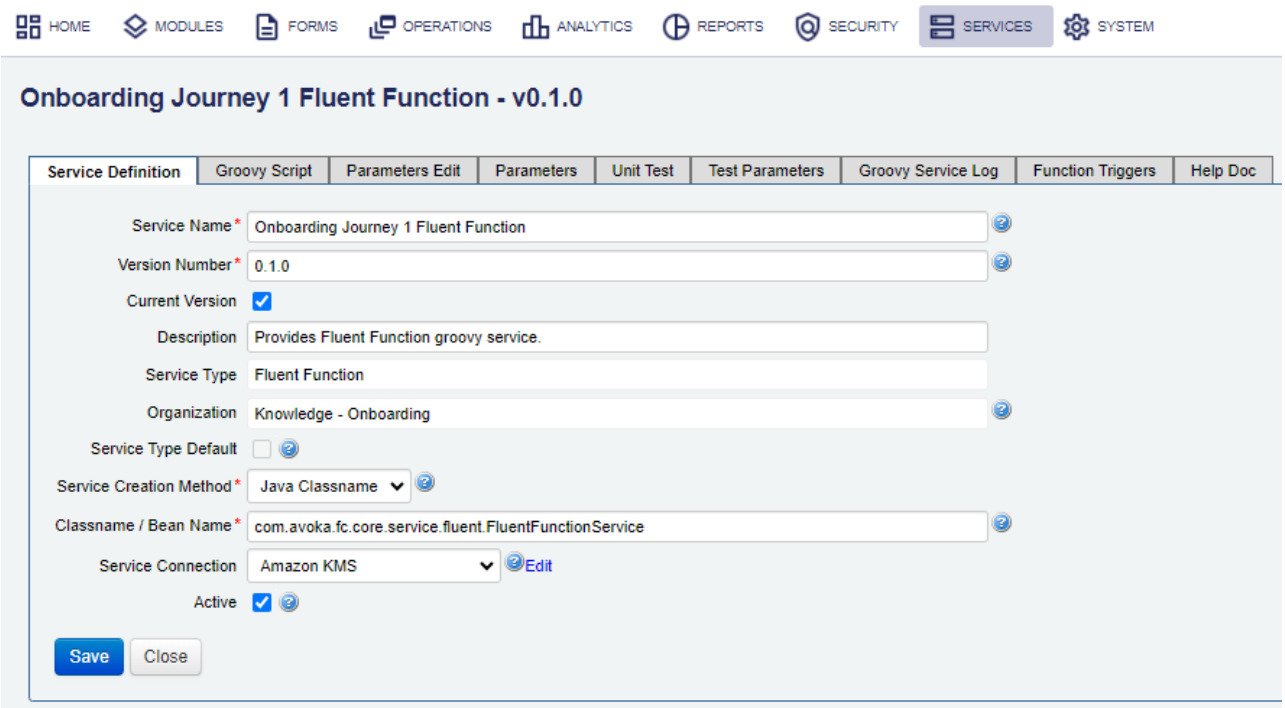
Next, learn how to configure Unit test parameters.

Configure a Transact Function Service Definition

Manager allows you to edit a Transact function's service definition so you can change a Transact Function signature or some other main parameters.

To configure the Transact function service definition:

1. Select Services > Transact Functions.
2. Locate a Transact function and click Edit.
3. Select the Service Definition tab.



4. Enter a new Transact function name in the Service Name filed. This name combined with the Version Number must be unique.
5. Enter a Transact function version number in the Version Number filed. Version number must adhere to semantic versioning[1] practices.

---

[1]Semantic Versioning is a formal convention for specifying compatibility using a three-part version number: major version; minor version; and patch. The patch number is incremented for minor changes and bug fixes which do not change the software's application programming interface (API). The minor version is incremented for releases which add new, but backward-compatible, API features, and the major version is incremented for API changes which are not backward-compatible. For example, software which relies on version 2.1.5 of an API is compatible with version 2.2.3, but not necessarily with 3.2.4.

6. Select the Current Version checkbox to allow Manager to use this function in a form.

   NOTE

   You can use this option to rollback a Transact function.

7. Provide a meaningful description of what this Transact function does and why in the Description field.
8. Select a type of a Transact function from the Service Type dropdown list, which can be one of the following:
   - Fluent Function - A Fluent Function, also known as a Form Function, provides an improved programming model for client applications to call server-side business functions. The Form Function is designed to make calls from form applications to other systems to execute a remote data look-up or some secure business logic on a remote server.
   - Delivery Function - A Delivery Function is used to deliver a form data entered by a user when a form is submitted. The form data includes a submitted form XML, a PDF receipt document and many also include user uploaded file attachments.
9. Specify the organization, which this function will be available to, in the Organization field. If the organization is not set, this function is globally available and can be used in any organization.
10. Select the Service Type Default checkbox to use this Transact function as the implementation of this Service Type. Only global services and Transact functions can be marked as the default.
11. Select the invocation method to create this Transact function from the Service Creation Method dropdown menu, which can be one of the following:
    - Java Classname - to create a new instantiate of this Class it as a new Java Object using `Class.forName()` method.
    - Spring Bean - to look it up from the Sprint configuration files.
12. Enter the Java Class name, for example, `com.avoka.fc.core.service.fluent.FluentFunctionService`, or the Spring Bean name, which is used to create this function, in the Classname / Bean Name field.
13. Select a service connection, which this function will use, from the Service Connection dropdown list or leave it blank if the Transact Function doesn't use any service connection. Click Edit to edit the service connection
14. Select the Active checkbox to make the Transact function active so it can be used in Manager and Maestro forms. It is selected by default.

NOTE

Clear the Active checkbox to rollback the configuration changes to the Transact function as it will become not available to any forms that use it.

15. Click Save to update the changes.

Next, learn how to edit Transact function's Groovy script.

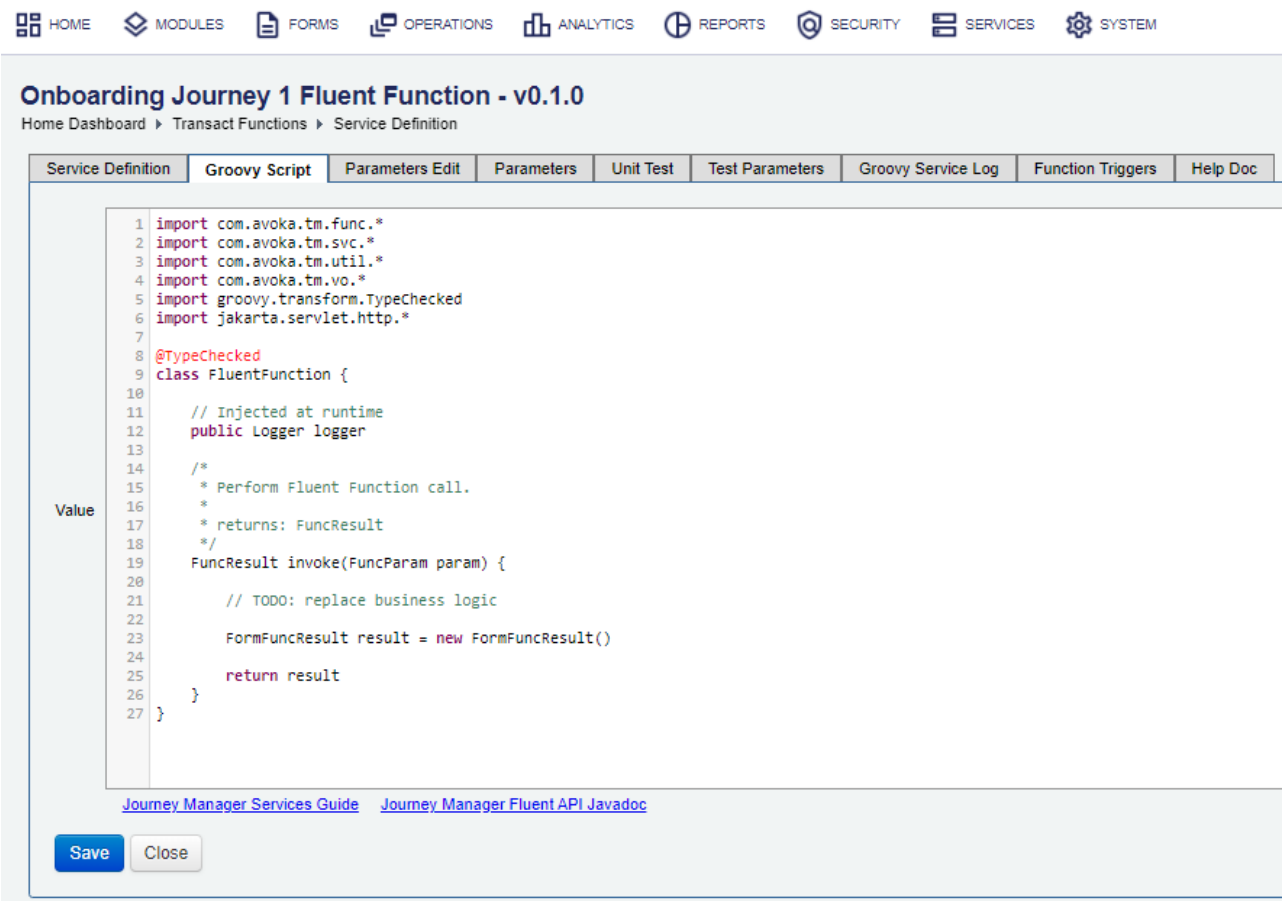
Edit a Transact Function Groovy Script

Manager allows you to edit a Groovy script, which implements the desired business logic of a Transact function. As you create a new Transact function, Manager generate a default Groovy script, so you must modify it.

> NOTE
>
> You can test your Groovy script or a part of the script using the Groovy console.

To edit the Groovy script:

1. Select Services > Transact Functions.
2. Locate a Transact function and click Edit.
3. Click the Groovy Script tab to view the current script.



While working on the script, you can use some basic editing options, such as:

- Line numbering.
- Ctrl-Z.

- Ctrl-C and Ctrl-V.
- Collapse and extend class, function and method definitions.

4. Click the documentation links to get help on how to write Groovy scripts and which APIs to use.
5. Click inside the Groovy script to start editing it. For example, you can replace the default script with the simple script shown below:

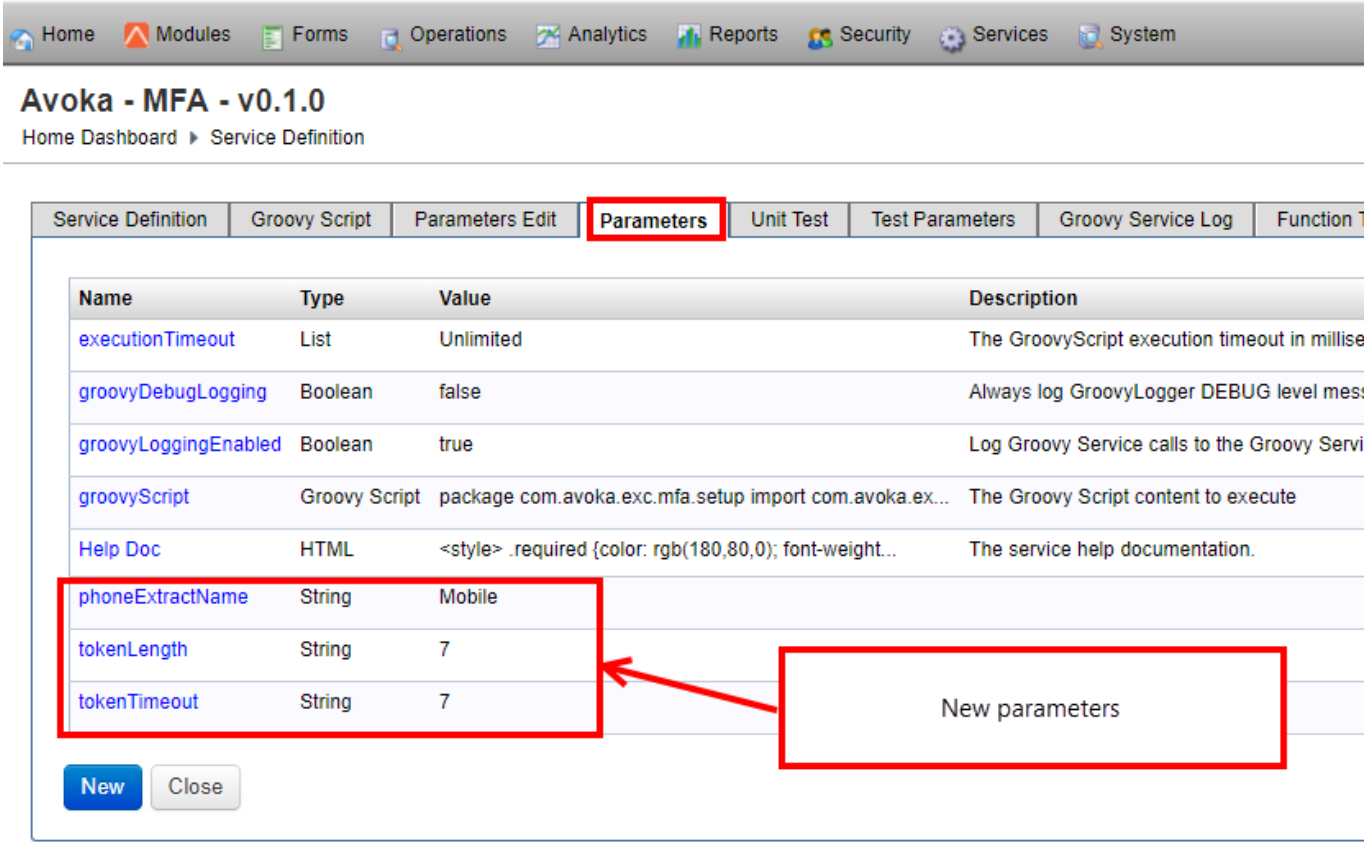
```
import com.avoka.tm.func.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import javax.servlet.http.*

class FluentFunction {
	// Injected at runtime
	public Logger logger
	/*
	* Perform Fluent Function call.
	*
	* returns: FuncResult
	*/
		FuncResult invoke(FuncParam param) {
			def formDoc = new XmlDoc(param.appDoc)
			formDoc.setText('/AvokaSmartForm/Applicant/FirstName', 'John')
			formDoc.setText('/AvokaSmartForm/Applicant/LastName', 'Smith')
			formDoc.setText('/AvokaSmartForm/Applicant/Email', 'jsmith@test.com')
			formDoc.setText('/AvokaSmartForm/Applicant/Mobile', '0401010101')
			FormFuncResult result = new FormFuncResult()
			result.includeFormData = true
		return result
	}
}
```

> NOTE
>
> We recommend writing a Groovy script in your favorite IDEA, such as Eclipse or IntelliJ, and then copy and paste it into the Groovy Script tab.

6. Click Save to update the changes.

As you are working on the script, we recommend Creating a Groovy unit test script right away to test its implementation.

If the Groovy script uses some configurable parameters, add them in the Parameters tab. The example below shows the new three parameters: `tokenTimeout, tokenLength, phoneExtractName`.



The Groovy script snippet below shows how to use the new three parameters.

```
int tokenTimeout = inputParams.svcDef.paramsMap["tokenTimeout"]
int codeLength = inputParams.svcDef.paramsMap["tokenLength"]
String ref = (String) inputParams.svcDef.paramsMap["phoneExtractName"]
```

NOTE

The parameters' values can be changed any time without any Groovy script modifications.

Next, learn how to configure Transact function parameters edit.

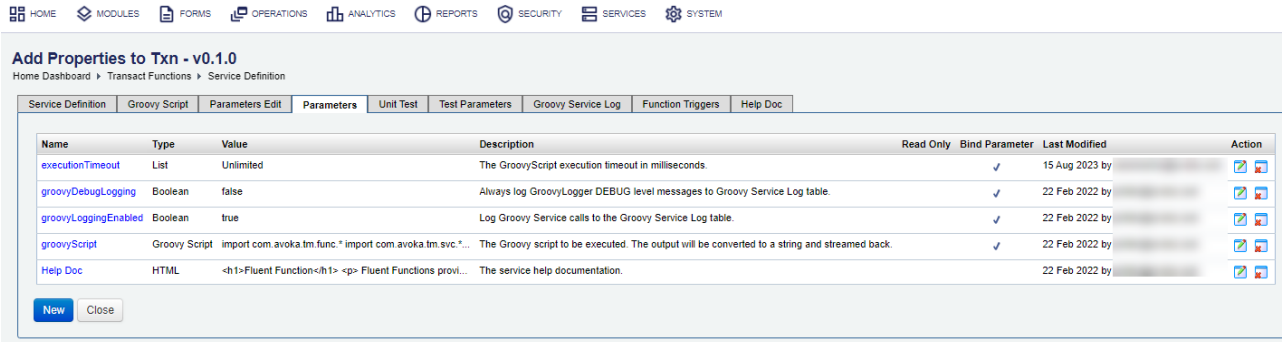
Configure Transact Function Parameters

Manager allows you to view, configure, create, and remove Transact function Groovy script parameters in the Parameters tab.

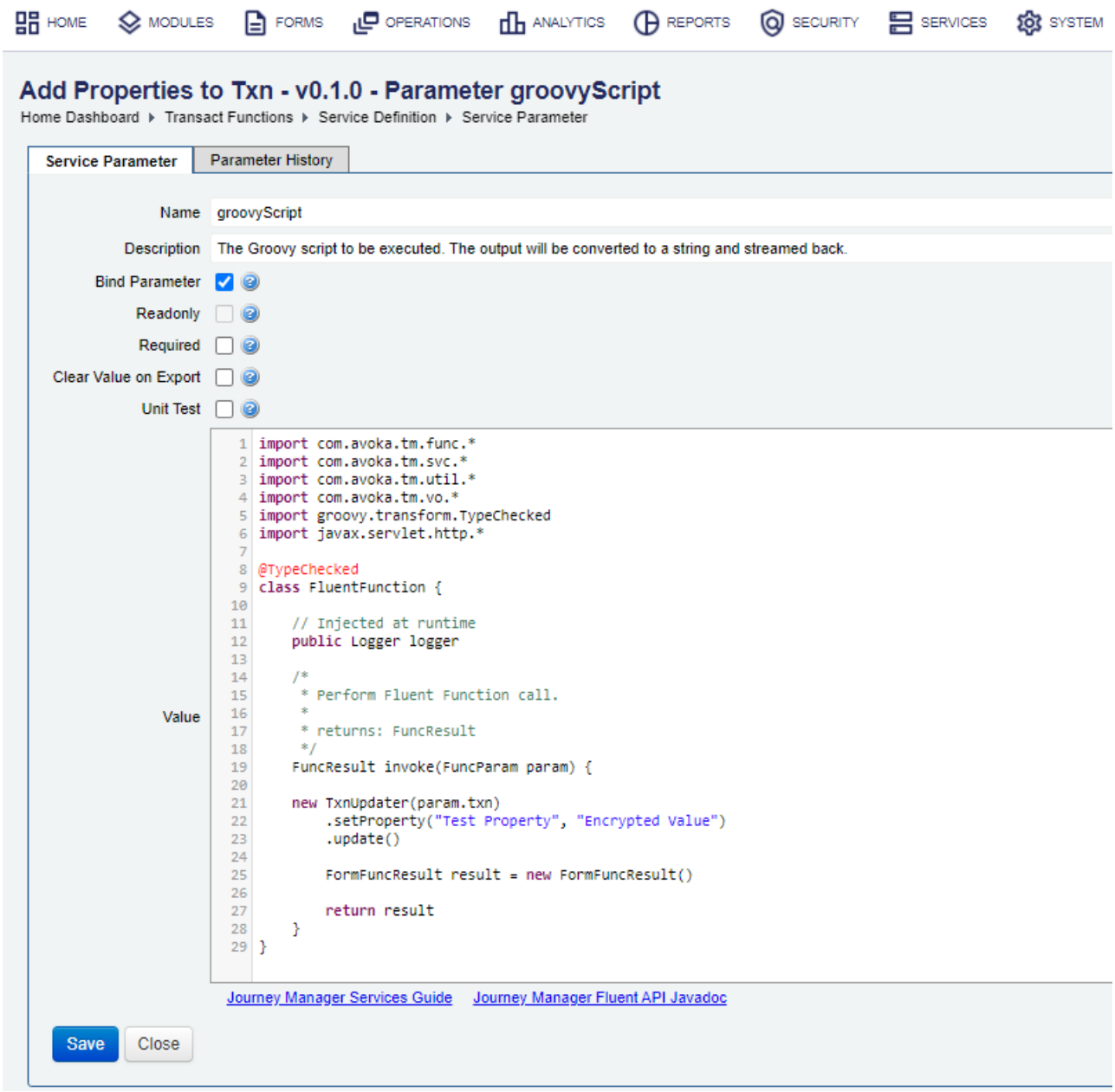
> NOTE
>
> You can update the same parameters in the Parameters Edit tab.

To configure Transact function parameters:

1. Select Services > Transact Functions.
2. Locate a Transact function and click Edit.
3. Click the Parameters tab to view, edit and add Transact function parameters.



4. Locate a parameter to modify and click Edit. For example, click `groovyScript` to edit its configuration, as shown below.



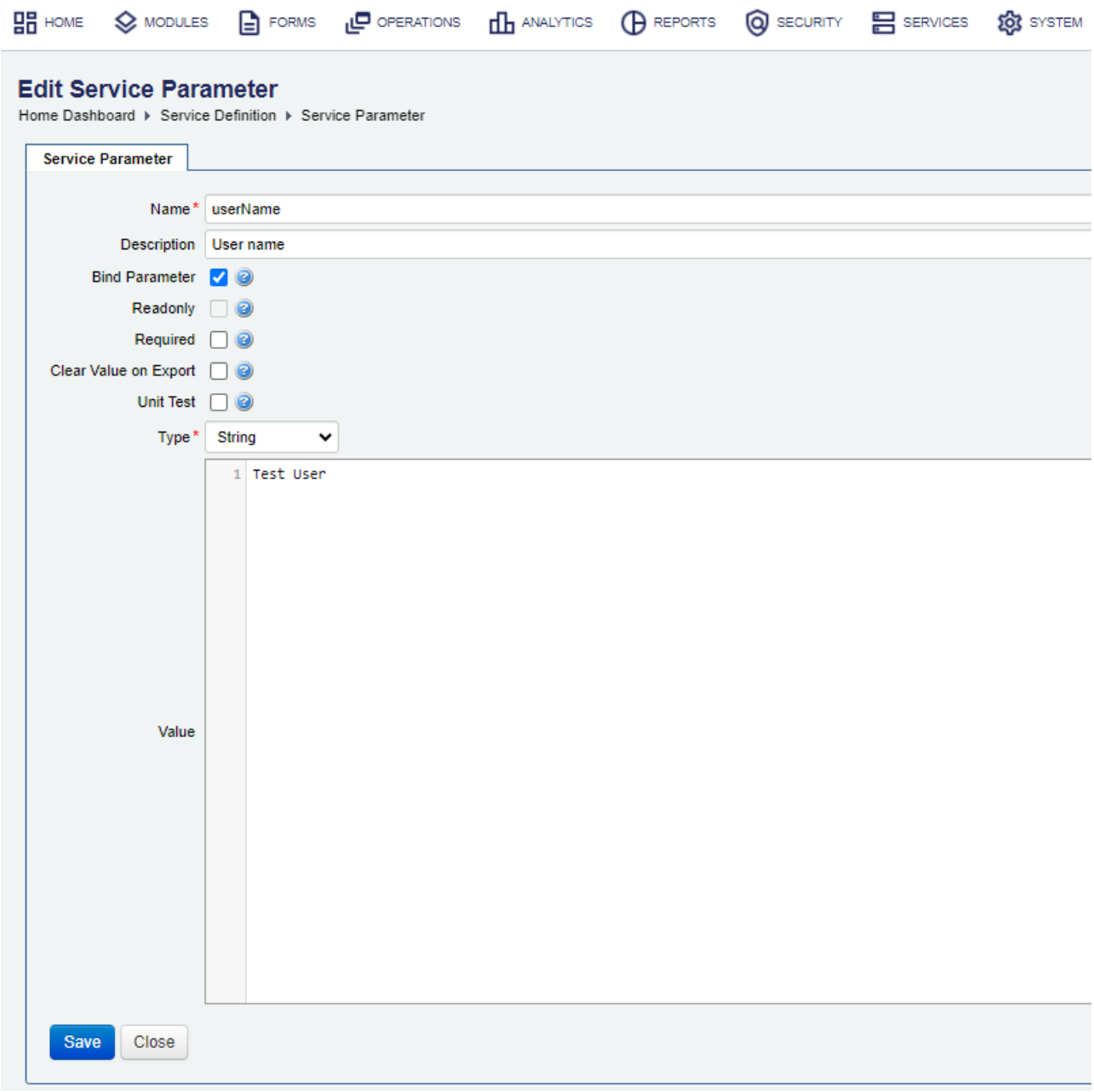
> NOTE
>
> For explanation of common parameters, see Configure Transact Function Parameters Edit. Otherwise, check the specific documentation of each service.

5. Click Save to update the changes.

To create a Transact function parameter:

1. Select Services > Transact Functions.
2. Locate a Transact function and click Edit.
3. Click the Parameters tab to view, edit and add Transact function parameters.

4. Click New to create a new parameter.



5. Update a Groovy script to make use of this parameter, for example `String UserName = (String) inputParams.svcDef.paramsMap["userName"]`.
6. Click Save to update the changes.

Next, learn how to configure Transact function Unit test.

Configure Transact Function Parameters Edit

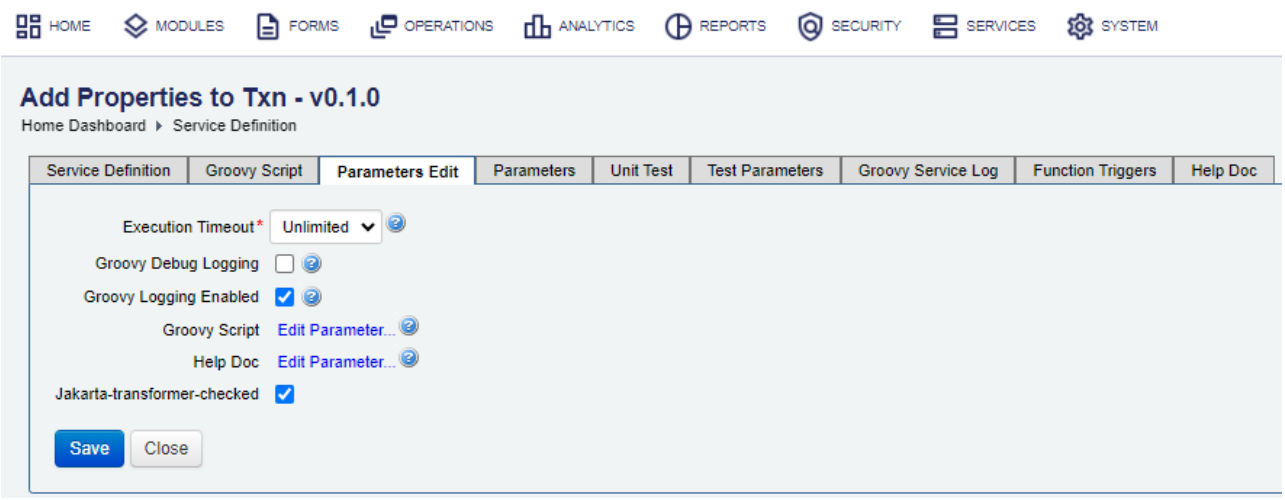
Manager allows you to view and configure Transact Function's parameters in the Parameters Edit tab.

> NOTE
>
> You can add new parameters as well as edit the existing ones in the Parameters tab. For more information, see Configure Transact Function Parameters.

To configure the parameters:

1. Select Services > Transact Functions.
2. Locate a Transact function and click Edit.
3. Click the Parameters Edit tab to view parameters available to the Transact Function.

   > NOTE
   >
   > The number of parameters and their types depend on the service implementation so the layout of this tab is different for each service.



4. Specify the script execution timeout in milliseconds in the Execution Timeout field. The pre-defined values are:
   - Unlimited - the service runs in the same thread that never times out and the forking of this thread is disabled. This is the default.
   - 10000
   - 30000
   - 60000

You can also define this parameter in the `service-def.json` file, as shown below:

```
{
   "name": "executionTimeout",
   "description": "The GroovyScript execution timeout in milliseconds.",
   "value": "60000",
   "type": "value",
   "bind": false,
   "required": false,
   "clearOnExport": false,
   "readOnly": false
}
```

5. Select the Groovy Debug Logging checkbox to log debug messages generated by the Groovy script to the Groovy Service Log. This option is disabled by default.

> WARNING
>
> Use the Groovy Debug Logging option only for troubleshooting of a Groovy script. We don't recommend enabling it in a production environment nor running a service with this option enabled for a prolonged period of time because it generates significant amount of verbose information in the `groovy_service_log` and `groovy_service_log_data` tables. This can deplete a database of free space and degrade the overall server performance.
> Also, the size of these tables impact the performance of the Manager UI, such as Transaction Groovy Log and Groovy Service Log, where the Groovy service log records are displayed.

To help you avoid enabling this option by mistake in a production environment, Manager checks if the Transact Manager module has the Production Mode option selected and then disable Groovy Debug Logging. | 23.04 This feature was introduced in 23.04

However, you can override this behavior by setting the `com.avoka.fc.core.overrideProductionGroovyDebugLogRestriction` property to `true` in the `standalone.xml` file. | 23.10 This feature was introduced in 23.10

> NOTE
>
> You must contact the Cloud Services team to change the `com.avoka.fc.core.overrideProductionGroovyDebugLogRestriction` property.

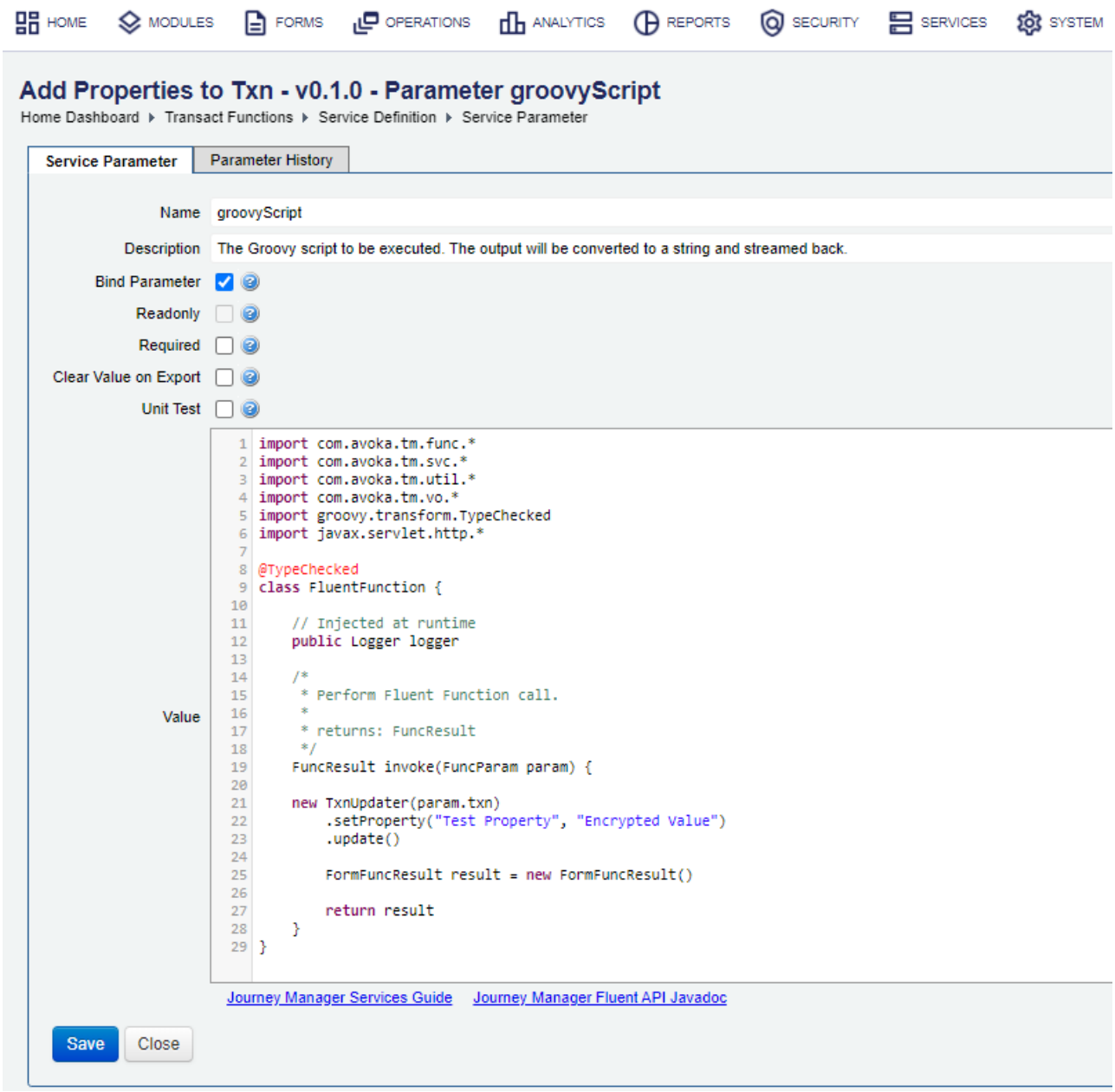
To sum it up:

- Groovy Debug Logging is disabled by default.
- If Production Mode is selected, the Groovy Debug Logging option is ignored. To enable Groovy debug logging, set the `overrideProductionGroovyDebugLogRestriction` property to `true`.
- If Production Mode is not selected, use the Groovy Debug Logging option to enable or disable Groovy Debug Logging.

6. Select the Groovy Logging Enabled checkbox to log messages, generated by the Groovy script, to the Groovy Service Log. It is selected by default.

> NOTE
>
> If a Groovy script is part of a Security Manager with the Enable Logging checkbox selected, then the Groovy scrip will log messages to the Security Manager Log, not the Groovy Service Log.

7. Click Groovy Script to edit the Groovy script and its parameters in the Service Parameter tab.



NOTE

You can use the Service Parameter tab to edit Groovy script parameters listed below.

8. Select the Bind Parameter checkbox to inject the Groovy script into the created service object. It is selected by default.

NOTE

The service class of this function must implement a setter method, for example void `setObject(Object o)`.

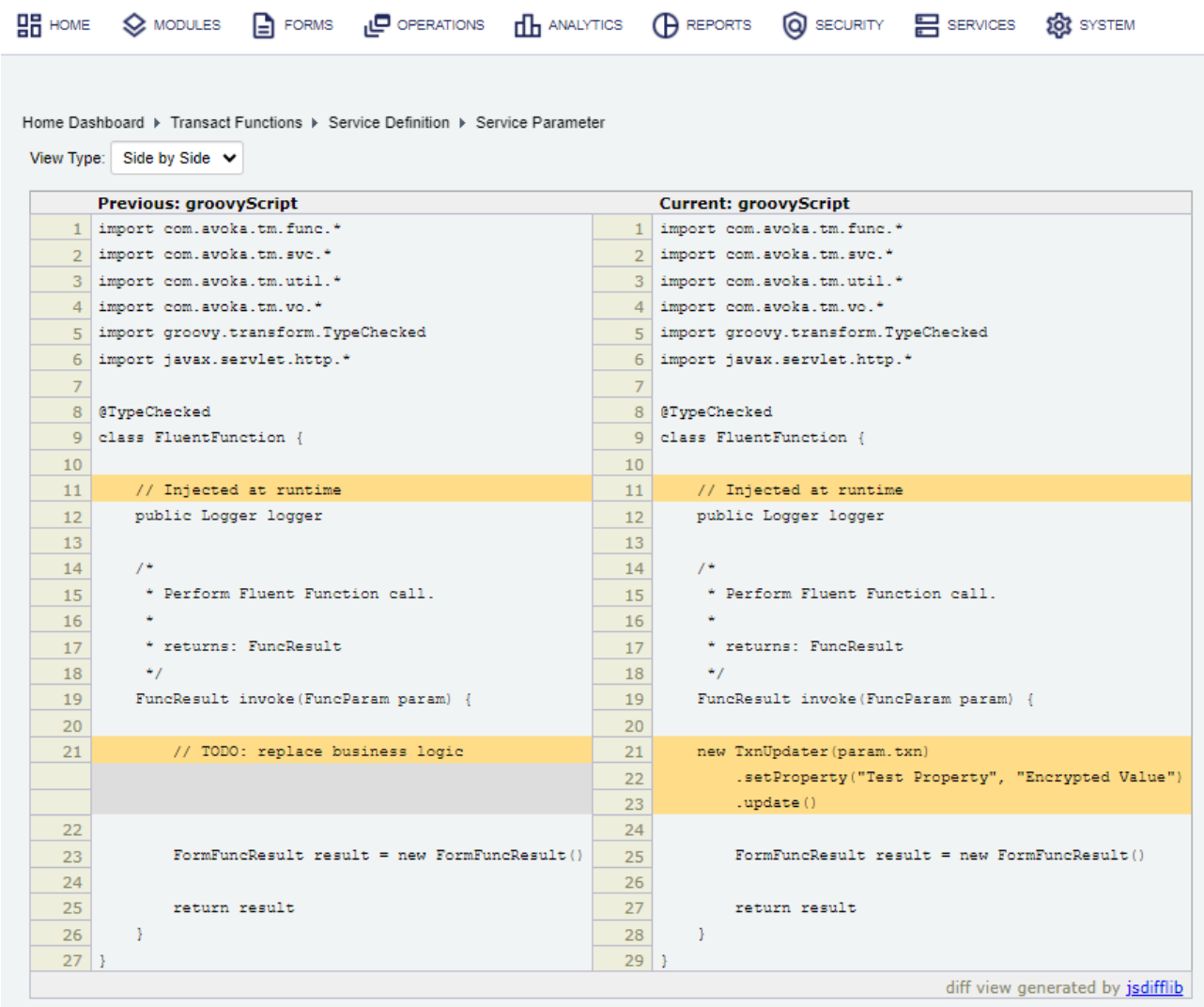
9. Select the Readonly checkbox to make the Groovy script read only, so it can't be edited.

NOTE

If you've selected this checkbox, you can't edit any of the following options.

10. Select the Required checkbox to make the Groovy script required to create a service object.
11. Select the Clear Value on Export checkbox to clear the Groovy script when the service definition is exported. This ensures that server specific configuration can't be accidentally used in other services.
12. Select the Unit Test checkbox to use this Groovy script as a Unit Test only. This is useful in mock-up testing.
13. Click the Parameter History tab to see Groovy script change history.

14. Click View Content Difference to check what was changed in the script. Select how you want to view the difference from the View Type dropdown list, which has the following options: Side by Side and Inline. This example below illustrates how the difference in the contents is shown in the Side by Side mode.



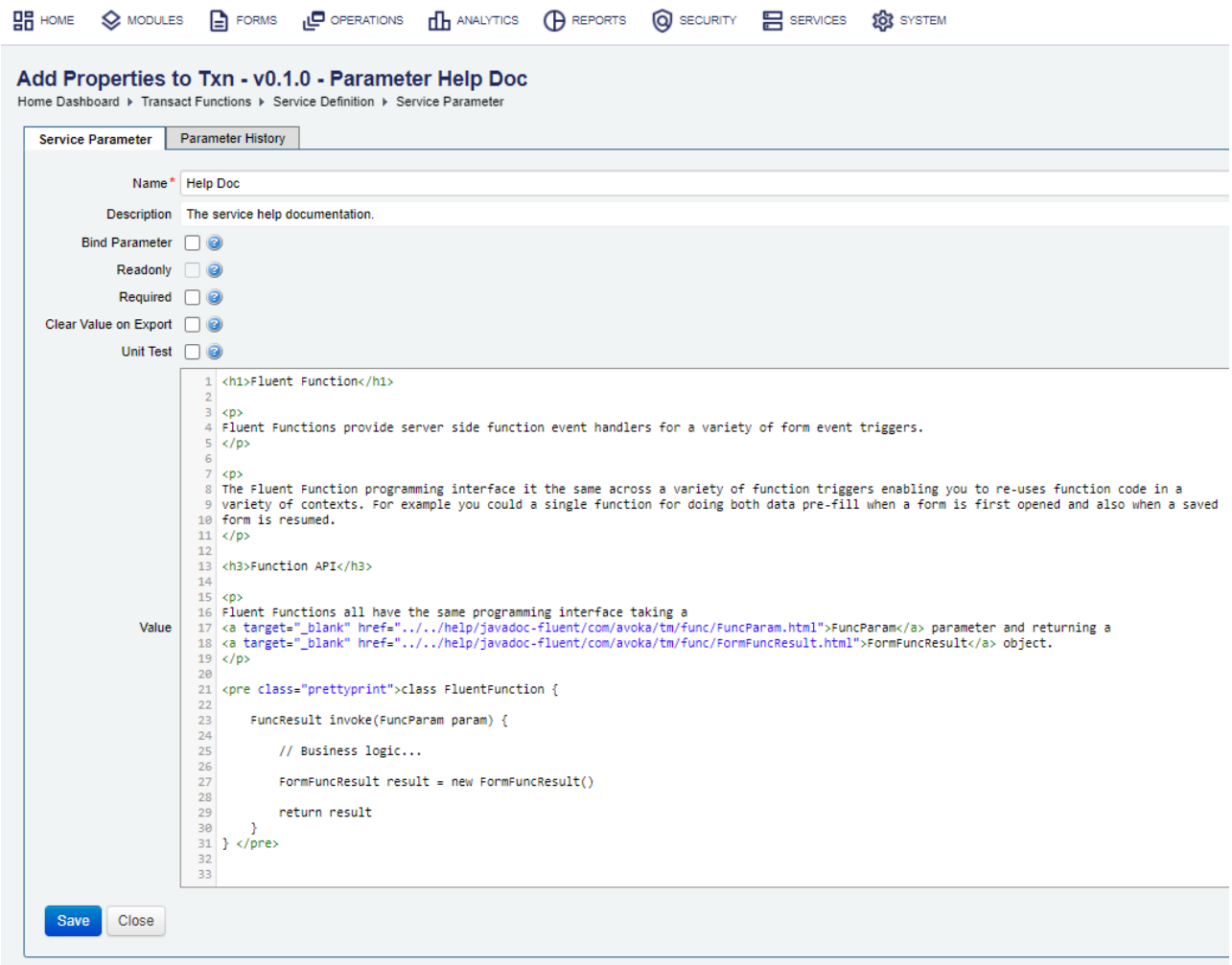

```
import com.avoka.tm.func.*
import com.avoka.tm.svc.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import groovy.transform.TypeChecked
import javax.servlet.http.*

@TypeChecked
class FluentFunction {

    // Injected at runtime
    public Logger logger

    /*
     * Perform Fluent Function call.
     *
     * returns: FuncResult
     */
    FuncResult invoke(FuncParam param) {

        // TODO: replace business logic
```

```
import com.avoka.tm.func.*
import com.avoka.tm.svc.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import groovy.transform.TypeChecked
import javax.servlet.http.*

@TypeChecked
class FluentFunction {

    // Injected at runtime
    public Logger logger

    /*
     * Perform Fluent Function call.
     *
     * returns: FuncResult
     */
    FuncResult invoke(FuncParam param) {

    new TxnUpdater(param.txn)
        .setProperty("Test Property", "Encrypted Value")
        .update()
```

```
        FormFuncResult result = new FormFuncResult()

        return result
    }
}
```

```
        FormFuncResult result = new FormFuncResult()

        return result
    }
}
```

15. Click Restore History Content to restore this history content.
16. Click Remove Service Parameter History to remove this service history.
17. Click Help Doc to view or edit the Groovy script documentation page in the Service Parameter tab.



18. Click the Parameter History tab to see Help Doc change history. To configure this tab, see the Groovy script Service Parameter History tab described above.
19. Check the Jakarta-transformer-checked status to see whether the Transact function or a service has been checked by the Jakarta Transformer scheduled service or not. Every time the Jakarta Transformer service runs as a scheduled job and transforms a Transact function, it selects its Jakarta-transformer-checked checkbox to mark it so it won't scan and transform it again during the next run. | 23.10 This feature was introduced in 23.10

> NOTE
>
> The Jakarta-transformer-checked checkbox is not shown if the Transact function has not been transformed.

> NOTE
>
> Clear the Jakarta-transformer-checked checkbox if you want the Jakarta Transformer service to run a transformation on this service again.

20. Select the Xml Included checkbox to include Form XML Data within a marshaled transaction object, identified as `formXml`. This is an optional parameter, so you need to create it when required. | 24.10 This feature was introduced in 24.10

> NOTE
>
> Delivery Functions include Form XML Data in a marshaled transaction object by default.

Next, learn how to configure Transact function triggers.

Configure a Transact Function Unit Test

It is a good development practice to create a unit test[1] for any Transact function Groovy script, so it can be thoroughly tested prior to using it. Manager provides Groovy unit testing capability to all Groovy services. When a Transact function is created, it comes with a default Groovy script, which is merely a template to help developers implement a new test logic. You need to change it for each Transact function that you have to test. Typically, a unit test does a lot of this:

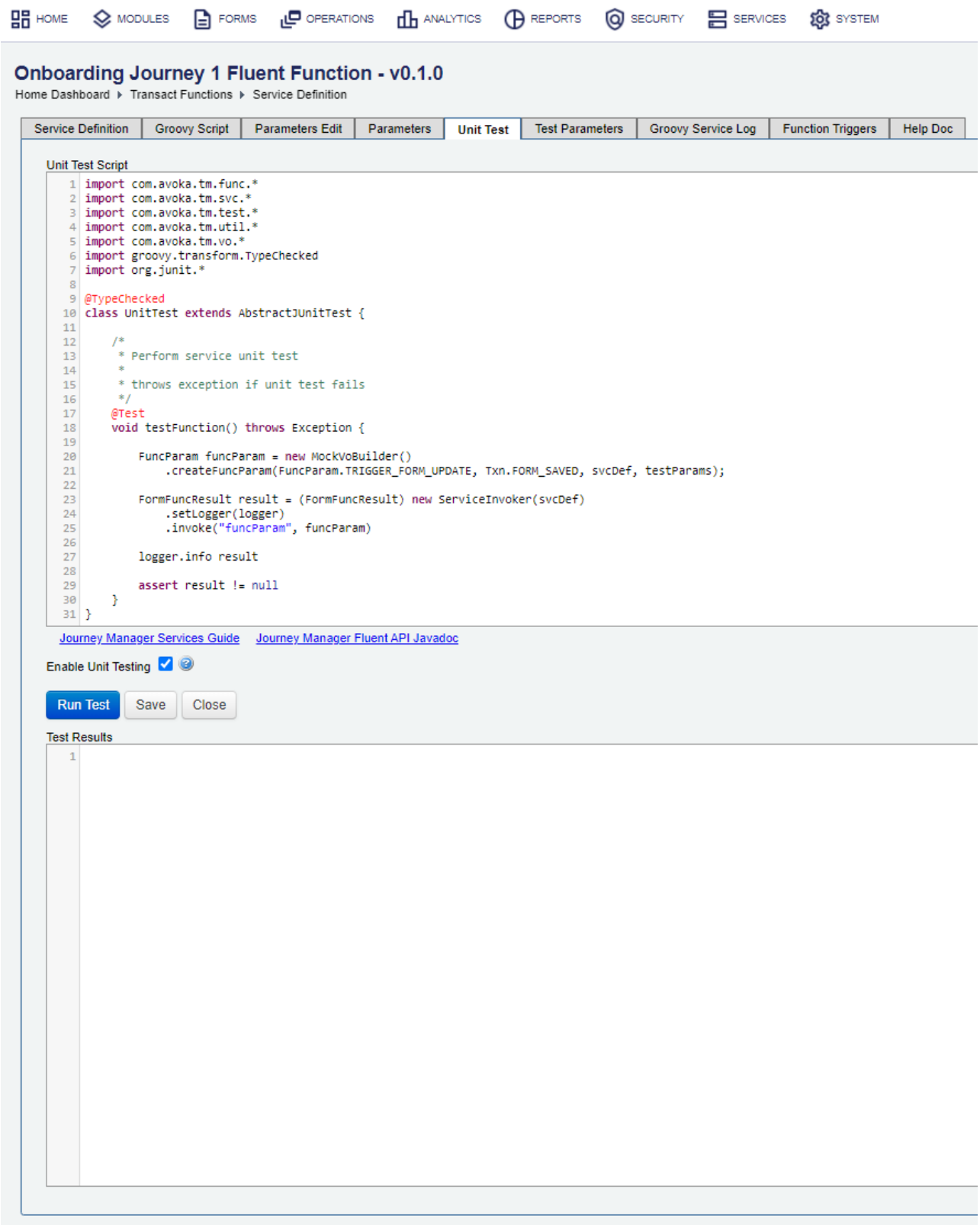
```
FormFuncResult result = (FormFuncResult) new ServiceInvoker(svcDef)
       .setLogger(logger)
       .invoke("funcParam", funcParam)
```

To configure the unit test:

1. Select Services > Transact Functions.
2. Locate a Transact function and click Edit.

[1]is a software development life cycle (SDLC) component in which a comprehensive testing procedure is individually applied to the smallest parts of a software program for fitness or desired operation. Unit Testing is a software testing method by which individual units of source code are tested to determine whether they are fit for use. For more information, see http://docs.groovy-lang.org/next/html/documentation/core-testing-guide.html.

3. Click the Unit Test tab.



4. Edit a Groovy unit test script as required.

5. Click on two help links below to get more help on how to write Groovy scripts and which Transact Fluent API to use:
    - Service guide
    - Transact Fluent API Javadoc[1]
6. Select the Enable Unit Testing checkbox to enable unit testing for this script.
7. Click Run Test to execute the Groovy test script. Test results, success or failure, is displayed in the Test Result field.

Next, learn about Transact function help documentation.

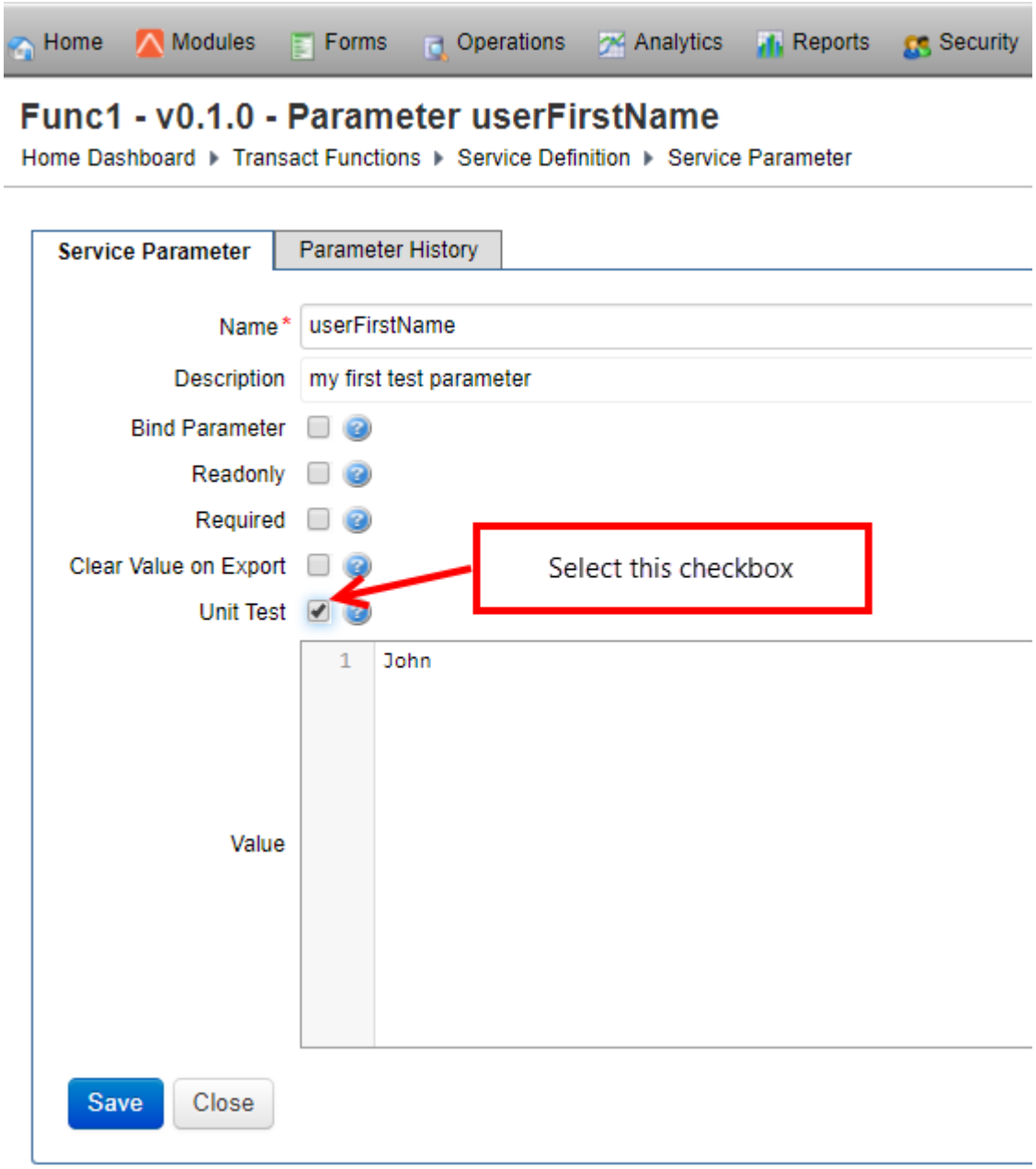
[1]Javadoc is a documentation generator created by Sun Microsystems for the Java language for generating API documentation in HTML format from Java source code.

Configure Transact Function Unit Test Parameters

Manager allows you to configure Groovy Unit Test script parameters, which you can dynamically update in the Test Parameters tab.

To configure the Unit test parameters:

1. Select Services > Transact Functions.
2. Locate a Transact function and click Edit.
3. Click the Test Parameters tab.
4. Click New to add and configure a new parameter, for example, `userFirstName`.



5. Select the Unit Test checkbox to make this new parameter available in the Unit Test tab.
6. Click Save to update the changes.
7. Verify that the `userFirstName` parameter now appears in the Test Parameters tab.
8. Update the Groovy unit test script to make use of this parameter.

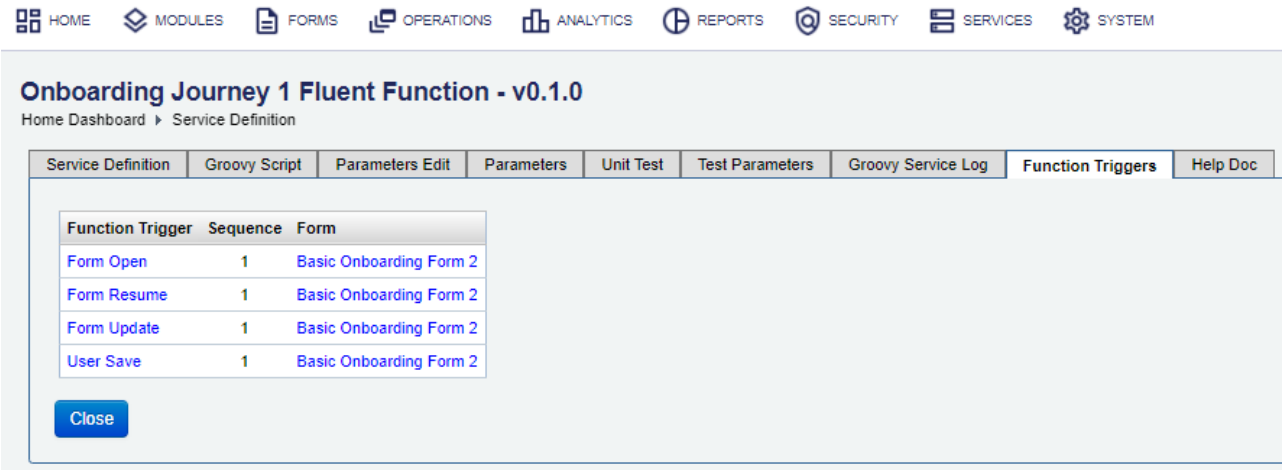
Next, learn how to run a Transact function Unit test.

Configure Transact Function Triggers

Journey Manager allows you to create and configure various Transact function triggers, which are methods that define when a Transact function is called. For more information on triggers, see Fluent Functions > Function Triggers.

To configure Transact function triggers:

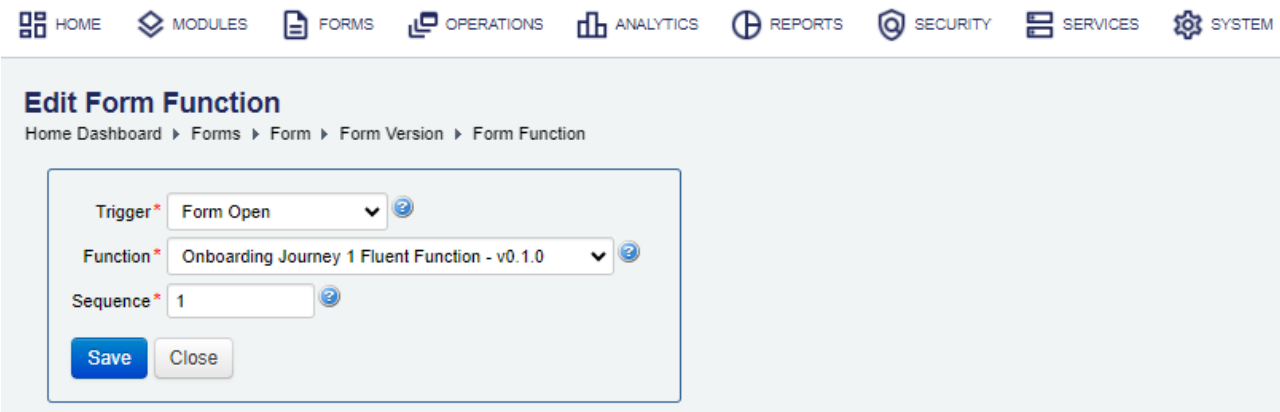
1. Select Services > Transact Functions.
2. Locate a Transact function and click Edit.
3. Click the Function Triggers tab to view all configured triggers for this Transact function.



The list displays the following details:

- Function Trigger: a name of a trigger, which can be one of the following:
  - Form Open is called when new form transaction is created by Manager, immaculately before it is rendered to the browser.
  - Form Resume is called by Manager when a saved form transaction is resumed, immediately before it is rendered to the browser.
  - Form Save Challenge is called when a save challenge is performed.
  - Form Update is called when the form makes an background update operation in a browser.
  - Form Ineligible is called when the form determines the user is ineligible to complete the application.
  - Form Function is called by Maestro form business rule scripts generally to get dynamic data from the server and perform any transaction updates.
  - User Save is called when the user explicitly saves and closes the form in a browser.

    - User Submit is called when the user explicitly submits a completed form application in a browser.
    - User Cancel is called when the user explicitly cancels and closes a form application in a browser.
    - Background Delivery is called to run a Delivery Function in a background, for example, when the user submits a form but the PDF receipt document will not be available yet. For more information, see Delivery Function.
  - Sequence: an order that functions with the same trigger are executed.
  - Form: a name of a form where the trigger is used.
4. Click a trigger name, for example, Form Open, to view and edit its configuration.



5. Select a trigger from the Trigger dropdown list to define when a Transact function is called. For more information, see Function Triggers.
6. Select a function, which is invoked on the form trigger event, from the Function dropdown list.
7. Specify the sequence order in which Transact functions with the same trigger are called.
8. Click Save to update the changes.
9. Click on a form link to view and edit its configuration.Edit a form which uses this Transact function trigger in the new window. For more information on form functions, see Form Functions Quick Start.

Next, learn how to edit a Transact function Groovy script.

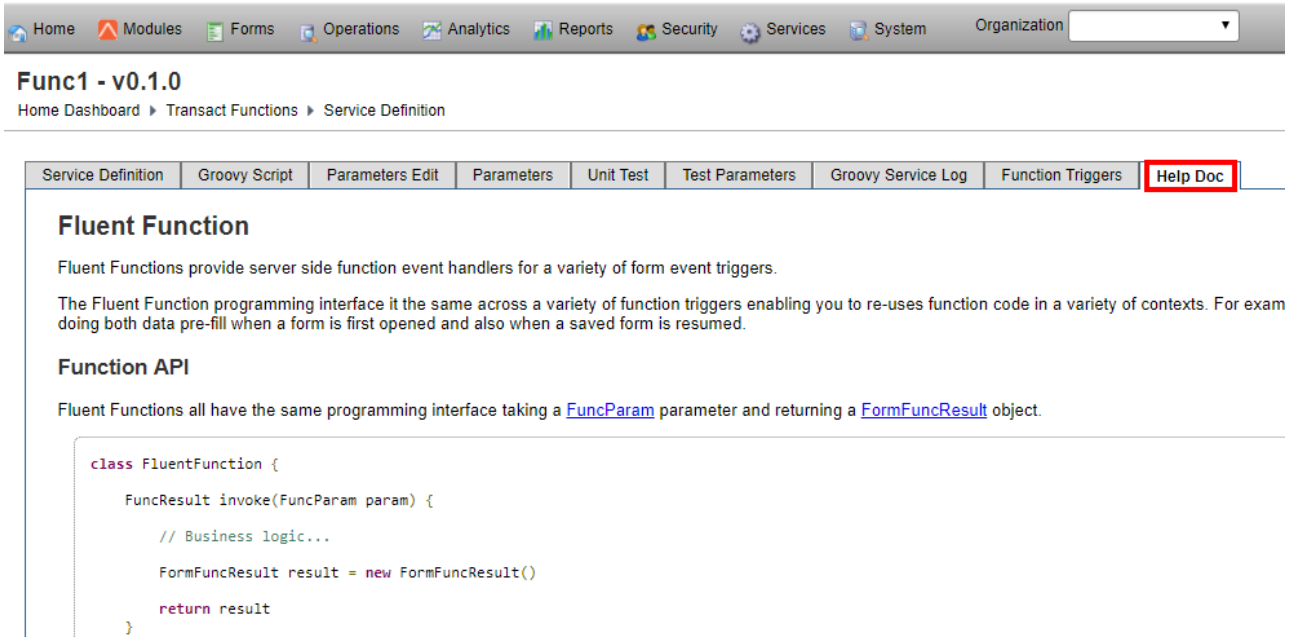
Transact Function Help Doc

Manager provides the Help Doc tab to view and edit a Transact function documentation, which can include:

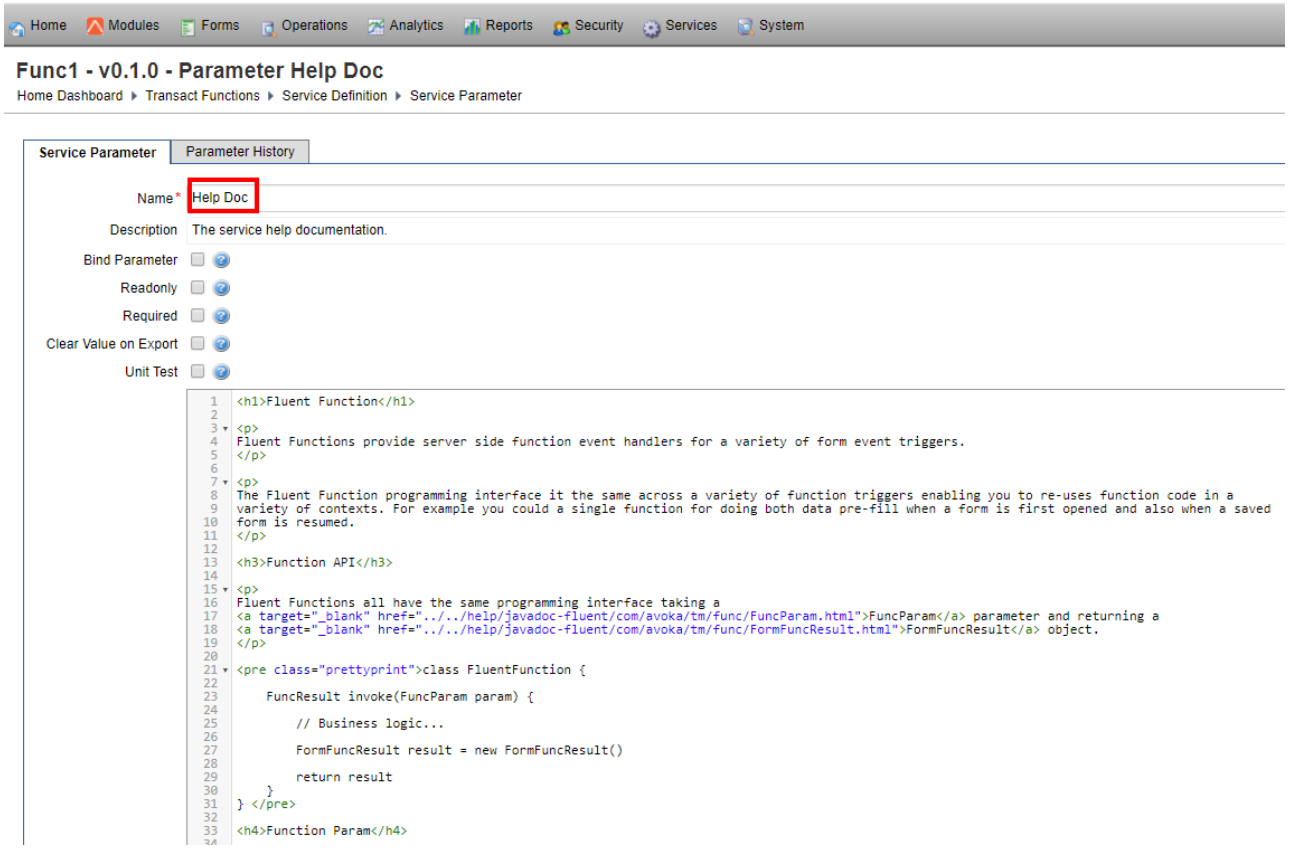
- Description of the function and how it is implemented
- References to Function APIs
- Triggers
- Exceptions
- Class definitions
- Code snippets and examples
- Test cases
- JSON definitions and examples.
- Some other specific information.

To view Transact function documentation:

1. Select Services > Transact Functions.
2. Locate a Transact function and click Edit.
3. Click the Help Doc tab.



4. To edit the help doc information, select the Parameters tab and click Help Doc.



5. Click Save to update the changes.

Next, learn how to configure Transact function parameters.

Server-Side Persistence Pattern

The main persistence model for Manager is the XML file. This is stored on the server, sent to the browser when a form is rendered, and sent back to the server when the form is submitted. It is also saved to the server in background save events.

However, there are cases when it may be useful to store information on the server which is not stored in the XML file. Some of these cases include:

- Storing results from a backend server call, such as an actual credit score. We want to record the actual score, but all we want to send to the form is a pass/fail. We don't want it to be in the XML because we don't want the end-customer to see their actual score, or the reasons why they failed.
- We want to temporarily store in-progress data that has been used in back-end service calls. This is to ensure that the end-user doesn't "game" the system, by changing the data in the form at different points in the journey.
- We want to store the results of a back-end service call for auditing or debugging purposes - but it's too much data to inject into the XML.
- There are probably many other cases.
- This data could be as simple as a string, or more complex like some JSON or XML.

There is a simple way to store and retrieve arbitrary string data against a transaction using Fluent service calls. Let's look at these in more detail.

Storing Data

You can use the `TxnUpdater.setProperty(String name, String value)` method to create or update a property value. This is illustrated below:

```
new TxnUpdater(txn)
    .setProperty("MyProp", "MyValue")
    .update()
```

Retrieving Data

You can use the `TxnQuery` class to search for a particular transaction, and then get the Property Map within the Txn value object to retrieve the particular property. This is illustrated below:

```
Txn txn = new TxnQuery()
    .setSubmitKey(…) // or similar to find the correct transactior
    .withPropertyMap() // to retrieve the properties
    .firstValue() //to find the first matching transactior
    // The returned txn contains an attribute propertyMap of type Map<String,String>
```

This is important to note that:

- Data stored in the Transaction Properties are encrypted and subject to data retention policies, just like XML. (In recent versions of Manager.)
- In some cases, it may be necessary to store data against a collaboration job rather than an individual transaction.
- The PropertyQuery service is used for locating configuration properties for Form Versions, Workspaces, etc. This is for static configuration properties, not for per-transaction or runtime properties.
- It is also possible to use the standard HTTP session to store temporary data. However, this data will not be persisted beyond an individual session. A transaction property is persisted in the database.

Next, learn how to view Transact functions.

Handling Form Function Exceptions in Maestro Applications

Manager comes with the Groovy[1] console enabling developers to rapidly develop and test Groovy scripts. Groovy console has an access to all services and libraries, so it can simulate the behaviors of any Groovy service.

Form Functions are easy to use for the happy path scenarios when no errors or exceptions are taken into consideration. The simplest implementation can look like this:

```
Transact.formFunction(ffName, '0.1.0', 'myMilestone',ffData).then(function (response))
```

However, in real life errors do happen, so Form Function call implementations must cater for all sorts of errors and exceptions that may occur and handle both the success and error return messages. There are several ways of doing this; the following pattern is recommended: the Form Function captures all responses and returns different flags, for success or error statuses, and a detailed error message. This article explains how to use this pattern and provides examples of both server and client source code.

Journey Manager - the Server Side

If there is going to be an error, most developers would expect it to happen in a Form Function that runs in Manager. This is where calls to other systems are made, and most data manipulation occurs, so any exceptions should be handled here. However, placing a try-catch block around the code isn't enough. We need a consistent way to return a function call result, a success or a failure, back to the Maestro application's JavaScript, so it can effectively handle it and possibly display to a form end user.

When you develop Transact Functions, try to make them as self-contained as possible with regards to how they send data back to the application. Maestro applications have limited data passed back by a Transact Function when an exception occurs in Manager. Usually, it means that something catastrophic happened on the server side.

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

To address this limitation, you should create consistently named variable pairs and return them in the `FormFuncResult` object; for example, `executionStatus` and `errorMessage`.

You can use the `executionStatus` variable to indicate whether the Transact Function call has followed a normal execution path or encountered a problem. A good development practice is to create a set of constants to use across your SDK projects. The example below uses constants as String objects. Condition codes are returned to indicate the following outcomes: success, problems with the input data, problems accessing third party APIs, or trapped exceptions.

You can use the `errorMessage` variable to hold an additional detailed message to compliment the `executionStatus` value. If the Transact Function doesn't capture an extra error message, this variable will be empty.

When an exception occurs, the Transact Function will catch the exception and populate the response object with one or both of `exceptionStatus` and `errorMessage`, so the Maestro application can better manage the user experience.

Maestro Application - the Client Browser Side

The `Transact.formFunction()` call from an application to a Manager service returns a promise. You can use this callback for successful, also known as happy path, results. However, this callback doesn't allow you to handle all possible conditions gracefully. You will need the second callback `catch()` added to the `then()`. This callback is executed if an error has occurred; for example, a Manager server failure or a network issue between the application and the server. The `catch (function(err))` input parameter is an error object containing a message which you can process and display to the end user.

The `catch()` block handles any rejections of the initial promise, specifically during the first `then()` callback. One use case is when a system-level error is detected in the Form Function on the server, the application can check this in the first callback and throw a `new Error` there. That Error object can then be processed in the `catch()` block.

Remember that `then()` and `catch()` promises are generic JavaScript structures and syntax, not Maestro specific.

Code Examples

The Click Rule JavaScript and Form Function Groovy Script below illustrate the exception handling pattern. They allow a user to select a test scenario, which can be one of the following options: `Success`, `DataError`, `SystemError`, `Exception`, and `TM Error`. The console log output shows the path of each scenario through the code.

The Click Rule JavaScript is shown below.

```
var ffName = "Form Function Full Example";
var ffData = {"testType":data.selectTest};

if(data.selectTest === "TM Error") {
  // Changing the name used will cause TM to report an error.
  ffName = "Will Not Work";
}

console.log("");
console.log("Start...");

Transact.formFunction(ffName, '0.1.0', 'myMilestone', ffData)
.then(function (response) {

  console.log("The Form Function was called, and returned something.");

  if(!Util.isBlank(response)) {
    if (!Util.isBlank(response.data) &amp;amp;&amp;amp; !Util.isBlank(response.data.ex-
ecutionStatus)) {
      switch(response.data.executionStatus) {

        case "DATA_ERROR":
          console.log("The return value status code indicates DATA_ERROR, with message: "
+ response.data.errorMessage);
          console.log("An exception is not being thrown based on design.");
          break;

        case "SYSTEM_ERROR":
          console.log("The return value status code indicates SYSTEM_ERROR, with message:
" + response.data.errorMessage);
          console.log("An exception can be thrown here if design dictates.");
          throw "SYSTEM_ERROR - " + response.data.errorMessage;
          break;

        case "SUCCESS":
          console.log("The return value status code indicates SUCCESS, with data: " +
JSON.stringify(response.data));
          break;

      }
    } else {
```

```
      console.log("An empty data object indicates an service problem.");
      console.log("An Exception/Error will be thrown manually from JavaScript.");
      throw "Empty Data Object";
    }
  } else {
    console.log("An empty response object indicates an infrastructure problem.");
    console.log("An Exception/Error will be thrown manually from JavaScript.");
    throw "Empty Response Object";
  }
}, function(err) {
  console.log("TM throwing an exception gets here - Using function(err)");
  console.log("Error Message: " + err.message);
}).catch(function(err){
  console.log("JavaScript throwing an exception gets here - Using promise's catch(func-
tion(err))");
  console.log("Error Message: " + err);
});

console.log("End of call.");
console.log("Calling Form Functions is an asynchronous operation.");
console.log("Execution continues here before the Form Function returns.");
console.log("");
```

The Form Function Groovy Script is shown below.

```
package com.avoka.funcs.fffullexample;

import com.avoka.tm.func.*
import com.avoka.tm.query.*
import com.avoka.tm.svc.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*

import javax.servlet.http.*

public class FormFunctionFullExample {

    // Injected at runtime
    public Logger logger

    /*
     * Perform Fluent Function call.
     *
     * returns: FuncResult
     */
    FuncResult invoke(FuncParam param) {

        def testType = param.params.get('testType').toString()

        logger.info("Test Type: ${testType}")

        FormFuncResult result = new FormFuncResult()

        def data = [
                rate   : 2.0
        ]

        // Choose execution path
        try {
            switch (testType) {
                case "DataError":
                    // Return data error
                    result.data.put("executionStatus", "DATA_ERROR")
                    String msg = "Invalid Input"
                    result.data.put("errorMessage", msg)
                    result.data.put("exchangeRate", {})
```

```
                    break

                case "SystemError":
                    // Return system error
                    result.data.put("executionStatus", "SYSTEM_ERROR")
                    String msg = "External System Down"
                    result.data.put("errorMessage", msg)
                    result.data.put("exchangeRate", {})
                    break

                case "Exception":
                    // Throw an unexpected exception
                    throw new Exception("FormFunctionFullExampleException")
                    break

                case "Success":
                default:
                    // Return sucessfully
                    result.data.put("executionStatus", "SUCCESS")
                    result.data.put("exchangeRate", data)
                    break

            }
        }
        catch (exception) {
            result.data.put("executionStatus", "SYSTEM_ERROR")
            String msg = "Exception Caught in Service - ${exception.toString()}"
            result.data.put("errorMessage", msg)
            logger.debug msg
            result.data.put("exchangeRate", {})
        }

        return result
    }
}
```

The various return statuses are explained below.

Success

The successful path, or the happy path scenario, is the result when everything in your application and Form Function works as expected. The Form Function validates the input values, any

external services return to the caller function, so everything is good. In the example above, this results in the “Success” case being triggered. The response object contains data to be displayed, and the response.data.executionStatus value is “SUCCESS”.

Note the script log output is asynchronous, which is the nature of `Transact.formFunction()` call.

DataError

Sometimes development of an application and a Fluent Function get out of sync, or more likely an external service’s logic implementation changes. This means that the inputs provided by the Maestro application pass its own validation, but they fail the validation of the Form Function or the third party service. In this case, the Form Function returns “DATA_ERROR” as “`executionStatus`”, and some useful error message as `errorMessage`. You should always try to catch and fix these `DataError` situations during your software development and test cycle.

Execution flow through both the Form Function and application JavaScript is often unrestricted, meaning no exceptions are thrown. The problem can be gracefully reported to the end user, so they can report it to a system administrator or developer.

SystemError

SystemError scenarios cover any error conditions more severe than data validations. In the example above, the Form Function calls an external API and the external system returns an error or throws an exception. The Form Function catches this error and returns the appropriate values in result.data.executionStatus and result.data.errorMessage to the application.

The notable difference lies in implementing the application’s JavaScript throw clause, which is used when a “SYSTEM_ERROR” value is returned in executionStatus. This moves the execution path into the catch() promise, where the error can be handled differently to a DataError in this pattern.

Exception

This execution path shows how an exception being thrown in the middle of the Form Function code should be captured by a try/catch and the result object populated accordingly. This scenario is otherwise handled as a `SystemError`.

Manager Error

Finally, there is the scenario where Manager has failed catastrophically. The example simulates this by using a Form Function that does not exist. Some infrastructure issues can cause similar

errors; for example, network is not available. There is nothing in the Form Function code to handle this kind of error, as it would never be called.

The application JavaScript has a code path that is executed only when this level of error happens, the second promise used for rejections. The console.log() outputs the only information available about the cause of the `error (err.message)`.

Conclusion

This error handling pattern demonstrates two things:

How to code for error conditions using exceptions and the additional promise functionality available;

A way to convey execution status between Manager and the Maestro application run on the end user's device. Using `executionStatus` and `errorMessage` is a simple, and yet powerful approach. It allows the Form Function to pass error related information back to the client while keeping all error handling on the service.

Next, learn about Groovy service log.

Validate Form Data Against a Schema

There are some scenarios where you need to validate submitted XML against an XML Schema. This below code snippet is an example of this in Groovy.

```
import javax.xml.XMLConstants
import javax.xml.transform.stream.StreamSource
import javax.xml.validation.SchemaFactory
import org.xml.sax.SAXException

// submissionString is a String variable containing the submitted XML

// the schema content is stored as a parameter on the Service Definition
def xsd = serviceDefinition.getServiceParameter('schema').getValue()


def factory = SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI)
def schema = factory.newSchema(new StreamSource(new StringReader(xsd)))
def validator = schema.newValidator()
validator.validate(new StreamSource(new StringReader(submissionString)))
```

Next, learn more about the server-side persistence pattern.

User Builder API Overview

The UserBuilder Fluent API is a great API to create new users, update or change user properties, such as their role or spaces assigned, in Manager.

Let's take a look at an example of how to create a new user.

Create New User

A simple code example could look like this:

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

Jser thisUser = new UserBuilder(“Maguire”)
                .setEmail(“johndoe@avoka.com”)
                .setFirstName(“John”)
                .setLastName(“Doe”)
                .setRoleNames([“Developer”, "Maestro Developer"] as Set)
                .setOrgNames([“ACME”] as Set)
                .setSpaceNames([“maguire”] as Set)
                .setUserType(User.TYPE_LOCAL)
                .setLoginName(“alanreiley@gmail.com”)
                .setPassword(“testP@sswd12”)
                .create()
// optionally use createOrUpdate() which will only update if the user already exists.
```

The above example will create a new user with a user name of johndoe@avoka.com, an email address of johndoe@avoka.com, has been assigned to the Maguire space, was granted Developer and Maestro Developer roles, Maguire space name, and the ACME organization. We had to add the as Set for the list types (which wasn't mentioned in the API documentation) so that Groovy knows to cast this as a Set. Otherwise, you will get an error. We also can choose between creating a Local account, LDAP-based, or an SSO-based users, as seen illustrated above with setUserType.

Updating an existing user is almost as simple, but comes with a caveat that you can easily overcome. The any property that is a Set, such as Roles, Spaces, and Organization, the update method replaces the existing properties assigned to that user. Let's take a look at updating an existing user's properties.

Update an Existing User

Groovy gives us a very easy way to convert our object to a string representation - JSON.

```
import com.avoka.tm.svc.*
import com.avoka.tm.vo.*

User user = new UserBuilder("Web Plug-in")
  .setLoginName("johndoe@avoka.com")
  .setRoleNames(["Developer","Maestro Developer","Maestro SCM Design","Maestro SCM Org
Library","Maestro SCM Project"] as Set)
  .update()
// Updates the user account specified by the loginName parameter
```

The above example will find the user: johndoe@avoka.com and updates their role assignment to become: Developer, Maestro Developer, Maestro SCM Design, Maestro SCM Org Library, Maestro SCM Project.

Use the Groovy Console

Remember to select "Commit DB Changes" if you're using this script in a Groovy Console or the DB write will be ignored and no user will be created.

Conclusion

The UserBuilder API is a very powerful tool for creating and modifying users and has become a very handy tool for user creation here within the Support Group. Some other helpful functions of the UserBuilder API are available to set the Account Status, remove an Org, and set the User Type to SSO.

Next, learn how to view Transact functions.

Access Form Properties with a Groovy Service

It can be the case that the operation of a Groovy service (prefill, delivery, dynamic data, etc.) must be tailored slightly for specific forms. One way of storing the data for these customization is at the form level, in form properties. Once the data has been stored in these form properties, it must obviously be accessible from within the service that requires the data.

Accessing properties from within a Groovy script can be achieved by using the PropertyValueDao class. Using this class you can make calls to the 'getPropertyValue' method, providing the name of the property you're looking for and the form object it's associated with. Most Groovy services in Manager, such as Prefill, Dynamic Data, Receipt Number, Form Saved, Submission Complete, have a predefined reference to the form object, in variable called 'form'. For these services properties for that form can be accessed with the following code:

```
import com.avoka.fc.core.dao.PropertyValueDac

PropertyValueDao propertyValueDao = new PropertyValueDao()

def theValue = propertyValueDao.getPropertyValue("The Property Name", form)
```

Delivery Processes on the other hand don't have this existing reference to the form object, but can acquire one easily from the 'submission' object as in the following example:

```
import com.avoka.fc.core.dao.PropertyValueDac

PropertyValueDao propertyValueDao = new PropertyValueDao()

//Notice 'form' has been substituted for 'submission.getForm()'
def theValue = propertyValueDao.getPropertyValue("The Property Name", submission.getForm
())
```

Next, learn how to view Transact functions.

Access Service Parameters of Another Groovy Service

The requirement was for a 'Form Prefill' service to be able to access the service parameters configured within an existing 'Delivery Process' service. The use case being that there was some common configuration between the services and that they wished to maintain these settings in a single location. As it turns out this functionality is built into the Manager's Java libraries and it was fairly straight forward to implement the solution. Start by adding the following imports to the top of your Groovy script:

```
import com.avoka.fc.core.service.IServiceDefinitionAware

import com.avoka.fc.core.service.ServiceLocator

import com.avoka.fc.core.entity.ServiceDefinition
```

Then, use the 'ServiceLocator' class to find the service with the parameters you want to access and retrieve its service definition:

```
IServiceDefinitionAware tmService = ServiceLocator.getServiceForName("The Existing Ser-
vice's Name")

ServiceDefinition serviceDefinition = tmService.getServiceDefinition()
```

Finally, get the service parameter you're looking for and where necessary access the value it contains:

```
def param = serviceDefinition.getServiceParameter("The Parameter's Name");

def paramValue = param.getValue();
```

Next, learn how to view Transact functions.

Create Anonymous Tasks with a Groovy Service

A task is a form that has been prefilled with data and then assigned to a user. The typical use case for a task is: A user opens and completes a blank form. On submission, a task is created for another user, containing the original user's submitted data. The second user then reviews or modifies the data before submitting for processing.

Tasks can be assigned to Manager users, in which case they are accessed when that user logs into a portal, or they can be assigned anonymously, in which case a saved form is generated, and the URL is sent via email. Anonymously assigned tasks can be created in a groovy script by using the `SubmissionTaskService` object.

The SubmissionTaskService API can be accessed at `<Manager server->/manager/admin/javadoc/com/avoka/fc/core/service/SubmissionTaskService.html`

The following groovy script can be used to assign anonymous tasks:

```
import com.avoka.fc.core.entity.*
    import com.avoka.fc.core.dao.*
    import com.avoka.fc.core.service.SubmissionTaskService
    import com.avoka.fc.core.service.SubmissionTaskService.SavedFormParam
    import com.avoka.fc.core.util.PortalUtils
    import com.avoka.core.util.StringTemplate
    import com.avoka.fc.core.service.EmailService


    def formUrl = ""

// Setup the saved form parameter DTO
    def param = new SavedFormParam()

// Setup the form parameter
    def form = DaoFactory.getFormDao().getFormByFormCode( serviceParameters.FormCode ?:
"" )
    if(!form){ throw new Exception("Form Code is invalid, form not found.") }
    param.form = form

// Setup the portal parameter
    // uses the portal provided, if none provided, used the one configured as a service
parameter.
    def portalName = serviceParameters.PortalName ?: ""
    def portal = DaoFactory.getPortalDao().getPortalByName( portalName )
    if(!portal){ throw new Exception("Portal Name is invalid, portal not found") }
    param.portal = portal

// Check the email serviceParameters
    if(serviceParameters.SendEmailFlag){
        if(!serviceParameters.ContactEmailAddress){ throw new Exception("Contact Email
Address must be provided"); }
        if(!serviceParameters.EmailSubject){ throw new Exception("Email Subject must be
provided"); }
        if(!serviceParameters.EmailMessage){ throw new Exception("Email Message must be
provided"); }
    }

// taskSubject
    param.taskSubject = ""
    // taskMessage
```

```
    param.taskMessage = ""

// saveChallengeAnswer
    param.saveChallengeAnswer = ""
    // submissionXml
    if(!serviceParameters.FormXml){ throw new Exception("Form Xml must be provided") }
    param.submissionXml = serviceParameters.FormXml

// Create the task
    def submissionTaskService = new SubmissionTaskService()
    def submission = submissionTaskService.createAnonymousSavedForm(param)

// Task created, send notification email if required
    //=====================================================================
    if(serviceParameters.SendEmailFlag){
        def emailService = new EmailService()

    // Get the URL to resume the saved form
        formUrl = PortalUtils.getFormSavedUrlWithReferenceNumber(portal, submission)

    // Create data map for use in Velocity templates
        def dataMap = [:]
        dataMap.put("formUrl", formUrl)

    def bodyStringTemplate = new StringTemplate(serviceParameters.EmailMessage)
        def emailBody = bodyStringTemplate.merge(dataMap)

        emailService.sendMessage(
            serviceParameters.EmailSubject,
            emailBody,
            null,
            serviceParameters.ContactEmailAddress,
            "",
            null)
    }

return formUrl
```

To run the script, you need to configure the following service parameters:

- ContactEmailAddress - <String> the email address that the URL of the created task is sent to

- EmailSubject - <String>
- FormCode - <String> The form code of the form used to generate the task. This form code must exist on the Manager server.
- FormXml - <String> The xml data used to prefill the generated task
- PortalName - <String> Then name of the portal that the form belongs to.
- SendEmailFlag - <Boolean> Select 'True' to send the email to the assigned contact
- EmailMessage - <String> A velocity template of the email body HTML. Velocity templates contain HTML with placeholders for dynamic content. In this case, a placeholder must be configured for 'formUrl', which will be populated with the link to the created task.

An example of an email message velocity template is:

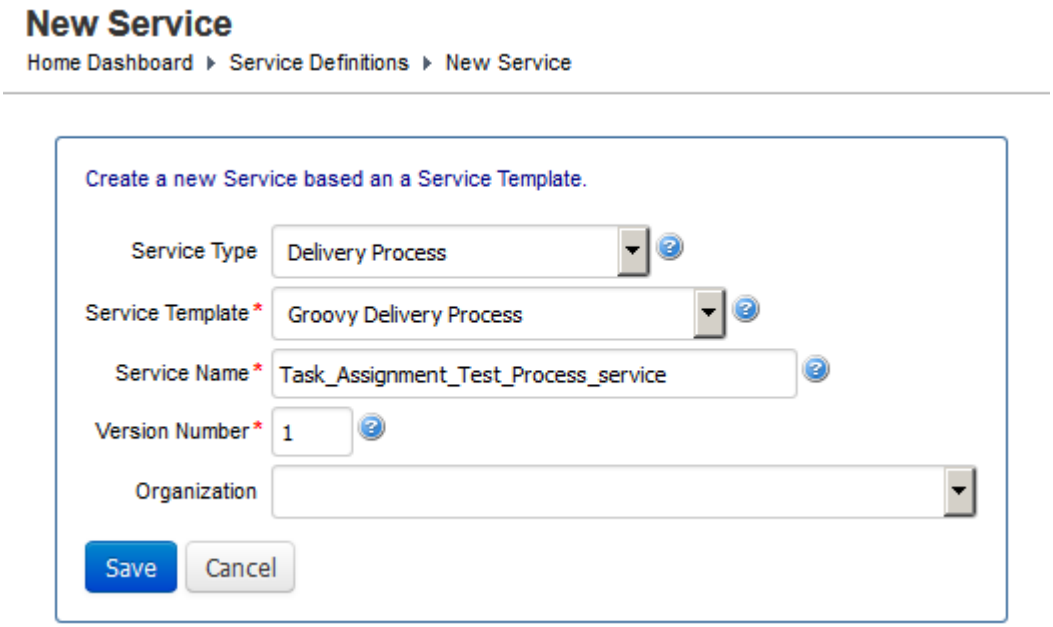
```
<html>
        <head>
            <style type="text/css">
                /* custom CSS here */
                body {
                    font-family: Verdana, Arial, Helvetica, sans-serif;
                }
            </style>
        </head>
        <body>
            <!-- custom Content here -->
            <div>
                This is my message.<br>
                Form Link: <a href="$formUrl">$formUrl</a>
            </div>
        </body>
    </html>
```

Next, learn how to view Transact functions.

Create Form Tasks with a Groovy Delivery Service

A form can be configured to create a Form Task on Submission via configuration of a Delivery Process, which is then assigned to the Form.

First, you need to create a delivery process, as shown below.

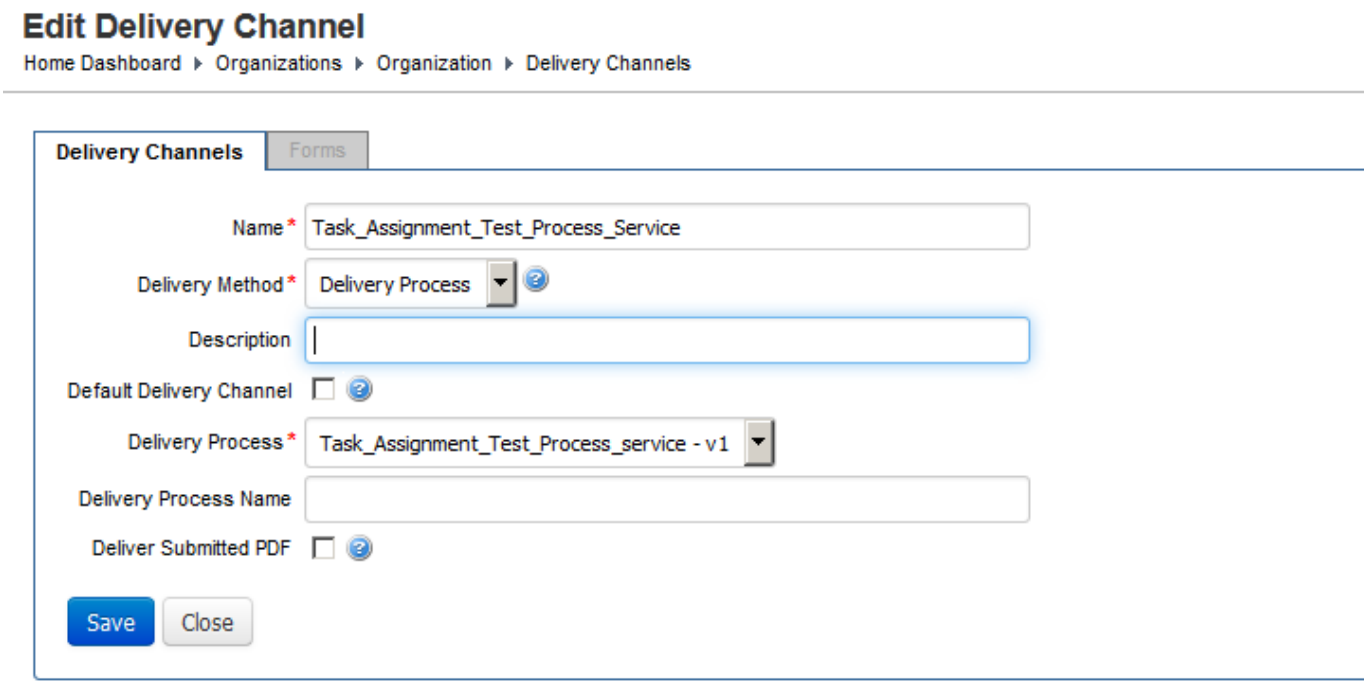


The Service Definition's Groovy Script service parameter is configured with Groovy Script to create a new Form Task using the Task Server createFormTask method.

The following script example shows the simplest example possible of the use of the TaskService to assign a task to a user based on the same form and submission data from the initial user submission. The task is assigned to a single user for review. In real world scenarios a different form would be used for the review or the submission Xml would be manipulated to set the form into review mode when rendered from a task.

```
import com.avoka.fc.core.service.ServiceFactory
import com.avoka.fc.core.dao.DaoFactory


def user = DaoFactory.getUserAccountDao().getUserAccountForLogin("reviewUser");

def taskService = ServiceFactory.getTaskService();
def submission = taskService.createFormTask(submission.form,
                                            "Please Review",
                                            "You have a task to review",
                                            null,
                                            user,
                                            null,
                                            submissionXml,
                                            null,
                                            null,
                                            null,
                                            null,
                                            null,
                                            null,
                                            true,
                                            "Please Review",
                                            "You have a task to review",
                                            false);
```

This Delivery Process method is then added to the Form's Organization Form Delivery Detail configuration. This will enable the Delivery Process to be available on the organization’s forms configuration.



Finally a Form can have its Details section configured with the Organization's Delivery Details for Production and Test Delivery (This is impacted by the Test Mode flag in the Details screen).

Home Dashboard ▸ Forms ▸ Form

Dashboard | Details | Flow Config | Email Verification | Form Versions | Abandonment | Page Tracking | Spaces

Form Display Name * Test Task

Form Code * template-test

Ref Form ID

Form Manager

Transaction Value

Active ☑

Test Mode ☐

Form Version Selector

Submission Expiry Date

Submission Expiry Days

Tracking Code

Use Receipt No. for Tracking Code ☐

Tracking Number Service

Receipt Number

Receipt Number Service

Receipt Number Pattern * ${formCode}-${nextSequence}

Next Sequence Number * 1

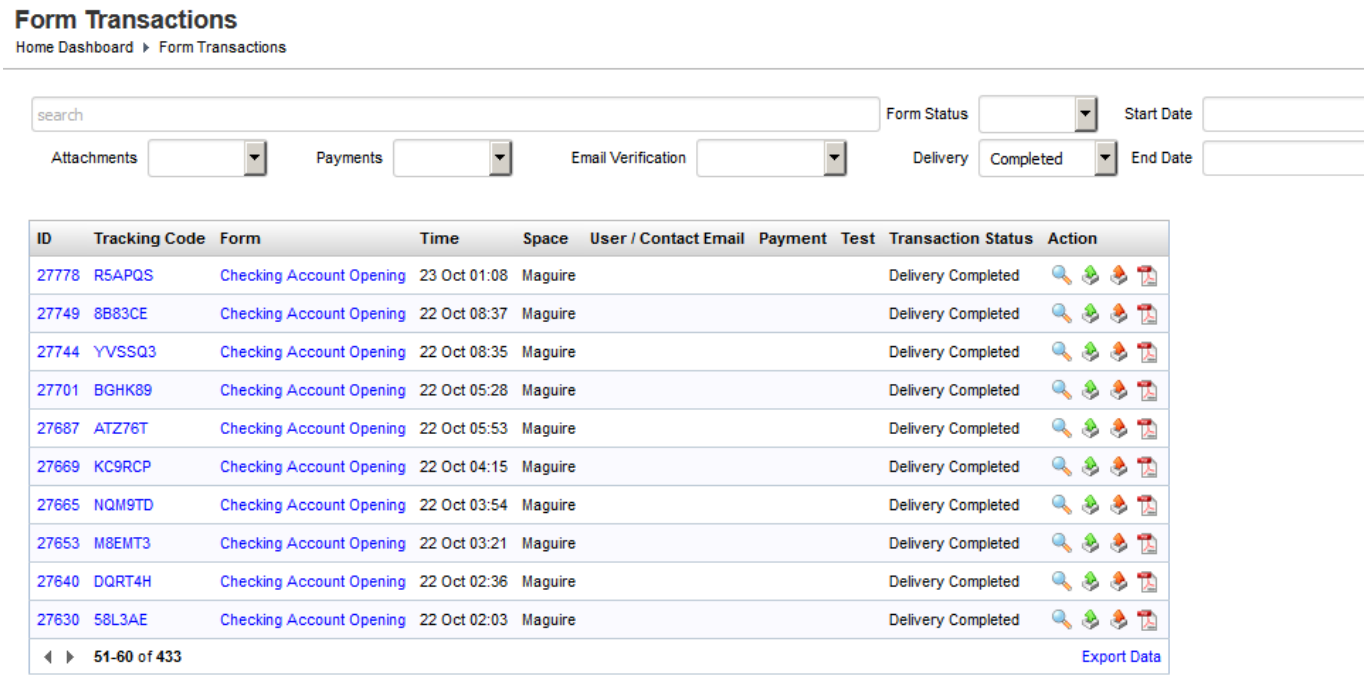
Delivery Channels

Production Delivery Task_Assignment_Test_Process_Service

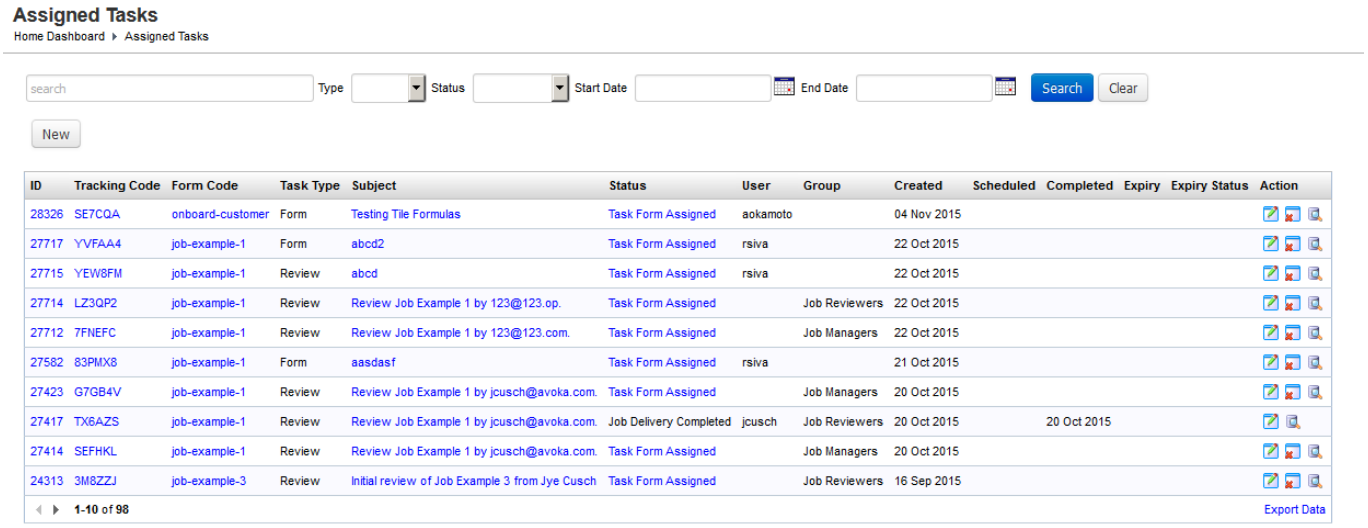
Test Mode Delivery

Validation Failure Delivery

Once a Form is submitted, the Transaction can be viewed from the Operations -> Form Transactions screen. This screen will show the Transaction Status which will include the Delivery Status, and also allow Delivery to be retried.



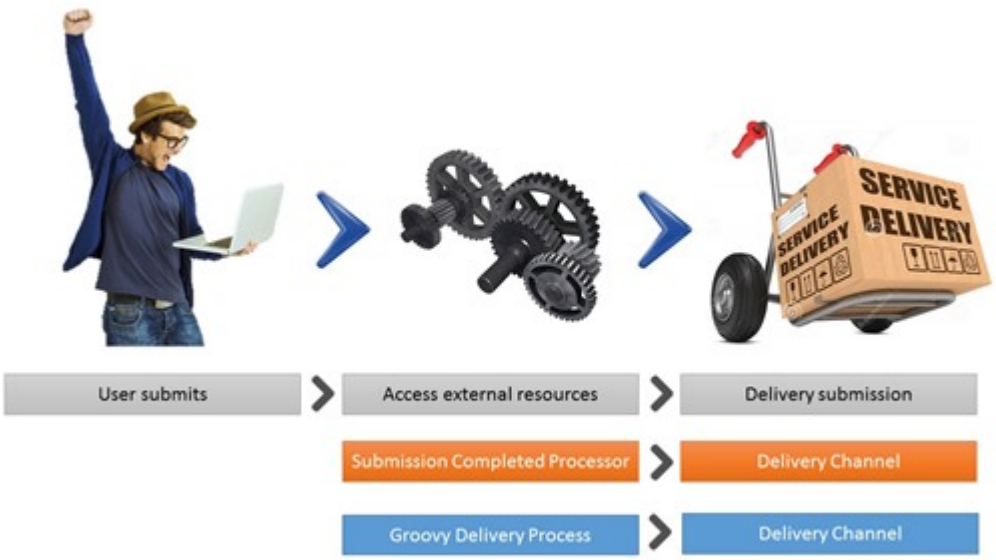
The Operations -> Assigned Tasks screen displays the lists of Assigned Tasks, the assignee and access to the transaction XML.



Next, learn how to view Transact functions.

Mix Standard Delivery Functions with Custom Groovy Processing

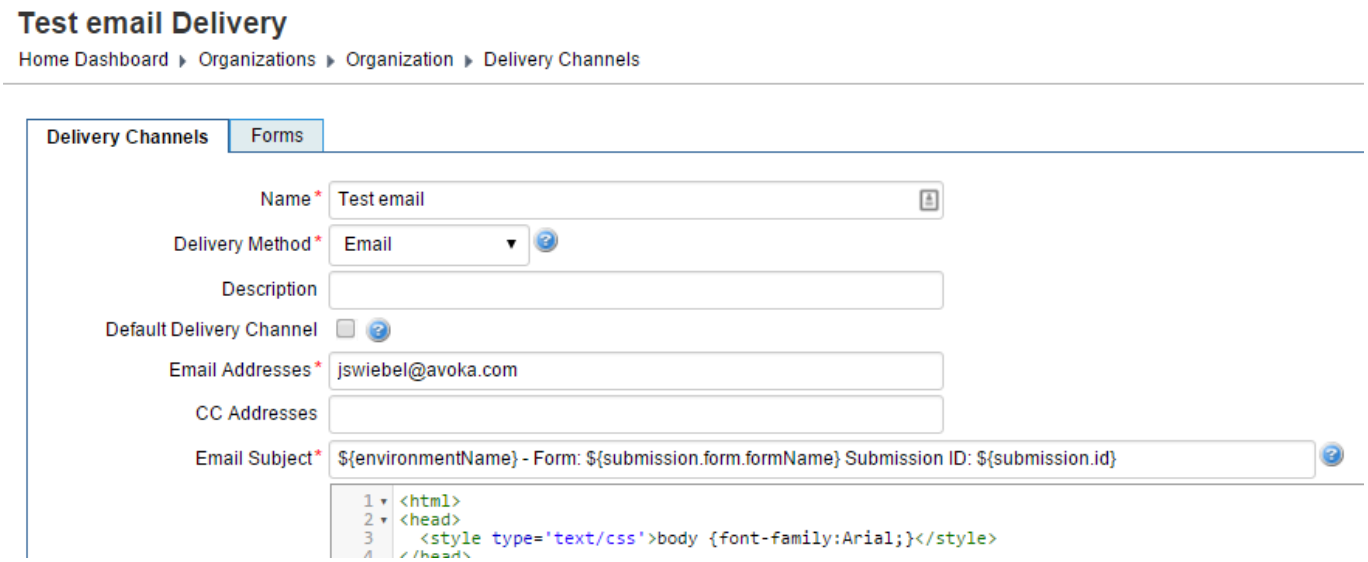
More than once will you be needing to access an external resource (eg web service, file share or perhaps an external mail server) to do something on submission. Typically you would create a Groovy delivery service and do whatever needs to be done in there. Sometimes this approach is a little problematic if your submission requires a delivery method other than Groovy, like Web Service, REST or Email. It's also possible that you already have an existing delivery channel that is used for other forms, but for this one form you need to do something extra before delivery.



The simple solution is to add the logic to the Submission Completed processor, your external resource will be accessed first and then the normal delivery process will kick. But what if the external resource is temporarily not available, the submission completed process will fail and unfortunately cannot be retried.

Let's leek at how the external resources is accessed from a Groovy delivery process, and how the process will change the delivery for that submission to the final (non-Groovy) delivery process.

For this we assume that we already have a non-Groovy delivery channel configured, called "Test Email", using the standard configuration.



Now, create a new Groovy delivery process, and in this you will be able to do whatever it is that needs be done using something external.

```
import com.avoka.fc.core.service.DeliveryResult
import com.avoka.fc.core.service.DeliveryResult.Status
import com.avoka.fc.core.dao.DaoFactory
import com.avoka.fc.core.dao.DeliveryDetailsDao
import com.avoka.fc.core.entity.DeliveryDetails
import com.avoka.fc.core.dao.ClientDao
import com.avoka.fc.core.entity.Form

// here starts the work
// step one, do your external resource stuff like call an external database
if (deliveryCheckpoint.doCheckpoint("Access External Resource", "Write db entry")) {
  try {
     // do the actual method call that does the thing
     storeToDb(.....)
     deliveryCheckpoint.completedCheckpoint("Access External Resource")
  } catch (Throwable e) {
    deliveryCheckpoint.errorCheckpoint("Access External Resource", e)
  }
}
// now make the submission available for Email delivery service
if (deliveryCheckpoint.doCheckpoint("Email Delivery", "Make submission available for
email delivery")) {
  try {
    // First get a new deliveryDetails object for the client (organisation) and the name
of the delivery service you'd like to use
    Form form = submission.getForm()
    DeliveryDetailsDao deliveryDetailsDao = DaoFactory.getDeliveryDetailsDao()
    DeliveryDetails deliveryDetails = deliv-
eryDetailsDao.getDeliveryDetailForClientAndName(
               form.getClient(), "Test email")

    // Set the delivery details on the submission
    submission.setDeliveryDetails(deliveryDetails)

    deliveryCheckpoint.completedCheckpoint("Email Delivery")
    // and return a ready status so that the submission will be ready for delivery via
the newly set delivery process
    return new DeliveryResult(Status.READY)
  } catch (Throwable e) {
    deliveryCheckpoint.errorCheckpoint("Email Delivery", e)
  }
```

```
}
```

Then, create a new Delivery Channel for your form's organisation, this channel needs to be of the type "Delivery Process" and make sure you configure it to use your newly created delivery Groovy process.

The last step is to change the delivery channel on your form to use the new Groovy one and not the old Test email service.

> NOTE
>
> Don't try to change the submission's Delivery details object.

If you try to manipulate the submission's delivery details be prepared for some funky result. Most likely the configuration of the delivery channel itself will be set in an inconsistent state. So don't do this.

If you want to re-trigger delivery, the submission will now go through the submission process twice, so be patient. Re-triggering the submission form Manager will now only execute the newly email delivery process, not the original Groovy service.

Next, learn how to view Transact functions.

Invoke Dynamic Data Service from a Groovy Service

Use Case

A Dynamic Data service is normally used in a Composer Form to perform a lookup or AJAX calls to get information into the form from various integration endpoints. You may come across some situation where you want to leverage existing dynamic data service in your groovy script. For example, I have come across a situation where I have unformatted address string that I want to validate against Veda address service. Manager provides two dynamic data services for Veda address lookup integration. "Veda Address Search" service is used for type ahead address lookup to return possible matches and "Veda Address Detail" service is used to get more granular data for selected or provided address such as Unit No, street no, street name, street type, suburb, etc. I wanted to leverage existing "Veda Address Detail" service for two purpose. One to check if address string is a valid address in Veda address database (that is confirmed by if DPID is returned in the response) and second to get granular information from unformatted address string.

Solution

Firstly you will need to obtain a reference to the service with ServiceLocator class and cast it to IDynamicDataService to invoke required service by its name. As Dynamic Data services are intended to work with AJAX requests, it needs HTTPServletRequest object to invoke service. In our case, we cannot use originating request object as we need to pass our own request parameter (i.e. address string). As the request object is mutable and cannot add new request parameters to it, we need to create a mock request using the `MockHttpServletRequest` class to fulfill our requirement. The code snippet is shown below.

```
import com.avoka.fc.core.service.ServiceLocator;
import com.avoka.fc.core.service.form.IDynamicDataService;
import org.springframework.mock.web.MockHttpServletRequest;
import groovy.json.JsonSlurper;

def concatedAddress = "300 George St Sydney NSW 2000";

def vedaAddressDetailsService = (IDynamicDataService) ServiceLocator.getServiceForName
("Veda Address Detail");
def mockRequest = new MockHttpServletRequest();
mockRequest.addParameter("addressString" , concatedAddress);
def vedaResponse = vedaAddressDetailsService.getFormData(mockRequest, null, null);
def addressJson = new JsonSlurper().parseText(vedaResponse);
```

Next, learn how to view Transact functions.

Dynamic Data Lookup Service

Introduction to Dynamic Data

The Dynamic Data service was introduced to provide back-end querying capability for Maestro forms so some data can be fetched from a Manager service. This allows for reuse of data and functionality to users and customers, saving time and money, while improving the end user experience in a secure and efficient way.

The Maestro widget maintains its focus on end-user ease-of-use, while providing amazing power and flexibility to the Form filling experience. The widget can appear as a Button or a Link and includes a Test facility so you can mock up the return data in Maestro before publishing to Manager.

On the Manager side, a simple to create Service Definition includes all the parameters required to expose a Groovy script which defines the activity of the Dynamic Data Service. Groovy is a language with a syntax very familiar to ActionScript and JavaScript programmers alike. You can find out more about Groovy here http://groovy.codehaus.org/, but this article assumes no prior knowledge of Groovy syntax or structure.

Handling Data

Dynamic Data can cater for single and multiple record return values, and as the Form designer, you need to know when to use the right output structures. Dynamic Data requires only the name of the Dynamic Data Service defined in Manager, but in reality needs an Input Block and an Output Block of widgets to be useful as a look up service. Data passed between the Composer generated SmartForm and Manager (and indeed the test data) is structured using the JSON format, a structure almost universally available on every platform known to mankind. While a complete description is beyond the scope of this article, JSON uses a simple key, value naming arrangement that can support deep nested structures and arrays.

Building the Composer Form

The example form referenced in this article uses an 'ABN Lookup Dynamic Data Service'.

Input Data

The example form takes an ABN number, collected using the ABN widget. The widget is placed inside a Block and configured with whatever other parameters are needed, like mandatory rules and visibility. The Block can be called anything you like and needs no other special

configuration. Note that any fields in the Input Block will be automatically bound and transmitted to Manager as input data. This doesn't mean that your Dynamic Data Service has to use them, just be aware that they are there.

Output Data

This example returns one field called CompanyName to a standard block. Note that you don't have to have your output fields in a block of their own, they can be in a block with other fields. We'll see later how the binding of the output data occurs.

The Lookup Widget

In Composer, locate the Dynamic Data Button widget, typically found in the Manager section of the widget palette, and drop it on the form where you need it to appear. The following dialog will appear.

Click the Browse buttons next to the Input and Result block entry areas to locate the parent Blocks of the Input and Output fields you defined earlier. Finally change the Button type as desired.

You don't need to change the Dynamic Data Service Lookup Name just yet, we'll be using the test mode to prove out our scenario first and then migrating it to Manager a little later on.

Save the dialog.

Testing the Lookup

Our next step will be to provide some test data so we can be assured of the JSON data format required to prove our scenario. Bring up the property editor for the Dynamic Data widget. In the Data section, check the Test Mode flag and enter the JSON formatted return value into the Test Data text area.

The format to use when formulating JSON notation is as follows:

```
{"KeyA":"ValueA","KeyB":"ValueB"}
```

which equates to:

`KeyA = ValueA`
`KeyB = ValueB`

This might appear in Composer as:

```
{"FieldNameA":"Field Value","FieldNameB":"Field Value"}
```

There are a few gotchas to be aware of:

Remember to use Field Names as the key, not labels. Don't be fooled into thinking the hierarchy uses field names, you'll need to look up the field names in the property editor for the field (right click the field in the Structure panel on the left, and select "Edit Properties...").
Values need to be formatted as appropriate for the field.

Here is an example of the JSON test data that I formulated for my specific test case. RegistrationNumber, VehicleDescription and RegistrationRenewalD are the three output fields of type text, text and date respectively.

```
{"CompanyName":"Temenos Journey"}
```

When the Dynamic Data Service component receives the results data from either test or Transaction Manager, it binds the associated value to the appropriate field by means of its name. It's this name binding that makes it possible to specify an output block that has other fields that aren't part of the result set.

After saving the properties and previewing the form, you should see your test data appear in output fields. If your data does not appear, or you get errors you'll need to resolve these before migrating the test data to Transaction Manager.

Building the Transaction Manager Dynamic Data Service.

Transaction Manager exposes a single endpoint for Dynamic Data Services, but determines which service to use by a single hidden field included in the input. Manager also secures the use of the Dynamic Data Service by embedding a hash key into the form when published to thwart possible misuse of the service data by unauthorised clients.

Dynamic Data Service Definitions use a combination of Service Definition configuration values and parameters to configure the service.

Copying an existing Dynamic Service Definition

Log in to Transaction Manager instance and select the System > Service Definitions menu item. Select Dynamic Data from the list of Service Types and click the search button. This gives you a list of the existing Dynamic Data Services already developed and you should go through some

to get a feel for how they work. For the purpose of this example we're going to go ahead and copy an existing service and then modify it to suit our needs.

On the Service Definitions page, click the Copy button. On the Copy Service Definition page, select Dynamic Data as the Service Type, "Groovy Dynamic Data" as the Existing Service Name and enter a new name for the service.

Modifying the Dynamic Service Definition

Now that the service exists, technically we could call it from a published SmartForm but the output data in the script isn't tailored to our needs, so we'll fix that up first. Edit the Service Definition for the new Dynamic Data service and click the Service Parameters tab. Click on the groovyScript parameter name, this will invoke the text area editor which has a handy code syntax highlighter to help you spot errors.

The script below is used by the 'ABN Lookup Dynamic Data Service' on Transaction Manager Demo. It uses the wslite library to call the Australian Government's ABN lookup service.

```
import wslite.soap.*

import net.sf.json.*

import net.sf.json.xml.*

def abn = request.getParameter('abn')

def client = new SOAPClient('http://abr.business.gov.au/abrxmlsearch/AbrXmlSearch.asmx')
def response = client.send(     """<soapenv:Envelope xmlns:soaper-
v="http://schemas.xmlsoap.org/soap/envelope/" xmlns:-
abr="http://abr.business.gov.au/ABRXMLSearch/">

   <soapenv:Header/>

   <soapenv:Body>

      <abr:ABRSearchByABN>

         <!--Optional:-->

         <abr:searchString>""" + abn + """</abr:searchString>

         <!--Optional:-->

         <abr:includeHistoricalDetails>N</abr:includeHistoricalDetails>

         <!--Optional:-->

         <abr:authenticationGuid>avoka-7c4df64c-aa6c-41e5-abf1-0ffe200a8833</-
abr:authenticationGuid>

      </abr:ABRSearchByABN>

   </soapenv:Body>

</soapenv:Envelope>"""

)

String organisationName =
```

```
response.ABRSearchByABNRe-
sponse.ABRPayloadSearchResults.response.businessEntity.mainName.organisationName;

def jsonResponse = new JSONSerializer().toJSON(["CompanyName": organisationName]);

return jsonResponse;
```

The important parts of the code to take note of are:

```
def abn = request.getParameter('abn')
```

This line searches the input block in the composer form for a field called 'abn', and saves it's value to a variable in the script called abn.

```
def jsonResponse = new JSONSerializer().toJSON(["CompanyName": organisationName]);
```

This line takes the organization name (saved in the variable 'organisationName', and inserts it into a JSON object, using "CompanyName" as a key. This will save the value returned by this script to a field named 'CompanyName' in the output block in the composer form.

Linking the Dynamic Data service in Transaction Manager to the Composer form.

The Composer form specifies the Dynamic Data service it will use by name, in the 'Dynamic Data Service Name' field. When publishing the form, remember to clear the 'Test Mode' button, or the service will not be called.

A Final Word - Testing Testing Testing

OK, so there's a good chance that when you got it all hooked up it didn't work. For debugging I highly recommend FireFox and Firebug or Chrome and it's Dev Tools for analysing page behavior. In the screenshot you'll notice there's a section at the bottom of the page where you can see the POST request to the Manager servlet that services the Dynamic Data Services requests. The response from that POST includes the data as formatted in my Test Mode data. Yours should too.

Bonus Points - Repeating blocks of results

Thankfully, the Composer developers catered for multiple results sets by allowing data to be returned in a JSON array. This is easily accomplished by wrapping the JSON response data in an array spec and separating records with commas, like so:

```
[{"record1":"value1"},{"record2":"value2"}]
```

On the Composer side, the output block must be Repeatable Content to get the Repeatable-Block field definition. Again, field binding is done by name with each record in the JSON array creating a new repeat section. Do be careful if switching between the two types (Repeating and Standard Blocks) since the Output Block specification differs depending on the type. In my example the Output Block specification for a single result is:

```
../VehicleDetails
```

If I were to change the block to Repeatable Content in response to some change in development, I must ensure my Output Block specification is updated and appears as follows:

```
../VehicleDetails[*]
```

Conclusion

Dynamic Data Services is a very important and strategic feature for Transact providing a mechanism to further enrich the user experience by including interactivity between Composer and Transaction Manager. The range of possible applications for this capability are untold with the flexibility and power in the current design.

Next, learn how to view Transact functions.

Check a Form Request IP Address

It may be necessary in certain instances to determine the IP address of the user who has requested a form, for example, so that internal users can view a different version of a form than external users, or so that users in different countries or areas see different form content.

The form must be pre-populated with either the IP address of the requesting user, or an indicator of what type of user they are, which has been determined using the IP address in a pre-populated Groovy script.

You can do this in Manager:

1. Create the Groovy prefill service:

```
// Script parametes include: form (Form), formRequestParams (Map<String, Object>),
request (HttpServletRequest), schemaSeed (Document), userAccountId (String), ser-
viceDefinition (ServiceDefinition), serviceParameters (Map<String, String>)
// Script return : Form XML data string value or return empty value and update the
schemaSeed Document parameter
def returnXml = '''
<prefillData>
    <value1>''' + request.getRemoteAddr() + '''</value1>
    <value2>''' + request.getHeader("X-Forwarded-For") + '''</value2>
</prefillData>
''';
return returnXml;
```

2. Create a Service Parameter containing the groovyScript. The groovyScript includes a request parameter. The request parameter is a HttpServletRequest Java object, which contains information about the request. The IP address that the request came from can be accessed using the following code:

```
request.getRemoteAddr()
```

   Depending how the TransactionManager server is set up, this may not always give you the user's address. For example, tm.demo.avoka.com sits behind a load balancer, which catches the user's HTTP request and uses it to create a new one. Using the above code will return the IP address of the load balancer, rather than the user. In this instance, the user's IP address must be accessed through the HttpServletRequest object's header, using the code:

```
request.getHeader("X-Forwarded-For")
```

3. Create the sample input XML. In the form version data config, go to the input XML Prefill Mapping tab click Edit Input XML
4. Create the XML Mapping. On the same tab click the Edit XML Mapping button
5. Update the form version to include the Form Prefill Data Service.

Next, learn how to view Transact functions.

Groovy Service Tips and Tricks

Many services and functions in Manager are written in Groovy[1]. This is because it gives you extra flexibility in implementing a wide range of customer requirements that often can be rather complex. To make it even easier for you, we provide access to most Manager functionality via Fluent APIs, which are available in the SDK for each product release,

When working with Groovy using our API, you may face some typical problems, so we have captured some useful Groovy tips and tricks below to help you solve them.

Get the version of Groovy

```
print GroovySystem.getVersion()
```

Get Date widget into a W3C dateTime

The W3C has a specification for date time formats which has resulted in the sublime XMLGregorianCalendar Java class. While Groovy provides some convenience mechanisms to work with Java types such as Date, an it's relatively easy to use format specifiers to produce compatible dateTime formats, I prefer the brute force code approach. Below is an example that takes a date from a Composer widget and converts it into W3C compatible dateTime via Calendar objects

```
javax.xml.datatype.DatatypeFactory.newInstance().newXMLGregorianCalendar(Date.parse
("yyyyMMdd","19710517").toCalendar())
```

Here's a similar method that takes a date object and produces the compatible dateTime format

```
def StringBuffer timestamp = new StringBuffer(
       new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSSZ").format(new Date())
)
```

[1]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

Manager included libraries

An easy way to check this is go to JBoss module folder for groovy, such as `C:\avoka\transact\manager\server\modules\org\codehaus\groovy\main`.

The file module.xml specifies like the following:

Manager 3.5

```
<module xmlns="urn:jboss:module:1.0" name="org.codehaus.groovy">
        <resources>
            <resource-root path="groovy-all-2.1.0.patch.jar"/>
            <resource-root path="groovy-wslite-0.7.1.jar"/>
            <resource-root path="http-builder-0.5.2.jar"/>
        </resources>

    <dependencies> <module name="com.avoka.core" />
            <module name="com.microsoft.sqlserver" />
            <module name="com.mysql" /> <module name="com.nuodb" />
            <module name="com.oracle" /> <module name="javax.api"/>
            <module name="javax.servlet.api"/>
            <module name="net.sourceforge.json" />
            <module name="net.sourceforge.saxon" />
            <module name="net.sourceforge.nekohtml"/>
            <module name="org.apache.camel.core" />
            <module name="org.apache.camel.spring" />
            <module name="org.apache.commons.codec" />
            <module name="org.apache.commons.collections" />
            <module name="org.apache.commons.lang" />
            <module name="org.apache.commons.logging" />
            <module name="org.apache.commons.io" />
            <module name="org.apache.httpcomponents"/>
            <module name="org.apache.log4j"/>
            <module name="org.apache.xml.xml-resolver"/>
            <module name="org.junit"/> <module name="org.opensaml" />
            <module name="org.slf4j" /> <module name="sun.jdk" />
            <module name="system" />
        </dependencies>
    </module>
```

The dependency section and resources section are libs that your groovy scripts can use.

Groovy script body building

You can use the following code template to develop a SOAP message in the Groovy Script Console in Manager

```
import wslite.soap.*
    def body = new SOAPMessageBuilder().build {
        version SOAPVersion.V1_2
        encoding('UTF-8')
        envelopeAttributes(
            'xmlns:_04':'http://cai.dataaction.com.au/Cai/Member/New/2011_04_15/',
            'xmlns:dat':'http://schemas.datacontract.org/2004/07/DataAction.Cai.Wcf',
            'xmlns:new':'http://cai.dataaction.com.au/Cai/Member/New/'
        )
        header{
            Action('soap-env:mustUnderstand':1,
xmlns:'http://schemas.microsoft.com/ws/2005/05/addressing/none','http://cai.dataaction.com.au/Cai/Member/New/2011_04_15/MemberContro...)
            To('soap-env:mustUnderstand':1,
xmlns:'http://schemas.microsoft.com/ws/2005/05/addressing/none','https://data-dfw-cai3.cai.dataaction.com.au:8443/cai/member/new/2011...)
        }
        body{
            '_04:CreateNewPersonalMember'{
                '_04:newMember'{
                    'dat:Organisation' ('Avoka')
                    'dat:RequestId'(UUID.randomUUID())
                }
            }
        }
    }
    print body
```

Capturing Groovy Script Output

Capturing Groovy Script Output with SystemOuptutInterceptor

```
import groovy.ui.SystemOutputInterceptor
    ...
    def outputString = ''
    // a simple closure to add the iterator to the string; returning false diverts the
output from system.out
    def interceptor = new SystemOutputInterceptor({ outputString += it; false})


    interceptor.start()
    // ...do some Groovy code that generates output
    interceptor.stop()


    print outputString
```

Use a Builder and a Map to Conveniently Return Dynamic Data

Dynamic Data Services require data in a specific format that can be a little tricky to format. It uses an array around JSON objects to parse and fill repeating groups of output fields in the form, for example:

```
[{"Name":"Value"},{"Name":"Value"}]
```

But that can be a real pain to get right if you're trying to do it using string concatenation and place-holders, error prone at best. This example produces a JSON object using a JsonBuilder with a Closure. Wraps it nicely inside an array and then returns as a string (to the Dynamic Data Service)

```
JsonBuilder + Map = Dynamic Data
    def json = new JsonBuilder({
            Name(Value)
        })
    def arr = [json.toString()]
    return arr.toString()
```

Mocking up Web Service SOAP Return Data

Developing services on Transaction Manager is not the friendliest or easiest of experiences simply because it's intent is to provide a lean administration system, not a development

enviornment. Yet there are times when the only way to get at the values you need are on Transaction manager, especially when connected to remote, thrid party systems. This is a simple example top mock up a web service response which you can then use to develop the response action like extracting data.

Mocking up a SOAP Response

```
def soapMessageText = '''
    <s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a-
a="http://www.w3.org/2005/08/addressing">
        <s:Header>
            <a:Actio s:mustUnder-
stand="1">http://cai.dataaction.com.au/Cai/Member/New/2011_04_15/MemberContro...
                CreateNewPersonalMemberResponse</a:Action>
        </s:Header>
        <s:Body>
            <CreateNewPersonalMemberResponse xmlns-
s="http://cai.dataaction.com.au/Cai/Member/New/2011_04_15/">
                <CreateNewPersonalMemberResult xmlns:b-
b="http://schemas.datacontract.org/2004/07/DataAction.Cai.Wcf"
                    xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
                    <b:ReceiptNumber>8071296</b:ReceiptNumber>
                    <b:RequestId>d9da9951-1a6f-4f49-bfc0-9d00fc44f491</b:RequestId>
                    <b:Body xmlns:c="http://cai.dataaction.com.au/Cai/Member/New/">
                        <c:MemberNumber>22562764</c:MemberNumber>
                    </b:Body>
                </CreateNewPersonalMemberResult>
            </CreateNewPersonalMemberResponse>
        </s:Body>
    </s:Envelope>

'''
    import wslite.soap.*
    import wslite.http.*
    def httpRequest = new HTTPRequest()
    def httpResponse = new HTTPResponse()
    def soapEnvelope = new XmlSlurper().parseText(soapMessageText)

    response = new SOAPResponse(httpRequest:httpRequest,
                                httpResponse:httpResponse,
                                envelope:soapEnvelope,
                                text:soapMessageText)

    print response.en-
velope.Body.CreateNewPer-
sonalMemberResponse.CreateNewPersonalMemberResult.Body.MemberNumber
```

Groovy is a very powerful and dynamic language, and there are several different ways to work with and manipulate objects of different types.

You also need to be aware that Transact Services implement static type checking, in order to improve error checking and security. This creates some restrictions in the coding styles you can use.

Typed Objects

The best way to represent objects is as classes. (Some will no doubt have alternate opinions or a different use case, and this is an arguable point.)

Some simple code looks like this:

```
class Customer {
    String first
    String last
}

Customer customer = new Customer()
customer.first = "Mary"
customer.last = "Smith"

assert "Mary" == customer.first
```

Using classes is a little more work, because you have to define the class. And you have to modify it if there are any changes to the data structure. (This is compared to collections, which are inherently dynamic.)

But the advantages are:

A very simple syntax: "customer.first"

If you are using an IDE (you should be) then the IDE will give you code completion when you are typing

Safety - if you accidentally type: "customer.furst", you will get an error. In a dynamic Groovy collection, you have just created a new entry called "first".

JSON

Groovy gives us a very easy way to convert our object to a string representation - JSON.

```
import groovy.json.JsonOutput

String str =  JsonOutput.toJson(customer)
println str
println JsonOutput.prettyPrint(str)
```

This will produce: `{"first":"Mary", "last", "Smith"}`

The pretty version will produce a more easily human readable form including line breaks and indentation.

Collections and JSON

You can alternately use collections - maps and lists - as an alternative to statically defined classes. Let's try this by converting our JSON string into a collection.

```
import groovy.json.JsonSlurper

def slurper = new groovy.json.JsonSlurper()
def customerMap = slurper.parseText(str)
```

This will convert the JSON sting into a collection. In this case, it's just a simple map, but if the JSON was more complicated, the map could contain other maps and lists, instead of just strings.

In regular Groovy, we can refer to the elements within the collection like this:

```
println customerMap.first
```

However, because Manager implements static type checking, you can't use this syntax directly. Because Groovy doesn't know what type "customerMap" is, it assumes it's just a regular object, which doesn't have a "first" attribute.

To fix this, you can either using casting (which gets a bit tedious, especially as your object graph gets deeper), or use the following alternate syntax:

```
println customerMap["first"]
```

Create an Object Based on a Collection

Groovy provides a very useful default constructor for any object based on a collection. This means that if we choose to go with typed classes, we can easily convert from our collection to a typed object.

The code below shows how we can get from a JSON string to a collection, and from a collection to a typed object. You could of course do this in a single line of code, but we've expanded for clarity.

```
import groovy.json.JsonSlurper

def slurper = new groovy.json.JsonSlurper()
def customerMap = slurper.parseText(str)
Customer customer2 = new Customer (customerMap)
```

Next, learn how to view Transact Functions.

HTTP Session Object Pattern

You can store and retrieve data against the http session object in the usual way. However, the session object is (by definition) temporary/volatile, and may "go away" under several circumstances, such as save-and-resume. Data stored in the session object cannot be easily viewed in Manager, and therefore makes debugging difficult. Don't mess with the session object itself, because it is used internally by Manager. For example, the following code will cause all sorts of problems:

```
request.getSession(false)?.invalidate() // DON'T do this!!!!
```

As a general recommendation, you should strongly consider storing and retrieving data using the Manager built-in services.

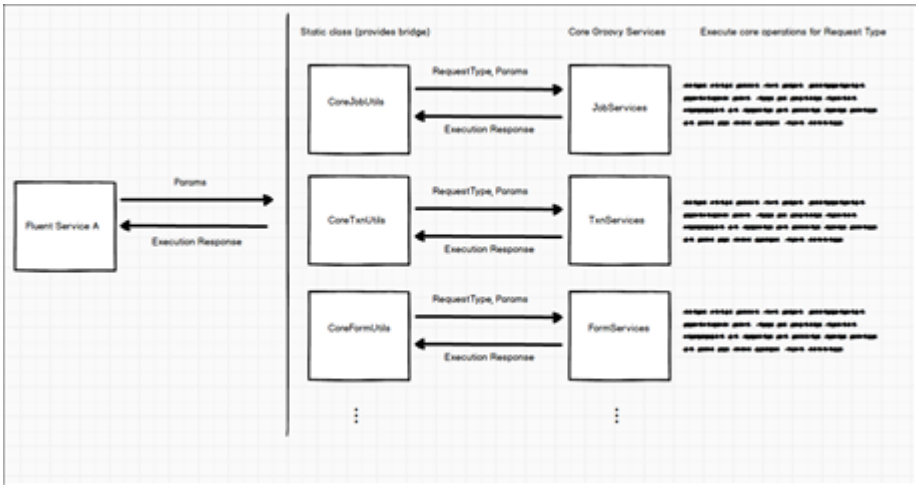
Next, learn more about the server-side persistence pattern.

Invoke Core Services Pattern

Fluent SDK provides great ability to execute Fluent APIs efficiently. Most of your project requirements should be served through Fluent APIs. However, you may find a situation where some of the operations that you were able to do earlier (prior Transact 17.10) using Core APIs are no longer supported by Fluent API. You should reach out to product team to discuss these requirements and create a request to add the support in Fluent API in future release. This may take some time due to product release cycle and your project cannot wait till then. This article explains the design pattern you should use to invoke Core APIs in your project and eventually replace when product team delivers that functionality with new release.

Let's take a use case for explaining this design pattern. In this example your project has a fluent service A needs to execute certain operations such as updating "Job Status" that is only available through Core API (at the time of writing this article). Fluent service does not allow you to import core classes. So you have to create a core groovy service by invoking "svc-scaffold-legacy" ant target in your sdk.

As illustrated in the diagram below, Fluent service may pass parameters that is required by core service to execute core operations. I have split core operations by categories such as Job, Transaction, Form etc as you may need certain operations under each category which are not available in fluent API yet.



Step 1

In this example below, I have created three static classes (which will be added dependencies in Fluent service definition). You can add more static classes as you need for other categories of Transact platform.

- CoreJobUtils - Provides all helper methods for collaboration job object (code is in Step 5)
- CoreTxnUtils - Provides all helper methods for Transaction object
- CoreFormUtils - Provides all helper methods for Form object.

Create these core services under projects "utils > core" package for better isolation. e.g. com.avoka.project.utils.core

Step 2

For consistency we will need to create a generic POJO that can be used as a wrapper for the response from core service. Let's called it "ExecutionResponse" groovy class and put that under "vos" package.

```
package com.avoka.cnb.cnb1.vos;

public class ExecutionResponse {

    public static final String STATUS_SUCCESS = "Success";
    public static final String STATUS_ERROR = "Error";

    private String status;
    private String errorMessage;
    private String serviceResponse;

    public ExecutionResponse() {}

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getErrorMessage() {
        return errorMessage;
    }

    public void setErrorMessage(String errorMessage) {
        this.errorMessage = errorMessage;
    }

    String getServiceResponse() {
        return serviceResponse
    }

    void setServiceResponse(String serviceResponse) {
        this.serviceResponse = serviceResponse
    }
}
```

Step 3

Fluent service needs to include required static class and ExecutionResponse in service definition so that it's available during execution.

```
{
    "name": "groovyScript",
    "filePath": "CNBProductSelectorPreprocessorAng.groovy",
    "fileIncludes": [
        "com/avoka/cnb/cnbl/utils/core/CoreJobUtils.groovy", "com/a-
voka/cnb/cnbl/vos/ExecutionResponse.groovy"
    ]
}
```

Step 4:

Create a core service by executing "svc-scaffold-legacy" target. Let's call it "CNB Job Services". This single groovy service will execute all core API operations by looking at "RequestType" request parameter which will be passed by Utils class in Step 4. This way we dont need to create multiple core groovy services for each missing operation. This service is responsible for creating "ExecutionResponse" object with relevant attribute values such as Status, error message and service's raw response. This will be then forwarded back to Utils class and eventually back to caller which is Fluent service.

```
package com.avoka.cnb.cnb1.services.core.job

import com.avoka.cnb.cnb1.vos.ExecutionResponse;

/* Generic Groovy script to perform all Job related operations that currently not sup-
ported by Fluent

   Script parameters include:
       parameters : Map<String, String>
       serviceDefinition : com.avoka.fc.core.entity.ServiceDefinition
       serviceParameters : Map<String, String>
       args : List<String>
       job : com.avoka.fc.core.entity.Job
       jobAction : com.avoka.fc.core.entity.JobAction

   Script return:
       script result
*/
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.fc.core.dao.DaoFactory
import com.avoka.fc.core.service.job.impl.JobControllerService
import com.avoka.tm.util.Contract

ExecutionResponse executionResponse = new ExecutionResponse();

String status = ""
String errorMessage = ""
String requestType = parameters["RequestType"];
Contract.notNull(requestType, "Request Type");

Map<String, Object> executionParams = parameters["ExecutionParams"];
String jobReferenceNumber = executionParams["JobReferenceNumber"];

def requestedJob = DaoFactory.getJobDao().getJobForReferenceNumber(jobReferenceNumber);
if(requestedJob) {
    if (requestType == "STATUS_UPDATE") {
        String newStatus = executionParams["JobStatus"];
        requestedJob.setStatus(newStatus);
        new JobControllerService().commitChanges();
        status = ExecutionResponse.STATUS_SUCCESS;
    }
```

```
} else {
    status = ExecutionResponse.STATUS_ERROR;
    errorMessage = "Unable to find a job with reference number ${jobReferenceNumber}";
}

executionResponse.setStatus(status);
executionResponse.setErrorMessage(errorMessage);

return executionResponse
```

Step 5

Add a static method in "CoreJobUtils.groovy" class to invoke core service to update Job status. This service is responsible for updating job status and return back "ExecutionResponse" object with execution status.

```
packagecom.avoka.cnb.cnb1.utils.core;
importcom.avoka.cnb.cnb1.vos.ExecutionResponse;
importcom.avoka.tm.svc.GroovyServiceInvoker;
importcom.avoka.tm.vo.SvcDef;
publicclassCoreJobUtils {
    /**
     * Updates the job status
     * @param svcDef Service that requests this activity to be added to the package life-
cycle
     * @param jobReferenceNumber Collaboration Job Reference Number
     * @param jobStatus Job Status
     * @return ExecutionResponse object contains status and error message
     */
    publicstaticExecutionResponse updateJobStatus (SvcDef svcDef, String jobRe-
ferenceNumber, String jobStatus) {
        Map<String, Object> reqParams = newHashMap<>();
        reqParams.put("RequestType", "STATUS_UPDATE");
        Map<String, Object> executionParams = newHashMap<>();
        executionParams.put("JobReferenceNumber", jobReferenceNumber);
        executionParams.put("JobStatus", jobStatus);
        reqParams.put("ExecutionParams", executionParams);
        Object activityResponseData = newGroovyServiceInvoker()
                .setServiceName("CNB Job Services")
                .setClientCode(svcDef.clientCode)
                .invoke(reqParams);
        return(ExecutionResponse) activityResponseData;
    }
}
```

Step 6

Now, let's invoke the utility method in the Fluent service itself that allows us to update job status by invoking actual core groovy service.

```
ExecutionResponse executionResponse = CoreJobUtils.updateJobStatus(svcDef, job.ref-
erenceNumber, Job.STATUS_CANCELLED);
if(executionResponse.status == ExecutionResponse.STATUS_SUCCESS) {
    logger.info "Cancelling job "+ job.referenceNumber + " due to edit package event"
} else{
    logger.error "There is a problem with cancelling job ${job.referenceNumber}. Error:
${executionResponse.errorMessage}";
}
```

Benefits

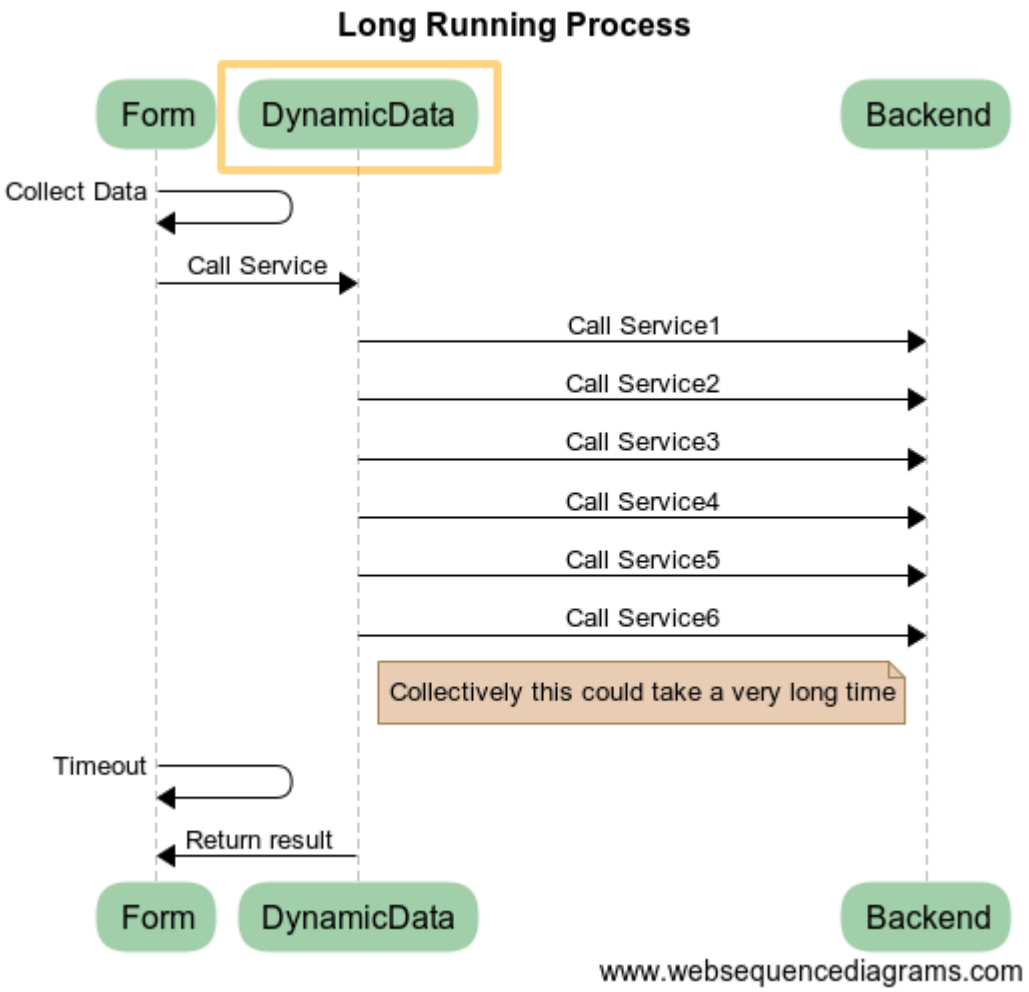
- Provides a controlled approach to isolate core and fluent services usage in your project.
- Utility classes lists all missing methods that eventually you need a support in Fluent API. This can also help you to find a usage of those methods in your project to understand the impact of a change. If you directly invoking core groovy service from fluent service, you have to rely on text search for service name in your project which can be messy in a big complex project.
- As we get fluent API support with an upgrade, you can find all usages and replace the code with Fluent API easily. This can also help you understand which area you have to focus on testing in order to make sure things work as expected. You can then delete that method from Util class. Ideally we don't want any methods in that class as support should be through Fluent.
- Fluent SDK combines all file includes in service definition into a single file during application deploy to Manager. Splitting into multiple Utils class provides less footprint in actual service output. Not something you need to worry about but still better to have less duplicate and lengthy code across file which provides better readability if someone viewing from Manager admin console.

Next, learn more about the server-side persistence pattern.

Long Running Service Pattern

Sometimes you may need to invoke a Dynamic Data Service from a form, and you know that it may take some time. Perhaps there is a back-end service that is slow, or perhaps you are calling several different services in order to get your answer.

It might look something like this:



There is two problems with this:

- Both Maestro and the Transact Manager web server (usually Apache) have a built-in service timeouts. If you exceed this timeout, you will receive an error in your application. This means that you will never actually receive a result.
- While the service is running, the user has no feedback on what is going on.

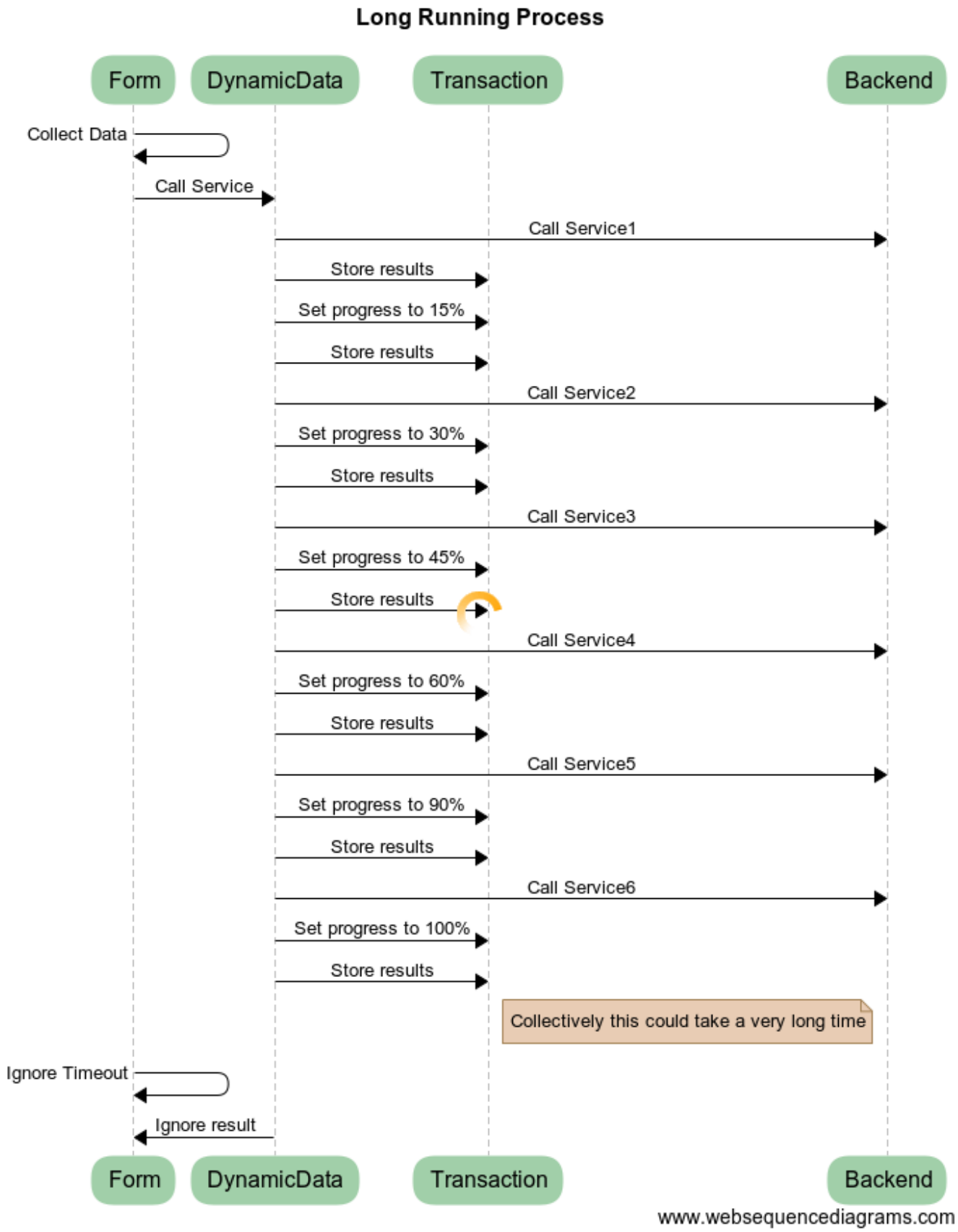
One solution would be to increase the timeouts. However, the timeouts are there for a reason - they are there to protect the application server from having a large number of long-running HTTP

connections open. Too many open connections, and the server will eventually run out of resources.

Here are two patterns that may be helpful to work around this problem.

Progress Polling

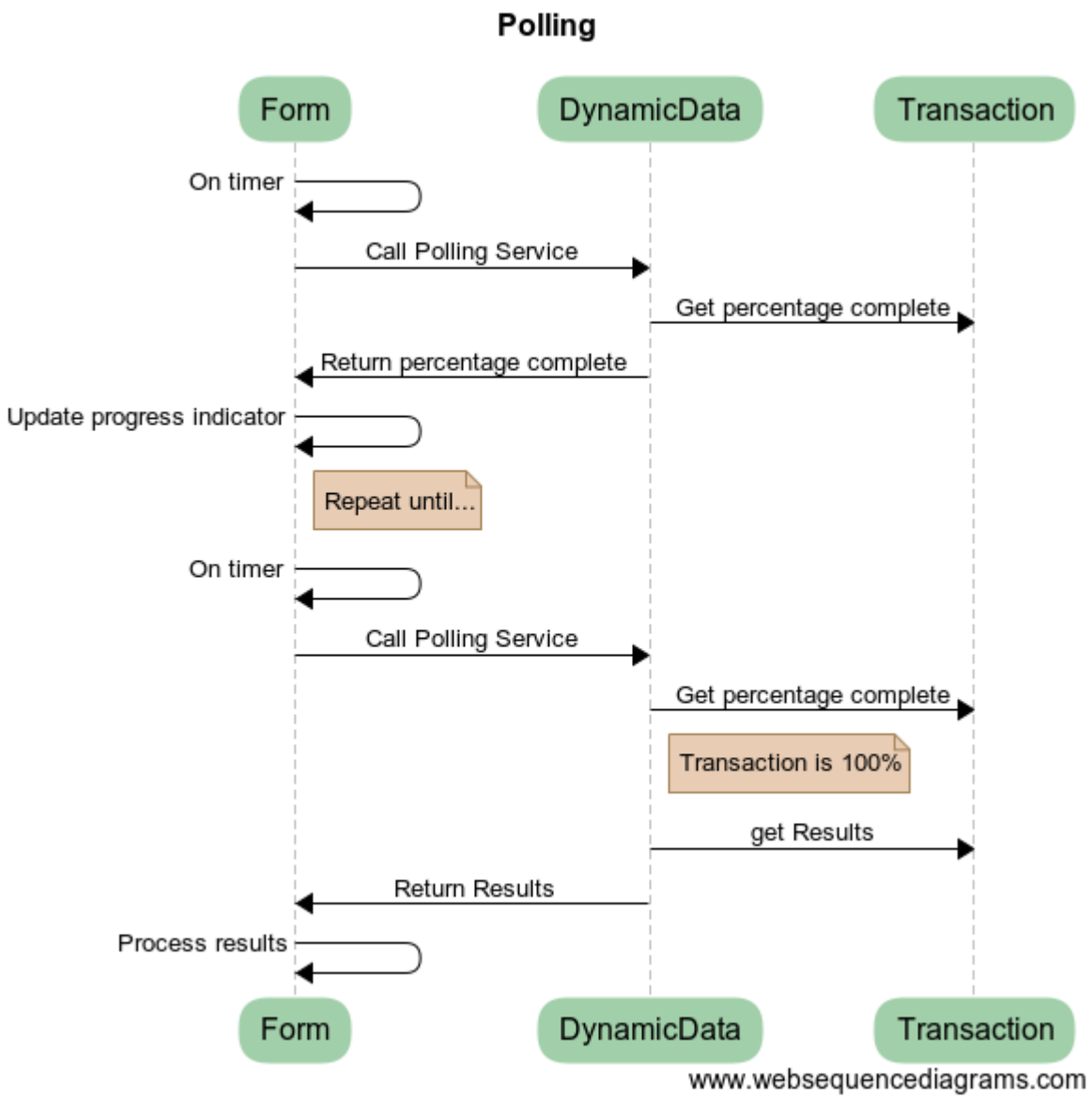
One of the useful things about the web (and Manager) is that services are all asynchronous. So we can modify the above sequence to look something like this:



We've basically made three changes:

1. Update a transaction property to represent the percentage progress of the long running process.
2. Store the intermediate results of the service calls in a transaction property (or multiple properties).
3. Ignore both the timeout and the result of the initial Dynamic Data Service call.

Then, in parallel, we will implement another Dynamic Data Service call. This will be on a timer, and will call the Manager server periodically.



The form will call the Polling Service periodically.

The polling service will check the percentage complete, and either;

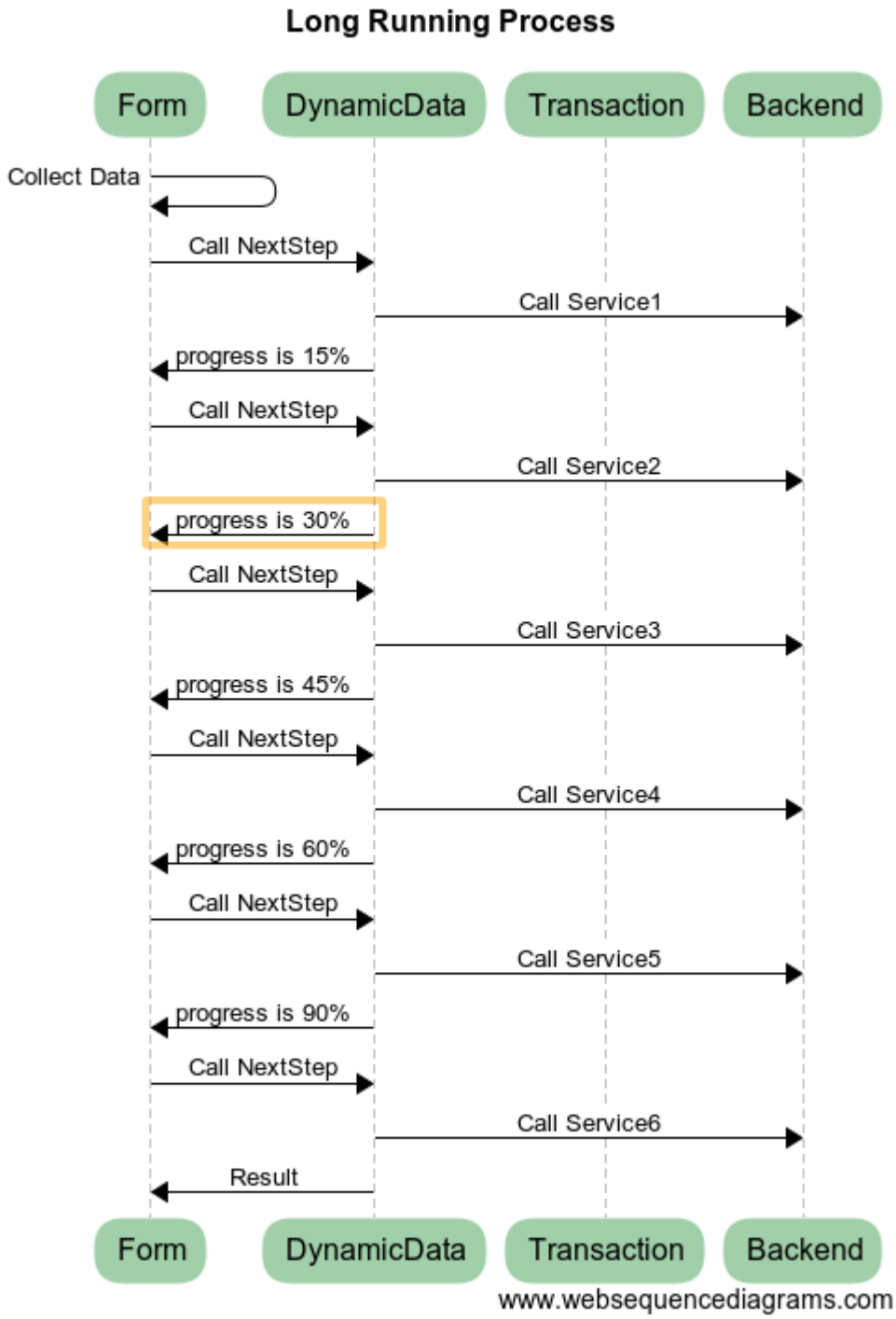
- Return the percentage complete, so that the form can update the progress meter (or similar) (This is shown in the top half of the diagram above.)
- If the process is complete, return the results, so the form can use them. (This is shown in the bottom half of the diagram above.)

This solves both problems.

Blind State Machine

Another way to solve this is to use a blind state machine.

This breaks up the sequence of execution, but does so in a way that the Form is unaware of the details or logic that is occurring on the server. This helps to make sure that the form cannot be compromised by a malicious user.



In this pattern:

- We break up the calls to ensure that none of them exceeds the timeout value.
- In the same way as the above solution, we are giving feedback to the form that allows it to update a progress meter.

- We are careful not to expose the internals of the state machine, or how its logic works. All we are doing is calling "NextStep" until we get back a result rather than a progress indicator.

Enterprise Service Bus

A third approach is to use an Enterprise Service Bus or similar architecture. In this case, we would use message queues or similar patterns to "kick off" a long running process. We would then using the polling pattern to discover when the process is complete.

Conclusion

Both the patterns described should be fairly easy to implement, and solve the required problem. The progress polling pattern is arguably superior, because it completely hides the server implementation from the Form.

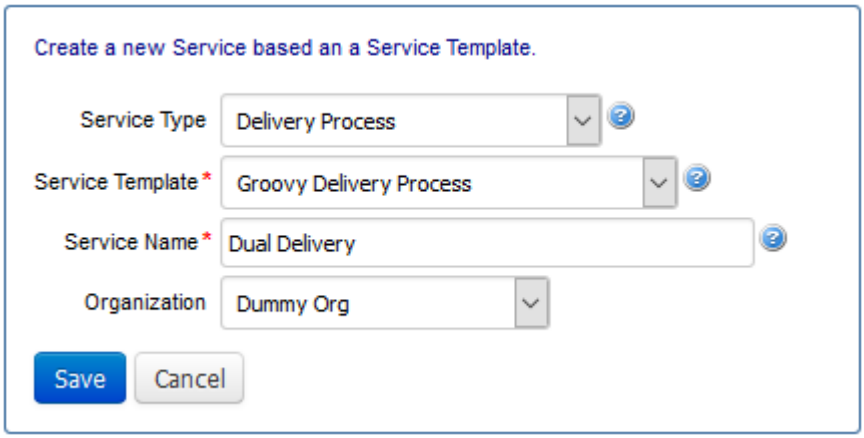
Next, learn how to view Transact functions.

Multi-Step Delivery Process

Manager already provides the capacity for multi-step delivery processes through the use of delivery checkpoints in the Groovy Delivery Process. However, for larger or more complicated delivery methods, it may be easier to separate delivery steps into individual delivery processes. This has the added advantage of allowing forms to call only the delivery method they need. For forms that require these multi-step delivery methods, it is possible to configure a Groovy delivery process to call other delivery processes. This allows form developers to create a type of landing point for form submissions that can branch off to any number of different delivery methods, providing a wide range of functionality for accepting form submissions.

Basic Groovy Delivery Process

First, lets make a basic Groovy Delivery Process. Log in to Manager, go to the service definitions page and click new and fill out the new service information.



The basic groovy delivery process will have the following code provided for you:

```
if (deliveryCheckpoint.doCheckpoint("Checkpoint A", "Description...")) {
        try {
                // TODO: provide your custom delivery code
                deliveryCheckpoint.compietedCheckpoint("Checkpoint A")
        } catch (Throwable e) {
                deliveryCheckpoint.errorCheckpoint("Checkpoint A", e)
        }
}
if (deliveryCheckpoint.doCheckpoint("Checkpoint B", "Description...")) {
        try {
                // TODO: provide your custom delivery code
                deliveryCheckpoint.compietedCheckpoint("Checkpoint B")
        } catch (Throwable e) {
                deliveryCheckpoint.errorCheckpoint("Checkpoint B”, e)
        }
}
```

When this process is called, after a form submission is made, it will first run the code in the try/catch block labled Checkpoint A and then the code in the next try/catch block labeled Checkpoint B. If the code in the try catch block should fail it will retry this code again in 15 mins.

The delivery retry delay and the number of retry attempts can be adjusted in the service parameters of the delivery process, the fields are retryDelayMins and maxDeliveryAttempts.

In this code even if Checkpoint A should fail Checkpoint B will still make an attempt to be delivered, this is known as a parrallel delivery method. You can also setup the delivery process to be serial, this is useful if you have delivery processes that rely on each other. The groovy code for a serial delivery looks like this:

```
if (deliveryCheckpoint.doCheckpoint("Checkpoint A", "Description...")) {
       try {
               // TODO: provide your custom delivery code
               deliveryCheckpoint.compietedCheckpoint("Checkpoint A")
       } catch (Throwable e) {
               deliveryCheckpoint.errorCheckpoint("Checkpoint A", e)
       }
}
if (deliveryCheckpoint.isCompleted("Checkpoint A") &&
    deliveryCheckpoint.doCheckpoint("Checkpoint B", "Description...")) {
       try {
               // TODO: provide your custom delivery code
               deliveryCheckpoint.compietedCheckpoint("Checkpoint B")
       } catch (Throwable e) {
               deliveryCheckpoint.errorCheckpoint("Checkpoint B”, e)
       }
}
```

This code won't call the second checkpoint until the first has been successfully completed. For more information, see Delivery Function.

Dual Delivery

It is possible,using groovy script, to call another process from within your delivery process, as shown below:

```
IDeliveryProcessService deliveryService = (IDeliveryProcessService) Ser-
viceLocator.getServiceForName('[SERVICE_NAME]'))
deliveryService.deliverSubmission(submission, new DeliveryDetails())
```

These two lines of code first locate the service by its name, then calls the service by passing the submission object and delivery details to the service. Using this code in a groovy delivery service it is possible to create a landing point for submissions that can be delivered in a variety of methods based on any number of factors, such as submission data or other successful delivery methods.

Let's look at an example below.

A form is submitted that calls a basic delivery process 'logger', which logs the form submission data for review by a backend system. The form is updated to include an email delivery function

only if a certain email is input into the form. To get this functionality to work a delivery process is created to make the initial 'logger' delivery and also check for the email and complete that delivery process if it is found. The snippet of the code is shown below.

Dual Delivery will look like this:

```
import com.avoka.fc.core.entity.DeliveryDetails
import com.avoka.fc.core.entity.Submission
import com.avoka.fc.core.dao.SubmissionDataDao
import com.avoka.fc.core.service.IDeliveryProcessService
import com.avoka.fc.core.service.ServiceLocator
import com.avoka.fc.core.service.submission.DeliveryCheckpointService
import com.avoka.fc.forms.api.DeliveryException

import java.util.List

if (deliveryCheckpoint.doCheckpoint("Checkpoint A", "Description...")) {
  try {
        // Call first delivery process
        IDeliveryProcessService deliveryService = (IDeliveryProcessService) Ser-
viceLocator.getServiceForName('Logger Delivery')
        deliveryService.deliverSubmission(submission, new DeliveryDetails())
        deliveryCheckpoint.completedCheckpoint("Checkpoint A")
   } catch (Throwable e) {
         deliveryCheckpoint.errorCheckpoint("Checkpoint A", e)
   }
}
if (deliveryCheckpoint.doCheckpoint("Checkpoint B", "Description...")) {
  try {
    def list = submission.getSubmissionExtractData();
    //Iterate through Submission Data
    for(def temp in list){
      //Check for Email Field with specifc Email value
      if(temp.getName() == "email" && temp.getValue() == "BackEnd@domain.com"){
              //Call Second Delivery process
              IDeliveryProcessService deliveryService = (IDeliveryProcessService) Ser-
viceLocator.getServiceForName('Email Delivery')
              deliveryService.deliverSubmission(submission, new DeliveryDetails())
        }
   }
    deliveryCheckpoint.completedCheckpoint("Checkpoint B"
   } catch (Throwable e) {
       deliveryCheckpoint.errorCheckpoint("Checkpoint B", e)
  }
}
```

Checkpoints

Delivering transactions is hard work, and often involves more than one step. Let's say our delivery job entails zipping up the transaction's bits (form receipts, XML data, attachments, and whatnot) into a zip file then sending that file to the bank by secure FTP. So that's two things: the zipping and the sending. Let's worry about what happens when the zipping goes well, but the sending not so much. Delivery processes are clever enough to retry themselves when they fail, but we don't want to do the zipping all over again (assuming we've saved the zipped file somewhere from the previous attempt). We just want to retry the sending bit. To handle that, we can use checkpoints.

Below is the code from a delivery service, which uses checkpoints. Instead of actually zipping and sending, I have decided to just flip a coin at each step to determine whether to fail or succeed. This allowed me to explore what happens with my transaction and my job when the delivery retries.

```
{
import com.avoka.tm.svc.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import groovy.transform.TypeChecked

@TypeChecked
class FluentDeliveryProcess {

    // Injected at runtime
    public Logger logger


    DeliveryResult invoke(SvcDef svcDef, Txn txn) {

        TxnCheckpointSvc checkpointSvc = new TxnCheckpointSvc(txn)

        Random random = new Random()

        Integer maxDeliveryAttempts = txn.getDeliveryMaxAttempts() ?: 10
        Integer previousDeliveryAttempts = txn.getDeliveryProcessAttempts() ?: 0
        Integer currentDeliveryAttempt = previousDeliveryAttempts + 1

        logger.info("this is attempt $currentDeliveryAttempt of $maxDeliveryAttempts")
        Integer attemptsRemaining = maxDeliveryAttempts - currentDeliveryAttempt
        boolean isLastAttempt = attemptsRemaining < 1

        if (checkpointSvc.doCheckpoint("Checkpoint A")) {
            try {
                logger.info("Trying the code before checkpoint A")
                def nextBoolean = random.nextBoolean()
                logger.info("nextBoolean: " + nextBoolean)
                if (nextBoolean) {                    logger.info("completing checkpoint
A")
                    checkpointSvc.complete("Checkpoint A")
                }
                else {
                    logger.info("Error before completing checkpoint A")
                    throw new RuntimeException("Error before completing checkpoint A")
                }
```

```
            } catch (Throwable e) {
                checkpointSvc.error("Checkpoint A", e.toString())
                return new DeliveryResultBuilder()
                    .setStatus(isLastAttempt ? Txn.DELIVERY_UNDELIVERABLE : Txn.DELIVERY_
ERROR)
                    .build()
            }
        }

        if (checkpointSvc.isCompleted("Checkpoint A") && checkpointSvc.doCheckpoint
("Checkpoint B")) {
            try {
                logger.info("Trying the code before checkpoint B")
                def nextBoolean = random.nextBoolean()
                logger.info("nextBoolean: " + nextBoolean)
                if (nextBoolean) {
                    logger.info("completing checkpoint B")
                    checkpointSvc.complete("Checkpoint B")
                }
                else {
                    logger.info("Error before completing checkpoint B")
                    throw new RuntimeException("Error before completing checkpoint B")
                }

            } catch (Throwable e) {
                checkpointSvc.error("Checkpoint B", e.toString())
                return new DeliveryResultBuilder()
                    .setStatus(isLastAttempt ? Txn.DELIVERY_UNDELIVERABLE : Txn.DELIVERY_
ERROR)
                    .build()
            }
        }

        return new DeliveryResultBuilder()
                .setStatus(Txn.DELIVERY_COMPLETED)
                .build()
        }
}
```

You can see that delivery service tracks when we've exhausted the configured number of retries and quits delivering by setting the delivery status to UNDELIVERABLE. Manager has a built in user group called Receive System Alerts – Delivery. When a transaction fails to deliver and

there are no remaining attempts, Journey Manager sends an email alert to all members of this group for the organization. In this way, the attention of a system administrator can be brought to the problem.

In some cases, the administrator can fix the problem and retry delivery. For more information, see Receive Delivery Escalation Alerts.

Next, learn how to view Transact functions.

Restore Undeliverable Transactions

Sometimes things don't go as planned, as illustrated with the following scenario. Earlier this week you deployed the form change that was requested, it all tested ok and you were quite happy that it all went smoothly. Until two days later, when you find out that your form change caused a downstream system to fail and all submissions for these two days can't be processed anymore. Quickly restoring the previous form version will at least stop the bleeding, but what to do with the submissions that are "stuck"? You know how to fix them up, a change to the delivery process would be simple. Unfortunately, this change would also impact the current submissions, and a change would require regression testing and all kinds of approvals.

To restore transactions that can't be delivered:

1. The easiest solution at this point is to create a new delivery channel within the same organisation, let's call this one "fixer upper delivery channel".
2. For this new delivery channel you can write the Groovy code that's necessary to "massage" the data for the stuck transactions so that the downstream process can handle it.
3. We assume that you have a list of tracking codes of the stuck transactions.
4. From the Groovy console you can run the following code that will change the delivery channel on these transactions and make them ready for delivery:

```
import com.avoka.fc.core.dao.SubmissionDac
import com.avoka.fc.core.dao.DeliveryDetailsDac
import com.avoka.fc.core.dao.ClientDac
import com.avoka.fc.core.service.DeliveryResult.Status
import com.avoka.fc.core.service.ServiceFactory
import com.avoka.fc.core.entity.Submission

// create your collection of tracking codes
String[] trackingCodes = ["TC-1", "TC-2", ...]

// Find the client and then create a new DeliveryDetails object for the delivery
channel for that client
client = new ClientDao().getClientByCode("Your Organisation code");
deliveryDetails = new DeliveryDetailsDao().getDeliveryDetailForClientAndName(cli-
ent, "fixer upper delivery channel")

// we need to get the submissions using the SubmissionDa
sDao = new SubmissionDao()

for (String trackCode in trackingCodes) {
    submission =  sDao.getSubmissionByTrackingNumber(trackCode)
    if (submission) {
        println submission.formName + " found";
        submission.setDeliveryDetails(deliveryDetails)
        submission.setDeliveryStatus(Submission.STATUS_Ready)

            println submission.trackingNumber  + " changed to new delivery process"
    }
}
ServiceFactory.getSubmissionService().commitChanges()
```

Next, learn how to view Transact functions.

Reload Form Submission Data Extracts from the Groovy Console

If a client requests new submission data extracts after a number of initial submissions have already taken place, you can add these new data extracts to the form data configuration and then reload the data extracts for every past submission of the form using the script below in the groovy script console. However, this only works on existing XPath locations, which are existing fields in the form.

```
import com.avoka.fc.core.dao.SchemaSeedDac
import com.avoka.fc.core.dao.FormDac
import com.avoka.fc.core.service.SubmissionService
import com.avoka.fc.core.service.ServiceFactory
import java.util.List;
import com.avoka.fc.core.dao.SubmissionDataDac
import com.avoka.core.util.XmlUtils
import org.apache.cayenne.BaseContext;
/*
 Reloads data extracts for said form
*/
//Set the form code to the form you want to update the extracts.
def formCode = "PUT-FORM-CODE-HERE"

def submissionDataDao = new SubmissionDataDao();

def formDao = new FormDao();

def schemaDao = new SchemaSeedDao()

        def submissionService = new SubmissionService();

def form = formDao.getFormByFormCode(formCode)

def curVersion = schemaDao.getSchemaForVersion(form.getCurrentVersion())

def submissions = form.getSubmissions()

/*
 Loop submissions and update extracts
*/

if(submissions) {
          submissions.each(){
            def submissionXml = submissionDataDao.getSubmissionXmlString(it)
            // only if we have a submission XML string value
            if(submissionXml?.trim()) {
              def document = XmlUtils.parseDocumentFromString(submissionXml)
              submissionService.populateSubmissionExtractData(document, it, curVersion)
            } else {
              println "Skipping blank submission XML for submission
```

```
trackingCode:${it.getTrackingCode()}"

    }
          }

}

BaseContext.getThreadObjectContext().commitChanges();
```

Next, learn more about the server-side persistence pattern.

Server-Side Persistence Pattern

The main persistence model for Manager is the XML file. This is stored on the server, sent to the browser when a form is rendered, and sent back to the server when the form is submitted. It is also saved to the server in background save events.

However, there are cases when it may be useful to store information on the server which is not stored in the XML file. Some of these cases include:

- Storing results from a backend server call, such as an actual credit score. We want to record the actual score, but all we want to send to the form is a pass/fail. We don't want it to be in the XML because we don't want the end-customer to see their actual score, or the reasons why they failed.
- We want to temporarily store in-progress data that has been used in back-end service calls. This is to ensure that the end-user doesn't "game" the system, by changing the data in the form at different points in the journey.
- We want to store the results of a back-end service call for auditing or debugging purposes - but it's too much data to inject into the XML.
- There are probably many other cases.
- This data could be as simple as a string, or more complex like some JSON or XML.

There is a simple way to store and retrieve arbitrary string data against a transaction using Fluent service calls. Let's look at these in more detail.

Storing Data

You can use the `TxnUpdater.setProperty(String name, String value)` method to create or update a property value. This is illustrated below:

```
new TxnUpdater(txn)
    .setProperty("MyProp", "MyValue")
    .update()
```

Retrieving Data

You can use the `TxnQuery` class to search for a particular transaction, and then get the Property Map within the Txn value object to retrieve the particular property. This is illustrated below:

```
Txn txn = new TxnQuery()
    .setSubmitKey(…) // or similar to find the correct transactior
    .withPropertyMap() // to retrieve the properties
    .firstValue() //to find the first matching transactior
    // The returned txn contains an attribute propertyMap of type Map<String,String>
```

This is important to note that:

- Data stored in the Transaction Properties are encrypted and subject to data retention policies, just like XML. (In recent versions of Manager.)
- In some cases, it may be necessary to store data against a collaboration job rather than an individual transaction.
- The PropertyQuery service is used for locating configuration properties for Form Versions, Workspaces, etc. This is for static configuration properties, not for per-transaction or runtime properties.
- It is also possible to use the standard HTTP session to store temporary data. However, this data will not be persisted beyond an individual session. A transaction property is persisted in the database.

Next, learn how to view Transact functions.

Tasks Overview

Journey Manager allows you to create tasks based on forms and assign them to users or groups. The task can have prefill data associated with it, and completes when one of the assignees completes the task form.

In Manager, a task is a combination of:

- A form or form-code.
- Some XML data, used to pre-populate the form.
- The ID of a user or group that the task is assigned to.

Internal tasks are also created when a user has not completed a transaction, for example, when they have saved but not submitted a form, submitted but not paid or added attachments, and so on.

A task consists of a form that needs to be filled in by a user. It may contain some prefill data for form customization, such as reference numbers, a task subject, and a task message that is used to identify the task to the user or other users, for example, a reviewer. When a task is created, Manager generates a submission object that is marked as a task and contains all task's related data. As the user opens and completes the task, the submission is updated as per its normal life cycle.

- Form - a basic task that can be created using either form XML or prefill XML data.
- Review - a task associated with a previous submission, which is often prefilled with the submitted data for review purposes. It doesn't accept prefill XML data.

Tasks can be assigned to either a single user or a group of users. The task status indicates whether the task is assigned, completed or has expired. Task expiration is controlled via an optional expiry date. When a task expires, it can invoke a Task Expiry Process service to notify an external system about the task expiry.

You can abandon a task if it's not required any longer or can't be performed.

As processing of tasks are based on collaboration jobs, we recommend you getting familiar with Collaboration Jobs, especially Action Properties.

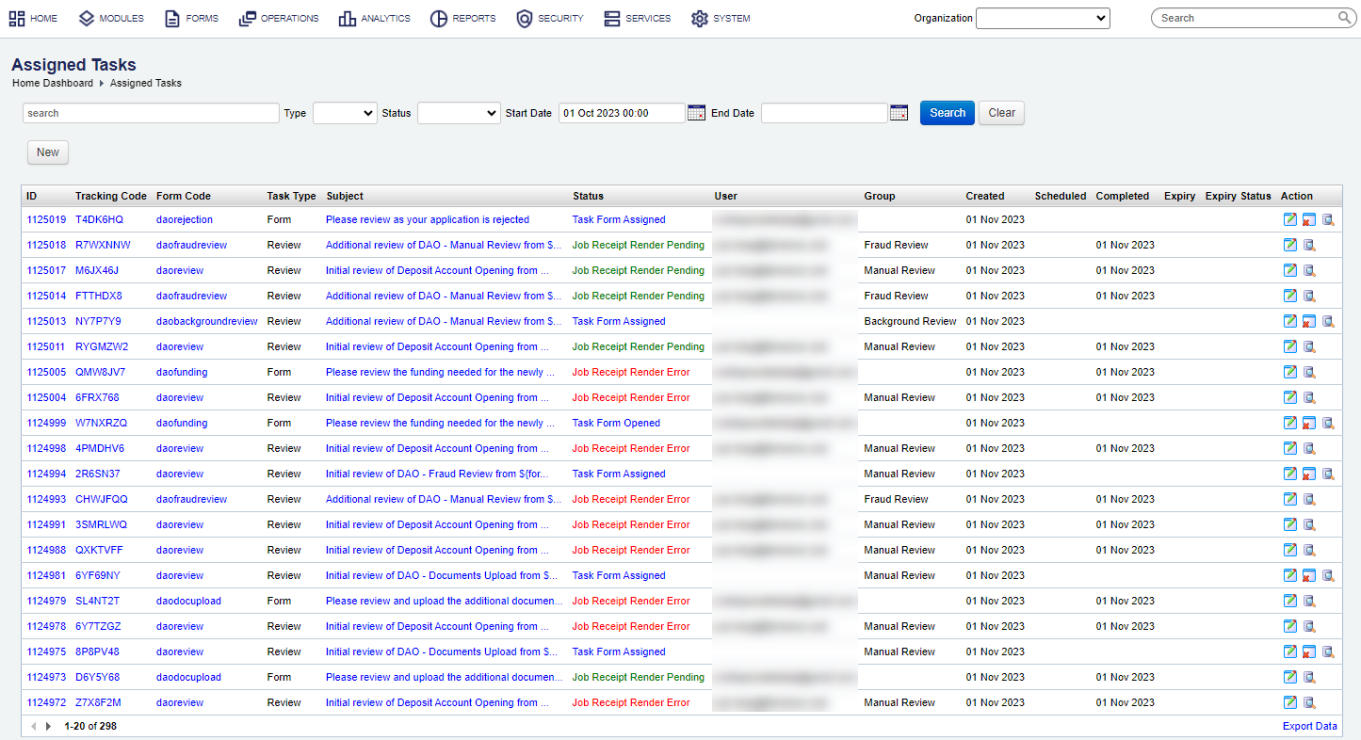
Next, learn how to view all tasks.

View Tasks

Journey Manager allows you to create and view tasks, including an option to search for associated transactions which helps you to identify and manage tasks.

To view tasks, select Operations > Assigned Tasks.

Tasks are shown page by page, with the maximum number of records per page configured via the user's preferences.



The list displays the following details:

- ID: a transaction ID of the submission generated by a task
- Tracking Code: a transaction tracking code
- Form Code: a form code of a form this task was assigned for
- Task Type: a task type, which can be one of the following:
  - Form - a basic task that can be created using either form XML or prefill XML data.
  - Review - a task associated with a previous submission, which is often prefilled with the submitted data for review purposes. It doesn't accept prefill XML data.
- Subject: a task subject describing a scope of the task o a user who will be doing it
- Status: a transaction status, which can be one of the following:
  - Abandoned
  - Assigned

    - Completed
    - Expired
    - Opened
    - Saved
    - Submitted
- User: a user name or user email address whom this task is assigned to. Leave it blank to assign it to a group.
- Group: a group this task is assigned to
- Created: a Task's creation date
- Scheduled: a Task's scheduled date
- Completed: a Task's completion date
- Expiry: a Task's expiry date
- Expiry Status: a Task's expiry status

To filter or search the tasks by one or more criteria, specify the following settings and click Search:

- A transaction ID, a tracking code, a task subject name or part of its name.
- Task type:
    - Form - a basic task that can be created using either form XML or prefill XML data.
    - Review - a task associated with a previous submission, which is often prefilled with the submitted data for review purposes. It doesn't accept prefill XML data.
- Status of a task:
    - Abandoned
    - Assigned
    - Completed
    - Expired
    - Opened
    - Saved
    - Submitted
- A time interval from Start Date to End Date when a task was created.

Click Clear to reset the search criteria.

You can also perform the following actions:

- Click New to create a task.
- Click Edit to update a task.
- Click Abandon Task to abandon a task.
- Click View Transactions Details to check transaction details of a task.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

NOTE

You can also find the task's transaction information using form transactions or transaction support search functionality.

Next, learn how to create a task.

Create a Task

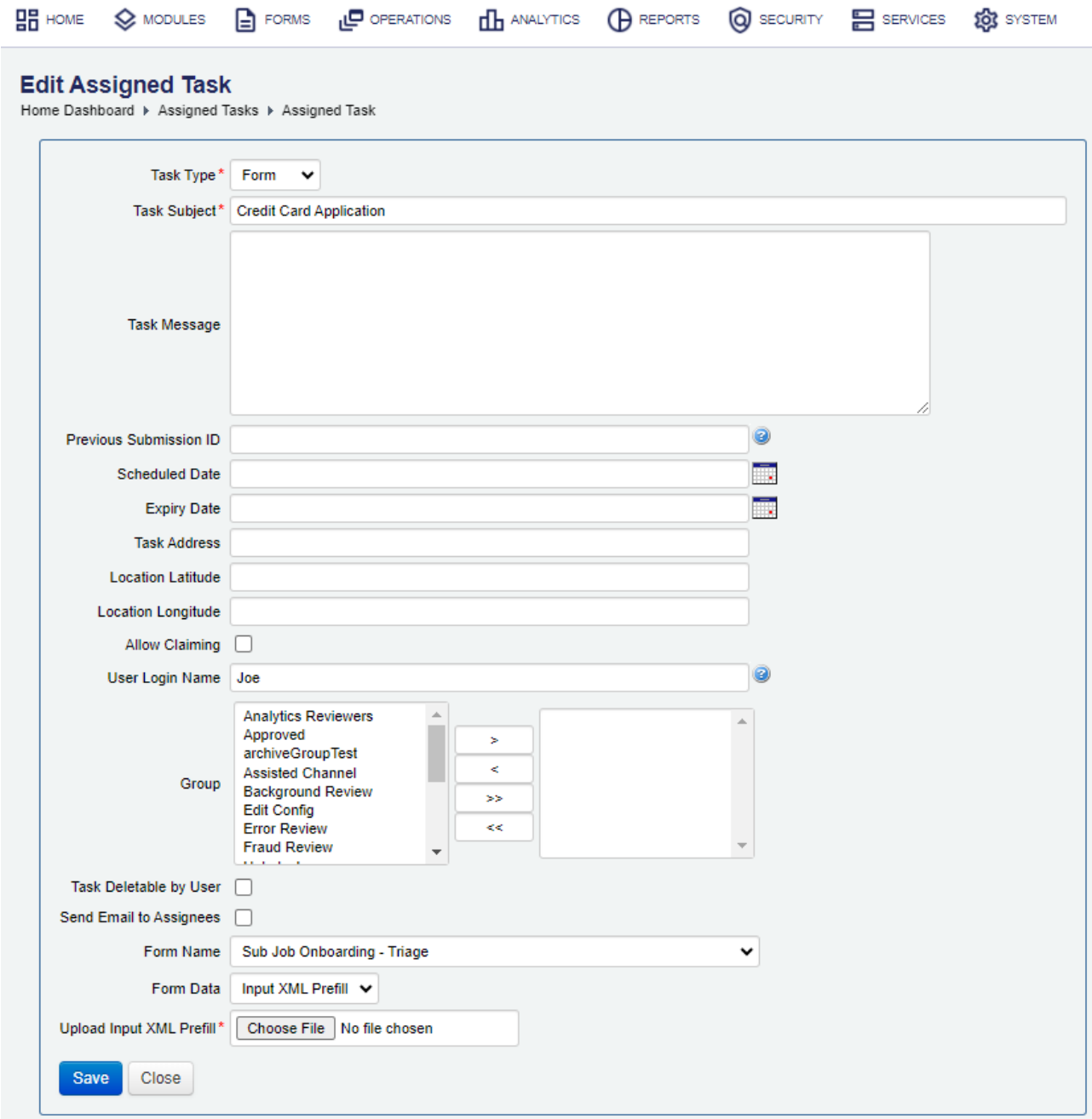
Manager allows you to create a task for a form to be filled or reviewed by a user, which can be an authenticated or anonymous user, or a group.

If a form, you are creating a task for, is a global form, you must choose a form space, which will be used to obtain a correct organization. | 23.10 This feature was introduced in 23.10

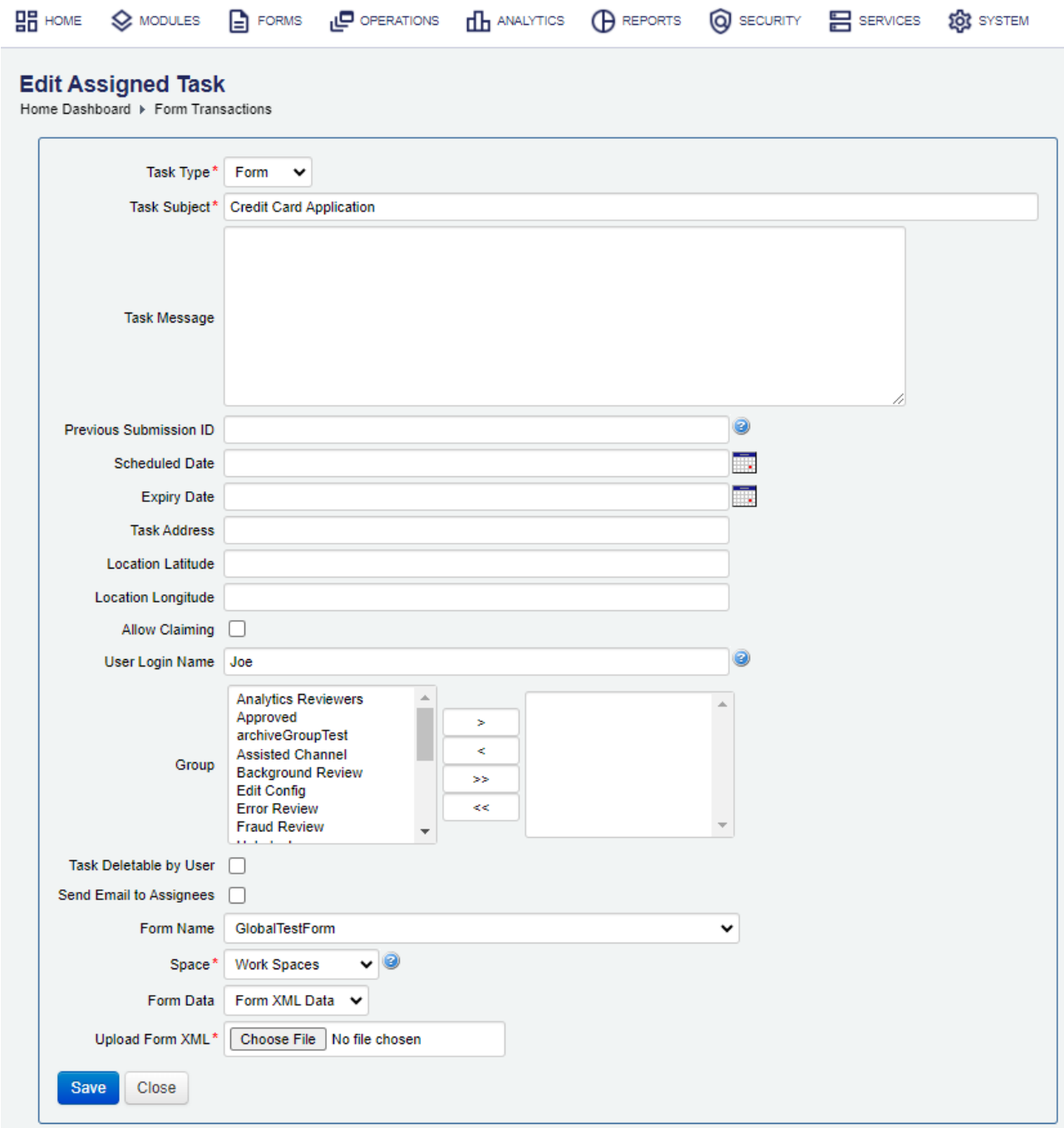
Manager uses the Job Task Assign service to create Form and Review tasks, so you can use Fluent API to configure it.

To create a new task:

1. Select Operations > Assigned Tasks.
2. Click New.



Create a task related to a standard form.



Create a task related to a global form.

3. Select a task type from the Task Type dropdown list. A task type is very useful to determine the type of the transaction and to distinguish between submissions and review tasks.
   - Form - a basic task that can be created using either form XML or prefill XML data.
   - Review - a task associated with a previous submission, which is often prefilled with the submitted data for review purposes. It doesn't accept prefill XML data.
4. Provide a brief description of a task in the Task Subject field.

5. Enter a task message that describes the task to a user who will perform it, for example, reference numbers. This information is shown to assignees of the workspace. You can use variables that will be dynamically substituted with configurable strings in runtime, for example, `Please review the Job Example 1 by Joe Public. $func.startSubmission().id()`
6. Enter the submission ID that the task is based on in the Previous Submission ID field. This for review tasks only.

> NOTE
>
> If the previous submission is of a global form, the Space dropdown list is shown and it's mandatory to select a form space. | 23.10 This feature was introduced in 23.10

> WARNING
>
> The experimental Global Forms feature has been temporarily disabled while we continue improving it. This feature will be disabled in releases starting from 24.10 and going back to 23.10.

7. If the assignee needs to be informed that the task should be done on a specific date, complete the Scheduled Date field.
8. Enter an expiry date in the Expiry Date field if you want the task to expire at a specific date and time. If a task is not completed before the expiry date, the transaction processing job will run in the background and change the task submission's form status to Expired. At this point, the default workspace will not allow assignees to complete the task. Expired task submissions can't be progressed any further.
9. Provide the physical address of a location in the Task Address field.
10. Enter latitude and longitude information in the Location Latitude and Location Longitude fields, if available. The latitude value needs to be in a range of `-90` to `90` and the longitude value between `-180` and `180`. This information will be displayed to the assignee. The default form spaces include Google Map support.
11. Select the Allow Claiming checkbox to put the task into a special claiming mode, where users have to claim the task before being able to work on it. In addition to being claimed, such tasks can be reassigned to other users by the claiming user or other space users with special privileges.
12. Provide a user's login name in the User Login Name field to assign the task to a specific user. Leave it blank if the task is assigned to a group, see below.

13. Select a user group name from the Group dropdown list so any member of the group is able to view and complete the task.

    NOTE

    Users joining a group will be able to view and work on all tasks existing for the group.

14. Select the Task Deletable by User checkbox to allow assignees to delete the task.

    NOTE

    Assignees will not be able to delete completed tasks to preserve the audit trail.

15. Select the Send Email to Assignees checkbox to send an email to the assignees to notify them of the new task. In this case, you can also fill in the Email Subject and Email Message fields. Otherwise, the default values are used.

    NOTE

    An email is sent to the current assignees only once. If new users join the group later, they will not receive a notification email.

16. Select a form, which needs to be filled to complete the task, from the Form Name dropdown list. The form can have payment and attachments associated with it. For review tasks, you can leave the dropdown blank, so Manager will use a form, which the original submission was based on. For more information, see the Previous Submission ID field.
17. Select a form space from the Space dropdown list.

    NOTE

    If a form you have selected from the Form Name dropdown list is a global form, the Space dropdown list is shown and it's mandatory to select a form space. | 23.10
    This feature was introduced in 23.10

WARNING

The experimental Global Forms feature has been temporarily disabled while we continue improving it. This feature will be disabled in releases starting from 24.10 and going back to 23.10.

18. Select a way to prefill the form, that will be presented to the assignees, from the Form Data dropdown list, which can be one of the following:
    - Input XML Prefill - for Form tasks to use standard XML prefill for forms, which are set up for XML prefill. Click Choose File to upload the prefill XML file in the Upload Input XML Prefill field.
    - Form XML Data - for Form tasks if you have a populated seed file structurally matching the form seed XML data. Click Choose File to upload the seed file using the Upload Form XML field.
    - Blank - for Review tasks the XML data of the previous submission is automatically used, if nothing else is specified.
19. Click Save to update the changes. A task submission will be created with the form status as Assigned, and users will be able to fill in the form and complete the task using the Workspaces associated with the form.

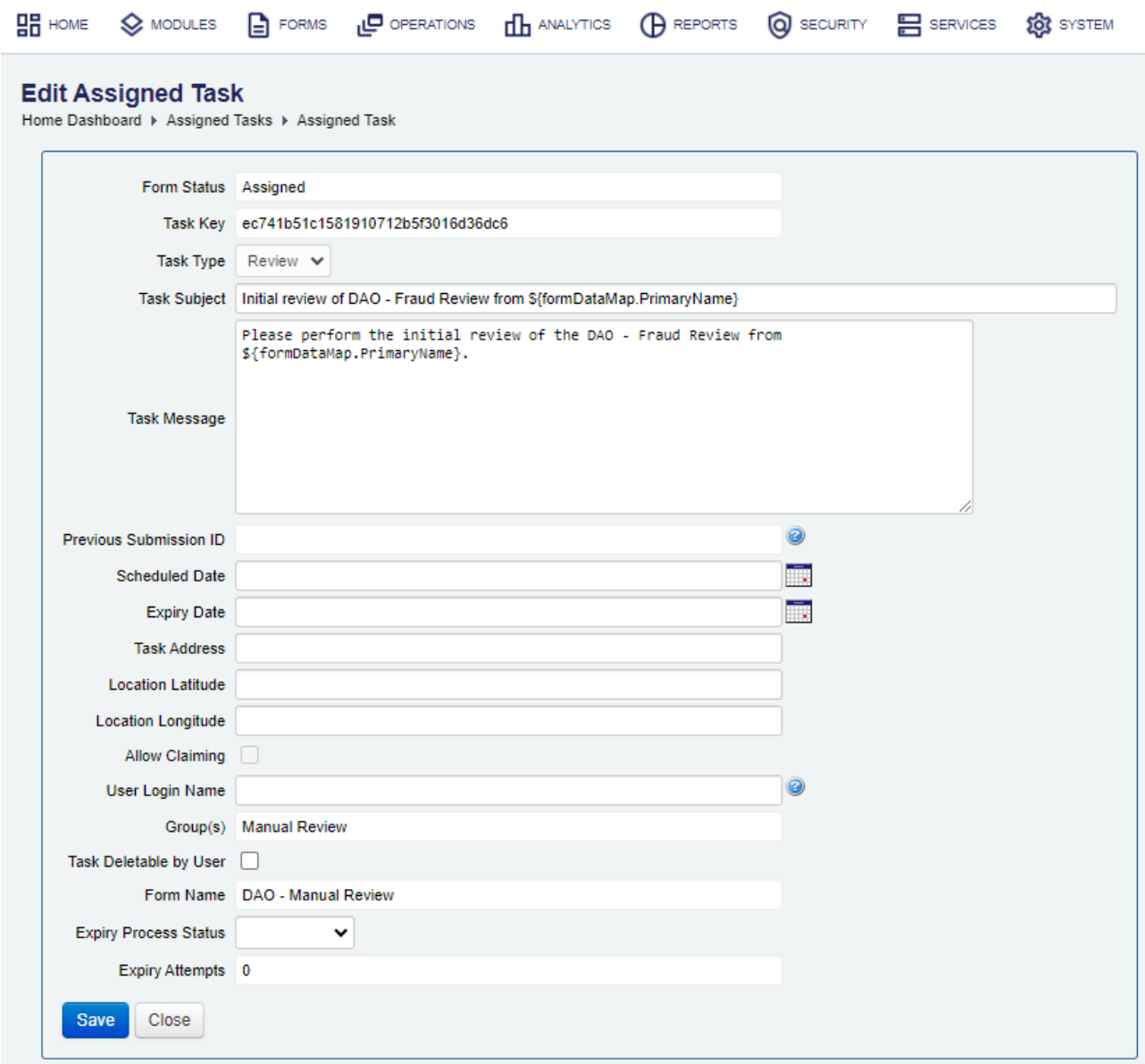
Next, learn how to view all tasks.

Configure a Task

Manager allows you to create new and configure existing tasks.

To configure a task:

1. Select Operations > Assigned Tasks.
2. Locate a task and click Edit.
3. Click Edit to view the existing configuration. For more information, see Create a Task.



4. Check the unique Task Key generated by Manager.
5. Select a status from the Expiry Process Status dropdown list, which can be one of the following:

- Required
- Error
- Completed

> NOTE
>
> This should not be necessary during normal operations because this status is updated by the expiry process service.

6. Expiry Attempts show the number of attempts taken at this task.
7. Click Save to update the changes.

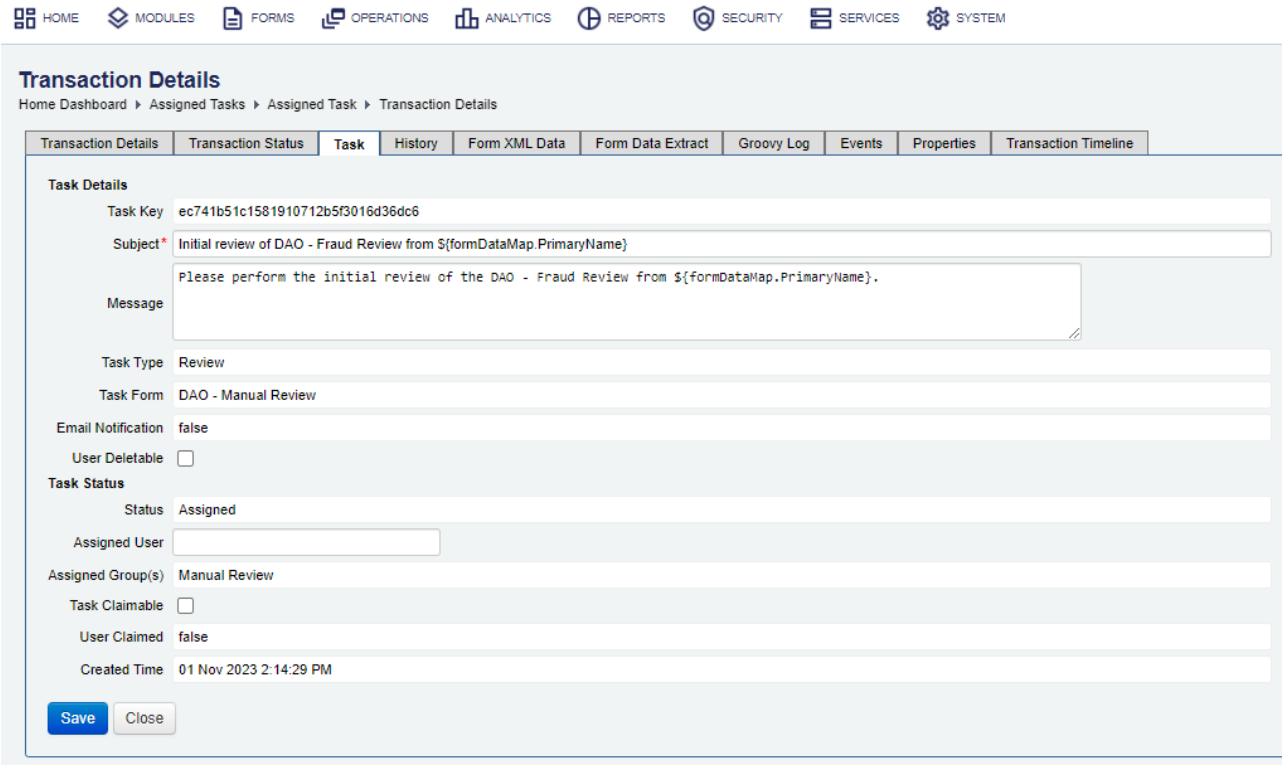
Next, learn how to view the task's transaction details.

View Task Transaction Details

Manager allows you to view transaction details of tasks. You can use this functionality to troubleshoot tasks and form submissions.

To view the task's transaction details:

1. Select Operations > Assigned Tasks.
2. Locate a task.
3. Click View Transaction Details and select the Task tab. For more information, see Create a Task.



4. Click Close.

> NOTE
>
> You can also find the task's transaction details using the transaction support search functionality.

Next, learn how to abandon a task.

Abandon a Task

Manager allows you to abandon a task when it's no longer required. Abandoned tasks will be purged.

To abandon a task:

1. Select Operations > Assigned Tasks.
2. Locate a task.
3. Click Abandon Task and confirm your decision.

Next, learn how to create a task.

Event-Based Architecture Overview

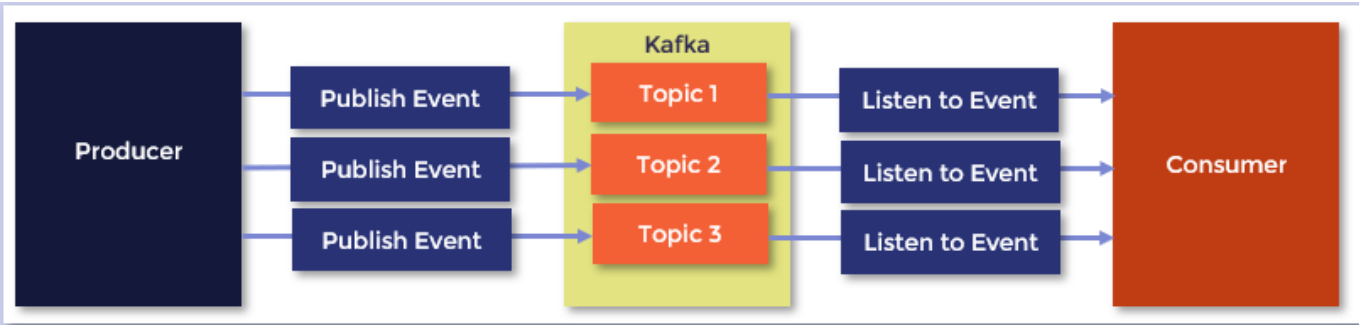
Manager comes with the Event-Based Architecture (EBA) that allows you to subscribe and publish various events to event topics run on Apache Kafka[1].

This makes it possible to decouple some Manager's core functionality so it can be externalized. For example, you may want to replace collaboration jobs workflow engine with an external product of your choice. This, in turn, improves integrations with various 3rd party systems.

We have introduced the Event-Based Architecture because it is based on the concept of events. This approach encourages you to think of events first, not the things and their states. An event is something that happens, for example, data can be an event. In Manager, most of our events are some sort of changes to data, for example, a form submission.

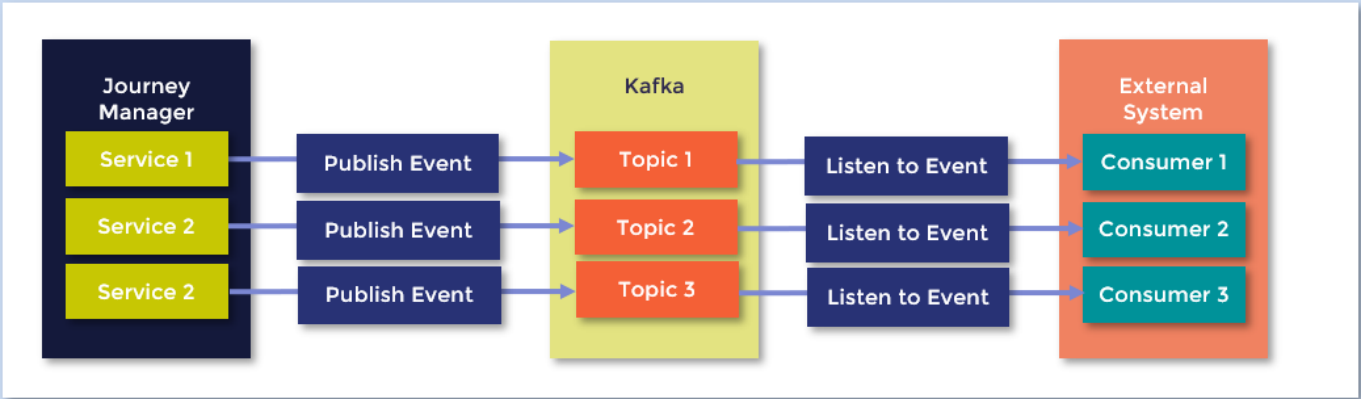
Our event-based architecture uses the Inbox and Outbox patterns. We've implemented theme using the Kafka Subscriber and the Kafka Producer, which are Kafka clients that listen to and publish events to different topics in a Kafka's cluster.

The simplified Kafka event flow is illustrated below:

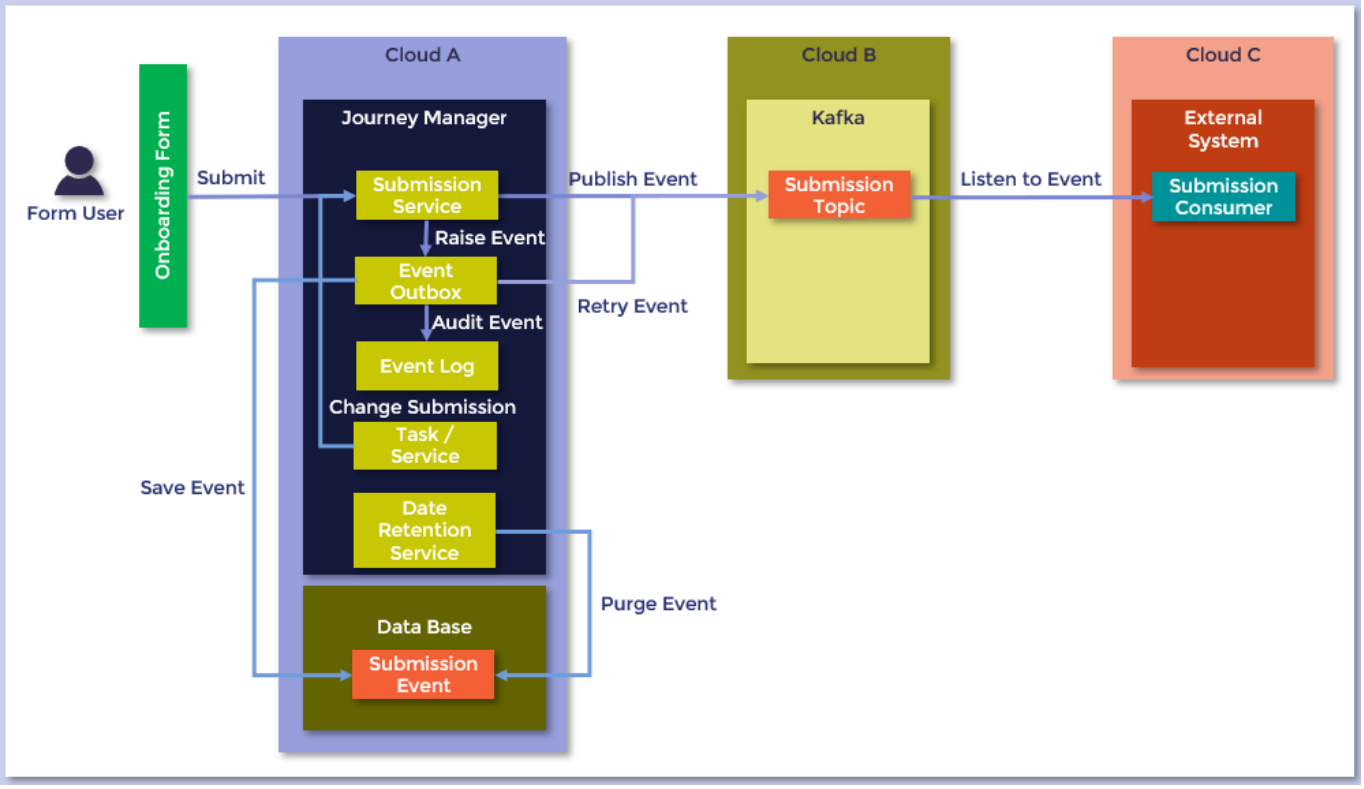


In terms of the event-based architecture, Journey Manager is a producer and any external system is a consumer of various events. This is illustrated below:

[1]Apache Kafka is a distributed event store and stream-processing platform, which provide a unified, high-throughput, low-latency platform for handling real-time data feeds. For more information, see https://kafka.apache.org/intro



The event publishing mechanism kicks in when submission data is created or modified. For example, when a form user clicks a Submit button on an Onboarding application form, the Submission Service raises an event. A similar event is raised if a service or a task updates or deletes submission data. This is illustrated in a diagram below:

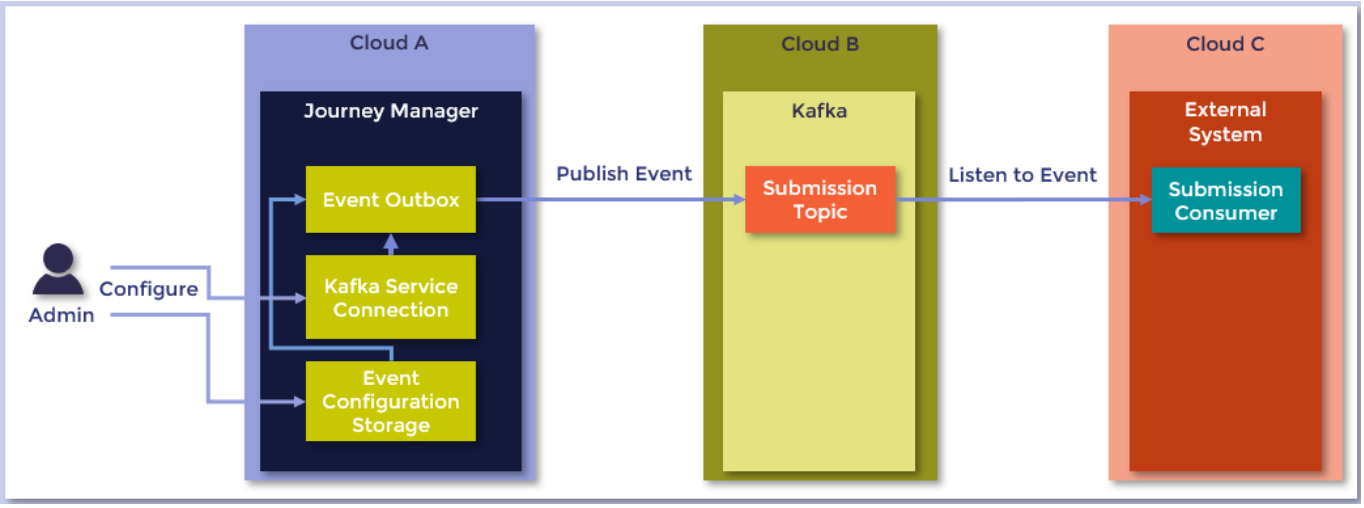


The Submission Service passes the new event to the Event Outbox component to publish it to a Kafka's topic. The Event Outbox also persists it in a local database to ensure the event will be published in case the first publishing effort fails. There are several other components involved to make sure event processing is audited and all data is purged according to your data retention management.

Let's have a detailed look at the sequence of events when a form user submits a form:

1. A customer fills in an onboarding form and submits details, as defined in the onboarding process, to Journey Manager's submission service.
2. Various events are triggered depending on the custom event configuration. Events are persisted in the database and sent the Event Outbox for publishing.
3. Auditing is also performed on the event generation.
4. Further changes to the submission may result in additional events being triggered with additional entries persisted in the database Event Outbox
5. Journey Manager publishes the event on the topic defined in the Event Configuration. If a connection error occurs, a retry will happen within a defined period of time and number of retries.
6. The event is removed from the database on successful delivery to the defined service connection, in this case it's the Kafka service connection.
7. Details of the triggered event are stored as a part of the Submission details in Event Log.
8. External systems, which have subscribed to the same Kafka topics that are defined in the Event Configuration, are notified when the trigger occurs

For the event-based architecture to work correctly, you need to configure Manager in a way that certain events are published in corresponding topics in the Kafka cluster. This is illustrated in a diagram below:



There's some configuration required to enable publish and subscribe event flows in Manager. Your Administrator or the Cloud Services team can easily perform it by following the configuration steps below. This is a common task that must be done after Manager is installed or upgraded.

Publishing Flow

To start generating events and publishing them to Kafka topics:

1. Create and configure a Kafka Service Connection.
2. Create a custom Event Configuration Storage service which uses the Kafka Service Connection.
3. Enable the Eventing Feature Enabled deployment property to activate the eventing framework, so all triggered events are stored in the Event Outbox.
4. The Event Outbox publishes the events to Kafka on a topic as configured in the Kafka Service Connection.
5. Any external system, subscribed to Kafka on the topics as defined in the Event Configuration, is notified when an event is triggered and published to Kafka by Manager.

Subscribing Flow

To subscribe to topics and start listening to events from Kafka:

1. Create a HTTP Endpoint service connection to point to your Manager REST endpoint.
2. Update the Event Inbox Processor Service with this service connection so it will be used to execute REST API calls.
3. Create a Kafka Service Connection to connect to a Kafka service with specified topics.
4. Create an Event Listener Configuration Service which uses the Kafka Service Connection to subscribe to these topics.
5. Check the service is running and listening to events.
6. The Event Inbox Processor consumes events that are added to its queue by Event Listener Configuration services. and executes REST API calls using properties defined in each topic.
7. An external system, which published this topic to Kafka, is notified when an event is consumed and expects the REST API has been made.

Next, learn about the event definition.

Event Definition

Manager comes with the Event-Based Architecture, which allows you to send and receive events that can have various definitions.

An event is defined using the JSON format, which has the following schema:

```
{
  "KAFKA_TOPIC" : {
  "entityName":"submission",
  "eventType":"INSERT",
  "entryConditions":[
    {
      "key":"formStatus",
      "previousValue":"[NULL]",
      "targetValueOperand":"E",
      "targetValues":[
        "Completed",
        "Saved",
        "Cancelled"
      ]
    },
    {
      "key":"formStatus",
      "previousValue":"In Progress",
      "targetValueOperand":"NE",
      "targetValues":[
        "Completed",
        "Saved",
        "Cancelled"
      ]
    }
  ]}
}
```

Schema Description

entityName -

entryConditions - In this example, there are the two condition blocks can only form the `AND` condition. However, in future, we could implement more complex logic.

key -

previousValue -

targetValueOperand -

targetValues -

if a target section is missing - i.e. a condition block only has:

```
{
            "key":"formStatus",
            "previousValue":"In Progress"
        }
```

This means that we're looking for a change only and do not want to worry about what the new value is, just that it has changed.

[NULL] is the null string identifier too to differentiate an empty string and null, although Oracle only has null's and doesn't recognise empty strings.

we receive an Object from Cayenne and that will already tell us what type it is from the database - all that's needed then is to instantiate a new instance and parse the user entered config String into an object for a generic compare.

Next, learn how to? .

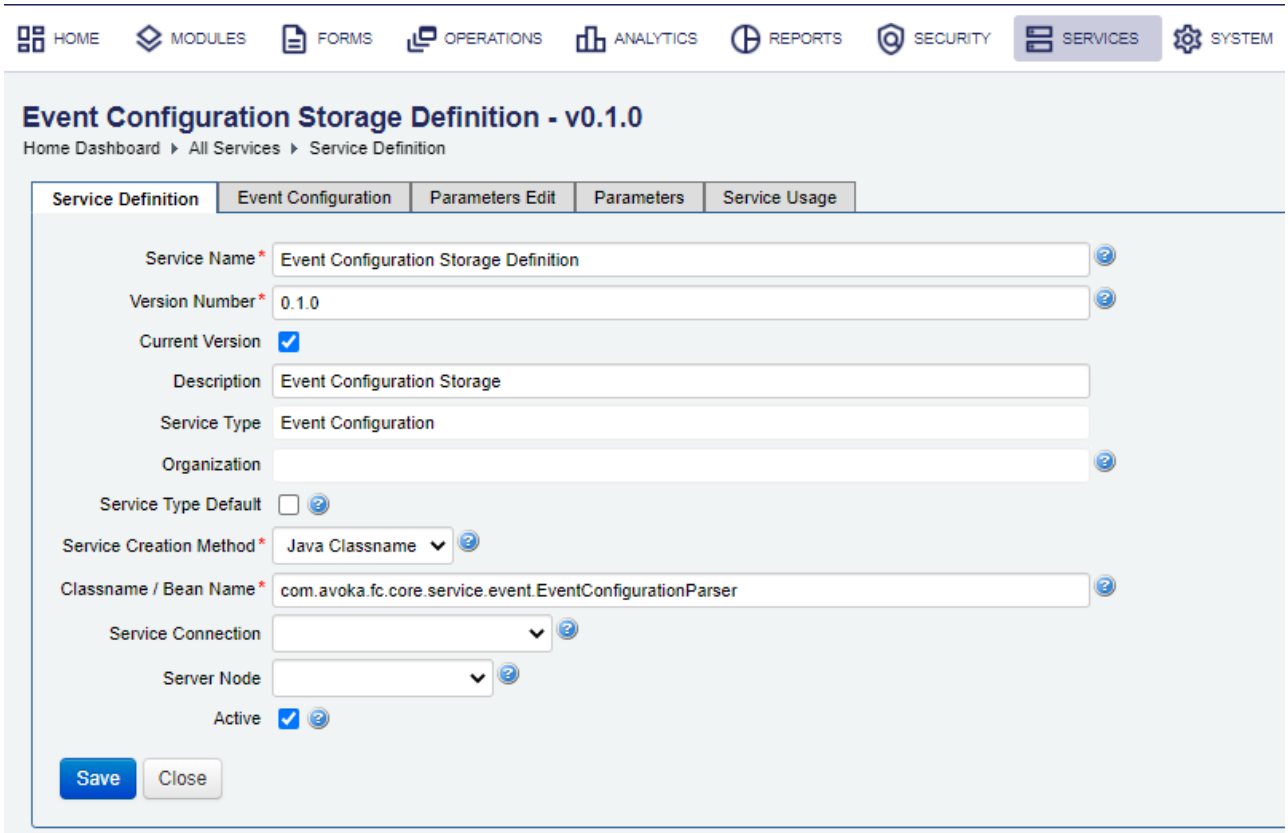
Event Configuration Storage Service

Manager comes with the Event Configuration Storage service, which allows you to publish various types of events to Kafka topics. This service is an integral part of the Event-Based Architecture.

> NOTE
>
> You may need to create and configure a Kafka service connection to use this service.

To configure the Event Configuration Storage service:

1. Select Services > Core Global Services.
2. Locate the Event Configuration Storage service and click Edit.
3. Configure the standard service settings using the Event Configuration service type as the Service Type and the Event Configuration Storage Definition template as the Service Template. Select a relevant service connection to use to connect to the external service.



4. Select the Event Configuration tab to specify filter criteria that are used to decide which Manager's entity to published to which Kafka topic.

5. Copy an paste an existing JSON configuration file or edit it using JSON syntax in the Event Configuration field. You must specify at least the following properties:
   - Topic name: a name of a topic configured in Kafka, for example, `KAFKA_TOPIC`.
   - Entity Name: a name of the Manager's entity that is published to the Kafka topic identified with the topic name, for example, `Submission`.
   - Entity Conditions: query to list entities to be published to the Kafka topic.

   For more information, see event definition.

6. Select the Auto Format checkbox to apply JSON file formatting, such as indentation.
7. Click Journey Manager Services Guide and Journey Manager Fluent API Javadoc links to access the relevant documentation.
8. Click Save to update the changes.

> NOTE
>
> You must enable the Eventing Feature Enabled deployment property to activate the eventing framework, so all triggered events are stored in the Event Outbox.

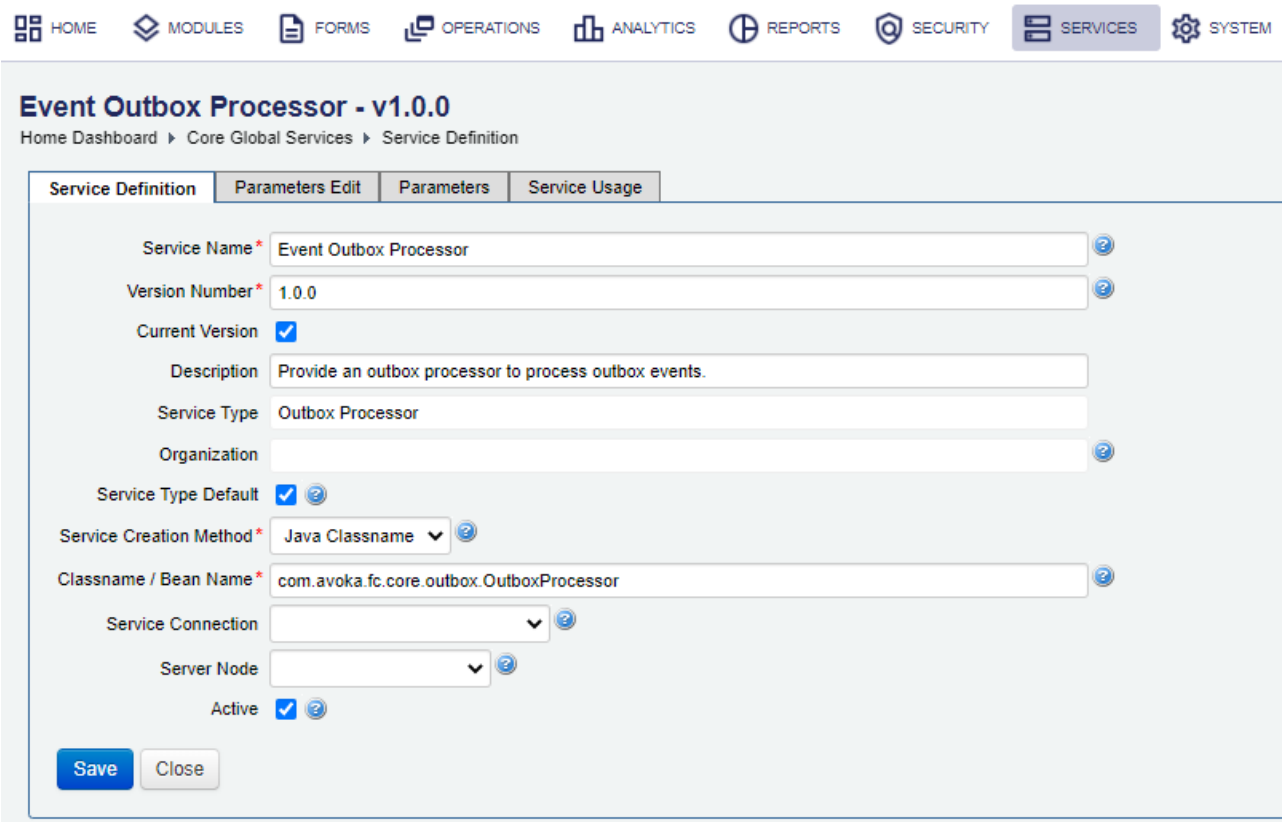
Next, learn how to view core global services.

Event Outbox Processor Service

Manager provides the Event Outbox Processor service to publish events into a Kafka topic. Normally, this service is called by the Event Outbox Processor job.

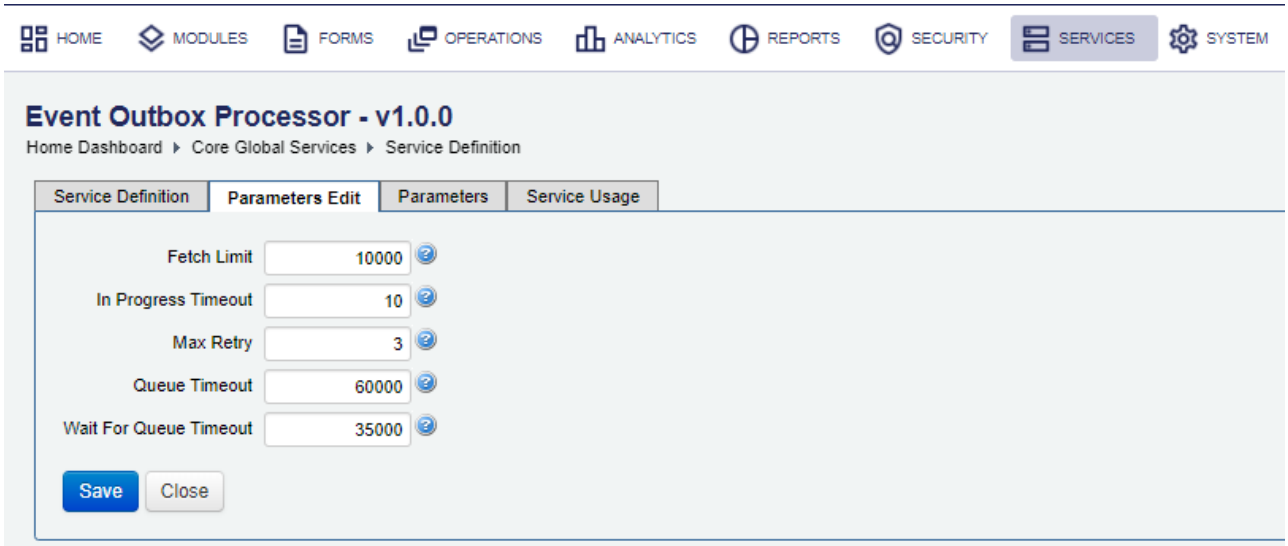
To configure the Event Outbox Processor service:

1. Select Services > Core Global Services.
2. Select the Outbox Processor from the Type dropdown list to shown only services of this type.
3. Locate the Event Outbox Processor service and click Edit.
4. Click the Service Definition tab to configure the service.



5. Click Save to update the changes.

6. Click the Parameters Edit tab to configure the service parameters.



7. Edit a maximum number of events to be processed in the Fetch Limit field. The default is 10000.
8. Specify how long to wait for an event to be processed (in milliseconds) in the In Progress Timeout field.

   NOTE

   The event's status is set to In Progress when Manager starts sending it to Kafka

9. Specify a maximum number of retries to process an event if it failed in the Max Retry field.
10. Specify how long to wait in the event queue (in milliseconds) in the Queue Timeout field.
11. Specify how long to wait to add an event to the event queue (in milliseconds) in the Wait For Queue Timeout field.
12. Click Save to update the changes.

The service adds an entry to the event log every time it runs. The entry can be one of the following types:

- Send Event/Delivered
- Send Event/Failed

The log entry example is shown below:

```
Event sent from Journey Manager, Event ID:10, Service Connection:KafkaConnection, Event
Topic:topic_1, Event Details:
{"entity":"Submission"}
```

Next, learn how to view all core global services.

Event Inbox Processor Service

Manager comes with the Event Inbox Processor service that consumes events that are added to its queue after being read by Event Listener Configuration services. The Event Inbox Processor executes REST API calls using properties defined in each topic and an endpoint configured in a service connection of the Event Listener Configuration services. The service runs in a background thread.

Each event contains the following parameters:

- `className`:
- `method`:
- `body`: a payload of the REST API call.

For more information, see REST API.

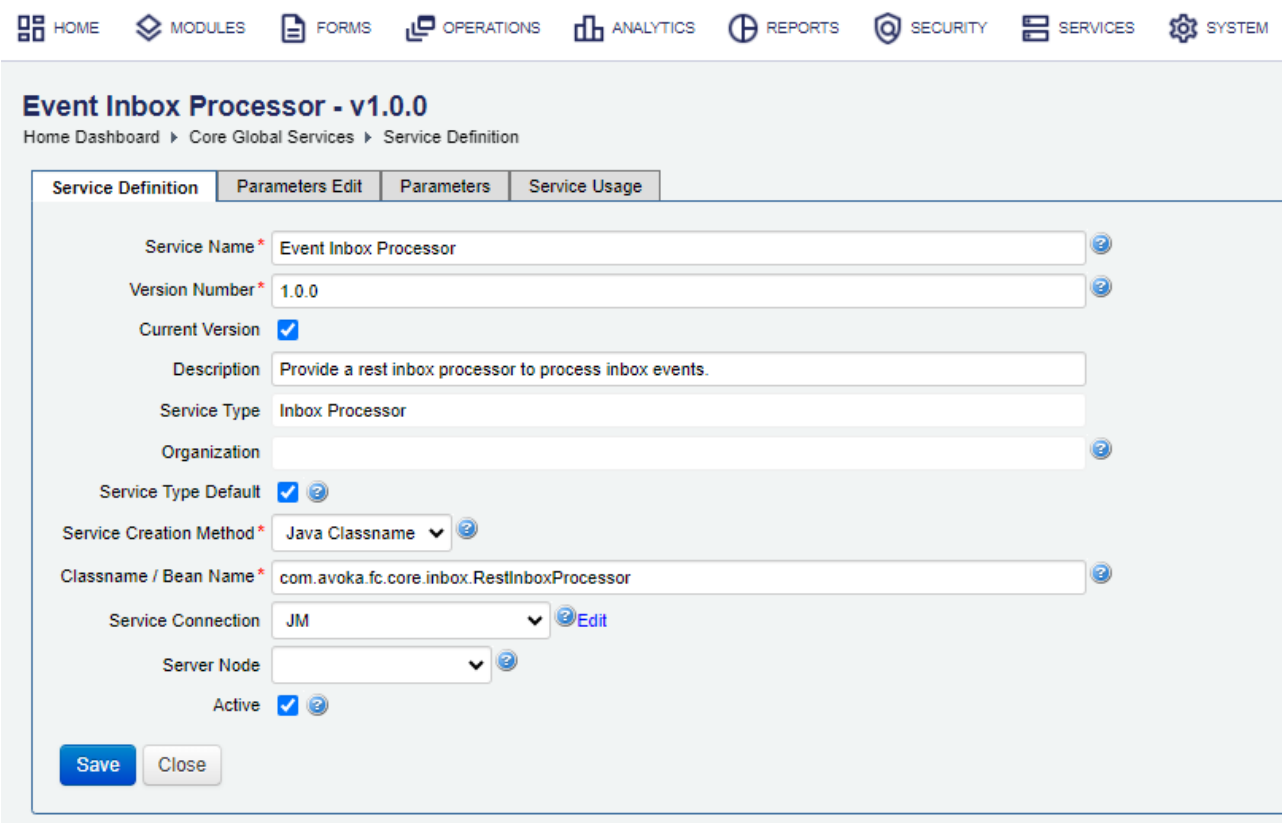
An example of an event is shown below:

```
{
"className": "txnupdater",
"method": "update",
"body": "{ \"setId\": \"2\", \"setDeliveryStatus\": \"Ready\" , \"setProperty\":
[\"named\", \"My Name\"]}"
}
```

To configure the Event Inbox Processor service:

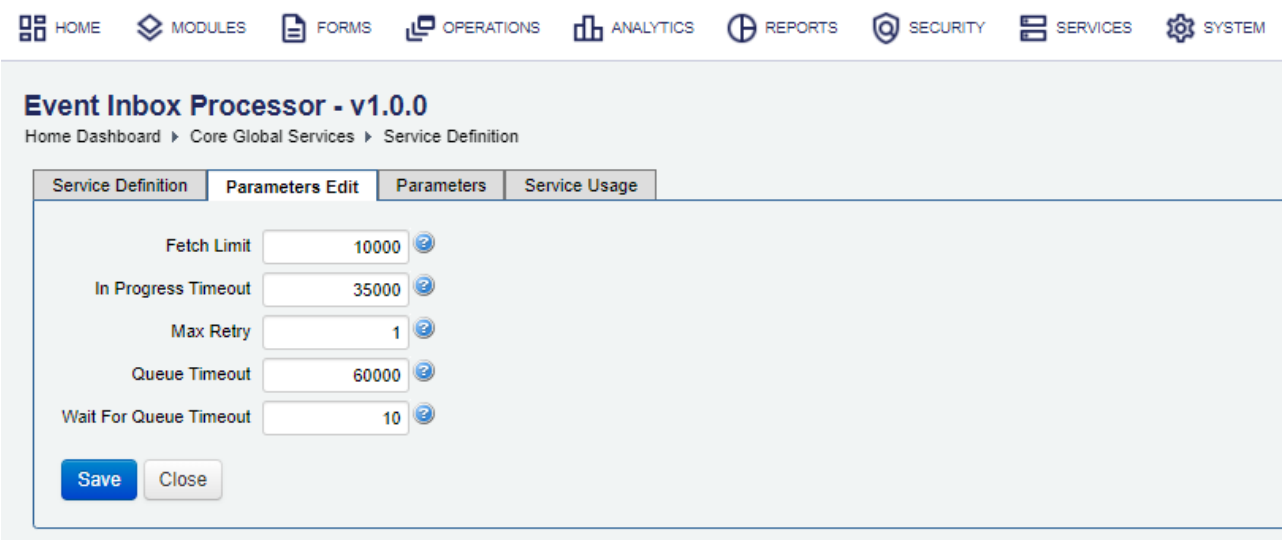
1. Select Services > Core Global Services.
2. Select the Inbox Processor from the Type dropdown list to shown only services of this type.
3. Locate the Event Inbox Processor service and click Edit.

4. Click the Service Definition tab to configure the service.



5. Select the HTTP Endpoint service connection from Service Connection dropdown list. In this example, it's called JM, but it could be any name you used to create it. If it doesn't exist, create a new HTTP Endpoint service connection using the following endpoint: `https://my-domain/manager/secure/rest/fluentapi`.
6. Click Save to update the changes.

7. Click the Parameters Edit tab to configure the service parameters.



8. Edit a maximum number of events to be processed in the Fetch Limit field. The default is 10000.
9. Specify how long to wait for an event to be processed (in milliseconds) in the In Progress Timeout field.

   NOTE

   The event's status is set to In Progress when Manager starts sending it to Kafka

10. Specify a maximum number of retries to process an event if it failed in the Max Retry field.
11. Specify how long to wait in the event queue (in milliseconds) in the Queue Timeout field.
12. Specify how long to wait to add an event to the event queue (in milliseconds) in the Wait For Queue Timeout field.
13. Click Save to update the changes.

NOTE

View transaction details to check REST API calls made by the Event Listener Configuration service.

Next, learn how to view all core global services.

Event Listener Configuration Service

Manager provides the Event Listener Configuration service, which is an active stream listener, to consume events as Kafka topics and add them to a queue. Then, the Event Inbox Processor reads the queue and executes REST API calls using properties defined in each topic and an endpoint configured in a service connection of the Event Listener Configuration.

You can have as many Event Listener Configuration services as required for your specific implementation. If there is no Event Listener Configuration service in your environment, create one using the following settings:

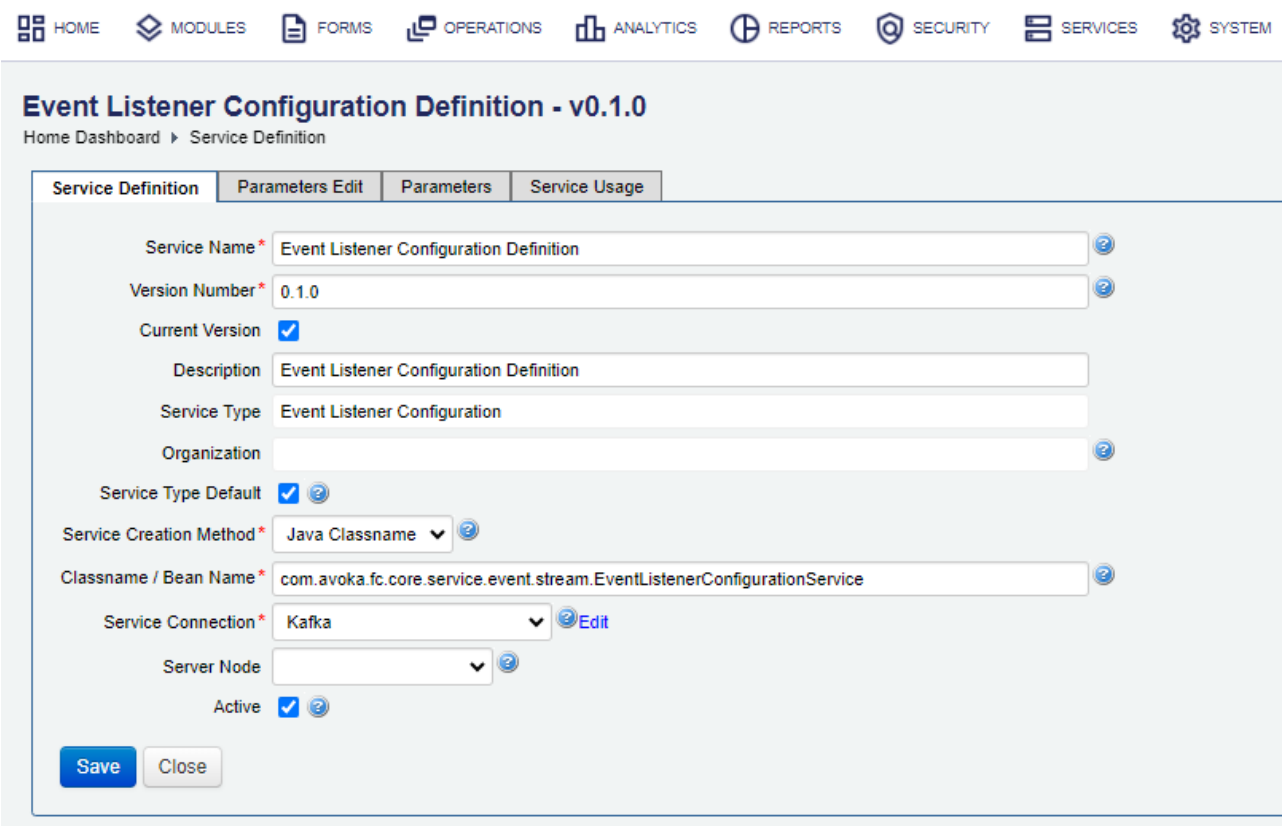
- Service Type: Event Listener Configuration
- Service Template: Event Listener Configuration Definition
- Name: a unique name
- Organization: your organization

After you have created the Event Listener Configuration service, it appears in the list of Event Listeners. It's is automatically started, so you see its status as Active.

To configure the Event Listener Configuration service:

1. Select Services > Core Global Services.
2. Select the Event Listener Configuration from the Type dropdown list to shown only services of this type.
3. Locate the Event Listener Configuration service and click Edit.

4. Click the Service Definition tab to configure the service.



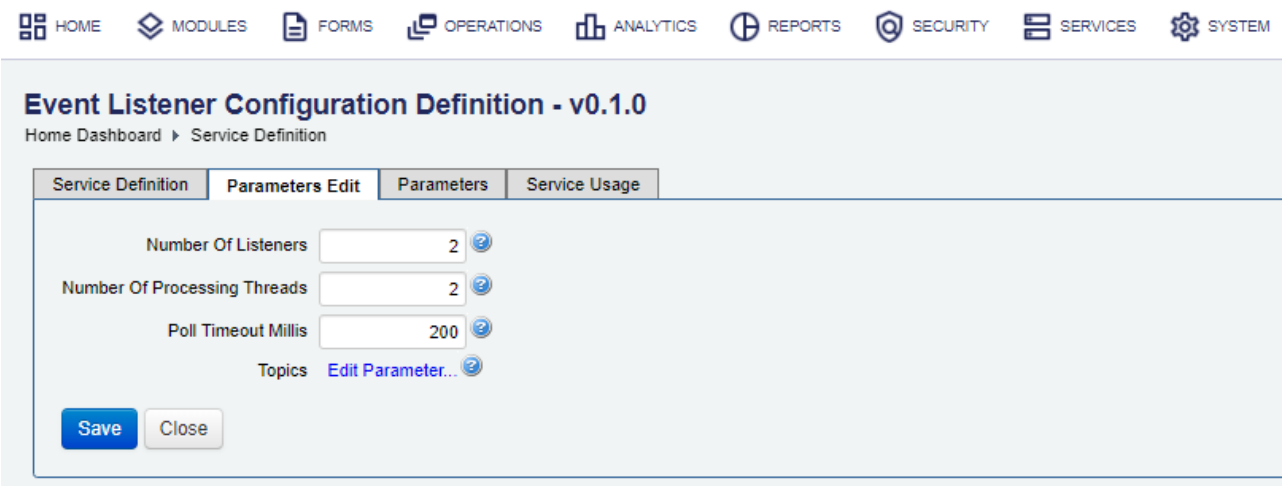
5. Select the Kafka service connection from Service Connection dropdown list. If it doesn't exist, create a new Kafka service connection. Make sure it connects to a Kafka producer with topics, which you specify as parameters in Topics, see below. This is important as, when started, a listener tries to subscribe to all configured Kafka topics.

> NOTE
>
> If you use VPN[1] to connect to external services, it must be configured correctly for the Kafka service connection to work correctly.

6. Click Save to update the changes.

[1]A virtual private network (VPN) is a mechanism for creating a secure connection between a computing device and a computer network, or between two networks, using an insecure communication medium such as the public Internet.

7. Click the Parameters Edit tab to configure the service parameters.



8. Edit a number of threads to be set to listen to Kafka topics in the Number of Listeners field. The default is 2.
9. Edit a number of threads which process stream records in the Number of Processing Threads field. The default is 2.
10. Specify the poll timeout (in milliseconds) in the Poll Timeout Millis field. The default is 200.
11. Click Topics to edit a list of Kafka topics in JSON format. For example, for one topic:

```
["KAFKA_TOPIC"]
```

Or, for multiple topics:

```
["KAFKA_TOPIC_1", "KAFKA_TOPIC_2"]
```

12. Click Save to update the changes.

The configuration changes are applied immediately to the listeners related to that configuration and then the listeners are shut down and started with the new configuration. For example, if you updated the number of listeners, all existing listeners are stopped and then the new number of listeners are started.

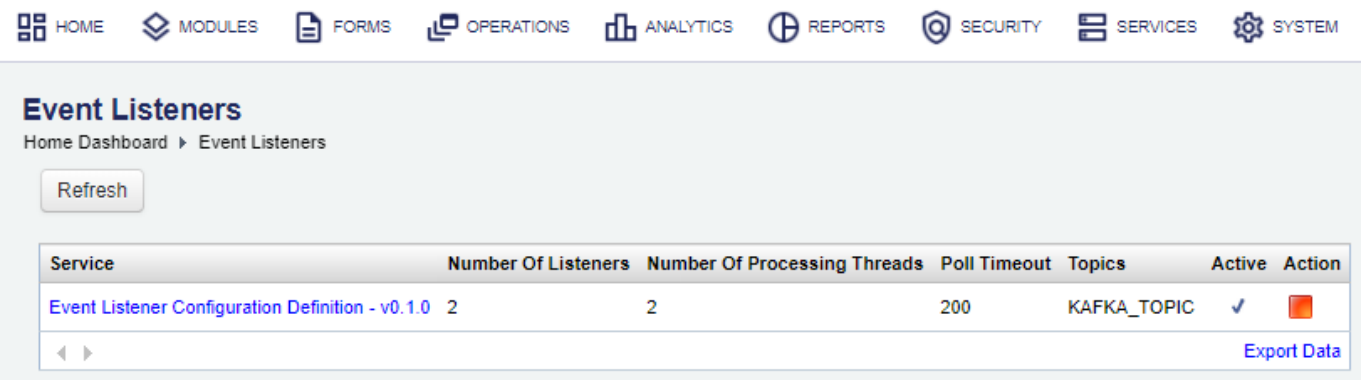
Next, learn how to view all core global services.

View Event Listeners

Manager allows you to view available event listener services, so you can easily find, manage, and update them.

To view all event listeners, select System > Event Listeners Status.

Event listeners are shown page by page, with the maximum number of records per page configured in the users preferences.



The list displays the following details:

- Service: a name of a service that implements a Kafka topic listener.
- Number of Listeners: a number of listeners.
- Number of Processing Threads: a number of threads that run listeners services to process stream records.
- Poll Timeout: a timeout (in milliseconds) to poll Kafka for new stream records.
- Topics: a list of Kafka topics in JSON format.
- Active: a status showing with whether an event listener is active or not.
- Action: click the green play icon to start a listener or click the red square icon to stop a listener.

To reload the list of event listeners, click Refresh.

You can also perform the following actions:

- Edit an event listener.
- Start / Stop an event listener.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn about event-based architecture.

System Event Publisher Overview

Journey Manager allows you to send server events and other information to external systems, usually clients, in near real-time using System Event Publisher (SEP). Clients can then use this functionality to monitor Manager from your external systems, such as Splunk, via a third-party message queue using standard protocols. Also, clients decide what type of data they want to collect, so they can:

- Monitor workload of their staff
- Build systems to track applications
- Monitor system or integration issues

System Event Publisher hooks into Manager's transaction processing engine, records events in memory and sends the applicable events to a queue. It all happens in the background to maintain performance of the user flow. Manager attempts to retry delivery of event messages in case of an error, but it can't be guaranteed, meaning, event messages are published on a best effort basis.

A Manager administrator can configure a System Event Publisher to publish Manager's events using Amazon AWS SQS, and, if required, authenticate a connection using the Amazon AWS Security Token Service[1]. You can further customize it to publish only events that are essential to be monitored within your external systems.

You can configure System Event Publisher to publish the following events to external systems:

- Submission audit log entries
- Submission Collaboration Jobs: steps, actions and action properties, and collaboration job update messages
- Submissions including data extracts, properties, milestones, and Groovy logs
- Submission update messages
- Submission form data mapping
- Submission error log entries
- Submission event log entries
- Submission Groovy service log entries
- Submission Groovy service log data

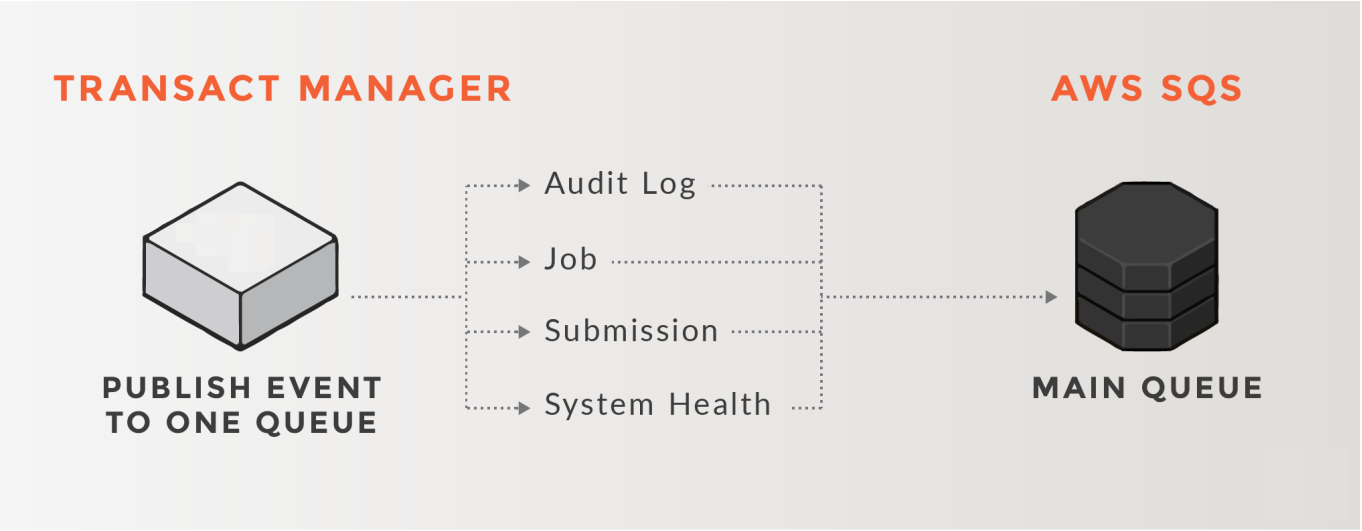
---

[1]The AWS Security Token Service (STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users).

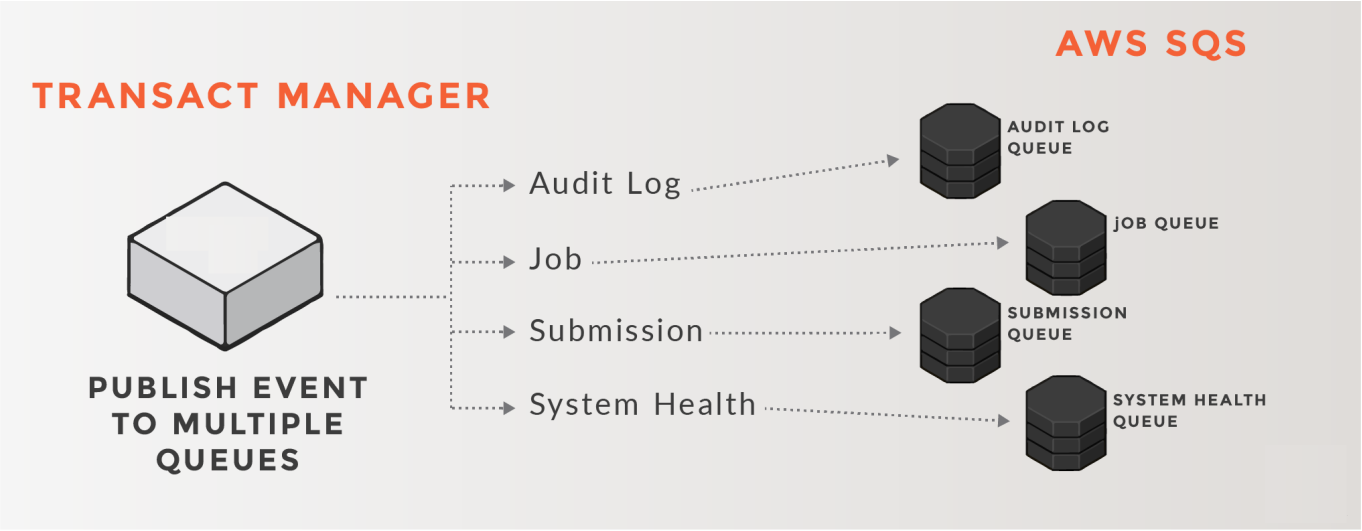
- Submission milestones
- Submission properties
- Transaction statuses
- System health
- User authentication events. | 18.11 This feature was introduced in 18.11.

System Event Publisher can publish events to the one default queue and up to three separate queues. For example, job, submission and system health events can be sent to three separate queues respectively. To achieve this, System Event Publisher should be configured differently. There are two distinct configurations:

A Single Queue



Multiple Queues



Next, learn how to configure system event publisher.

Configure System Event Publisher

Manager version 18.05 and higher comes with the System Event Publisher, which you can use to publish events into various queues. Initially, Manager supported only AWS queues, but starting from the release 20.05, it also supports Azure queues.

> NOTE
>
> A Manager administrator can configure the System Event Publisher.

To start using the System Event Publisher:

1. Configure a service connection to connect to a queue. Use an AWS SQS service connection or AWS STS service connection for AWS cloud, or an Azure service connection for Azure cloud.
2. Configure a system event publisher service: an AWS SQS System Event Publisher service for AWS cloud or Azure System Event Publisher Service for Azure cloud.
3. Configure the external to Manager processes and systems to transfer, store and report on the raw data according to your requirements.

Once the configuration is complete:

- Each time a Manager event takes place, the System Event Publisher formats the event relevant details as a JSON[1] message and places it on the internal queue.
- If opted for the system health message to be published, the System Event Publisher formats it as a JSON message with the system health message details and places it on the internal queue each hour.
- The internal queue is published immediately to the queueing system, such as AWS SQS or Azure Queue.

Next, learn about different types of System Event Publisher events.

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

System Event Publisher Events

The System Event Publisher publishes the following Manager events:

- Audit Log event
- Job event
- Submission event | 18.11 This feature was updated in 18.11.
- System Health event
- User authentication event | 18.11 This feature was introduced in 18.11.

The System Event Publisher uses the following generic JSON[1] message format for publishing events:

```
{
        "environmentName": "Engineering Test 1",
        "systemId": "c41d354134397c0362846d5efd7f34f4",
        "serverNode": "ip-192-168-120-149.us-west-2.compute.internal - TM Server",
        "threadName": "TM_Worker-1",
        "auditLog": {
                ...
        }
}
```

```
{
        "environmentName": "Engineering Test 1",
        "systemId": "c41d354134397c0362846d5efd7f34f4",
        "serverNode": "ip-192-168-120-149.us-west-2.compute.internal - TM Server",
        "threadName": "TM_Worker-1",
        "job": {
                ...
        }
}
```

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

NOTE

The event message contains only one child object component, such as `auditLog`, `job`, `submission` or `systemHealth`.

Next, learn how to configure System Event Publisher.

System Event Publisher Audit Log Events

Manager allows you to configure a [%TransactVariables.SystemEventPublisher%] to publish [%TransactVariables.AuditLog%] events.

> NOTE
>
> Select the Audit Log Publish checkbox in the AWS SQS System Event Publisher service configuration to publish this event.

System Event Publisher sends out Audit Log events in the following JSON[1] format.

```
{
        "environmentName": "server1.com",
        "systemId": "c41d354134397c0362846d5efd7f34f4",
        "serverNode": "ip-192-168-120-149.us-west-2.compute.internal - TM Server",
        "threadName": "TM_Worker-4",
        "auditLog": {
                "id": 517,
                "eventTime": "2018-05-13T00:08:47+1000",
                "user": "administrator",
                "eventType": "View",
                "entityName": "Submission",
                "entityId": 93,
                "message": "Submission XML viewed by administrator"
        }
}
```

Next, learn about other events.

[1] JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

System Event Publisher Job Events

Manager allows you to configure a [%TransactVariables.SystemEventPublisher%] to publish job events.

> NOTE
>
> Select the Job Publish checkbox in the AWS SQS System Event Publisher service configuration to be able to publish this event.

System Event Publisher sends out job events in the following JSON[1] format.

```
{
        "environmentName": "server1.com",
        "systemId": "c41d354134397c0362846d5efd7f34f4",
        "serverNode": "ip-192-168-120-149.us-west-2.compute.internal - TM Server",
        "threadName": "TM_Worker-4",
        "job": {
                "id": 3,
                "eventTime": "2018-05-10T21:37+1000",
                "jobKey": "19677169dac21105ce99be73d13bf3eb",
                "jobName": "Customer Onboarding Job",
                "jobRefNumber": "5VFL8Q7",
                "clientCode": "maguire",
                "status": "Completed",
                "stepName": "Application Completed",
                "stepStatus": "Completed"
        }
}
```

The `status` is a job status.

- In Progress - the collaboration job is active and in progress.
- Completed - the collaboration job has been completed. This happens when a endpoint step has completed.

---

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

- Expired - this happens when an endpoint expiry collaboration job has been automatically expired by Journey Manager.
- Cancelled - the collaboration job has been manually canceled by an administrator.
- Error - the collaboration job has a critical or unrecoverable error that requires an administrator intervention before it can be run again.
- Paused - the collaboration job has been paused temporarily.

The `stepStatus` is a job step status.

A job step status can be one of the following:

- In Progress- the step is in progress now.
- Completed - the step is completed.
- Expired - the step is expired.
- Cancelled - the step has been manually cancelled by an administrator.

Next, learn about other events.

System Event Publisher Submission Events

Manager allows you to configure a System Event Publisher to publish submission events.

> NOTE
>
> Select the submission checkboxes in the AWS SQS System Event Publisher service configuration to publish these events.

System Event Publisher sends out submission events in the following JSON[1] format.

[1] JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

```
{
        "environmentName": "server1.com",
        "systemId": "c41d354134397c0362846d5efd7f34f4",
        "serverNode": "ip-192-168-120-149.us-west-2.compute.internal - TM Server",
        "environmentName": "Engineering Test 1",
        "threadName": "default task-87",
        "submission": {
                "id": 17253,
                "eventTime": "2018-10-05T13:15:30Z",
                "submitKey": "ade3e4ebfe7016c8c10a3b5c0ee107e7",
                "requestKey": "28673f7e0c92d9d7ebc8bd80af8318af",
                "trackingCode": "Y85C5H",
                "clientCode": "maguire",
                "formCode": "CCA-RB",
                "formStatus": "Saved",
                "deliveryStatus": "Not Ready",
                "user": "john.smith@maguire.com",
                "jobRefNumber": "RJ6GJR9",
                "formDataMap": {
                        "First Name": "John",
                        "Last Name": "Smith",
                        "Email": "john.smith@amcore.com",
                        "Mobile": "0417 298 273"
                },
                "errorLog": [
                {
                        "id": 1850,
                        "errorTime": "2018-05-10T20:36+1000",
                        "errorType": "Email",
                        "message": "SmtpEmailService: SMTP Host=null"
                }
                ],
                "eventLog": [
                {
                        "id": 69378,
                        "eventTime": "2018-10-05T13:15:30Z",
                        "eventType": "INFO",
                        "message": "Email sent to: test_user1@company.com"
                }
                ],
                "groovyServiceLog": [
```

```
                {
                        "id": 74308,
                        "invokeTime": "2018-10-05T13:15:30Z",
                        "serviceName": "IDV Service",
                        "versionNumber": "1.0.2",
                        "durationMs": 26,
                        "status": "Completed",
                        "groovyLog": [
                                "13:15:30,056 INFO  Applicant CIN: 121483-38390\r",
                                "13:15:30,094 INFO  IDV Status: Completed"
                        ]
                }
        ],
                "milestone": [
                        "IDV" : "Completed"
                ],
                "property": {
                        "CIN": "121483-38390"
                },
                "requestLog": {
                        "id": 32813,
                        "requestTimestamp": "2018-05-10T20:34+1000",
                        "requestKey": "6e29b53f7b8cc4e5bf39148be987ef69",
                        "sessionId": "oC20STVJ4tvEOPl4jP256zhucOAX8YH5AIv7gzFj",
                        "ipAddress": "114.141.100.202",
                        "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:60.0) Gecko/20100101
fox/60.0"
                }
        }
}
```


For more information on the form status and delivery status values, see data retention status.

If the `formStatus` is Abandoned, an additional `formAbandonmentType` indicating the reason for abandonment is included.

A form abandonment types can be one of the following:

- Bounced
- Started
- Cancelled
- Ineligible

- Saved
- Submitted

The following is an example of an abandoned transaction with an additional `formAbandonmentType`:

```
...
        "formStatus": "Abandoned",
        "deliveryStatus": "Not Ready",
        "formAbandonmentType": "Cancelled",
        "formDataMap": {
                ...
        }
...
```

Manager includes submission PII data purging information in a JSON message. | 18.11 This feature was introduced in 18.11.

> NOTE
>
> This event is always published by the AWS SQS System Event Publisher service.

You can use this functionality to:

- Monitor submission PII data purging in Manager.
- Build external systems to monitor the progress of submission PII data purging.

The JSON format is the same as for submissions events with the following extra fields:

- scheduledSubmissionPurge
- scheduledPiiDataPurge
- piiDataDeletedTime

Before purging

```
{
        ...
        "deliveryStatus": "Completed",
        "scheduledPiiDataPurgeTime": "2018-10-23T15:12:22+1100",
        "scheduledSubmissionPurgeTime": "2019-01-14T15:12:22+1100",
        "user": "test_user1@company.com",
        ...
}
```

After purging

```
{
        ...
        "deliveryStatus": "Completed",
        "scheduledPiiDataPurgeTime": "2018-10-16T15:16:50+1100",
        "piiDataPurgedTime": "2018-10-16T15:16:59+1100",
        "scheduledSubmissionPurgeTime": "2019-01-14T15:12:23+1100",
        "user": "test_user1@company.com"
        ...
}
```

Next, learn about other events.

System Event Publisher System Health Events

Manager allows you to configure a [%TransactVariables.SystemEventPublisher%] to publish [%TransactVariables.SystemHealth%] events. Usually, an hourly message is sent, similar to System Health.

> NOTE
>
> Select the System Health Publish checkbox in the AWS SQS System Event Publisher service configuration to publish this event.

System Event Publisher sends out System Health events in the following JSON[1] format.

[1] JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

```
{
        "environmentName": "server1.com",
        "systemId": "c41d354134397c0362846d5efd7f34f4",
        "serverNode": "ip-192-168-120-149.us-west-2.compute.internal - TM Server",
        "threadName": "TM_Worker-1",
        "systemHealth": {
                "id": 392781,
                "eventTime": "2018-10-05T13:00:00Z",
                "abandonCount": 0,
                "deliveryCount": 0,
                "deliveryErrorStateCount": 0,
                "deliveryReceiptRenderCount": 0,
                "errorCollabJobCount": 0,
                "errorCount": 0,
                "errorCriticalCount": 0,
                "errorDeliveryCount": 0,
                "errorDynamicDataCount": 0,
                "errorEmailCount": 0,
                "errorFwSyncCount": 0,
                "errorGroovyCount": 0,
                "errorLivecycleCount": 0,
                "errorManagementCount": 0,
                "error_paymentGatewayCount": 0,
                "errorReceiptRenderCount": 0,
                "errorRenderCount": 0,
                "errorRestSoapCount": 0,
                "errorScheduledJobCount": 0,
                "errorSecurityManagerCount": 0,
                "errorSubmissionCount": 0,
                "errorUnclassifiedCount": 0,
                "eventCount": 0,
                "formRenderCount": 0,
                "fwSyncCount": 0,
                "groovyServiceCount": 0,
                "jobCreatedCount": 0,
                "saveCount": 0,
                "scheduledJobCount": 0,
                "submissionCount": 0,
                "userReceiptRenderCount": 0
        }
}
```


Next, learn about other events.

System Event Publisher User Authentication Events

Manager allows you to configure a System Event Publisher to publish user authentication events. You can use this functionality to monitor Manager for fraudulent and suspicious activity.

NOTE

Select the User Auth Publish checkbox in the AWS SQS System Event Publisher service configuration to publish this event.

System Event Publisher sends out user authentication events in the following JSON[1] format.


```
{
        "environmentName": "server1.com",
        "systemId": "f121e2b022fb852999c3be5f2ef7067f",
        "serverNode": "127.0.0.1 - TM Server",
        "threadName": "default task-47",
        "userAuth": {
                "id": 1234,
                "user": "test_user1@company.com",
                "portal": "Maguire",
                "eventTime": "2018-10-15T11:27:33+1100",
                "eventIpAddress": "127.0.0.1",
                "eventSessionId": "nILJPPBTZB-NuNU9tukO_l5UAgeH4fRpqLR2KzeJ",
                "eventUserAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/01.0.0101.100 Safari/537.36",
                "eventType": "Login Success"
        }
}
```


A user login event type can be one of the following:

- Account Locked Temp.
- Login Failure.
- Log Out.

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

- Login Success.
- Password Reset.
- Session Expiry.

Next, learn about other events.

Security Managers Overview

NOTE

The Security Managers online documentation has been replaced with a PDF version. If you have access to the Temenos instance of Microsoft Teams, you can download the Security Managers PDF from Teams; otherwise, contact your customer support manager (CSM) to request a copy of the PDF.

Journey Manager is designed with a robust security architecture in mind that ensures a safe and secure foundation for users to access forms and applications hosted on the server, as well as call a set of REST API. The security architecture is implemented using Security Managers, which are software components within Manager that handle users request access to different form spaces and modules. Users can be external or internal, and they can require access to form spaces or a combination of form spaces and modules in different environments.

Another important use of security managers is authentication of REST API calls. As Manager provides a set of REST API allowing integration with external systems, it is important to make communication secure. By default, REST API calls use basic authentication, but you can configure any required authentication mechanism, such as OAuth or SSO, for REST API endpoints.

You do it by configuring a security manager for a form space or a module that exposes REST API endpoint. When a request is sent to the REST endpoint, a user account, which identifies this call, needs to pass through Spring Security Session Management. The user must be authenticated with the form space's or module's security manager by using Single Sign-On (SSO), such as ADFS, oAuth2 or custom SSO.

Security Managers are responsible for:

- User authentication[1]
- User authorization[2]

[1]User authentication is the verification of an active human-to-machine transfer of credentials required for confirmation of a user's authenticity.
[2]User authorization is interested in who the user is, and is used to determine what functions, actions, data, or other parts of an application the user has access to.

- User self-registration
- Session management[1]

Security Manager use the latest technologies, such as TLS, SSL, SHA and AES, to allow them fulfill their responsibilities.

Additionally, Manager has the following built-in features to harden security of applications it runs:

- TLS encryption for data in transit.
- Form XML data at rest is stored as Encrypted BLOB values using AES 256-Bit encryption.
- Passwords are stores as SHA 512 hash values.
- SMTP Relay over TLS for integration with bank's email systems.
- TLS, SSL mutual authentication and IP Whitelisting for internal integrations.
- Role based permissions can be integrated with authentication systems, such as ADFS, WS-Trust, OAuth2, and SAML.

The security architecture enables developers and system administrators to implement best security practices, such as:

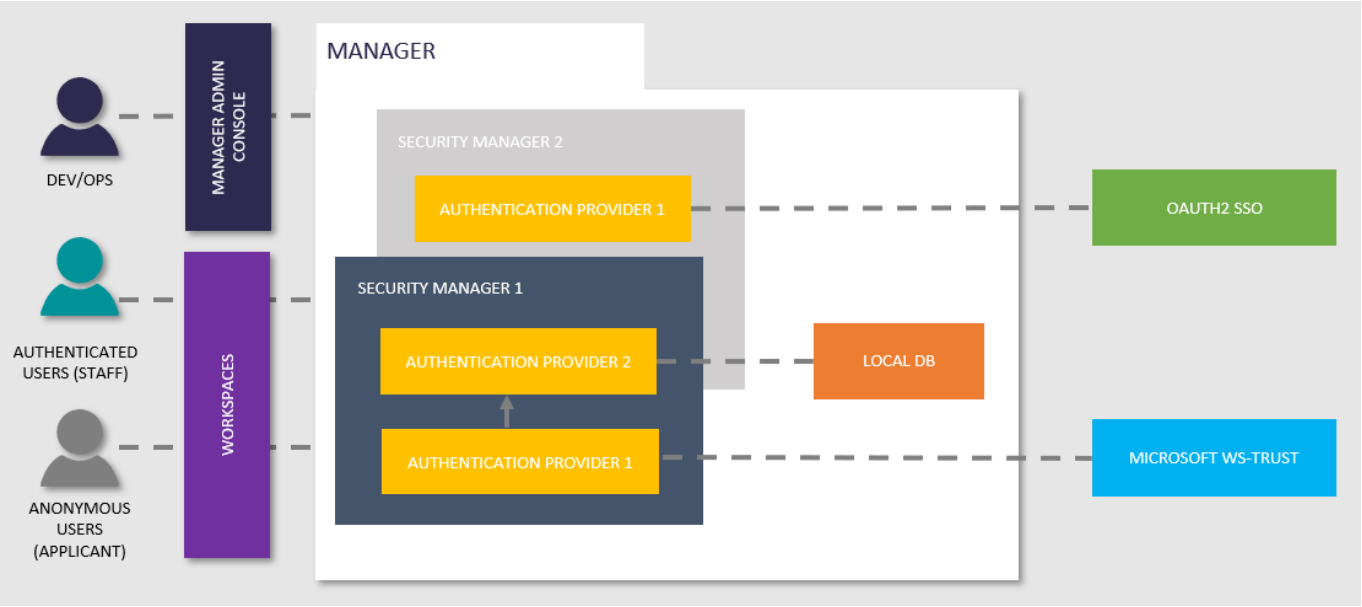
- Never log PII[2] information to the Groovy service log.
- Never turn on debug level of logging in production environments.
- Never store passwords in the clear formats.
- Never log passwords to the log files.
- Never pass internal security data to the client, for example, by adding them in the XML files.

---

[1]Session management is the rule set that governs interactions between a web-based application and users. Browsers and websites use HTTP to communicate, and a web session is a series of HTTP requests and response transactions created by the same user. Since HTTP is a stateless protocol, where each request and response pair is independent of other web interactions, each command runs independently without knowing previous commands. In order to introduce the concept of a session, it is necessary to implement session management capabilities that link both the authentication and access control (or authorization) modules commonly available in web applications.

[2]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

A security manager contains one or more authentication providers that can be chained. That is, if one authentication provider can't authenticate a user due to some reason, the next authentication provider is called and so on until the user is authenticated or no authentication providers are left. Providers can be local or delegated, as shown below.



An authentication provider's main job is to authenticate a user that will result in allowing or denying user access to a resource, which is often a form. Security Managers can be exported and imported across different Manager instances to make it easier to implement security requirements.

Manager comes with the following ready-to-use security managers:

- Local Security Manager
- LDAP Security Manager
- Microsoft ADFS Security Manager
- Microsoft WS-Trust Security Manager
- SSO Security Manager
- OAuth2 SSO Security Manager
- Fluent Microsoft ADFS Security Manager | 19.11 This feature was introduced in 19.11.
- Fluent SSO Security Manager | 19.11 This feature was introduced in 19.11.
- Fluent OAuth2 SSO Security Manager | 21.11 This feature was introduced in 21.11.

SSO security managers, such as Microsoft ADFS, have specific Groovy scripts, which you can configure to acquire a SSO token or call a chain of authentication providers. It allows you to create various SSO and login flows.

SSO Flow

1. SSO Get Auth Token. The result is passed into the authentication providers. If there's no result or a local authentication provider exists, it can go to the login flow
2. Authentication provider list
3. SSO authentication OK response
4. SSO re-validation

Login Flow

1. Login page
2. Authentication Providers, for example, WS-Trust and LDAP or Local DB

Manager controls access to the Manager Dashboard using permissions and organization based filters, whereas access to forms and content on the form space is controlled using groups, permissions, and user account based filtering.

Download the Security Managers PDF

The Security Managers PDF is available to download from the Temenos instance of Microsoft Teams.

1. Login to Microsoft Teams.
2. Go to the Temenos Journey Manager (TJM) team, and select the Temenos Document Share channel.
3. Select Files, and browse to the `SecurityManager (INTERNAL USE ONLY)` folder.
4. Download the Security Managers PDF for your version of Manager (`Security-Managers.pdf`).

Next, learn about organizations.

User Accounts Overview

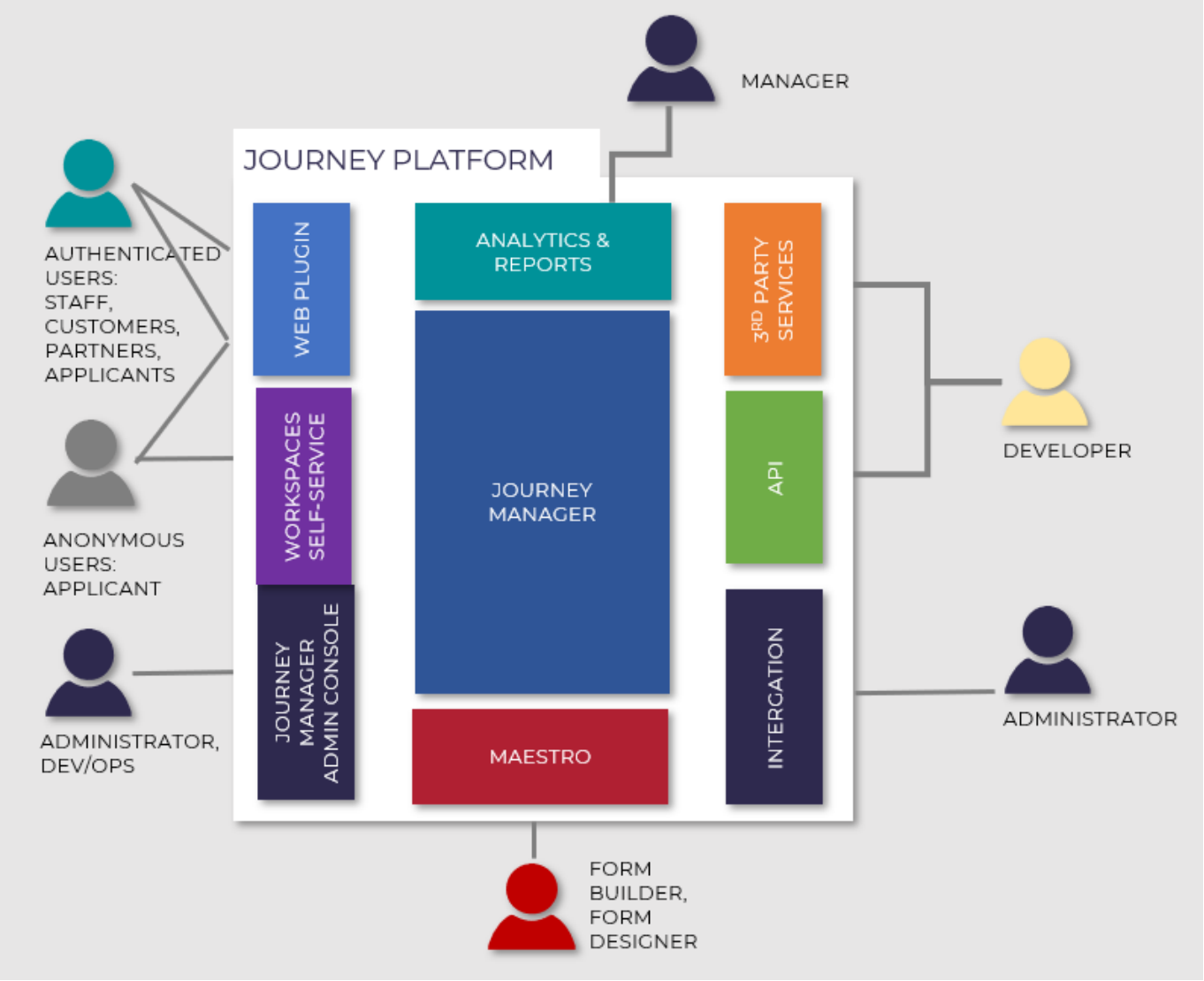
Journey Manager defines user accounts to authenticate and authorize users to log on to the Journey Manager Home Dashboard and one or more form spaces. to access assigned tasks, or Workspaces. This is controlled by assigning users to form spaces, where the Manager dashboard is one of form spaces.

There are two types of the users that can access forms:

- Authenticated users
- Anonymous users (unauthenticated or public users)

For more information, see Authenticated vs Unauthenticated Transactions.

The diagram below illustrates various user types and what they can access in the Temenos Journey Manager.



Each user account can belong to one or more roles. Roles are collections of user accounts with similar access rights. Similarly, user accounts can belong to any number of groups, which group user accounts into non-permission based collections, for example, administrators who are interested in receiving certain email notifications.

An administrator is a user account that is associated with the Manager dashboard form space. A form space user is a user account that is associated with at least one form space. Note that these categories are not mutually exclusive: An administrator can also be associated with arbitrarily form spaces, and can have global access or be restricted to data associated with one or more specific organizations.

Manager controls access to certain content or actions using permissions. When requesting restricted content, Manager only allows this if the required permissions are present. Permissions can be granted to one or more roles, and a role is characterized by the permissions it contains. If a user account belongs to several roles, they have all permissions that are set for at least one of their roles.

As users make changes to Manager's configuration, these changes are tracked and written to the audit log. This makes it possible to view an object's history and review a specific user's changes.

A user account is part of the Journey Manager security management system, which allows for fine-grained control based on user's functions, roles, groups, and access permissions. To provide flexibility around the security configuration, Manager supports one or multiple security managers. A security manager contains such settings as how users are authenticated, lockout and password settings, session timeout, SSO configuration and user self-registration settings.

Each security manager contains one or more authentication managers responsible for authenticating a user during login. Each form space can be associated with its own security manager or use the default security manager defined in the system.

Manager provides the easy-to-use interface to create, modify, and deactivate user accounts, as well as manage access permissions, user roles and groups. It also comes with the bulk import functionality to create multiple users, including user's details, roles, groups and access configurations, which significantly simplifies user provisioning task.

Manager doesn't impose any limits on the number of user accounts you can have. However, as customers[1] handle user authentication in different ways, there might be some limit in the

---

[1]A customer is a person who uses Temenos Journey Manager to create, host and manage application forms. Other names for a customer is a client.

Security Managers the customers use, such as SSO, LDAP and so on, or a physical limit in the database. Most customers only authenticate their own users, but some might also authenticate form applicants raising transactions. In this case, they could theoretically have thousands of users.

Manager doesn't have any built-in functionality to purge old or inactive user accounts automatically, so you should periodically audit user accounts to ensure their number doesn't grow beyond a point that impedes the system performance. For more information, see data retention management.

Authenticated Users

Before users can log in to a form space, which requires authentication, they must have valid user accounts. Manager allows an administrator to create a user and grants access to one or more form spaces. The user receives an email with login credentials, such as a username and password, to access forms hosted on the form space.

Manager allows authenticated users to login to a form pace using credentials by delegating authentication and authorization to the 3rd party system, such as Active Directory, LDAP[1] or SSO[2]. Manager can also maintain local users, such as contractors that are not in the organization's LDAP system.

The authenticated users can be:

- Business users internal to a company
- Government departments
- Staff and contractors that work in the field

NOTE

Authenticated users have access to the advance features of the secured portal such as Form, Task and History pages.

---

[1]Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol network.

[2]Single sign-on (SSO) is a property of access control of multiple related, yet independent, software systems. With this property, a user logs in with a single ID and password to gain access to any of several related systems.

Anonymous Users

Anonymous (unauthenticated or public) users can access forms if its form space is configured to allow anonymous access. An example of such a form space is the Web Plug-in form space. Members of the public click on a link from a company website which opens an unsecured form page in the portal. The available list of forms is manually maintained by the company on their website.

Using anonymous users has the following advantages:

- Removes the overhead of maintaining users and passwords
- Minimizes privacy issues related with storing public users' data, such as PII[1]

> NOTE
>
> Anonymous users don't have access to the advance features of the secured portal such as Form, Task and History pages.

An anonymous user can still use the forms, but with the following limitations:

- If they save a form, an email will be sent to them with a link that will open the saved form
- Tasks are emailed to a user with a link to open the task
- Job statuses can be emailed out the user as the job progresses

Anonymous users must provide their email address. Optionally, a security question can be set to open saved forms and tasks.

You can grant standard roles to users to control their access to and define capabilities at their form spaces. Each form space has a set of roles that can be granted to a user. You can also create a new role as needed.

A typical form user must only be granted access to a form space where forms are hosted. This allows self-registered users immediate access to forms. However, most users must be granted access to:

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

- Form spaces
- Modules
- Organizations
- Roles
- Groups

Quite often you may want to restrict user's access to a limited number of forms or even one form that are hosted on a form space. You can achieve this by creating a form group, adding a user to that group and assigning that group to a form you want to restrict an access to. Alternatively, you can create a special form space, assign the form to this form space and then assign this form space to that user. For more information, see how to provision a user account.

When working with user accounts, you should consider the environment that users are accessing, which can be production, development, or testing environment. Manager is designed to be customizable to accommodate enterprise's security configurations and policies. The following users require access to a production environment:

- Form users to access, complete and submit forms
- Operational staff to monitor and manage transactions and collaboration jobs
- Administrators to manage the environment and keep it secure
- System managers to view and analyze transaction reports
- Reviewer or manager to approve or reject customer form applications

The following users require access to a development and test environments:

- Administrators to manage the environment and keep it secure
- Form builders to create and configure forms
- Testers to test forms

Users can only log into a form space if their user accounts are assigned to that form space. It is not possible to assign a form space to a user account when the form space is only accessed anonymously.

The following users should have access to form spaces:

1. Form users to log into a form space.
2. Form Builders to log into one or more form spaces:
    - Maestro
    - Manager
    - Journey Analytics
    - Business Reports

    - Salesforce
    - Workspaces
3. Administrators to log into Manager.
4. Testers to log into various form spaces.
5. Operators to log into Manager.
6. Managers to log into one or more form spaces:
    - Journey Analytics
    - Business Reports
    - Salesforce
    - Workspaces

A standard user account contains the following information:

- Credentials - username and password
- User type - security managers used in your form space, which can be Local, LDAP or SSO
- User profiles - user contact details. A user can have more than one user profile containing different contact details
- Form spaces - assigned to the user to authorize access to specific form spaces
- Organizations - assigned to the user to authorize access to specific organizations. Applies to modules, but not relevant for form spaces
- Roles - grants individual roles which specify fine-grained permissions to access Temenos Journey Manager platform and form spaces
- Groups - belongs to an individual group for authorizing access to restricted forms, receive alerts and notifications pertaining to a group

Manager enforces some extra security rules to ensure that users cannot access restricted content, such as organizations, groups, users, forms, transactions and collaboration jobs. | 19.05
This feature was introduced in 19.05.

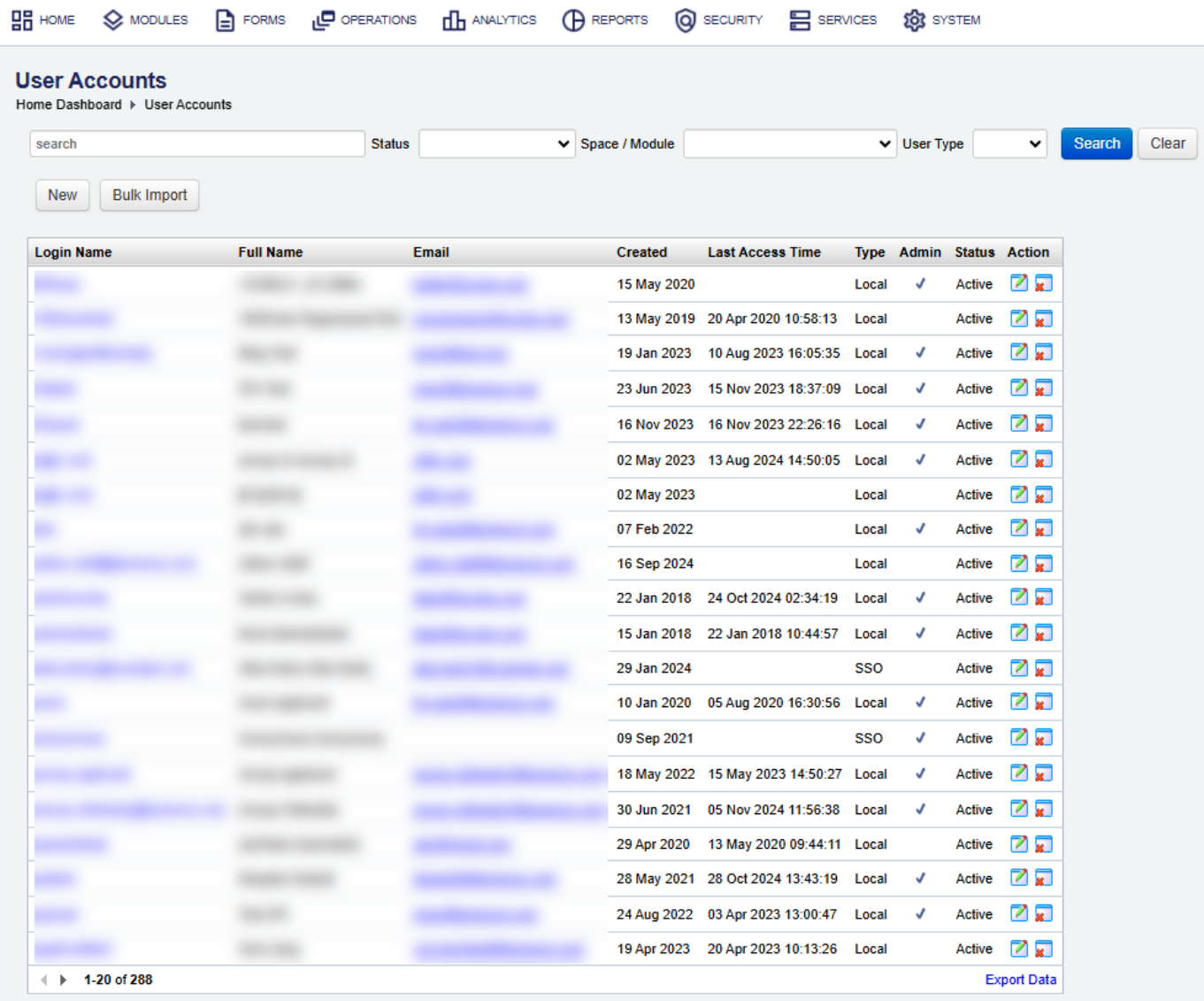
Next, learn how to view user accounts.

View User Accounts

Manager allows you to view the list of registered user accounts as well as manage them.

To view a list of user accounts, select Security > User Accounts.

User accounts are shown page by page, with the maximum number of records per page configured via the user's preferences.



Login Name	Full Name	Email	Created	Last Access Time	Type	Admin	Status	Action
			15 May 2020		Local	✓	Active	
			13 May 2019	20 Apr 2020 10:58:13	Local		Active	
			19 Jan 2023	10 Aug 2023 16:05:35	Local	✓	Active	
			23 Jun 2023	15 Nov 2023 18:37:09	Local	✓	Active	
			16 Nov 2023	16 Nov 2023 22:26:16	Local	✓	Active	
			02 May 2023	13 Aug 2024 14:50:05	Local	✓	Active	
			02 May 2023		Local		Active	
			07 Feb 2022		Local	✓	Active	
			16 Sep 2024		Local		Active	
			22 Jan 2018	24 Oct 2024 02:34:19	Local	✓	Active	
			15 Jan 2018	22 Jan 2018 10:44:57	Local	✓	Active	
			29 Jan 2024		SSO		Active	
			10 Jan 2020	05 Aug 2020 16:30:56	Local	✓	Active	
			09 Sep 2021		SSO	✓	Active	
			18 May 2022	15 May 2023 14:50:27	Local	✓	Active	
			30 Jun 2021	05 Nov 2024 11:56:38	Local	✓	Active	
			29 Apr 2020	13 May 2020 09:44:11	Local		Active	
			28 May 2021	28 Oct 2024 13:43:19	Local	✓	Active	
			24 Aug 2022	03 Apr 2023 13:00:47	Local	✓	Active	
			19 Apr 2023	20 Apr 2023 10:13:26	Local		Active	

The list displays the following details:

- Login Name: a name used to authenticate a user to log onto Manager. It can be a name or an email address
- Full Name: a user's first and last name
- Email: an email address to send notification messages

- Created: date when the user account was created
- Last Access Time: the date and time when the user accessed the account last time
- Type: a type of a user account:
  - Local: user accounts managed by Manager. This is the default user type.
  - LDAP: user accounts managed externally by a Manager's LDAP[1] security manager.
  - SSO: user accounts managed externally by a Manager's SSO[2] security manager.
- Admin: shows whether a user has the Admin permission or not.
- Status: a status of a user account

  A user account status can be one of the following:

  - Active
  - Inactive
  - Locked
  - Locked Temporarily
  - Pending
  - Rejected
- The total number of user accounts configured in Manager. You can see this information on the bottom left corner of the window. In this example, Manager has 178 registered users.

To filter or search the user accounts by one or more criteria, specify the following settings and click Search:

- User account's name or part of a name.
- Status

  A user account status can be one of the following:

  - Active
  - Inactive
  - Locked
  - Locked Temporarily

---

[1]Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol network.
[2]Single sign-on (SSO) is a property of access control of multiple related, yet independent, software systems. With this property, a user logs in with a single ID and password to gain access to any of several related systems.

    - Pending
    - Rejected
- Space:
- User Type:
    - Local: user accounts managed by Manager. This is the default user type.
    - LDAP: user accounts managed externally by a Manager's LDAP[1] security manager.
    - SSO: user accounts managed externally by a Manager's SSO[2] security manager.
- Organization

Click Clear to reset the search criteria.

> NOTE
>
> A check mark against a user in the Admin column means the user has access to the Transact Manager form space.

You can also perform the following actions:

- Create a user account.
- Configure a user account.
- Remove a user account.
- Bulk Import user accounts.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to bulk import user accounts.

---

[1]Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol network.

[2]Single sign-on (SSO) is a property of access control of multiple related, yet independent, software systems. With this property, a user logs in with a single ID and password to gain access to any of several related systems.

Create a User Account

Manager allows you to create user accounts, so you can set up a new or migrate an existing user from another environment. Creating a user is the first step in setting up a user account, so you also need to assign roles, groups, organizations, and form spaces to the user account before the user can perform actions in Manager. Then, the user can create and configure a user profile to adjust it to the user's needs.

NOTE

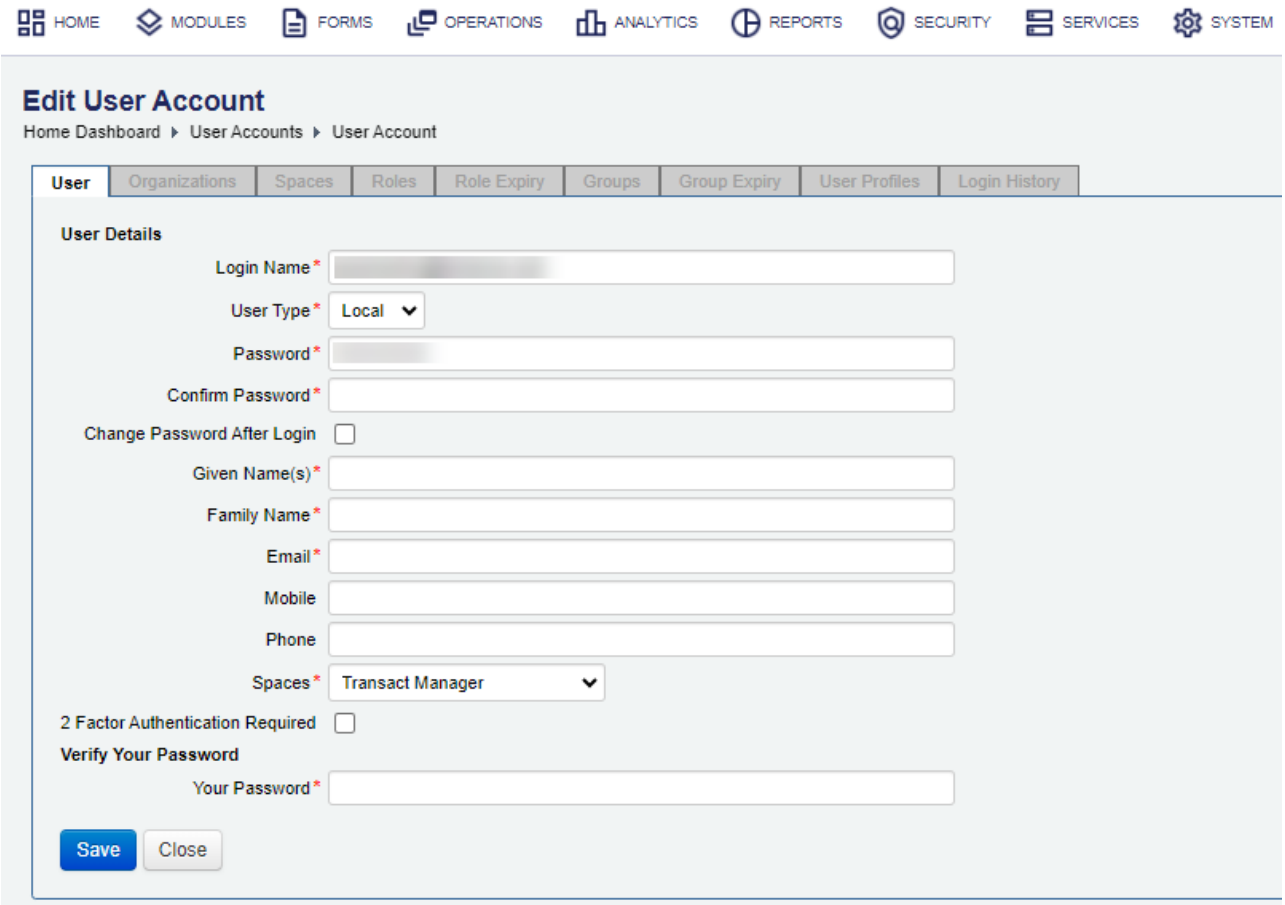
Use bulk import to create multiple users.

NOTE

You must be an administrator to create a new user account.
Only a Maestro Organization User Manager can create a new user for their assigned organizations.

To create a user account:

1. Select Security > User Accounts.
2. Click New.



3. Enter a login name for a new user in Login Name. The user can use this login name to log into Manager, Journey Maestro, Composer and form spaces, but it must be the same as the Login Name used for the Maestro cloud instance. Otherwise, users will not be able to update Manager form versions built in Maestro.

   NOTE

   It is recommended to use an email address as a login name.

4. Select a user type from the User Type dropdown list. Configuration is different for each user type, so use the specific details as shown below.

- Local: user accounts managed by Manager. This is the default user type.
- LDAP: user accounts managed externally by a Manager's LDAP[1] security manager.
- SSO: user accounts managed externally by a Manager's SSO[2] security manager.

5. Enter a user account password in the Password field.

   The standard Journey Manager password requirements are:

   - Must be at least 12 characters in length.
   - Must not exceed 40 characters. | 20.05 This feature was introduced in 20.05.
   - Must contain at least one or two letters and one digit.
   - Must contain at least one non-alphanumeric character, such as `%`, `&`.
   - Must contain mixed case, one upper case, and one lower case.
   - Must not be a password existing in the Illegal Password Values list.

   NOTE

   An administrator can customize all above password requirements, except for the not exceeding 20 characters, for the Local Security Manager.

6. Enter the exact same password as was just entered in Password in Confirm Password.
7. Select the Change Password After Login checkbox to force a new user to change their password on a first login. The user is unable to access restricted content until the initial password has been changed.
8. Enter a user's first name and middle names (if any) in Given Name(s).
9. Enter a user's last name in Family Name.
10. Enter a valid user's email address in Email.
11. Enter a user's mobile number in Mobile. Only accepts numeric and must be at least 10 digits.
12. Enter a user's phone number in Phone. Only accepts numeric and must be at least 10 digits.

---

[1]Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol network.

[2]Single sign-on (SSO) is a property of access control of multiple related, yet independent, software systems. With this property, a user logs in with a single ID and password to gain access to any of several related systems.

13. Select a form space, which you want the user to log onto, from the Spaces dropdown list. For example, select the Transact Maestro to provide the user with the access to the Maestro shared environment only. You will also need to assign the user to the other Transact Maestro form spaces for compatibility mode. For Maestro version 5.0, you need to add the Transact Maestro 5.0 form space. This is the same for all other versions of Maestro as they have their own form spaces.

    NOTE

    You can assign more form spaces from the Spaces tab.

14. Select the 2 Factor Authentication Required checkbox to enable two-factor authentication (2FA)[1] for better protection of user logging to a module or a form space. If a module or a form space has a security manager that requires 2FA, then the 2 Factor Authentication Required is selected by default. Then, when a user tries to log in to a module or a form space, the additional Verification Code field is displayed. | 18.11 This feature was introduced in 18.11..
15. Enter your password in the Your Password field.
16. Click Save to update the changes.

Next, learn how to configure a user account.

[1]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.

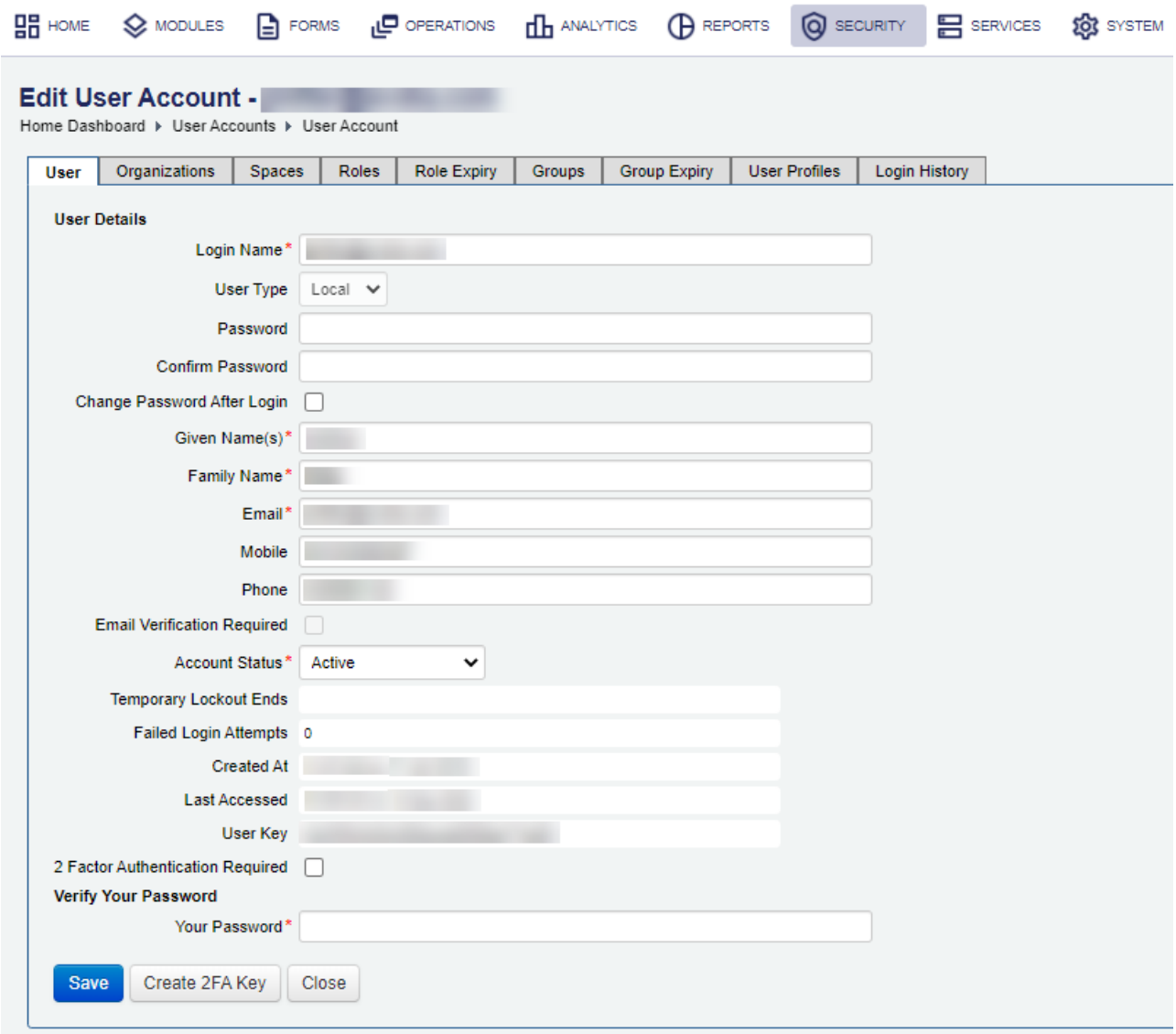
Configure a User Account

After you have created a user, Manager allows you to configure the user account to make sure the user can use Temenos Journey Manager platform products, such as Manager, Maestro, Exchange, and Analytics. You can also use this functionality to:

- Reset a password
- Change a user account status
- Change user's contact details
- Send or resent 2FA keys to configure 2 Factor Authentication
- Deactivate a user account

To configure a user account:

1. Select Security > User Accounts.
2. Locate a user account and click Edit.
3. Select the User tab to update user's details. For explanation of parameters, see Create a User Account.



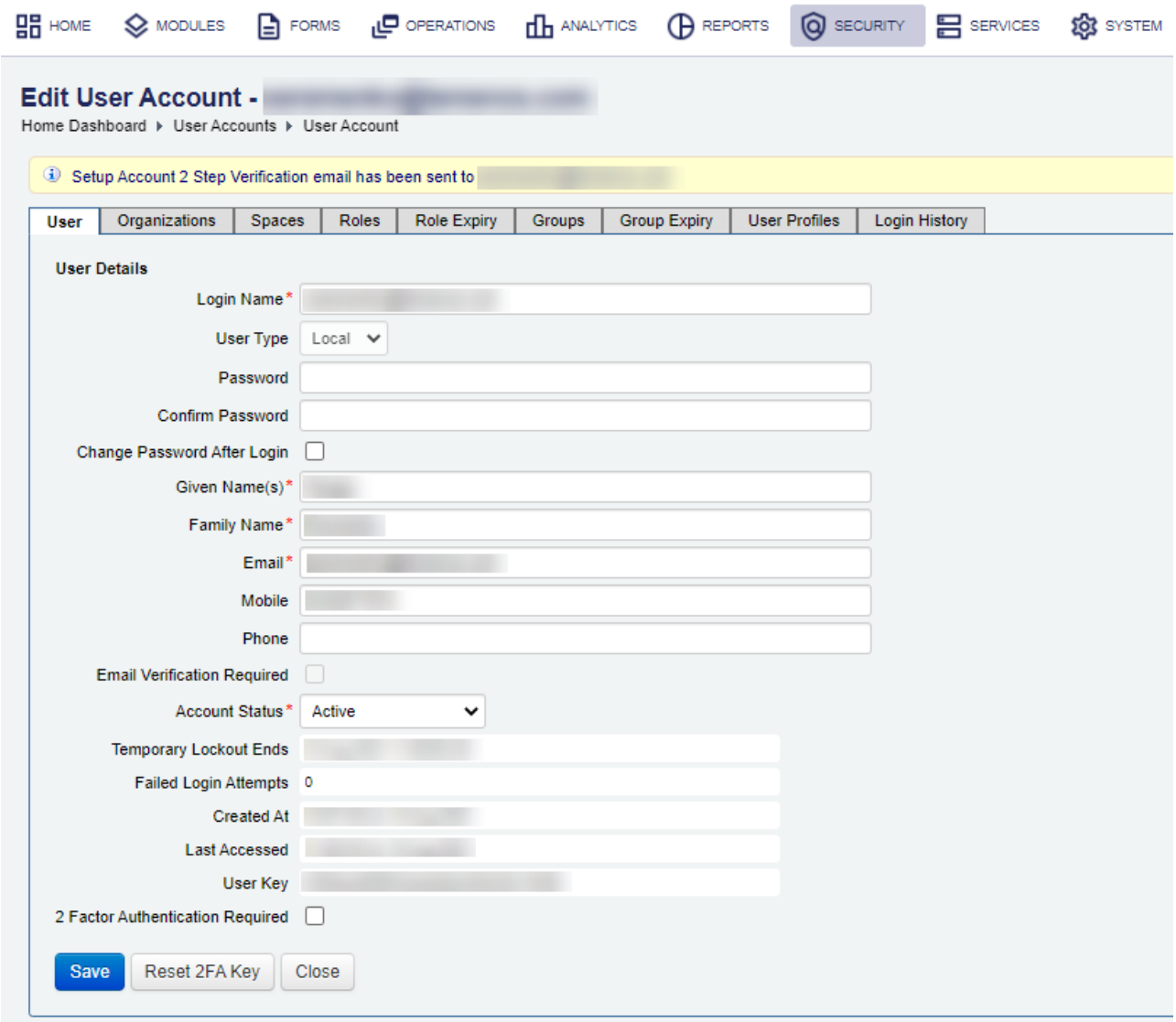
4. Update a user's password or user's details as required. For more information, see create a user account. .
5. Select a user account status from the Account Status dropdown list, if user's status changed, for example, the account was locked.

   A user account status can be one of the following:

   - Active
   - Inactive
   - Locked
   - Locked Temporarily
   - Pending
   - Rejected

6. Click Create 2FA Key to generate a 2FA[1] key and send it as an email to the user to complete the account configuration. | 18.11 This feature was introduced in 18.11. Use this functionality to send your verification code, for example, when you want to set up the 2-step verification in Journey Space. If you have already configured 2FA, the Reset 2FA Key button is shown. Click to Reset 2FA Key send a 2FA key to the user again.

[1]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.



NOTE

The Create 2FA Key button or the Reset 2FA Key button is shown only when the 2FA is enabled in a Security Managers that is used to authenticate a form space the user is assigned to.

NOTE

You must select the 2 Factor Authentication Required checkbox for the 2FA process to work.

NOTE

After you requested the 2FA key, don't forget to follow the instructions in the email to finalize the 2FA configuration. Otherwise, the configuration won't be complete. If you didn't receive the email with the QR code, contact your Administrator to resend the 2FA key.

7. Click Save to update the changes.
8. Assign roles to a user account.
9. Assign organizations to a user account.
10. Assign form spaces to a user account.
11. Assign groups to user account.
12. Configure user role expiry.
13. Configure user group expiry.
14. Click Save to update the changes.

Next, learn how to create a user profile.

Assign Roles to a User Account

Manager allows you to assign roles to a user. You can use this functionality to:

- Assign roles to a new user.
- Grant or revoke roles to an existing user.

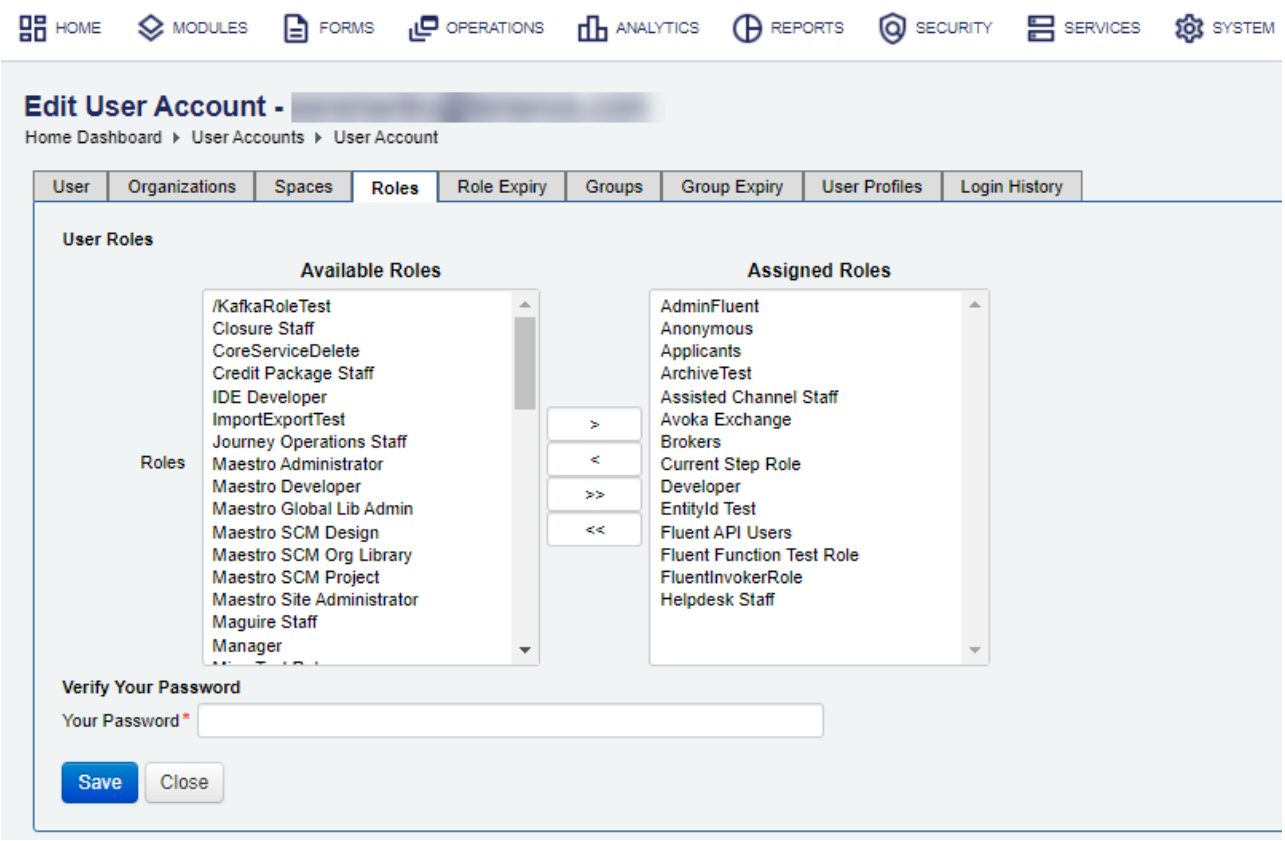
Each role has a set of permissions associated with it, so by assigning a role, you enable certain product functions to this user. Consider the following before assigning roles to the user:

- As a minimum requirement, for most form users, such as your customers and applicants, logging into the form space grants them access, allowing them to select and complete forms. No roles are needed to be assigned to these user accounts.
- The types of activities the user is required to perform in each form space. Roles are specific to each form space.
- What each of the individual permissions associated with a role is allowing the user to do. Each role has a specific set of permissions controlling what is permissible within each space.
- The role of the administrator is a specialized role which gives the user permission to manage the security in Manager.

To assign roles to a user:

1. Select Security > User Accounts.
2. Locate a user account and click Edit.

3. Select the Roles tab to see the currently assigned roles.



4. Select one or more roles from Available Roles that you want to assign to the user.

> NOTE
>
> Use the `Shift` and `Ctrl` keys to select multiple spaces on a Windows PC.

5. Click > to move the selected roles from Available Roles to Assigned Roles.
6. Click < to revoke the selected roles from Assigned Roles.
7. Click >> to assign all roles.
8. Click << to revoke all roles.
9. Enter your password in Your Password to make changes to the roles.
10. Click Save to update the changes.

> NOTE
>
> A user must log out and then log back into the form space for the changes to take effect.

NOTE

A user account's assigned roles can be changed at any stage.

NOTE

If you suspect that a user account has been compromised, remove all their roles and form spaces and make the user account inactive.

From the Roles tab, a global administrator will see and can assign all roles to a user account (provided their own roles contain the permissions around role assignment). An organization administrator, however, can only see and assign roles that are organization assignable roles. These are roles where the Organization Assignable checkbox is ticked. Any roles where the Organization Assignable checkbox is not ticked are hidden from and are unavailable to an organization administrator. Both global and organization administrators can change the role assignments for their own user account. An organization administrator assigning roles to their own account, the same rule applies where they can only see and assign organization assignable roles.

Next, learn how to assign organizations to a user.

Assign Organizations to a User Account

Manager allows you to assign organizations to a user so you can control user access to forms belonging to a certain organization. Most administrators are assigned to specific organizations as organization administrators to provide system administration these organizations.

Consider the following before assigning or re-assigning organizations to a user account:

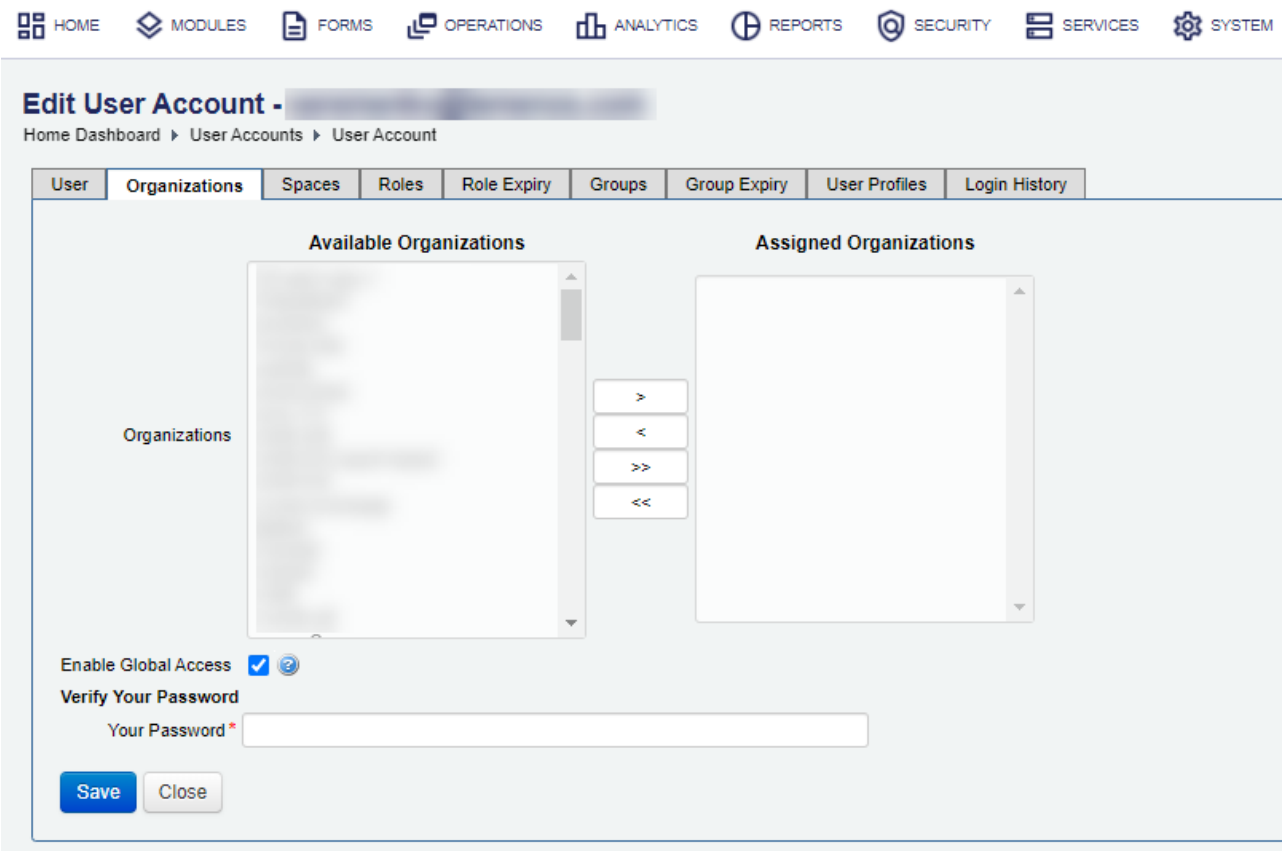
- Form builders must be granted access to organizations containing the forms they need to create or update.
- Administrators can be either an organization administrator (for most) or a global administrator. An organization administrator is only allowed to administer on behalf of specific organizations.
- Organization administrators have restricted access privileges, so they can only see the subset relevant to the specific organizations assigned to their user account.
- Organization administrators can only assign roles, which are organization specific, to other administrators. This is to prevent them from promoting themselves or others to become a global administrators.
- A global administrator is not constrained in any way and therefore can perform operation on behalf of all organizations.
- Operators are either granted access to specific organizations or to all organizations. Like administrators, operators monitor submissions and collaboration jobs on behalf of the specifically assigned organizations or all organizations, when globally enabled.
- Form space users don't need to assign any organizations to their user accounts as form spaces do not support restrictions based on organizations. When a user is granted access to a form space, they normally view and access all forms pertaining to any organization registered to host forms within that space, unless other restrictions are placed such as form groups.
- Within development and testing environments, it is often necessary to allow users a range of functions. This may be managed according to your enterprise's security policies.

NOTE

Assign selected individual organizations to a user account to provide a fine-grained control of user access to the system.

To assign organizations to a user account:

1. Select Security > User Accounts.
2. Locate a user account and click Edit.
3. Select the Organizations tab to see the currently assigned organizations.



4. Select one or more organizations from Available Organizations that you want to assign to the user account. This list depends on whether you are logged in as a global administrator or as an organization administrator. Global administrators have all organizations listed in available organizations section, but only those organizations granted to an organization administrator are listed.

   NOTE

   Use the `Shift` and `Ctrl` keys to select multiple spaces on a Windows PC.

5. Click > to move the selected organizations from Available Organizations to Assigned Organizations.
6. Click < to revoke the selected organizations from Assigned Organizations.
7. Click >> to assign all organizations.
8. Click << to revoke all organizations.
9. Enter your password in the Your Password field to make changes to the organizations.

10. Select the Enable Global Access checkbox to allow only a global administrator to grant the global administrator privilege to another user. In this example, the organizations in the Assigned Organizations section will revert to the Available Organizations section and it is no longer possible to assign organizations individually to this user account. That is, either this user account is for a global administrator or for an organization user. For a global administrator user account, it is assumed that all organizations are assigned to it.

    NOTE

    This option is only available to a global administrator.

    NOTE

    If you disable global access for your own user account, it can only be reinstated by another global administrator.

11. Click Save to update the changes. Now, it is possible to reassign individual organizations to this user account.

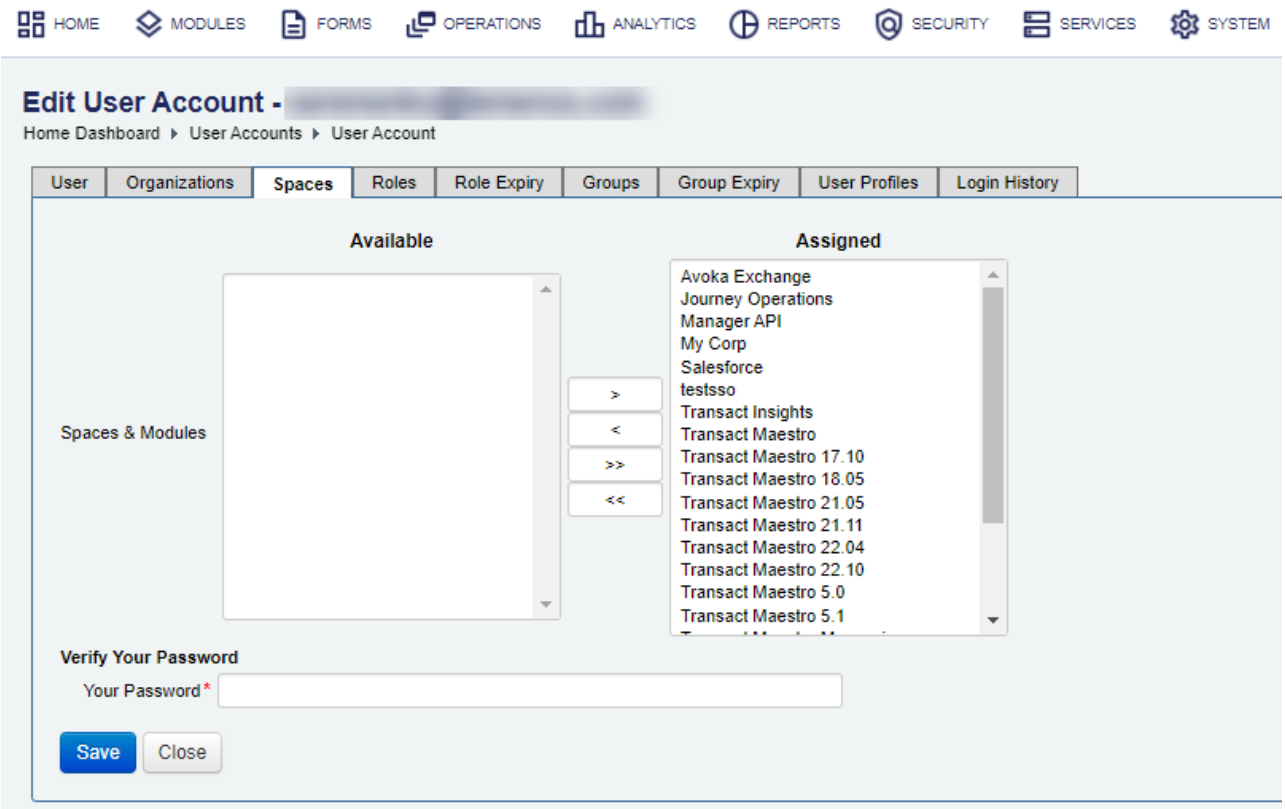
Next, learn how to view all user accounts.

Assign Form Spaces to a User Account

Manager allows you to assign form spaces to a user to grant or remove user access to form spaces

To assign form spaces to a user:

1. Select Security > User Accounts.
2. Locate a user account and click Edit.
3. Select the Spaces tab.



4. Select one or more spaces from Available Spaces that you want to assign to the user account. This list depends on whether you are logged in as a global administrator or as an organization administrator. Global administrators have all spaces listed in the Available Spaces & Modules section, but only those spaces granted to an organization administrator are listed.

   NOTE

   Use the `Shift` and `Ctrl` keys to select multiple spaces on a Windows PC.

5. Click > to move the selected form spaces from Available Spaces to Assigned Spaces.

6. Click < to revoke the assigned form spaces from Assigned Spaces.
7. Click >> to assign all form spaces.
8. Click << to revoke all form spaces.
9. Enter your password in the Your Password field to make changes to the form spaces.
10. Click Save to update the changes.

Next, learn how to assign organizations to a user.

Provision a User Account

NOTE

User Account Provisioning is a process that ensures user accounts are created, given proper permissions, changed, deactivated, and removed.

Provisioning users in Journey Manager is a frequent task, which an administrator performs to create new or migrate existing users from different environments. This process is straightforward, but you have to understand what roles, groups and permissions a user should have to utilize the product efficiently and securely. You also need to figure out what organizations and form spaces the user will need to access to be able to do the work. After that, setting up a user is as simple as following these steps:

1. Create a new or configure an existing user account.
2. Assign roles to the user account.
3. Assign organizations to the user account.
4. Assign form spaces to the user account.
5. Assign groups to the user account.
6. Configure user role expiry to harden the Manager security.
7. Configure user group expiry to harden the Manager security.

You can use the bulk import functionality to create multiple users including user's details, roles, groups and access configurations. This significantly simplifies user provisioning task. Finally, if a user is no longer required, you can deactivate it.

There are the following differences in provisioning different users, which are described below.

Provision a User Account for Accessing a Maestro Shared Environment Only

You may want to create users that can access and use only one product. Maestro is a good example as it is a product that can be run by itself and is used primarily by form developers, who don't need to use other products, such as Analytics.

NOTE

Only a Maestro Organization User Manager can create a new user for their assigned organizations.

To configure a user:

1. Configure a user as described above.
2. Assign the Maestro Developer and Maestro Administrator roles to the user.
3. Assign the Transact Maestro form space to a user account. You will also need to assign the user to the other Transact Maestro form spaces, such as Transact Maestro 5.0, Transact Maestro 5.1, Transact Maestro 17.05, Transact Maestro 18.11, for compatibility mode. For Maestro version 5.0, you need to add the Transact Maestro 5.0 form space because each Maestro has its own form space.

   NOTE

   There are multiple roles for Maestro users. Manager access is not required, so do not assign Manager roles.

Provision a User Account with the Organization User Manager Role

Manager has a set of roles that can be assigned to a user. One of these roles is the Organization User Manager role, which allows a user with this role to create new users, edit user details, and reset passwords.

To configure a user as the Organization User Manager:

1. Configure a user as described above.
2. Assign the Organization User Manager role to the user.
3. Assign the following form spaces to the user:
   - Maestro
   - Manager
   - Maestro 5.0
   - Maestro 17.10
   - Maestro 18.05

   NOTE

   If you use Maestro prior to version 18.05, you only need to assign the form spaces up to the version of Maestro you are using. For example, if you use Maestro 17.10, you need to assign both the Transact Maestro 5.0 and the Transact Maestro 17.10 form spaces to the user.

4. Assign the required organizations to the user.

NOTE

Do not assign global access to the user account.

Restrict User's Access to Forms

You can restrict user's access to several forms or even one form that are hosted on a form space using special groups or form spaces.

To restrict an access using a group:

1. Create a form group.
2. Add a user to that group.
3. Assign that group to a form you want to restrict an access to.

To restrict an access using a form space:

1. Create a special form space.
2. Assign the form to this form space.
3. Assign this form space to that user.

Troubleshooting

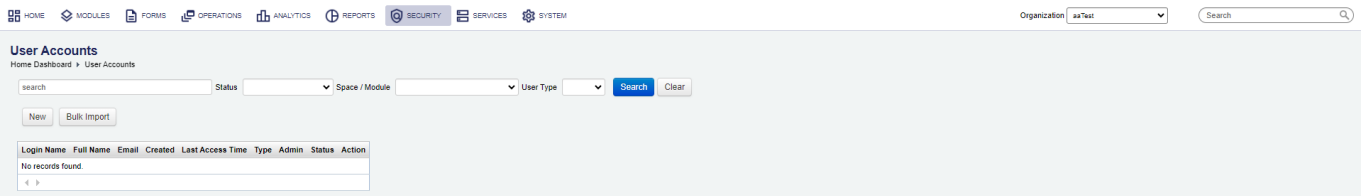
Sometime, after you have created a user account in Manager, it's still not visible while searching for it. There are a few primary causes, which are described below.

NOTE

An Organization User Manager (Org User Manager) is responsible for administering Manager users for a company. They will be contacted first before raising a support request.

General Cause 1

Incorrect organization is selected in the Organization dropdown list of the Manager menu bar, as shown below.



Solution

When you do not see a drop down field like above, and usually see a fixed Organization name, it implies you have access to only one organization. Hence, this scenario is not applicable, please move on to next general cause.

If you see a drop down field like the above image, it implies you have access to multiple organizations. Please select an appropriate organization which is also assigned to the new user. After selecting the appropriate organization if you do a user account search the user shall be visible.

You can also leave the Organization dropdown list blank. In this case, you will see the users for all organizations that you have access to in the search results. Hence, if unsure of user's assigned organizations please keep this field blank.

General Cause 2

The user account was created by someone and assigned to an organization which you don't have access to.

Solution

After checking the general cause 1, if you still cannot see the user in search results, please ask your Org User Manager to assist you in this regard. The Org User Manager should be able to assign the correct organization to the user. Please ensure that you have access to the same organization as the new user to make them visible in your search results.

General Cause 3

You have just created the user account but you cannot see it in search results . Usually, the user account was not assigned any organization.

Solution

In this scenario, the newly created user account does not belong to any organization hence, it is not visible to users who have access to particular organizations.

However, users with global access (generally, Org User Manager) can see the user and should be able to assign an appropriate organization to the new user account so that it is visible to you.

Provide your Org User Manager with the username of the newly created user so the account can be searched. Org User Manager should be able to edit the newly created user account and assign the organization as per business requirements.

Since you have created the user it is very likely you have administration rights for the organization, please ensure that the new user was assigned to your organization to make them visible in your search results.

> NOTE
>
> If the user is still not visible after above troubleshooting, raise a request with Temenos Support team using our support portal.

Next, learn how to view user accounts.

User Profiles Overview

Manager uses user profiles to represent various user contact details in a user account. These contact details are predominately used for identification purposes and prefilling forms. User contact details may be maintained by an administrator from within Manager or by a user with an account that supports user profile management. Users and Manager administrators can create and manage one or more user profiles for a user account.

When logged into a form space (and if the standard **Account** tab is available), users can:

- Update their contact details
- Create more than one user profile
- Elect another user profile to be the default, when more than one user profile exists
- Delete a user profile

An administrator is also able to view and manage user profiles.

When you create a user account, it automatically creates a default user profile named My Profile, which contains a copy of the following user contact details:

- Given Name
- Family Name
- Email
- Mobile

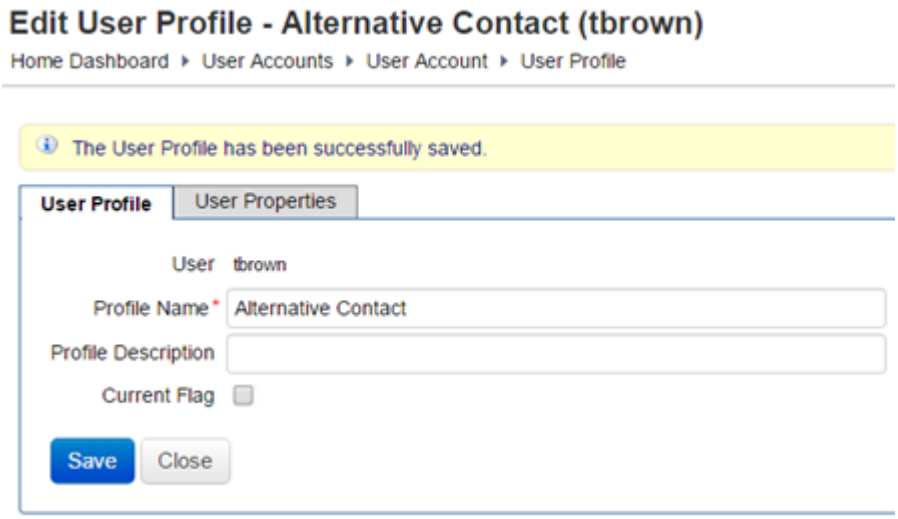
Next, learn how configure a user profile.

Create a User Profile

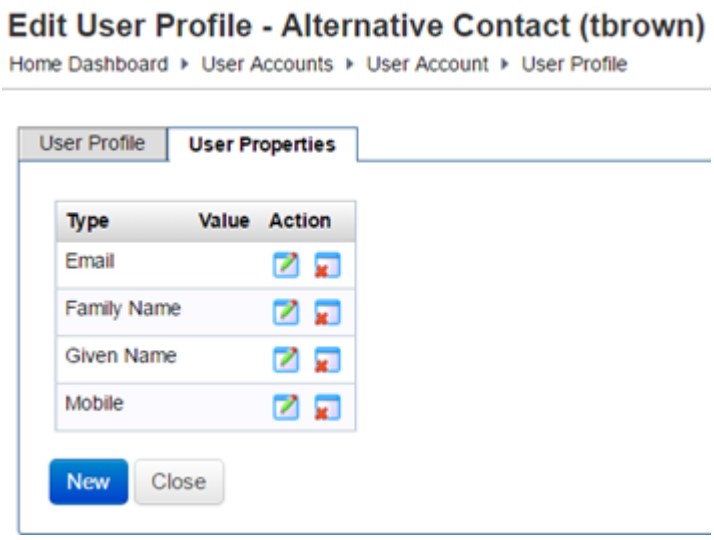
Manager allows you to create several user profiles for the existing user account.

To create a new user profile:

1. Select Security > User Accounts.
2. Locate a user account and click Edit.
3. Click the User Profiles tab to see all user profiles for this user account. The current user profile has a check mark against it it it is used as the default user profile for form prefill.
4. Click New.



5. Enter a unique profile name in the Profile Name field.
6. Enter a description in the Profile Description field.
7. Select Current Flag to make this the current user profile for this user account.
8. Click Save to update the changes.
9. Select the User Properties tab to list all user properties for this user profile, with blank values in the Value column.



10. Click Edit to set values for each of the user properties.
11. Select a property type and enter a value for the user property in Value, for example an email address.
12. Click Save to update the changes.

You can remove a user property, if it is not required. However, it's not recommended to delete a user property as this can cause the user profile to be inconsistent with other user profiles. If you want to remove the user property, leave its value field blank.

For prefilling a form with the user's contact details sourced from a default user profile, you need to map each data item, such as Given Name and Family Name, to target locations using form data configuration.

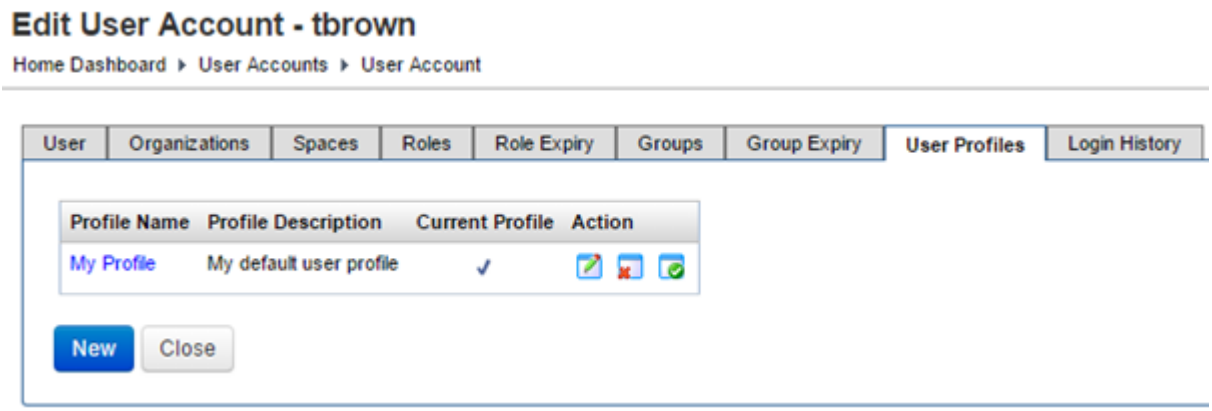
Next, learn about user property types.

Configure a User Profile

Manager allows you to configure and update the user profile of a user account.

To configure a user profile:

1. Select Security > User Accounts.
2. Locate a user account and click Edit.
3. Click the User Profiles tab to see all user profiles for this user account. The current user profile is selected in the Current Profile column, so it is used as the default user profile for form prefill.



4. Click Edit to update the profile.
5. Click Remove to delete the profile.
6. Click Set as default profile to set the profile as default, when there are more than one user profile for this user account.

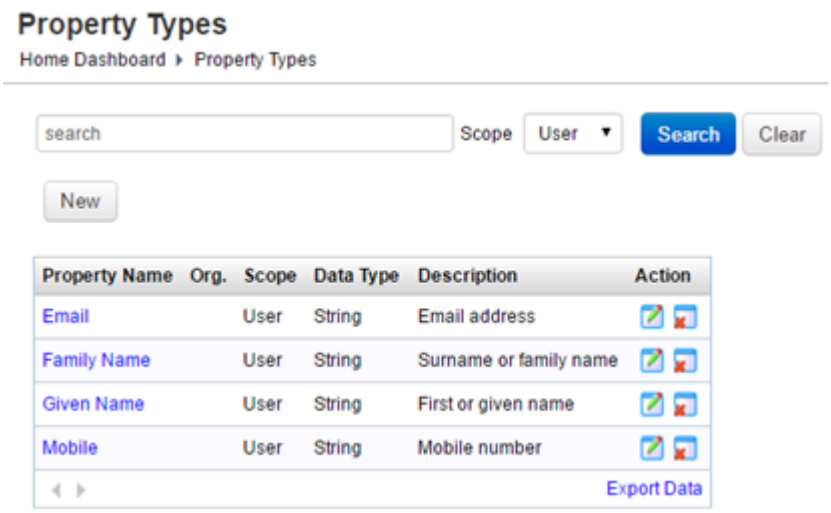
Next, learn how to create a user profile.

User Property Types

A user profile is comprised of a set of fields that are defined and maintained as user property types in Manager. Property types define user property scope.

Out of the box, Manager comes with a standard set of user profile fields where each corresponds to a predefined property type of the user property scope. Each user property type is available to all user profiles for maintaining a user's contact details both in Manager (by an administrator) and the assigned form spaces (by a user).

Manager uses the following standard property types:



- Email
- Family Name
- Given Name
- Mobile

An administrator can customize a user profile by adding some new details that you want users to provide. This is done by creating a new property type with a property scope of User.

> NOTE
>
> Only update the contact details of a user via the user profile of a user account.

Administrators can update the Given Name, Family Name, Email, and Mobile details that are registered directly against a user account. However, these changes won't be automatically reflected in the corresponding fields against the default user profile because that user contact information is always sourced from the user's default user profile and not from the contact details.

Next, learn how to create a user profile.

Remove a User Account

If a user account is not required any more or you want to stop the user from logging into any spaces or modules, Manager provides you with two options:

- Deactivate the user account.
- Remove the user account.

> NOTE
>
> We recommend deactivating a user account rather than removing it.

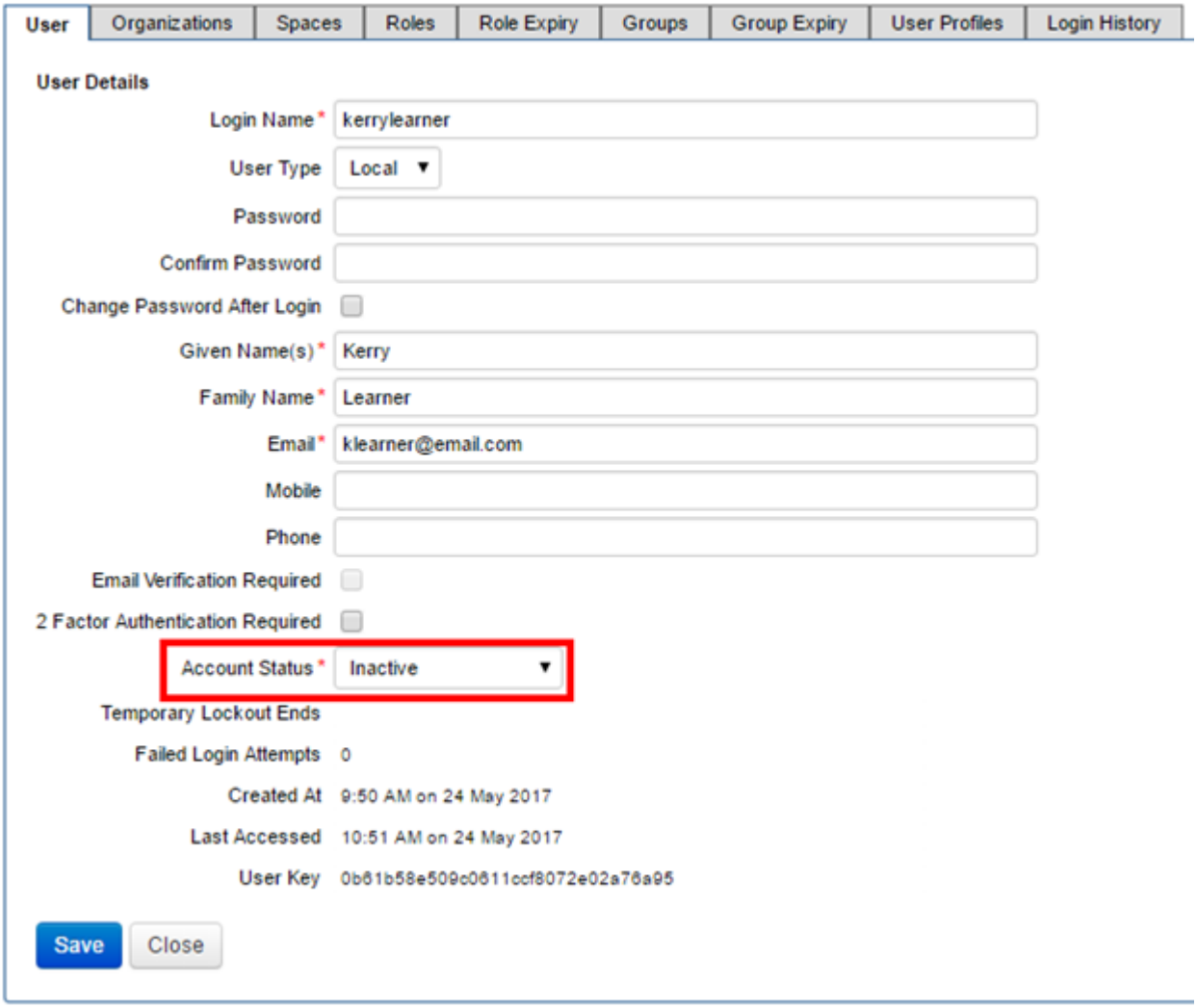
You can also configure your security manager in such a way that it automatically makes a user account inactive when the user hasn't logged onto the server for a specified duration of time.

Deactivate a User Account

You can deactivate a user account to prevent a user from logging in successfully using their credentials. Their details, however, will be preserved in Manager. Submissions made by a user are preserved even after their user account is deleted, but the submissions will no longer associated with a user account. Deactivation of a user account can be reversed by reactivating the user account.

To deactivate a user account:

1. Select Security > User Accounts.
2. Locate the user account and click Edit.
3. Click the User tab and select Inactive from the Account Status dropdown list.



4. Click Save to update the changes.

NOTE

To reactivate the user account, select Active from the Account Status dropdown list.

Remove a User Account

NOTE

When you remove a user account, Manager irrevocably removes all data associated with the user account, such as transactions and collaboration jobs. This action can't be rolled back.

To remove a user account:

1. Select Security > User Accounts.
2. Locate the user account and click Remove.
3. Click OK to confirm.

NOTE

You can't remove the user account you are currently logged in as.

NOTE

Even after a user account has been deleted, the changes that the user made to the system are still visible in the audit log.

Next, learn how to create a user account.

Bulk Import of User Accounts

Manager allows you to import multiple user accounts specified in an Excel spreadsheet to speed up user creation process. A bulk upload spreadsheet can include the following user's details:

- Login Name
- Given Name
- Family Name
- Email
- Password
- Change Password
- Portals
- Send User Enrollment Approval Email
- Groups. This includes access check that an imported user can access a group. | 20.05 This feature was introduced in 20.05.
- Global Access. This includes access check that an imported user is global. | 20.05 This feature was introduced in 20.05.
- Org codes | 20.05 This feature was introduced in 20.05.
- Roles | 20.05 This feature was introduced in 20.05.
- Mobile

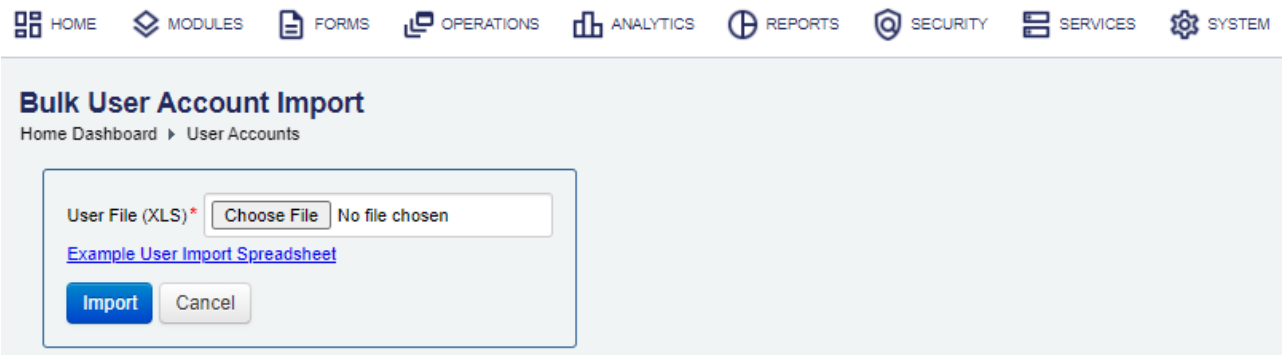
You should consider the following before running a bulk import:

- Existing users are ignored and won't be updated.
- Only local users will be created.
- Duplicates user accounts, identified by their Login Names, won't be imported.
- Users can only be associated with form spaces used to host forms, but not other spaces such as the Manager console. You can manually assign the missing form spaces after the importing.
- If a user who does the bulk import is not a global administrator, then it can't grant global access to users being imported.

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

To bulk import user accounts from an Excel spreadsheet:

1. Select Security > User Accounts.
2. Click Bulk Import.



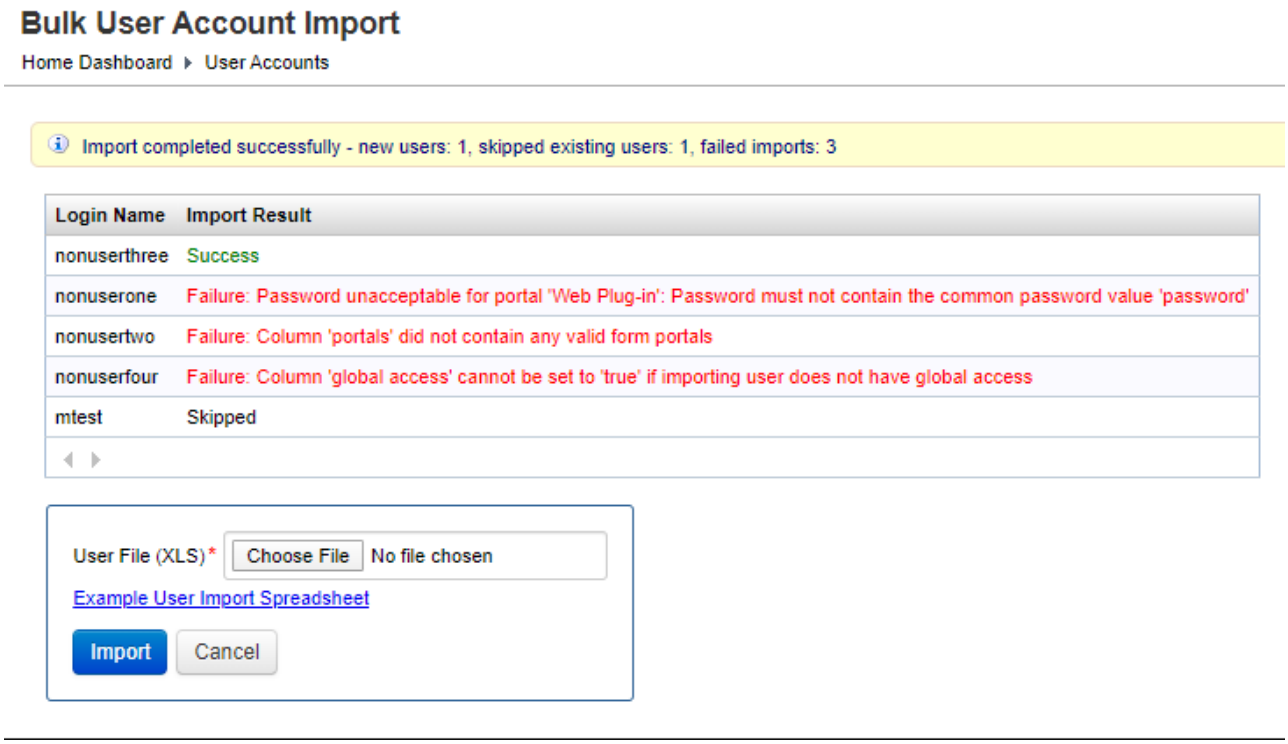
3. Click Example User Import Spreadsheet to download a sample XLS[1] file to assist you in entering user account details.

> NOTE
>
> If the XLS file doesn't follow the expected format, the bulk import will fail. Specifically, the header column names need to be correct and in a certain order.

4. Click Choose File and locate an XLS file you wish to import.

[1]is a file extension for a spreadsheet file format created by Microsoft for use with Microsoft Excel. XLS stands for eXceL Spreadsheet. Microsoft Excel files use a proprietary format for storing Microsoft Excel documents.

5. Click Import.



6. Check the import results. The successfully imported user accounts are displayed at the top. They are followed by users for which an error occurred, such as a mandatory column not being set. User accounts that were skipped because they already exist are displayed at the bottom.
7. For each error, you may choose to revise and fix your XLS import file and re-import it. Alternatively, you can create each user account manually.

NOTE

You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

Next, learn how to configure a user account.

Migrate User Accounts

Manager provides a mechanism to migrate user accounts from one environment to another, which significantly saves time to set up new servers. You can also configure and test user accounts in different environments before migrating them over to production.

To migrate user accounts:

1. Export local user accounts from a source environment
2. Check an exported XLS file and adjust it as required, for example, remove users you don't need in a new environment
3. Bulk import user accounts into a target environment using the exported XLS file
4. Adjust user configuration if needed

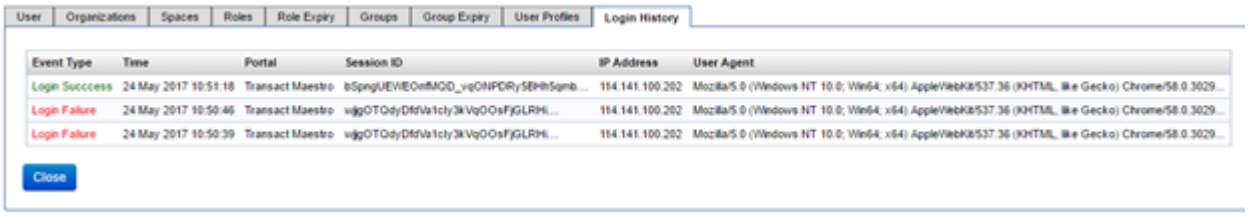
Next, learn how to create a user account.

View User Login History

Manager allows you to view a list of all login attempts by each user or all users.

To view a user's login history:

1. Select Security > User Accounts.
2. Locate a user account and click Edit.
3. Select the Login History tab to see each login attempt and its details:



- Event Type is the status of the login attempt.

  A user login event type can be one of the following:

  - Account Locked Temp.
  - Login Failure.
  - Log Out.
  - Login Success.
  - Password Reset.
  - Session Expiry.
- Time is the date and time of the login attempt.
- Portal is the space or portal a user accessed or attempted to access.
- Session ID is a user's session ID[1] of the login attempt.

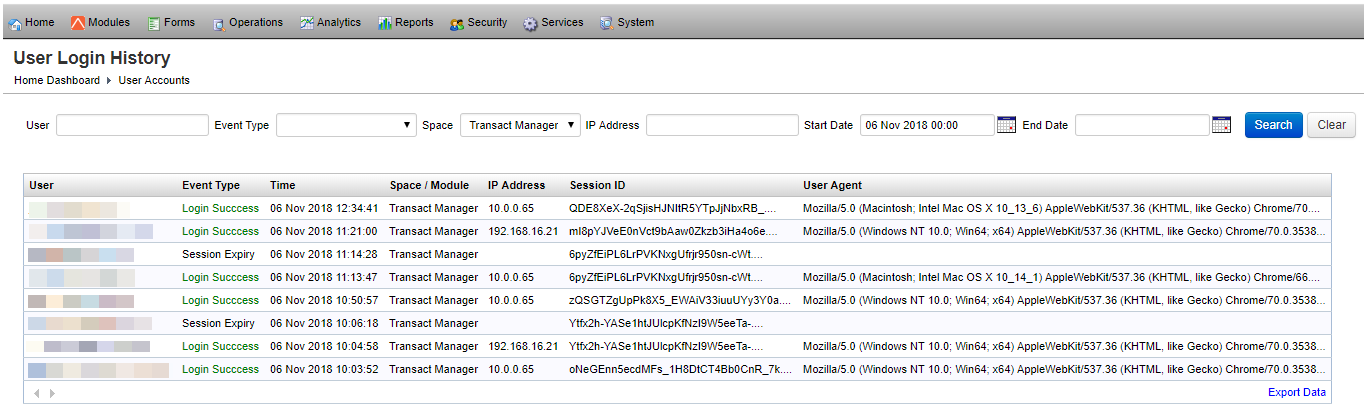
[1]Session ID is a unique number that a Web site's server assigns a specific user for the duration of that user's visit (session). The session ID can be stored as a cookie, form field, or URL (Uniform Resource Locator). Some Web servers generate session IDs by simply incrementing static numbers.

- IP Address is a user agent's IP address[1] used for this login attempt.
- User Agent is a user agent[2] used for this login attempt.

4. Click Close.

To view all users' login history, select Security > User Login History.

User login history records are shown page by page, with the maximum number of records per page configured in the users preferences.



To filter or search the user login history by one or more criteria, specify the following settings and click Search:

- User account's name or part of a name.
- Event Type.

  A user login event type can be one of the following:

  - Account Locked Temp.
  - Login Failure.
  - Log Out.
  - Login Success.
  - Password Reset.
  - Session Expiry.
- Space.

---

[1]IP address is a unique string of numbers separated by full stops that identifies each computer using the Internet Protocol to communicate over a network.
[2]User agent is software (a software agent) that is acting on behalf of a user. One common use of the term refers to a web browser that retrieves, renders and facilitates end user interaction with Web content.

- IP Address is a user agent's IP address[1] used for this login attempt.
- Start Date.
- End Date.

Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to view user accounts.

[1]IP address is a unique string of numbers separated by full stops that identifies each computer using the Internet Protocol to communicate over a network.

User Visibility Rules for Restricted Content

Manager implements rules to ensure that users of the Workspaces portal can't access restricted content, such as organizations, groups, users, forms, and form categories configurations. The rules are summarized below:

1. A global user, with the global access enabled, will see all organizations, groups, users, forms, and form categories information, if they are assigned to the Workspaces portal.
2. A user belonging to an organization without any form categories won't see any categories.
3. A user belonging to many organizations will see all the categories associated with these organizations.
4. A users without an organization won't see any organizations, groups, users, forms, and form categories information.
5. A user can see organizations they are assigned to and should be part of the Workspaces portal.
6. A user can see forms available to the organizations where the user has access to and is part of the Workspaces portal.
7. A user can see users available to the organizations where the user has access to and is part of the Workspaces portal.

Manager also maintains a separate set of rules for restricting:

- User access to collaboration jobs
- User access to transactions

The Workspaces APIs use these rules to filter the category, form, group, org, user, and transaction query responses.

Next, learn more about user transaction security.

User Visibility Rules for Collaboration Jobs

Manager allows a user to view collaboration jobs in the Workspaces portal and retrieve them using the APIs, if the following rules apply:

1. A user must have the Collaboration Job View permission for the form space to see collaboration jobs.
   - Only collaboration jobs with the In Progress status are further selected.
   - Only the collaboration jobs that are in the same form space as the user are selected. If this is not defined, this rule is not applied.
2. Manager further refines rules to select a subset of collaboration jobs from the collection above:
   - The collaboration jobs have no groups assigned or if there are groups assigned, the user is in one of those groups and
   - The user does not have global access then the user must be in a organization that the collaboration job is in.

To explain how these rules work, we will list several users, collaboration jobs, and their combinations showing whether a user can view these collaboration jobs or not.

The first table shows users with the access to four organizations, some of which have global access, and two groups.

	Org-1	Org-2	Org-3	Org-4	Global Access	Group-1	Group-2	Permission
User-1								Collaboration Job View
User-2								Collaboration Job View
User-3								Collaboration Job View

User-4					Collaboration Job View
User-5					Collaboration Job View
User-6					Collaboration Job View
User-7					Collaboration Job View
User-8					Collaboration Job View
User-9					

The second table shows whether a user can see transactions or not. collaboration jobs belonging to one of the two organizations and two groups.

	Org1	Org2	Group1	Group2
Job1				
Job2				
Job3				
Job4				
Job5				

NOTE

A collaboration job is limited to one organization but can have any number of groups.

This table shows whether a user can see collaboration jobs or not.

	Job1	Job2	Job3	Job4	Job5
User1					
User2					
User3					
User4					
User5					
User6					
User7					
User8					
User9					

This table helps you understand why users might not see some or any collaboration jobs in their work spaces, so you can review the user role, group, and permission configurations to resolve the access issues. We have illustrated this with the following examples:

- The User 1 starts working on the Job 1 collaboration job, but can't continue with it. The User 3 has right permissions to work on the Job 1, so the User 3 can complete with the work the User 1 started.
- The User 1 can't see the Job 3 collaboration job, event though it is in the same Org1 organization, because it is assigned to the Group 3.
- The User 9 can't see any collaboration jobs, because this user doesn't have the Collaboration Job View permission.

Next, learn how to view roles.

User Visibility Rules for Transactions

Manager doesn't maintain relationships between users and organizations to determine which transactions a user can see when using the Workspaces portal or the APIs. Instead, Manager defines a set of rules to filter out transactions that a user can see:

1. Manager applies the Help Desk View permission to select an initial collection of transactions. A user can see the whole or part of this collection depending on the following conditions:
    - A user with the Help Desk View permission can see transactions with a group, user or both.
    - A user without the Help Desk View permission by default can only see transactions if they are allocated to it or if they are in the same group.
    - A user with the Help Desk Authenticated Edit permission can see other users' authenticated transactions.
2. Manager applies more rules to select a subset of transactions from the collection above:
    - A global user, with the global access enabled, can see all above transactions.
    - Otherwise
        - A user with a limited number of organizations can see transactions from above for those organizations only.
        - A user which is not global and has no organization can see only own transactions from above.

To explain how these rules work, we will list several users, transactions and their combinations showing whether a user can view these transactions or not.

The first table shows users with the access to organizations, some of which have global access, groups and the Workspaces permissions assigned to them.

	Org-1	Org-2	Org-3	Org-4	Org-5	Org-6	Global Access	Group1	Group2	Group3	Group4	Permission
User-												

1		
User-2		
User-3		
User-4		Help Desk Authenticated Edit
User-5		Help Desk View
User-6		
User-7		

The second table shows transactions belonging to an organization, a group and having a user assigned.

	Org-1	Org-2	Org-6	Group1	Group2	Group3	Group4	User-1	User-2	User-6	User-7

Anon Txn
User1 Txn
User2 Txn
Group1 Txn
Group2 Txn
Group3 Txn
Group4 Txn

This table shows whether a user can see transactions or not.

	Anon Txn	User1 Txn	User2 Txn	Group1 Txn	Group2 Txn	Group3 Txn	Group4 Txn
User1							
User2							
User3							
User4							

User5
User6
User7

This table helps you understand why some users don't see some or any transactions in their space, so you can review the user role, group and permission configurations to resolve the access issues. We have illustrated this with the following few examples:

- The User 1 creates an anonymous transaction by filling a form, but can't continue with it, so the User 1 calls the help desk. The User 5 is a help desk user with the Help Desk View permission, so the User 5 can see the transaction and therefore can help the User 1 to complete it.
- The User 7 can see the Group4 Txn, because it's assigned to this user.
- The User 3 can't see any transactions, because there are no transactions belonging to Org3 organization, and the user doesn't have a global access and doesn't belong to any group.

Next, learn how to view roles.

Roles Overview

A role in Manager contains a set of permissions that allow a user to perform specific functions within a form space or module. Manager has the following set of standard roles automatically created and configured during its first start-up. You can use the standard roles, customize the standard roles, or create new roles.

Root Administrator

This is the most powerful role in Manager. Root administrators are assigned all permissions related to the Manager console and can access all restricted content. Critical operations, such as removing organizations and editing deployment properties, should be accessible to only these users. Root administrators can create new roles and customize any existing ones.

Developer / System Manager

These two roles have identical permissions. Developer/System Manager users have a wide range of permissions available to them to manage Manager configurations. Users in these roles can edit organizations, forms, and all related entities. Developers and System Managers are also able to view requests, view submissions as well as view a wide range of system configuration settings. These roles are extremely powerful but cannot access submission data and user accounts and are not able to modify roles.

Operations

Organization operation administrators can view submissions (included submitted XML data) and reports as well as various logs. Organization operation administrators are given permission to handle support requests submitted by space users.

However, they cannot access forms and associated configurations, nor can they access the details for user accounts stored in Manager.

Organization User Manager

Organization user administrators can create and configure users for their organization. This is a role that might be used in addition to other roles to give existing administrators access to user accounts, though it can also be used as a stand-alone role.

System Support

The system support role can be used by the support staff for debugging purposes. By default, this role has extensive read-only permissions to aid in debugging. However, XML data submitted by users cannot be viewed.

Transaction Data Access

This role provides access to transaction data entered by users and allows administrators to edit this data and make submission status changes. As this role provides access to sensitive information, this role should be assigned carefully.

Next, learn how to view roles.

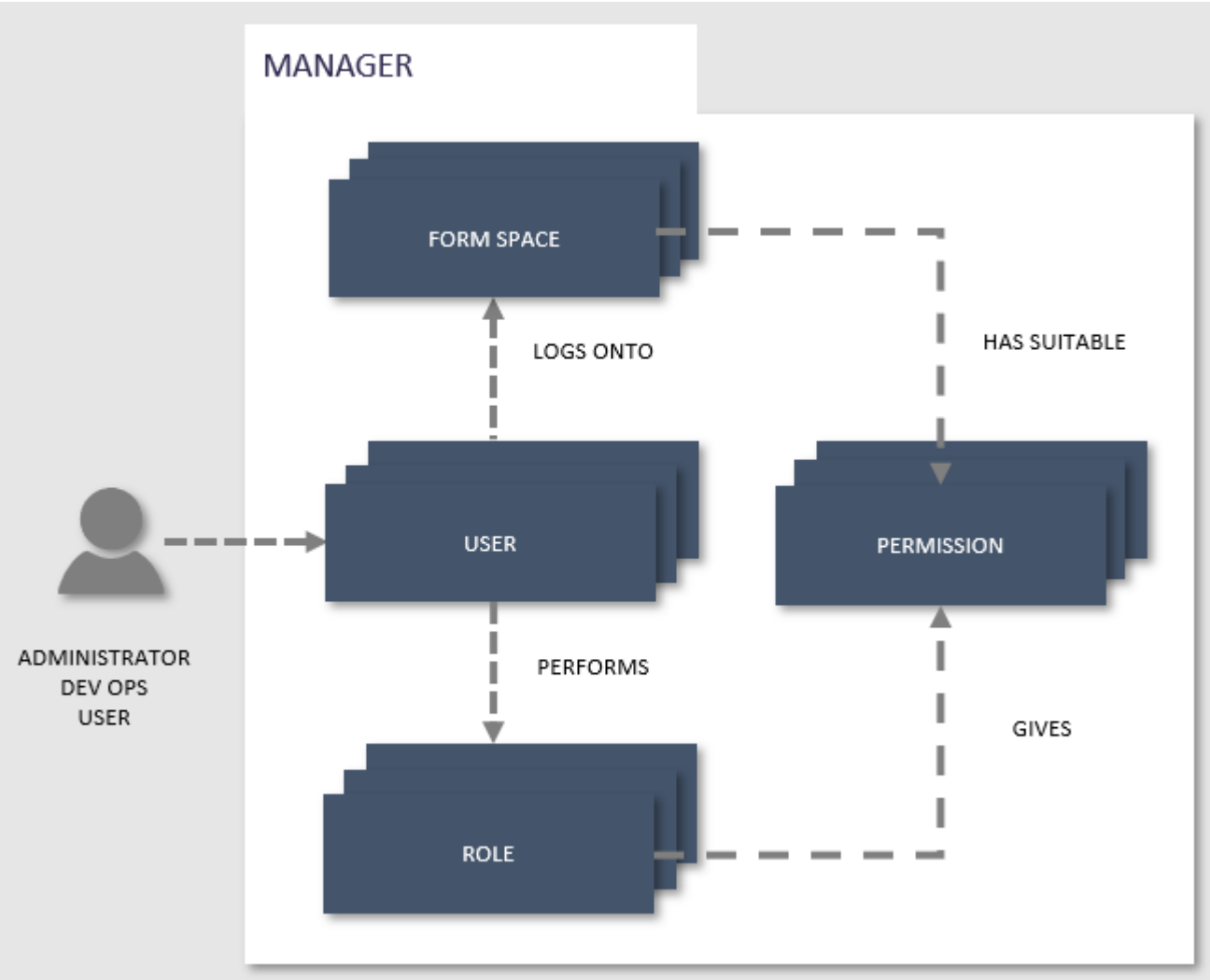
Standard Roles

Manager provides roles and permissions to control user access to form spaces and modules. You can use this functionality to grant users specific access that matches the functions they are permitted to perform within each space or module. Manager checks what privileges a user has when logged into a form space or module. The roles granted to a user determine these privileges and controls what they can see and do within that space or module.

Before either adding extra roles or granting roles to your users, it is important to understand the following concepts:

- A user, via their user account, can be granted one or more roles.
- A role sanctions a group of permissions related to a function that is to be performed within a space or module. That is, a role is a group of permissions that is relevant only to the intended space or module.
- Each form space or module have a list of valid permissions available to assign to a role. That is, the role may only grant the types of access relevant to that space or module.
- Each role is mostly for one form space or module, but can be for more than one.
- Each permission is normally for one form space or module, but can be for more than one.

The diagram below depicts the conceptual data model for users logging into spaces or modules and their required roles and permissions.



Out of the box, Manager comes with the standard set of roles. Each role is already configured with its permissions for each standard form space and module. It is possible to alter the standard roles as well as add new ones. However, it is recommended to preserve the standard set and, only when essential, add new roles. It is also important to understand what privileges each of the roles provide before assigning them to your users.

Manager has the following standard roles:

Role	Description
Administrator	This is the most privileged role. There should be minimal need to use this role once the system has been installed, especially not in Production. The exception would be for the duration needed to perform a deployment where organizational or user roles need to be altered or imported. Regular form upgrades can be done with the System Manager role.

Avoka Exchange	Allows a user to review and install Exchange components.
Developer	Allows users to develop and manage applications. This role is identical to the System Manager role but has been retained for historical reasons. The Developer role should not be used in production environments. You should use the System Manager role for system configuration and installation in production.
Maestro Administrator	Allows a user to use the developer functions, plus create and delete projects, import and export files and libraries.
Maestro Developer	Allows a user to use most of Maestro and the Maestro editor functionality. This is the default role, which is granted to both Form Builders and Template Designers.
Maestro SCM Design	Allows a user to use Maestro SCM with Designs.
Maestro SCM Org Library	Allows a user to use Maestro SCM with Organization Libraries.
Maestro SCM Project	Allows a user to use Maestro SCM with Projects that include project's branches and tags.
Maestro Site Administrator	Allows a user to administer a Maestro instance at a global level.
Maguire Staff	Allows a user to use Journey Space form space.
Manager	Allows users to perform operations, monitoring and management of submission transactions and collaboration jobs as a Manager. This role has limited access to customer submission data, such as receipts.
Operations	Allows a user to monitor and manage form transactions
Organization Administrator	Allows a user to test Users and Permissions.
Organization User Man-	Allows a user to manage user accounts for their organization.

ager	
REST Delivery	Allows a user to perform submission delivery by invoking REST Delivery Service.
Security Manager Role	Allows a user to test Security Manager Permission.
Service Test Administrator	Allows a user to test Users and Permission service Definition.
ServiceUser	Allows scheduled Groovy services run as ServiceUser, which defaults to a admin level role. This role can be limited to enhance the security around these services, if required. | 21.11 This feature was introduced in 21.11.
System Manager	This is a role for users who are almost as privileged as an administrator, except they cannot view submission data or create users and roles. This role should only be used where extensive system access is required.
System Support	This is a role for users to be able to do basic troubleshooting and triage of issues. This is the least privileged role in Manager.
Test Role	Allows a user to run test cases.
Transact Insights	Allows a user to use Journey Analytics.
Transact Insights Administrator	Allows a user to change global settings for Journey Analytics.
Transaction Data Access	Allows users to be able to view the form submission XML and receipt data, which is sensitive personal data that needs to be protected.
Journey Space Staff	Allows Journey Space users to use the portal
Workspaces Staff	For business Journey Workspaces Overview users.
Workspaces Api Test Role	Allows a user to test Workspaces API Test.

NOTE

To see this list in Manager, see view roles.

The roles depend on the functions that each team member performs. The following table is an example of how your team can be set up with specific roles:

Teams	Development	UAT	Staging	Production
Implementation				
Form Builder	System Manager. Transaction Data Access	System Manager. Transaction Data Access	No Access	No Access
Manager Developer Tech Lead	Administrator	Administrator	Administrator Transaction Data Access	No Access
Manager Tester	Operations	Operations	Operations	Operations
Operations				
System Administrator	Administrator	Administrator	Administrator	System Manager. Organization User Manager
Form User Support Operations	Operations. System Manager	Operations. System Manager	Operations. System Manager	Operations. System Manager
Temenos Teams				
Support Engineers. Tech Lead	Administrator	Administrator	System Manager	System Operator
Development	Developer	Developer	No Access	No Access

Teams

Next, learn about permissions.

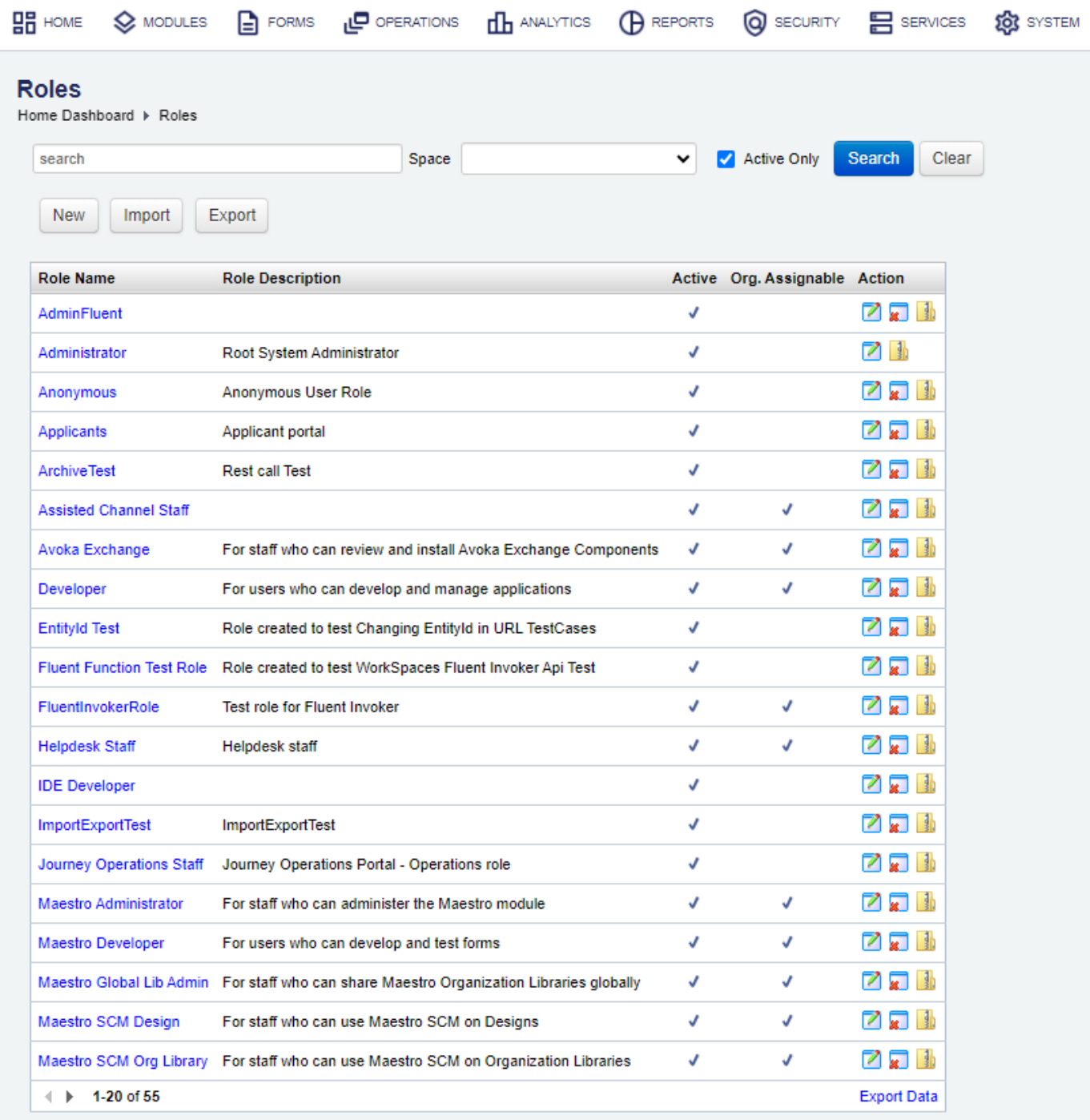
View Roles

Manager allows you to view available roles, so they you can grant or revoke a user's roles and privileges.

> NOTE
>
> You must be a Journey Manager global administrator to view all existing roles.

To view all configured roles, select Security > Roles.

Roles are shown page by page, with the maximum number of records per page configured via the user's preferences.



The list displays the following details:

- Role Name: a name of a role.
- Role Description: a meaningful description of a role.
- Active: whether a role is active or disabled.
- Org. Assignable: whether an organization administrators can assign this role to user accounts.

To filter or search the roles by one or more criteria, specify the following settings and click Search:

- Role's name or part of a name
- Space
- Active Only

Click Clear to reset the search criteria.

You can also perform the following actions:

- Create and configure a role
- Remove a role
- Export roles
- Import roles

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

> NOTE
>
> Organization administrators can view, edit, import and export only organization assignable roles, which are any organizations having Org. Assignable column ticked.

Next, learn how to assign permissions to a role.

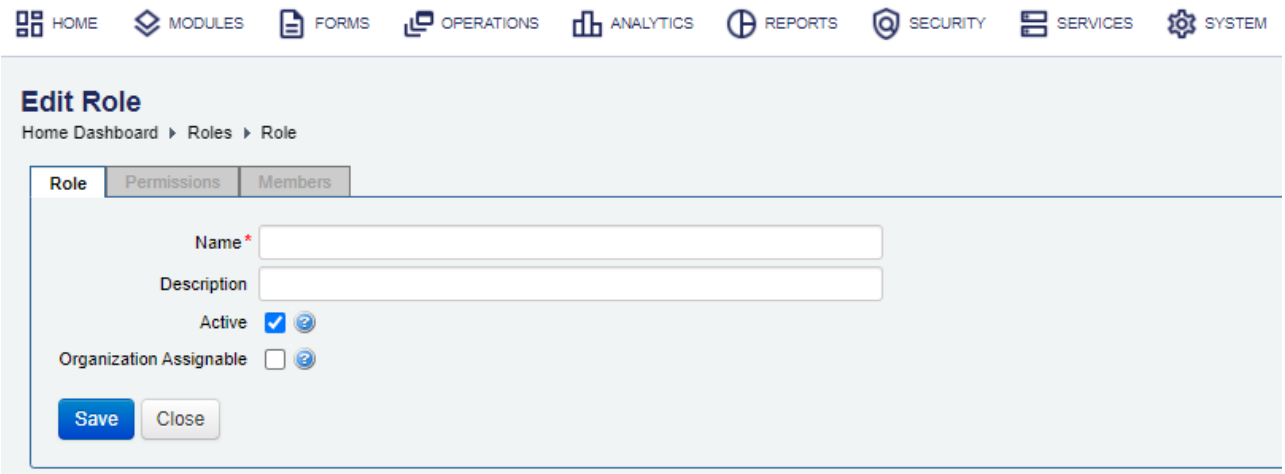
Create a Role

Manager allows you to create new roles. You can use this functionality to:

- Accommodate new customer requirements.
- Adjust existing roles to the customer organization hierarchy.

To create a new role:

1. Select Security > Roles.
2. Click New.



3. Enter a unique name for the new role in the Name field.
4. Enter a meaningful description if a role in the Description field.
5. The Active checkbox is selected by default, but you can clear it to disable this role for the time being.

   NOTE

   Inactive roles can't be assigned to user accounts until they have been reactivated.

6. Select the Organization Assignable checkbox to permit organization administrators to assign this role to user accounts.

NOTE

If you want only global administrators to be able to assign this role to user accounts, don't select Organization Assignable, so an organization administrator won't see it in their list of roles. If you don't select it, even organization administrators with the permission to assign roles will be unable to see or choose the role. For example, a global administrator role with all privileges will most likely not be organization assignable.

7. Click Save to update the changes.
8. Assign permissions to the new role.

NOTE

The Submission Viewer role can only be assigned to a user account by a global administrator.

Next, you will need to assign permissions to a role.

Assign Users to a Role

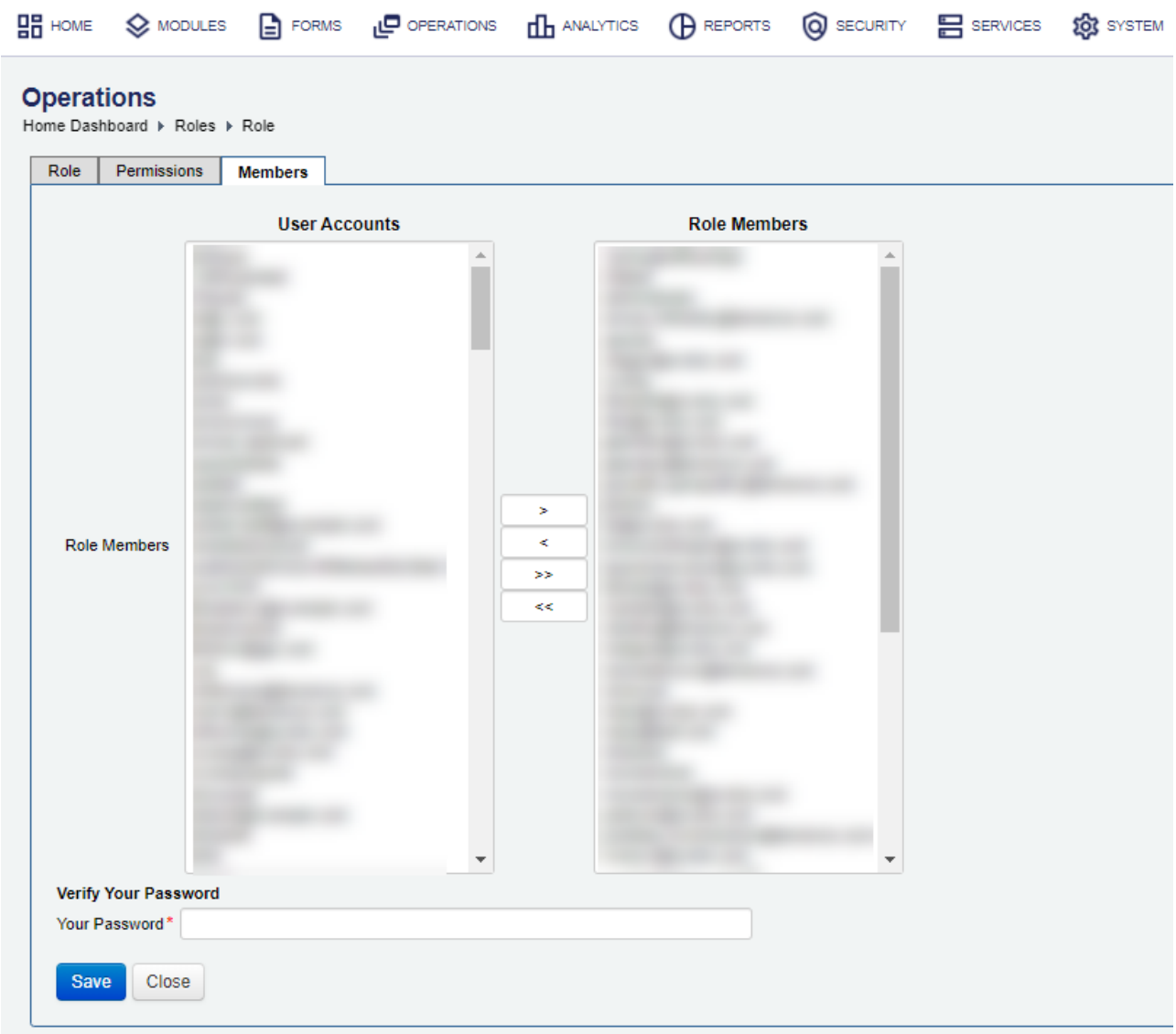
Manager allows you to assign user accounts to a role. These users are known as role members. You can also assign role members from their individual user accounts.

> NOTE
>
> You must be a Journey Manager administrator to create or configure this functionality.

After you have created a role, you can assign users to it. As an ongoing maintenance, you can assign new members or remove existing members for an existing role.

To assign users to a role:

1. Select Security > Roles.
2. Locate a role and click the Members tab to view role members. The Role Members column displays users that are currently assigned the selected role and the User Accounts column displays users you can assign to the selected role.



3. Select one or more users from User Accounts to add them to the role or remove them from it.

> NOTE
>
> You can use the Shift and Ctrl keys to select multiple permissions on Windows PC.

4. Click > to move the selected user accounts from User Accounts to Role Members.
5. To revoke any assigned user accounts, select the user accounts from Role Members and click <.
6. To assign all user accounts, click >>.
7. To revoke all user accounts, click <<.
8. Provide your password in the Your Password field, for security reasons.

9. Click Save to update the changes.

NOTE

All user accounts assigned to a role are loaded when the user logs in. Any changes will be visible to users as soon as they log out and then log back into the space.

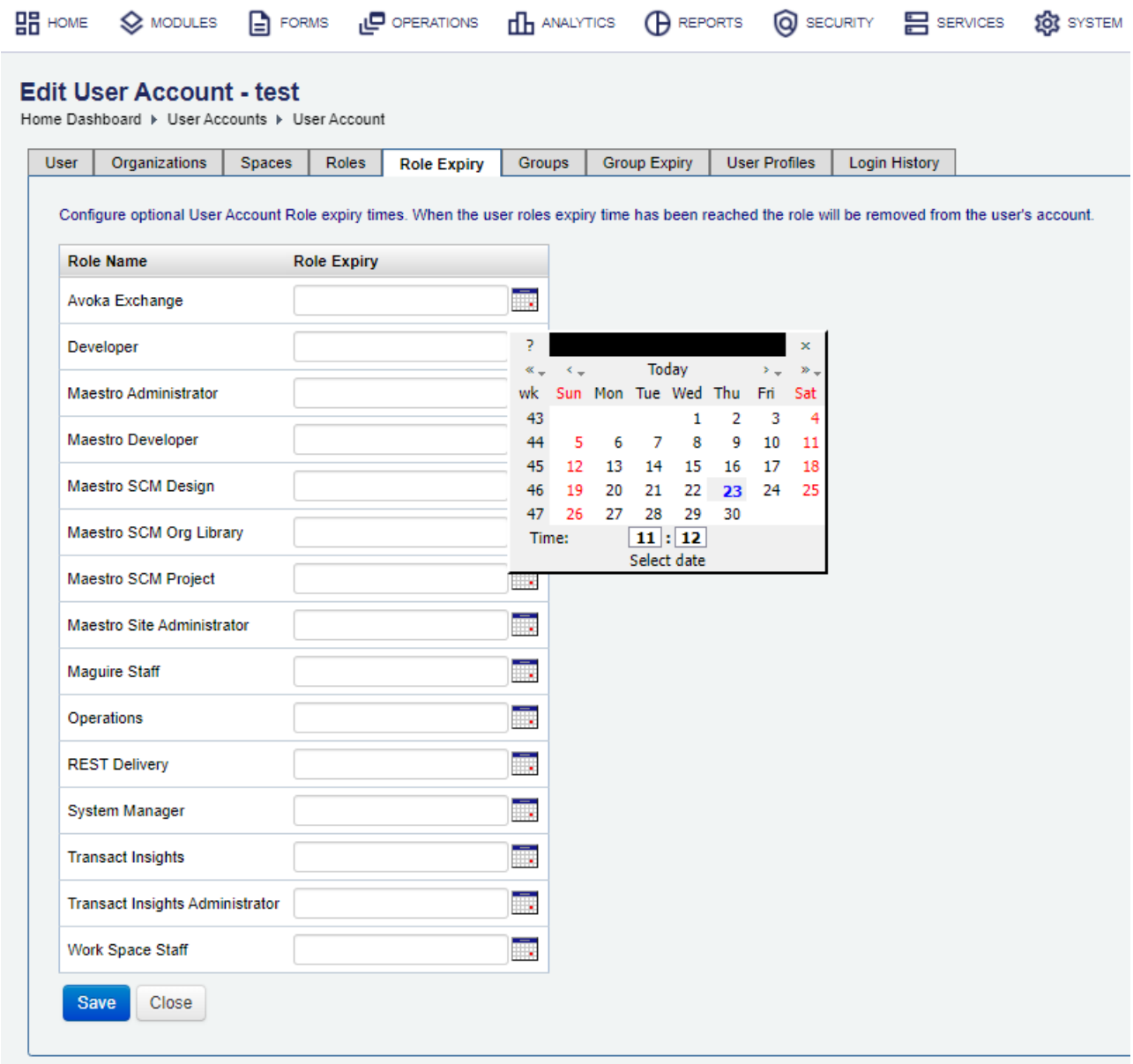
Next, learn how configure role expiry.

Configure Role Expiry

Manager allows you to enforce an expiry time to a user's role. You can use this functionality to temporarily assign a role to a user account to allow a support team to assist in debugging a problem.

To enforce a time limit on a user's role:

1. Select Security > User Accounts.
2. Locate a user account and click the Role Expiry tab.



All roles assigned to this user account are listed under Role Name column. Any temporary role assignments show an expiry date in the Role Expiry column.

3. Click Calendar in the Role Expiry column.
4. Select a date and time for when this user's role membership must expire in a Calendar pop up. You can select the time first, as the pop up automatically closes on selection of a date.

   NOTE

   To make a user's role membership permanent again, remove the expiry date and time from the Role Expiry column.

5. Click Save to update the changes. When the expiry time arrives, the Security Policy Manager background job removes this user's role membership from the user account.

Next, learn how to create a user account.

Remove a Role

If a role is not required any more, Manager provides you with two options:

- Deactivate a role.
- Remove a role.

NOTE

In general, we recommend deactivating a role rather than removing it.

Deactivate a Role

When you deactivate a role, Manager does the following:

- The role details are preserved, so the role can be reactivated if needed. On reactivation, a role can once again be assigned to a user account.
- User accounts remain assigned to the role, but its permissions are revoked.

To deactivate a role:

1. Select Security > Roles.
2. Locate the role and click Edit.
3. Clear the Active checkbox.
4. Click Save to update the changes.

NOTE

To reactivate the role, select its Active checkbox.

Remove a Role

When you remove a role, Manager irrevocably removes all data associated with the role, so it can not be rolled back.

NOTE

You can't remove the system-generated Administrator role.

To remove a role:

1. Select Security > Roles.
2. Locate the role and click Remove.
3. Click OK to confirm.

Next, learn how to create roles.

Export Roles

Manager allows administrators to export roles from one environment and import them into another. You can use the export functionality to:

- Back up or archive roles.
- Re-create or migrate roles in another Manager.
- Promote one or several roles to other environments.

To export one or several roles:

1. Select Security > Roles.
2. Click Export Archive to export one role or click Export to export all roles.

Manager exports the role into a group archive ZIP file, which will be downloaded to your PC. The ZIP file contains a `role-config.xml` file with all roles' definitions and permissions.

> NOTE
>
> To avoid any lockout issues, the Administrator role is not included in the export.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to import roles.

Import Roles

Manager allows you to import a role as a ZIP file. You can use the import functionality to:

- Roll back one or several roles.
- Re-create one or several roles in another Manager.
- Promote one or several roles to other environments.

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

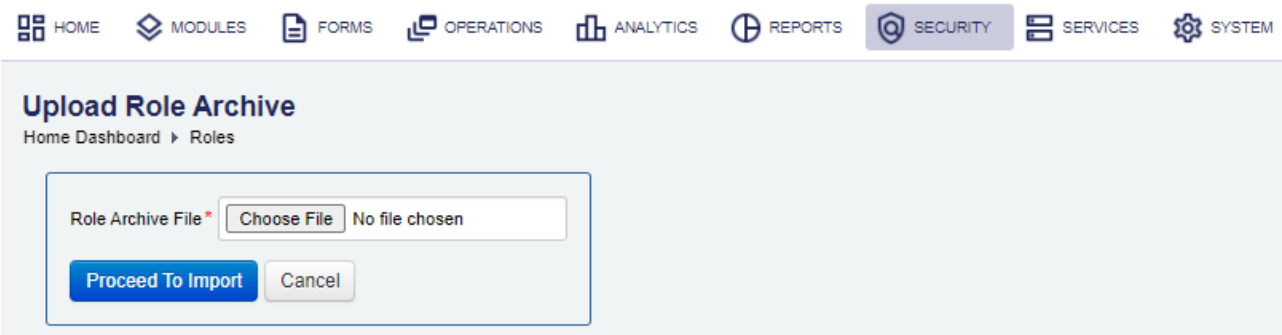
You can import a role using one of following options:

- UI
- SDK
- API

Import Using UI

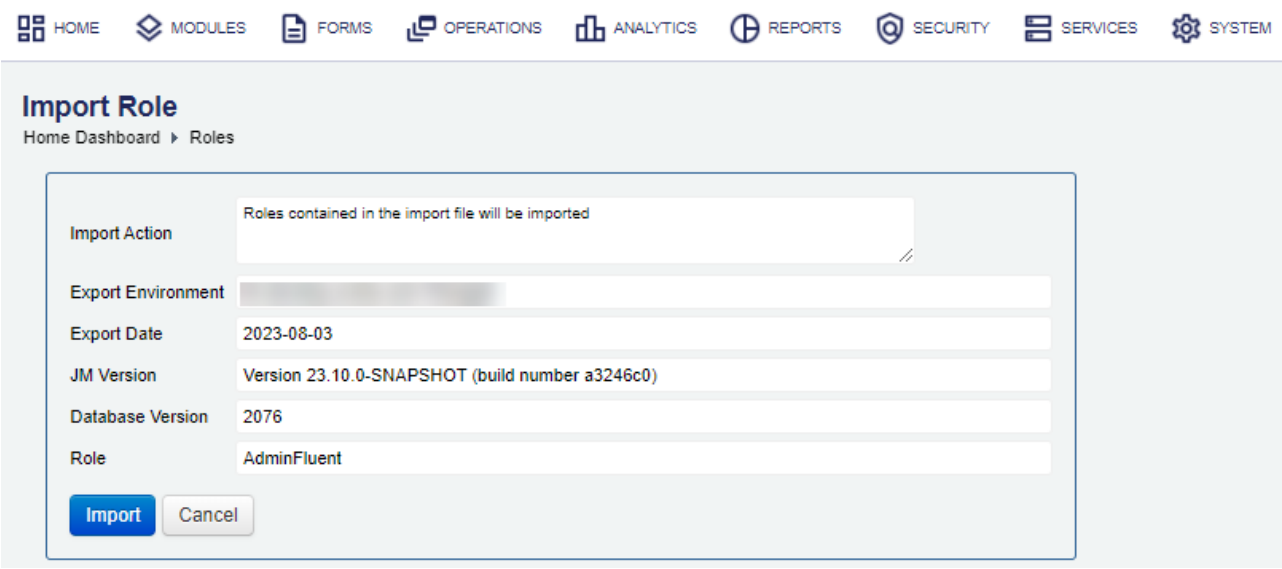
To import a role:

1. Select Security > Roles and click Import.

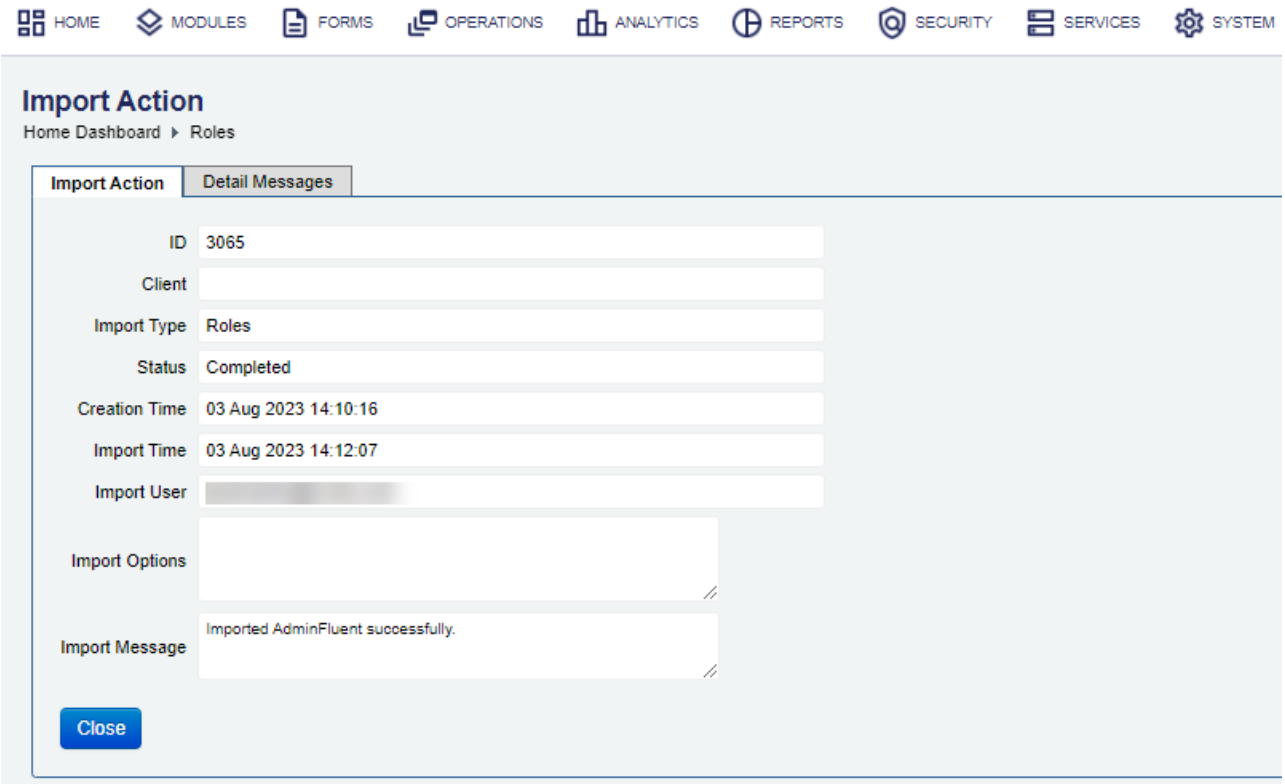


2. Click Choose File and locate a role archive ZIP file, which has been previously exported.
3. Click Proceed To import.

4. Review the Import Options.



5. Click Import to complete the process. The expected result page is shown below.



6. Select the Detail Messages tab to see more information about the import. This tab pops up open if the import operation has any issues you may need to check and resolve.
7. Press Close and check the list of roles to verify the imported roles have been added.

NOTE

You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

Import Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:archive-deploy goal to import a role.

Import Using API

The Journey Manager REST API provides the endpoint URL to import a role.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s):://<JM_SERVER>/manager/secure/rest/archive/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as Base64[1].

`--form ‘archiveFile=@“/Users/my-role.zip”’` - a path to an archive file to be imported.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

Example

```
curl --location --request PUT ‘https://<JM_SERVER>/manager/secure/rest/archive/v1/’
\
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
--form ‘archiveFile=@“/Users/my-role.zip”’
```

Next, learn how to migrate roles.

[1]Base64 is a binary to a text encoding scheme that represents binary data in an American Standard Code for Information Interchange (ASCII) string format.

Migrate Roles

Manager provides a mechanism to migrate roles from one environment to another. This allows you to develop and test roles in different environments before migrating them over to the production environment. It also saves time to set up the roles in new Manager environments.

NOTE

Role migration doesn't include user accounts and user data associated with the role.

NOTE

It is recommended that only global administrators migrate roles, otherwise only organization assignable roles are included.

To migrate one or several roles:

1. Export roles.
2. Import roles to another Manager environment.

NOTE

You can use the same process to perform a role rollback.

Next, learn how to assign permissions to a role.

Groups Overview

Manager allows you to create and use groups to implement a fine-grained control of user access to your environment. You can use this functionality to:

- Control user access to forms and their resources
- Add users to groups to enforce required user functions and responsibilities
- Extend the privileges defined through roles and their permissions
- Create working groups to allow users to collaborate to review, approve or reject application forms. For example, a reviewer who approves or rejects an application form must belong to a reviewer form group.

Manager uses the following pre-defined group types to create new groups:

- Alert groups - to notify and alert users about certain events
- Form groups - to restrict user access to forms or tasks that are associated with this group
- Job groups - to restrict access to job coordinators

Manager can use dynamically loaded groups when users are managed externally using LDAP[1] or SSO[2] authentication. Manager loads the user's group associations at login time. For LDAP users, groups are usually defined as a distinguished name, such as `"CN=mygroup,OU=Distribution Groups,OU=MyBusiness,DC=mycompany"`. Manager parses these group name, so `"CN=mygroup"becomes "mygroup"`. For SSO users, the group names can be passed in while creating a SSO token.

Next, learn how to view groups.

---

[1]Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol network.

[2]Single sign-on (SSO) is a property of access control of multiple related, yet independent, software systems. With this property, a user logs in with a single ID and password to gain access to any of several related systems.

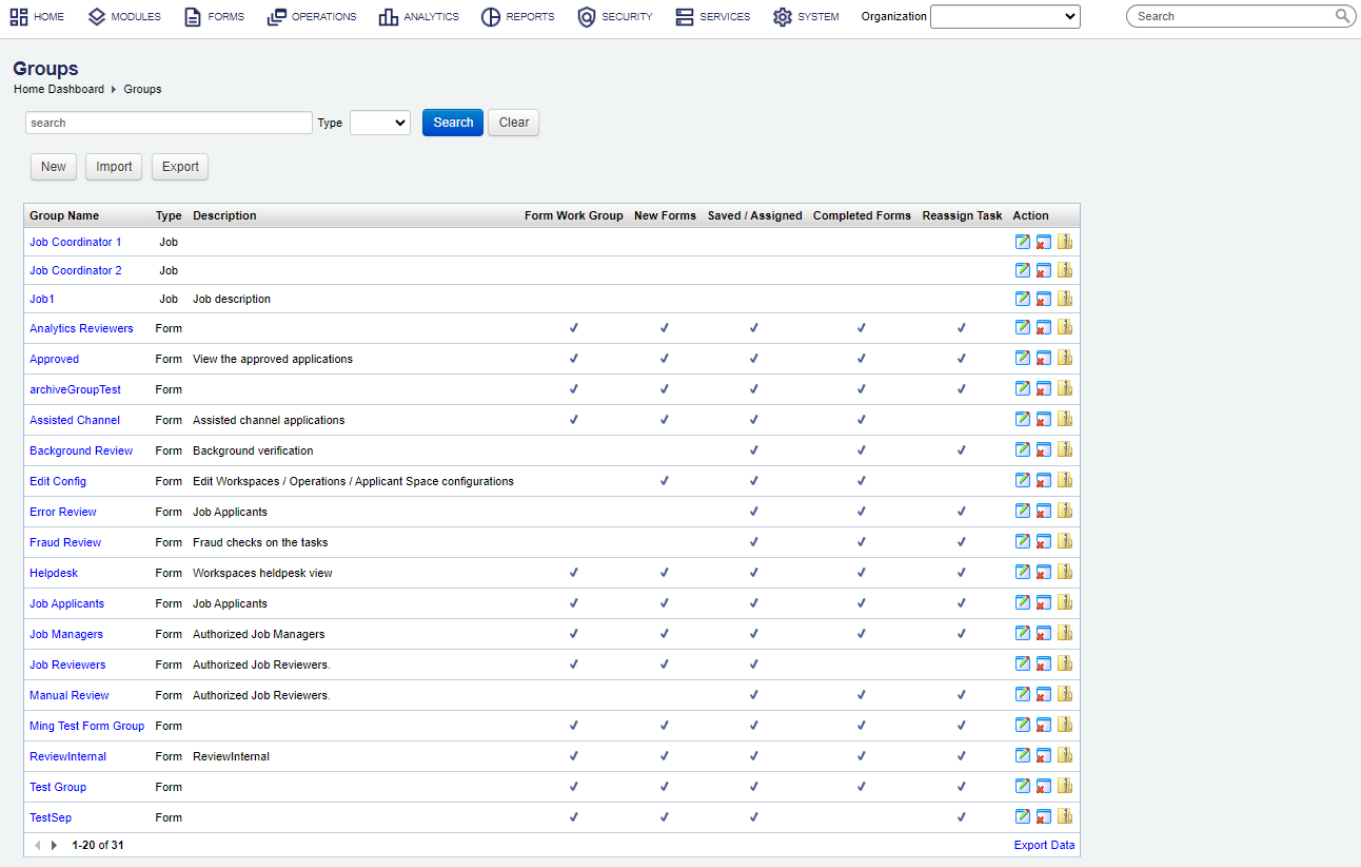
View Groups

Manager allows you to view the list of available groups.

> NOTE
>
> You must be an Administrator to view the groups.

To view all configured groups, select Security > Groups.

Groups are shown page by page, with the maximum number of records per page configured in the users preferences.



Group Name	Type	Description	Form Work Group	New Forms	Saved / Assigned	Completed Forms	Reassign Task	Action
Job Coordinator 1	Job							
Job Coordinator 2	Job							
Job1	Job	Job description						
Analytics Reviewers	Form		✓	✓	✓	✓	✓	
Approved	Form	View the approved applications	✓	✓	✓	✓	✓	
archiveGroupTest	Form		✓	✓	✓	✓	✓	
Assisted Channel	Form	Assisted channel applications	✓	✓	✓	✓		
Background Review	Form	Background verification			✓	✓	✓	
Edit Config	Form	Edit Workspaces / Operations / Applicant Space configurations		✓	✓	✓		
Error Review	Form	Job Applicants			✓	✓	✓	
Fraud Review	Form	Fraud checks on the tasks			✓	✓	✓	
Helpdesk	Form	Workspaces heldpesk view	✓	✓	✓	✓	✓	
Job Applicants	Form	Job Applicants	✓	✓	✓	✓	✓	
Job Managers	Form	Authorized Job Managers	✓	✓	✓	✓	✓	
Job Reviewers	Form	Authorized Job Reviewers.	✓	✓	✓			
Manual Review	Form	Authorized Job Reviewers.			✓	✓	✓	
Ming Test Form Group	Form		✓	✓	✓	✓	✓	
ReviewInternal	Form	ReviewInternal	✓	✓	✓	✓	✓	
Test Group	Form		✓	✓	✓	✓	✓	
TestSep	Form		✓	✓	✓		✓	

The list displays the following details:

- Group Name: a name of a group.
- Type: a type of a group, which can be one of the following:
  - Alert- notifies and alert users about certain events. A user must be assigned to a group of this group type.
  - Form- restricts user access to forms or tasks that are associated with this group.

    - Job - restrict access to job coordinators when overseeing jobs that they are responsible for.
- Description: a description of a group.
- Form Work Group: whether users of this group can share the form with work group members.
- New Forms: whether users of this group can open new forms or not.
- Saved / Assigned: whether users of this group can open a saved and assigned forms and tasks or not.
- Completed Forms: whether users of this group can open submissions of completed forms.
- Reassigned Forms: whether users of this group can reassign tasks to other users.

To filter or search the groups by one or more criteria, specify the following settings and click Search:

- Group's name or part of a name.
- Type of a group

Click Clear to reset the search criteria.

You can also perform the following actions:

- Create and edit a group
- Remove a group
- Export a group
- Import a group

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to assign groups to a user.

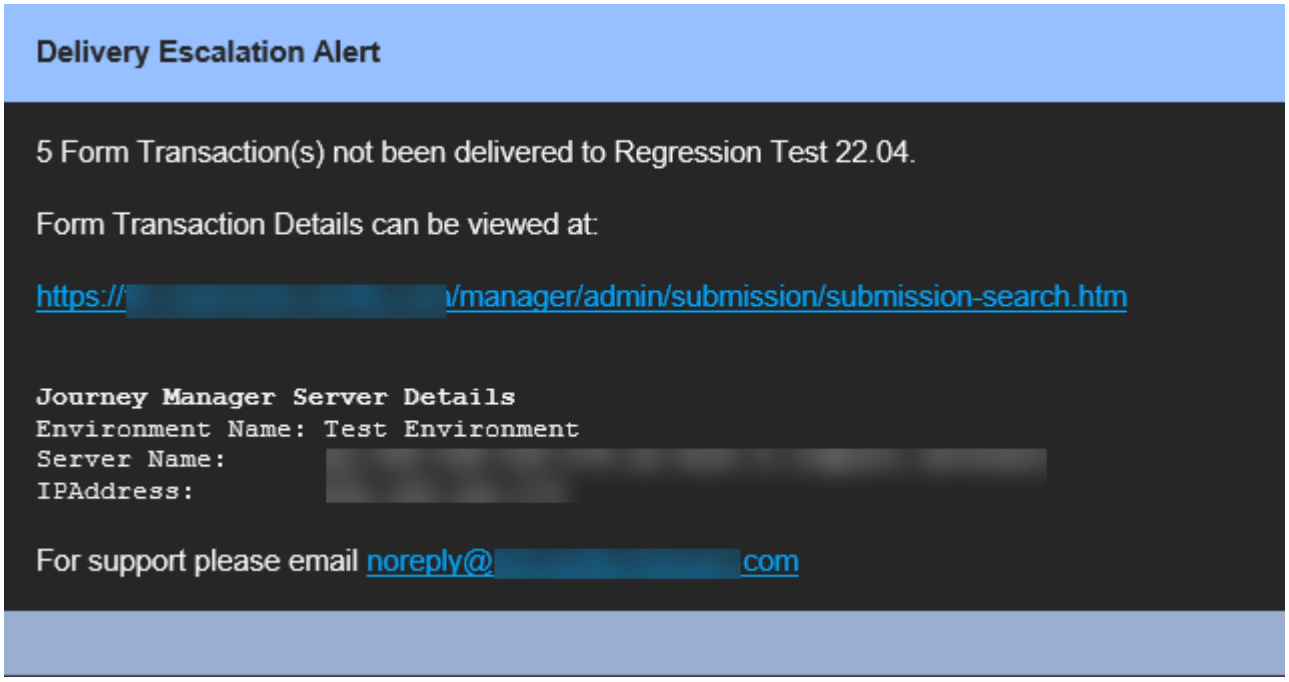
Alert Groups

Manager uses the following predefined alert groups to notify relevant users, such as IT operations, of various important events and errors, which require their attention. Users are notified by emails. Each email message contains one event for each alert group a user is subscribed to. Therefore, this might generate a significant amount of emails if users are assigned to many alert groups. In this case, the users should review the assigned groups and revoke one they don't need.

Receive Delivery Escalation Alerts

The submissions with secure email delivery have not been acknowledged after a certain period of time. Scheduled Job is Delivery Escalation, Context is Organization Users and Global Users. Users who receive alerts when a transaction delivered by secure email has not been acknowledged yet.

The out of the box delivery escalation alert message is shown below:



Receive Outstanding Payment Alerts

There is a submission with an outstanding payment for their organization. This group is relevant only if administrators want to monitor submissions that are abandoned at the payment stage and contact an applicant to provide support. Event is Scheduled Job and Payment Reminder, Context is Users in group and their scope including global users. Event is Fraud and Duplicate

Payment Detection, Context is Transaction. Users who receive email alerts reporting transactions that appear to have been abandoned at the payment stage.

Receive Promotion Alerts

Ends alert emails when a form is promoted. Event is Form promoted, Context is Form Organization Users and Global Users. User to notify when a form belonging to their organization is promoted.

Receive System Alerts

The system monitoring jobs find what important system activities are impacted. Event is Internal Exception Raised, Context is Global. Users who will be notified when the System Monitoring job finds that important system jobs are impacted.

Receive Submission Updates

A submission processing status has been updated. This group is relevant only to form users, but not administrators. Event is WebService Call and FormsServerWebService::UpdateProcessingStatus, Context is Transaction User. Users who will be notified when the processing status for a submission they have made has been updated. This group is relevant only to form space users, but not administrators.

Receive System Alerts - Database Integrity

Database Integrity reports that are generated by the Database Integrity Checker job and sent to all subscribed users. | 23.04 This feature was introduced in 23.04

Receive System Alerts - Delivery

Submission delivery fails. Event = Exception during delivery, Context = Transaction Organization Users but NOT Global User who are not part of the Organization. Users who will be notified when submission delivery fails.

Receive System Alerts - Receipts

Submission receipt generation fails. Event is Exception during receipt generation, Context is Transaction Organization Users but not a Global User who are not part of the organization. Users who will be notified when submission receipt generation fails.

Receive User Account Creation Alerts

A user has self-registered, creating a user account on a form space. Event is Account Creation, Context is All Users in Group but not Organization Specific. Users who will be notified when a user has applied for an account on a user space.

Custom Alerts

If you don't find a predefined alert that satisfies your needs, you can always create a custom one.

Alternatively, you can use system event publisher to send external notification events by publishing them to Azure or AWS SQS queues. For example, you can configure system event publisher to publish a notification each time there is a problem with the address component in a form and the problem is being recorded in the error log. In this case, you need to make sure that the error you want to log is one that is configurable in system event publisher.

> NOTE
>
> You can't change or delete the predefined alerts groups.

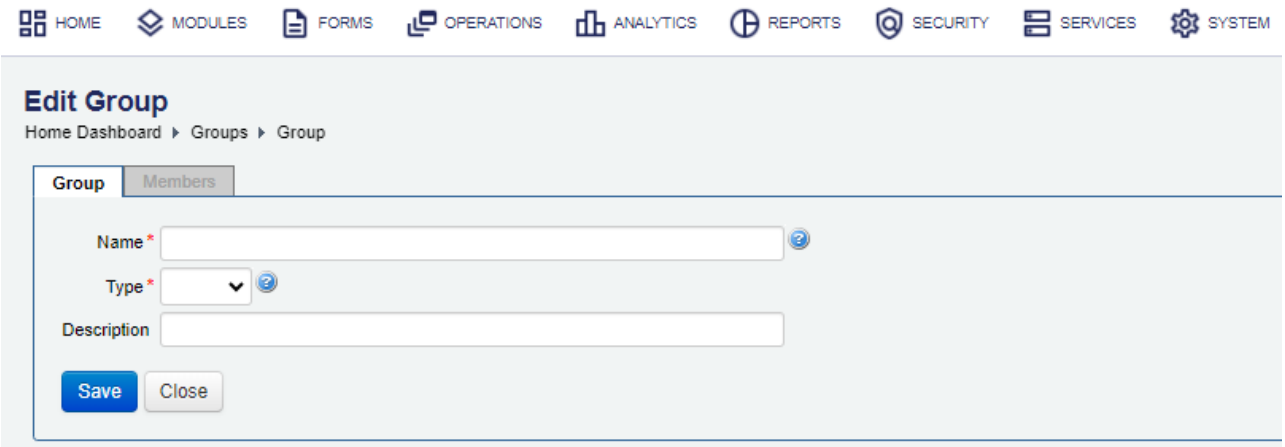
Next, learn about form groups.

Create a Group

Manager allows you to create a group to a set of users with certain privileges, such as form or job access.

To create a group:

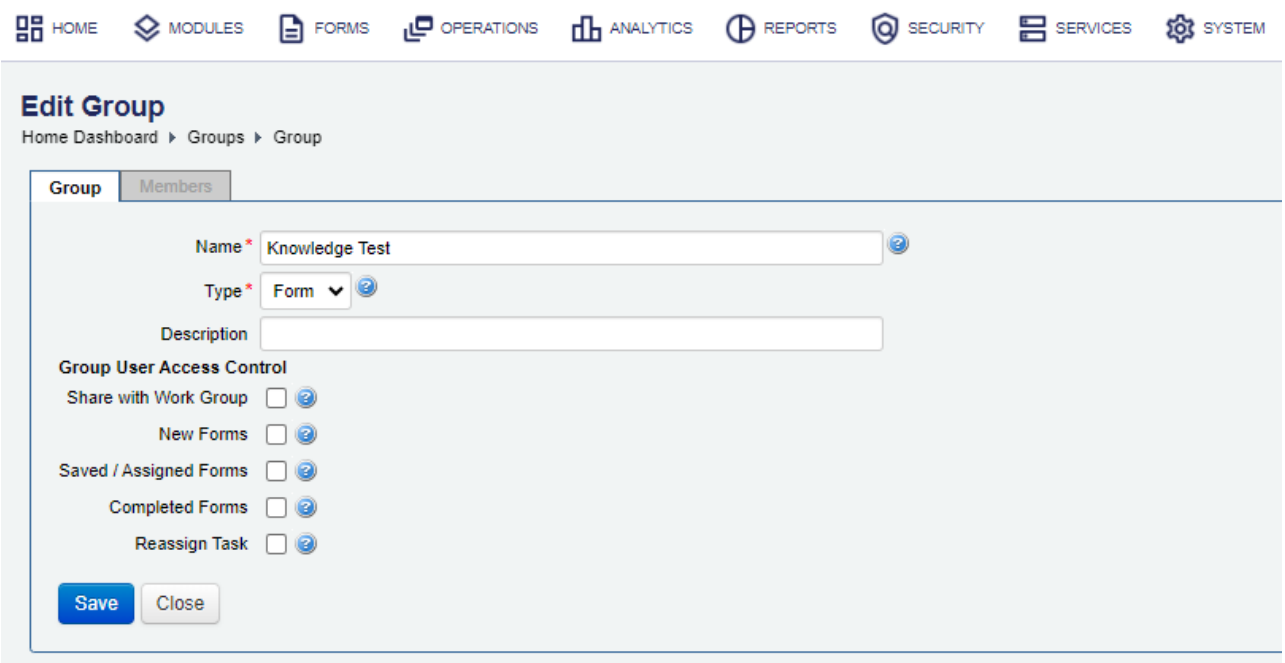
1. Select Security > Groups.
2. Click New to create a new group or locate a group and click Edit.



3. Enter a new group name in the Name field.
4. Select a group type from the Type dropdown list, , which can be one of the following:
   - Alert- notifies and alert users about certain events. A user must be assigned to a group of this group type.
   - Form- restricts user access to forms or tasks that are associated with this group.
   - Job - restrict access to job coordinators when overseeing jobs that they are responsible for.

> NOTE
>
> Job groups are automatically created by the Job Controller definition of a collaboration job., so you don't need to add them manually. However, you need to wait until each Job group is automatically generated before assigning members to the group.

5. Enter a group description in the Description field.
6. Complete the additional Group User Access Control section, which is shown for the Form group type only.



7. Select Share with Work Group to allow users of this work group to share a form.
8. Select New Forms to allow users to actively start filling in forms belonging to the group. Otherwise, they can't access the forms at all, except when tasks are assigned to the group and the group is configured to share tasks and incomplete submissions.
9. Select Saved / Assigned Forms to allow sharing of tasks assigned to the group as well as incomplete submissions for these forms.
10. Select Completed Forms to allow sharing of the list of completed submissions and access to receipts.
11. Select Reassign Task to allow members to reassign task submissions to other users within the form work group.
12. Click Save to update the changes.
13. Assign user accounts to this group.
14. Configure form access in the Group Access tab for each restricted form.
15. If the new group is to limit user access to a new form, make sure only the Authenticated Access checkbox is selected in a form space where the form is hosted. For more information, see Assign Form Spaces to a Form.
16. Click Save to update the changes.

NOTE

In the case of a Form group, it is now possible to restrict individual forms to be accessed only by members of this Form group. By default, a form is publicly available to all users logged in to a Form Space. Once a form is restricted to at least one Form group, then it is only listed for the members belonging to the Form groups in the Forms tab. For more information, see Assign Form Spaces to a Form.

NOTE

Any forms restricted to this Form group are now listed on the Forms tab from the Edit Group page.

Next, learn how to view groups.

Remove a Group

Manager allows you to delete a group when it is not required any more or you want to do it due to some security considerations.

To remove a group:

1. Select Security > Groups.
2. Locate a group and click Remove.
3. Click OK to confirm.

> WARNING
>
> You can't undo this action.

Next, learn how to view groups.

Assign Users to a Group

Manager allows you to assign users to a group. You can use this functionality to:

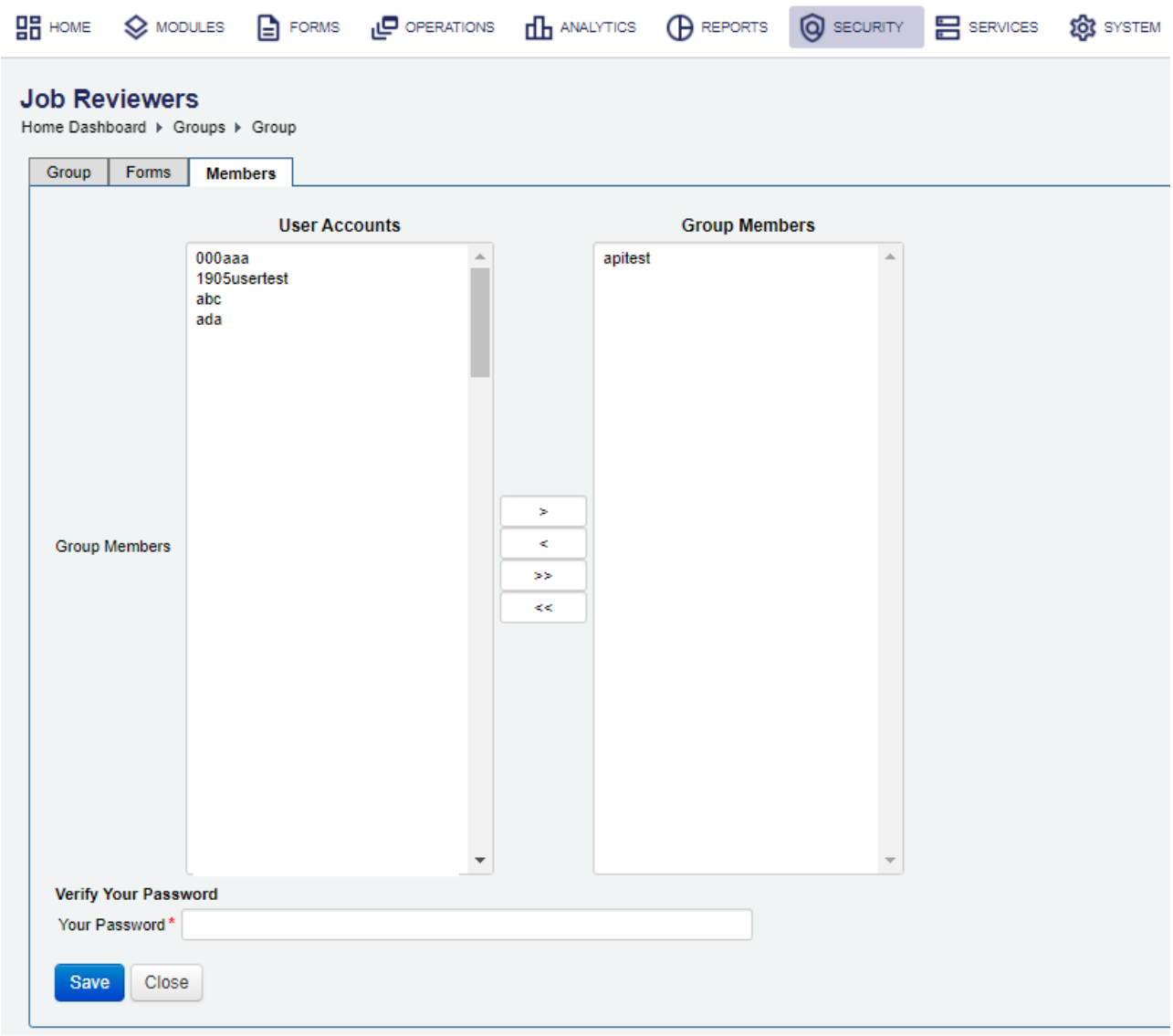
- Assign several user accounts to a newly created group
- Change an existing group's membership after you added or removed some user accounts

To assign user accounts to a group:

1. Select Security > Groups.
2. Locate a group and click Edit.
3. Select the Members tab to see a list of the assigned user accounts in Group Members.

   NOTE

   This list is empty for a newly created group.



4. Select one or more user accounts from User Accounts and click > to assign them to this group.

> NOTE
>
> You can use the `Shift` and `Ctrl` keys to select multiple user accounts on Windows PC.

5. To revoke assigned user accounts, select the user accounts from Group Members and then click <.
6. Click >> to assign all user accounts.
7. Click << to revoke all user accounts.
8. Provide your password in the Your Password field, for security reasons.

9. Click Save to update the changes.

NOTE

You can also manager a user group membership from the user account.

NOTE

If this is a form group, check which forms this group has been given privileged access to.

NOTE

You can also check any form's group access.

Next, learn how to assign groups to a user.

Assign Groups to a User

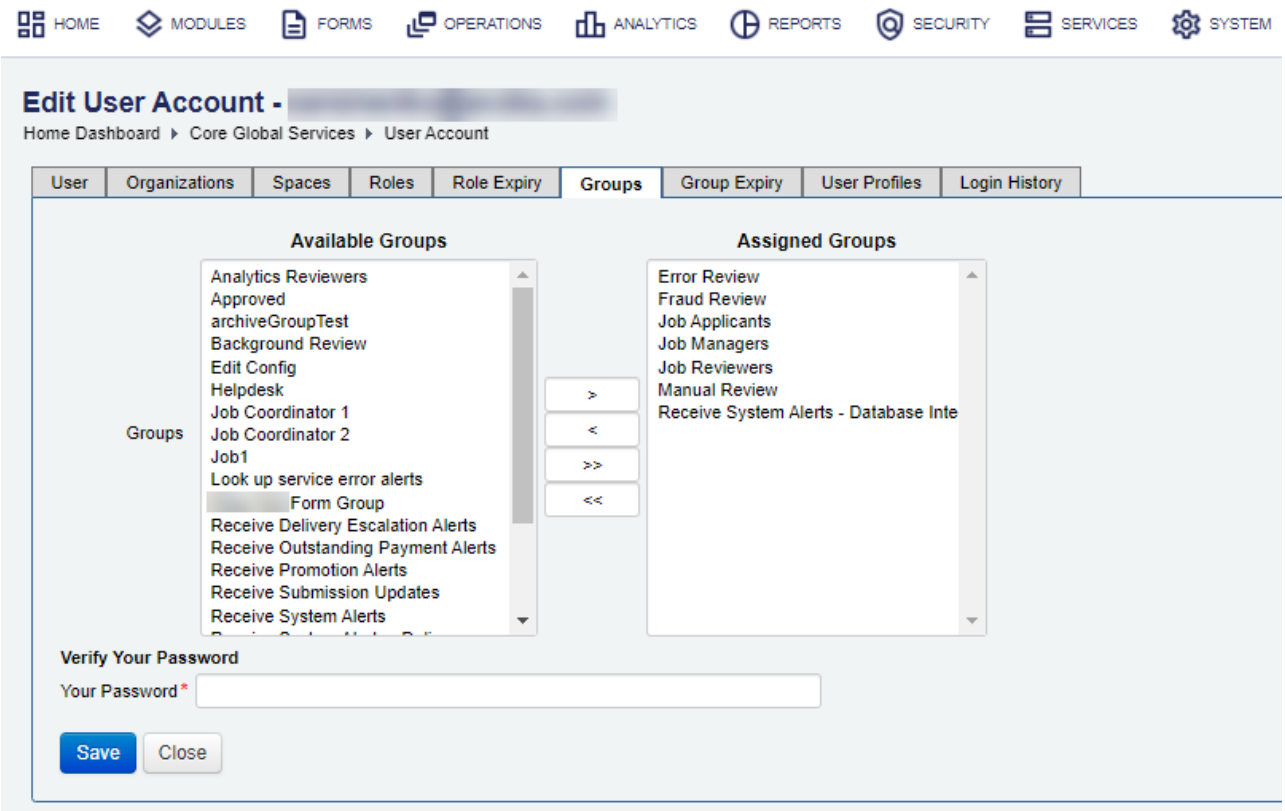
Manager allows you to assign groups to users, so they can perform relevant actions or receive various notifications pertinent to each group. You can use this functionality to:

- Assign several user accounts to a newly created group
- Change an existing group's membership after you added or removed some user accounts

Assign Groups

To assign groups to a user account:

1. Select Security > User Accounts.
2. Locate a user account and click Edit.
3. Select the Groups tab to see the assigned groups for this user account, which are listed in Assigned Groups.



4. Select one or more groups from Available Groups and click > to move them to Assigned Groups.

> NOTE
>
> You can use the `Shift` and `Ctrl` keys to select multiple groups on Windows PC.

5. Click >> to assign all groups to the user.
6. Provide your password in the Your Password field, for security reasons.
7. Click Save to update the changes.

Revoke Groups

When a user doesn't wish to be part of a group, you can revoke that grop.

To revoke groups from a user account:

1. Select the groups from Assigned Groups and click < to revoke any assigned groups.
2. Click << to revoke all groups from the user.
3. Provide your password in the Your Password field, for security reasons.
4. Click Save to update the changes.

> NOTE
>
> You can also assign users to a group.

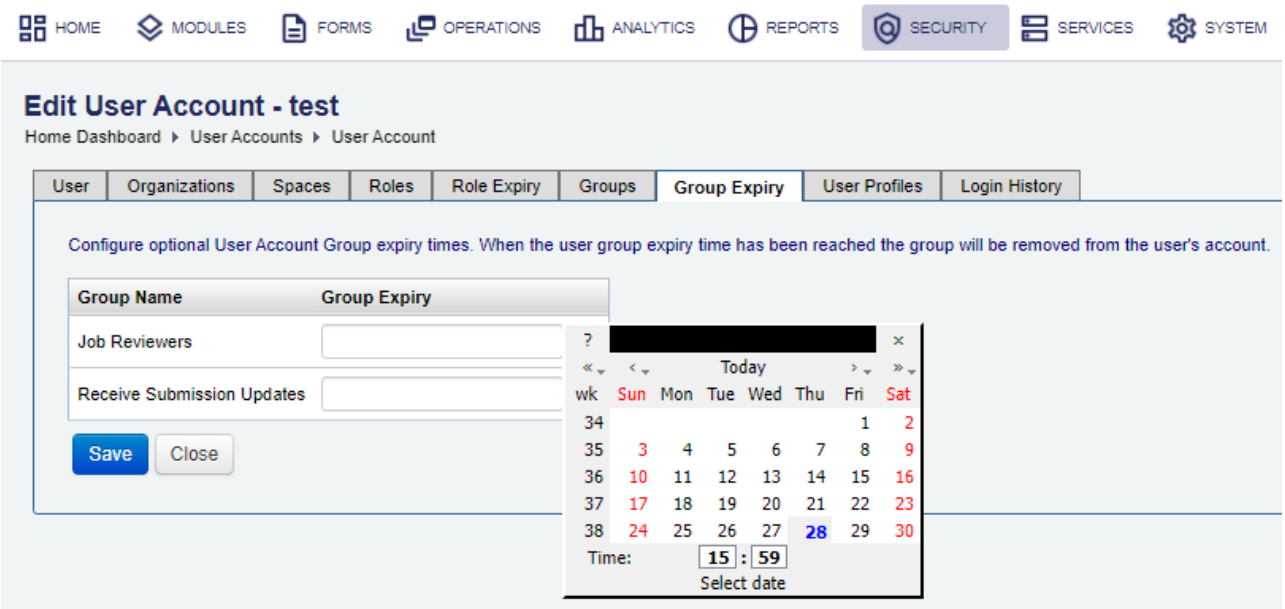
Next, learn how to configure group expiry.

Configure Group Expiry for a User Account

Manager allows you to enforce an expiry time to a user's group. You can use this functionality to temporarily assign a group to a user account to allow a support team to assist you in debugging a problem.

To enforce a time limit on a user's group:

1. Select Security > User Accounts.
2. Locate a user account and click Edit.
3. Select the Group Expiry tab to see groups that the user is assigned to in the Group Name column. Any temporary group assignment show an expiry date in the Group Expiry column.



4. Click Calendar in the Group Expiry column to make a user's group membership temporary.
5. Select the time and date when the user's group membership should expire. Select the time first, as the pop up automatically closes on the selection of a date.

> NOTE
>
> To make a user's group membership permanent again, remove the expiry date and time from the Group Expiry column.

6. Click Save to update the changes. When the expiry time comes, the Security Policy Manager background job removes the user's group membership from their user account.

Next, learn how to view groups.

View Forms Assigned to a Group

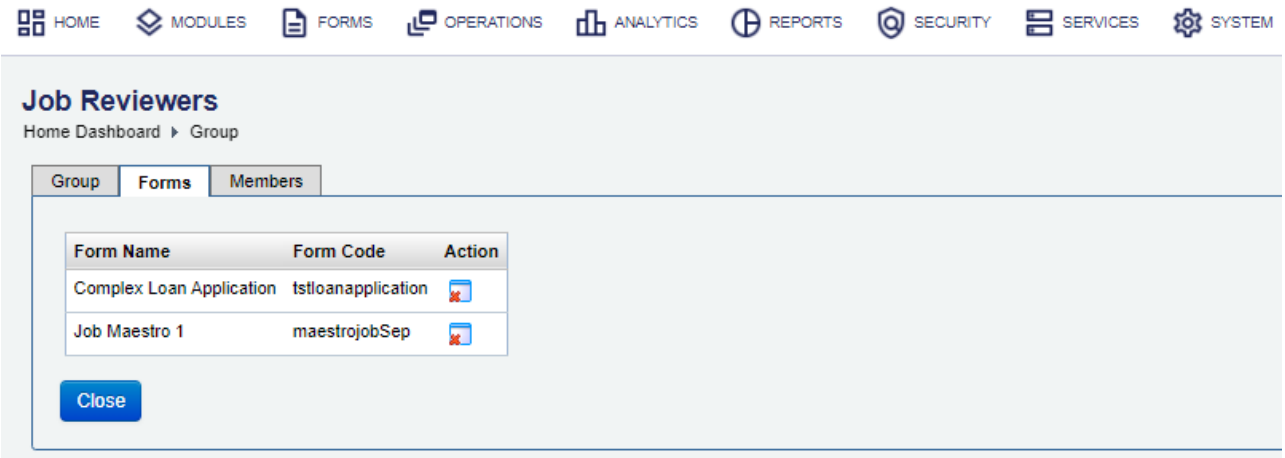
Manager allows you to restricts access to a form so only users assigned to the group will be able to access the form. A newly created group doesn't have any forms assigned to it, so you have to do it manually.

To assign forms to a group:

1. Create a new group.
2. Assign user accounts to the group.
3. Assign this group to a form. When a group is granted access to at least one form, it is listed in the Forms tab from the Edit Group page.

To view forms assigned to a group:

1. Select Security > Groups.
2. Locate a group and click Edit.
3. Select the Forms tab to see a list of forms assigned to this group.



4. Click Remove to remove group access to this form.

To add a form to the group, see Configure Form Group Access.

Next, learn how to configure a group expiry.

Migrate Groups

Manager provides a mechanism to migrate groups from one environment to another. This mechanism allows you to develop and test groups in different environments before migrating them over to the production environment. It also saves time to set up the groups in new Manager environments.

When a group is migrated, the following group properties are included:

- Group name
- Group description
- Group's user control

> NOTE
>
> Group migration doesn't include user accounts and forms associated with the group.

To migrate one or several groups:

1. Export groups.
2. Import groups to another Manager environment.

> NOTE
>
> You can use the same process to perform a role rollback.

Next, learn how to assign groups to a user.

Export Groups

Manager allows administrators to export groups from one environment and import them into another. You can use the export functionality to:

- Backup or archive groups
- Re-create or migrate groups in another Manager
- Promote one or several groups to other environments

To export one or several groups:

1. Select Security > Groups.
2. Click Export Archive to export one group or click Export to export all groups.

Manager exports groups into a group archive ZIP file, which will be downloaded to your PC.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to import groups.

Import Groups

Manager allows you to import a group as a ZIP file. You can use the import functionality to:

- Roll-back one or several groups
- Re-create one or several groups in another Manager
- Promote one or several groups to other environments

Manager runs a virus scanning service before an import file is processed, so the antivirus service must be properly configured and enabled. If a virus is detected, the import operation fails and a corresponding record is added to the Error Log. | 23.04 This feature was introduced in 23.04

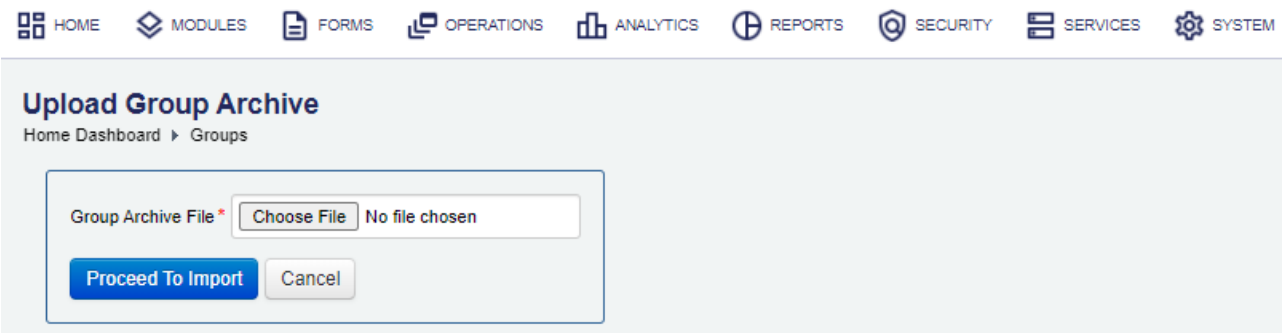
You can import a group using one of following options:

- UI
- SDK
- API

Import Using UI

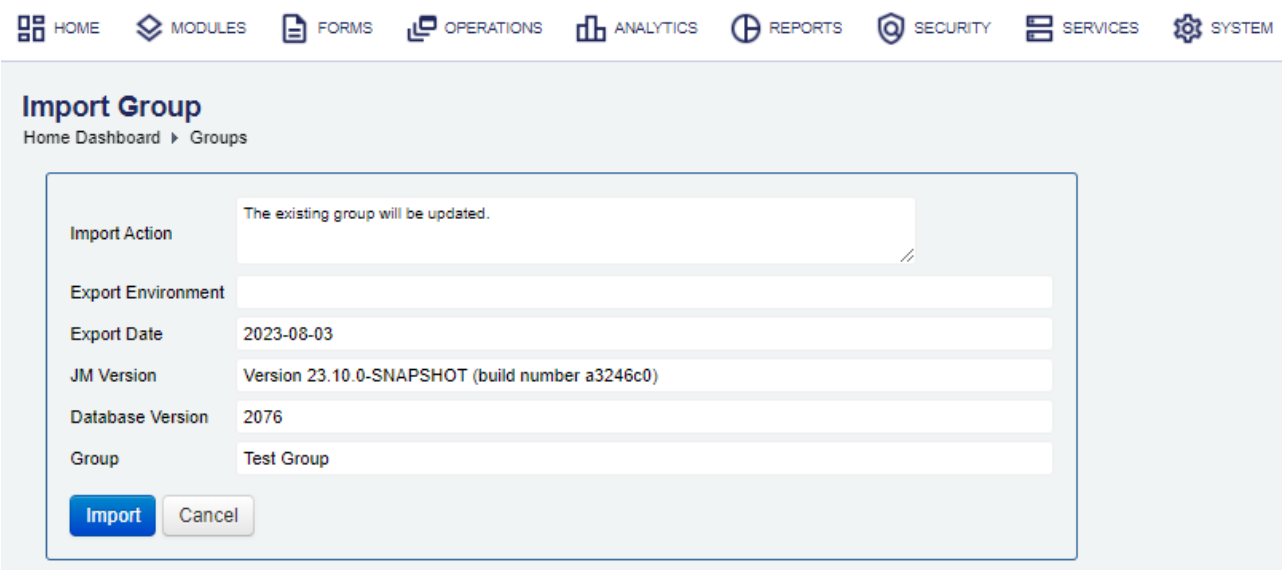
To import a group:

1. Select Security > Groups and click Import.

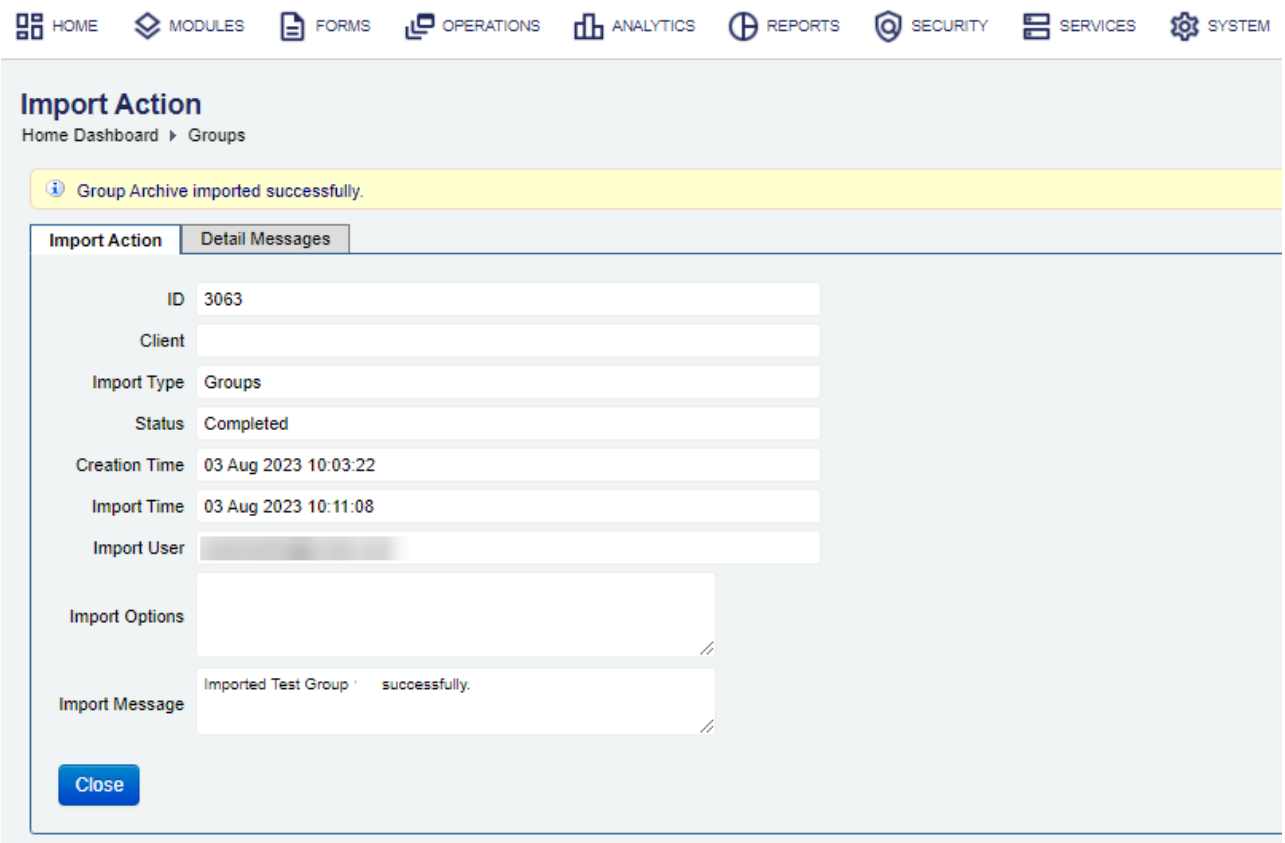


2. Click Choose File and locate a group archive ZIP file, which has been previously exported.
3. Click Proceed To import.

4. Review the Import Options.



5. Click Import to complete the process. The expected result page is shown below.



6. Click Close and check the list of groups to verify the imported groups have been added.

NOTE

You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

Import Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:archive-deploy goal to import a group.

Import Using API

The Journey Manager REST API provides the endpoint URL to import a group.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s)://<JM_SERVER>/manager/secure/rest/archive/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as Base64[1].

`--form ‘archiveFile=@“/Users/my-group.zip”’` - a path to an archive file to be imported.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

Example

```
curl --location --request PUT ‘https://<JM_SERVER>/manager/secure/rest/archive/v1/’
\
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
--form ‘archiveFile=@“/Users/my-group.zip”’
```

Next, learn how to migrate groups.

[1]Base64 is a binary to a text encoding scheme that represents binary data in an American Standard Code for Information Interchange (ASCII) string format.

Permissions Overview

Manager comes with a number of permissions to control user access to restricted content, such as form spaces and modules. Permissions are specific to each module and a form space, which of them has the standard set of pre-defined permissions. However, you can define your own permissions and implement security rules.

Permissions are mapped to standard roles which are in turn related to certain product features. By assigning roles to users, they are granted the necessary permissions to use the associated features. This is known as role-based access control.

> WARNING
>
> A set of permissions is associated with each role. Assigning several roles to a user grants that user all the permissions associated with all the assigned roles. Take care to review the complete set of permissions granted to a user to ensure they don't have any permissions they shouldn't have.

The Manager Home Dashboard has the following permissions that govern user access:

- View: gives users read-only access to the content.
- Edit: allows users to modify the content.
- Remove: allows users to delete the content.

By granting the Edit and Remove permissions, it doesn't automatically grant the View permission. For each of Edit and Remove permissions, you need to include the View permission to the role. For example, to allow users to create new or modify existing organizations in Manager, their role must require the Organization View and Organization Edit permissions.

Manager has different sets of permissions for the following modules:

- Manager
- Maestro
- Insights
- Exchange
- Workspaces

Next, learn about standard roles.

Core Permissions

Manager provides the a broad range of permissions to control user access to restricted content. The permissions are mapped to the standard roles, so you know what roles to assign to users to allow them the use of certain product functionality. This is listed in the table below:

Permission	Description	Role
Admin Directory	This is permission is required to access the Manager console. Users accessing the Manager console must be granted at least one role with this permission, otherwise they will not be able to use it.	System Manager Organization User Manager Transaction Data Access Operations System Support Administrator Developer
Audit View	Provides access to the system audit log and user login history.	System Manager System Support Administrator Developer
Bill Edit	Allows users to generate bill records using Fluent API. | 24.10 This feature was introduced in 24.10	System Manager Operations Administrator Developer

Bill View	Allows users to view bill records using Fluent API. | 24.10 This feature was introduced in 24.10	System Manager Operations Administrator Developer
Collaboration Job View	Provides read-only access to collaboration jobs, including viewing details about a collaboration job and its steps, actions and events. This permission is also found in each of the form spaces. However, it would grant different types of privileges for a form space.	System Manager Operations System Support Administrator Developer
Collaboration Job Edit	Allows users to manage collaboration jobs by pausing, triggering job processing, editing job properties and action details.	System Manager Operations Administrator Developer
Collaboration Job Remove	Allows users to delete collaboration jobs. Only canceled collaboration jobs can be deleted.	System Manager Operations Administrator Developer
Core Service Edit	Allows users to edit Core services' settings and options, for example, Remove Service, as well as some collaboration job related services. This permission is not assigned to any of the standard Manager roles, so you must use it with care and only when absolutely necessary, as there is a risk some essential core services, such	System Manager Administrator Developer

	as the Transaction Processor service, can be deleted. You should seek advice from the Support Team before assigning this permission to a user.	
Data Retention View	Allows users to view global data retention settings.	System Manager Operations System Support Administrator Developer
Data Retention Edit	Allows users to modify and apply global data retention settings.	System Manager Administrator
Db Version Update View	Allows users to view the database version history.	System Manager Administrator
Delivery Details View	Allows users to view the list of delivery channels and trigger delivery actions for submissions manually.	System Manager Operations System Support Administrator Developer
Delivery Details Edit	Allows users to create, edit and delete delivery channels.	System Manager Administrator Developer

Delivery Details Remove	It is not currently in use, so apply the Delivery Details Edit permission instead.	System Manager Administrator Developer
Deployment Property View	Allows users to view the list of system deployment properties and their values.	System Manager System Support Administrator Developer
Deployment Property Edit	Allows users to edit deployment property values, access and modify email settings.	System Manager Administrator Developer
Document Type View	Allows users to view document types and their details.	System Manager Operations System Support Administrator Developer
Document Type Edit	Allows users to create and edit document types.	System Manager Administrator Developer
Document Type Remove	Allows users to remove document types.	System Manager

Email Delivery Acknowledge	It is related to delivery of type Email Secure, where administrators receive an email notifying them of new submissions, asking them to acknowledge delivery by accessing a page on Manager and confirming. This permission controls access to this page. Must be assigned to only administrators involved in processing secure email delivery.	Administrator Developer System Manager Operations
Email Queue View	Allows users to access the system email queue, but does not allow them to view each email details. You need to assign the Email Queue Edit permission for that.	Administrator Developer System Manager Operations
Email Queue Edit	Allows users to create, edit and access email queue details.	Administrator Developer System Manager Operations
Error Log View	Provides access to the error log and error details. It also enables links to error log entries from various Manager tabs.	Administrator Developer System Manager Operations
Event Log View	Provides access to the event log and event details. It also enables links to event log entries from various Manager tabs.	Administrator

Export Application Package	Allows users to export application packages, including form bundles and services, for organizations they have access to.	Developer System Manager Operations Administrator Developer
Export Form	Allows users to export a form and all its versions.	System Manager Operations Administrator Developer
Export Form Version	Allows users to export a single form version.	System Manager Operations Administrator Developer
Export Groups	Allows users to export one or more groups.	System Manager Operations Administrator Developer
Export Metadata	Allows users to export the set of global metadata tags.	System Manager Operations Administrator Developer
Export Organization	Allows users to export an organization.	
Export Portal	Allows users to export a form space.	System Manager

		Operations Administrator Developer System Manager
Export Roles	Allows users to export one or more roles	Operations Administrator Developer
Export Services	Allows users to export one or more service definitions.	
Export Security Managers	Allows users to export one or more security managers[1].	System Manager Operations Administrator Developer
Form View	Allows users to view forms and most of their configuration in a read-only mode. Displays the Recent Forms card on the Journey Manager Home Dashboard. Enables links to forms from various Manager tabs.	System Manager Operations Administrator Developer

[1]Security managers are the Journey Manager's components that are responsible for User Authentication, User Authorization, User Self-Registration, and Session Management. Security mangers are designed to handle various sets of users requiring access to different form spaces and can be associated with one or more form spaces. Users can be external or internal, and they can require access to form spaces or a combination of form spaces and modules in different environments. You can find security managers under Security > Security Managers.

	NOTE If you need to view form version or form data configuration details, you need to add the Form Edit permission.	
Form Edit	Allows users to view all details about a form as well as create and edit forms. It includes editing form versions and form data configuration.	System Manager Operations Administrator Developer
Form Service Edit	Allows administrators to manage service definitions related to form development. This permission is a less powerful than the Service Edit permission.	System Manager Operations Administrator Developer
Form Test	This permission exposes links to access forms on a form space for testing. It also allows users to access the PDF receipt test harness, such as a link in the Form URLs section of the form edit page.	System Manager Operations Administrator Developer
Form Data Retention	Allows users to edit form specific data retention settings.	System Manager Operations Administrator Developer
Form Remove	Allows users to delete forms, form versions, form promotion entries and form data extract mappings.	System Manager Operations

Groovy Console	Allows access to the Groovy console.	Administrator Developer System Manager Operations
Groovy Log View	Allows access to the Groovy service log.	Administrator Developer System Manager Operations
Group View	Allows users to view the list of groups, but it does not allow access to group detail information, so you should add the Group Edit permission for that.	Administrator Developer System Manager Operations
Group Edit	Allows users to create, edit and view group details.	Administrator Developer System Manager Operations
Group Remove	Allows users to remove groups.	Administrator

Group Assign	Allows users to assign users to a group. It is also necessary to access the Forms panel on the group edit page.	Developer System Manager Operations Administrator
Import Action View	Allows access to the import log. It does not include read-only access to import action details.	Developer System Manager Operations Administrator
Import Action Edit	Allows users to view and edit import log details . It is also necessary to import form and form space archives exported from another Manager server, along with the import permission corresponding to the entity to be imported.	Developer System Manager Operations Administrator
Import Action Remove	Allows users to delete entries from the import log. Only completed import actions can be deleted.	Developer System Manager Operations Administrator
Import Application Packagev	Allows users to import application packages, such as form bundles and services, for organizations they have access to.	Developer System Manager Operations Administrator Developer

Import Composer Form	Allows users to import forms published by Composer and view the import details.	System Manager Operations Administrator Developer
Import Form	Allows users to import form version archives created by exporting a form version from another Manager server. to import form version archives successfully, users also currently need the Import Action Edit permission.	System Manager Operations Administrator Developer
Import Groups	Allows users to import group archives created by exporting one or more groups from another Manager server.	System Manager Operations Administrator Developer
Import Metadata	Allows users to import metadata tag archives created by exporting one or more metadata tags from another Manager server. In addition, this permission is needed to upload files with values for metadata tags of types List and List Hierarchy.	System Manager Operations Administrator Developer
Import Organization	Allows users to import an organization archive, which contains organization configuration but no forms, exported from another Manager server. Only administrators with access to more than one organization can see the import button.	System Manager Operations Administrator Developer
Import Portal	Allows users to import a space archive, which contains form	System Man-

	space configuration but not the backing WAR file, exported from another Manager server as well as space WAR files. To import space archives successfully, users also currently need the Import Action Edit permission.	ager Operations Administrator Developer System Manager
Import Roles	Allows users to import role archives, which contains roles and associated permissions, exported from another Manager server.	Operations Administrator Developer System Manager
Import Security Manager	Allows users to import security manager archives exported from another Manager server.	Operations Administrator Developer System Manager
Import Services	Allows users to import service archives, which contains service definitions and their service connections, exported from another Manager server.	Operations Administrator Developer System Manager
Import TPac	Allows users to import TPacs directly into Manager.	Operations Administrator Developer
Licensing Edit	Allows users to edit system licensing information and configure how licensing report data will be published to Avoka.	System Manager

		Operations Administrator Developer System Manager
Licensing Report View	Allows users to access the Transact Licensing and Job Licensing reports, otherwise these reports don't appear in the Reports menu.	Operations Administrator Developer System Manager
Metadata Tag View	Allows users to view metadata tags and their details.	Operations Administrator Developer System Manager
Metadata Tag Edit	Allows users to create and edit metadata tags.	Operations Administrator Developer System Manager
Metadata Tag Remove	Allows users to delete metadata tags.	Operations Administrator Developer System Manager
Metadata Value View	Allows users to view values configured for metadata tags on the organization or form version levels.	Operations

Metadata Value Edit	Allows users to create and edit values configured for metadata tags on the organization or form version levels.	Administrator Developer System Manager Operations
Metadata Value Remove	Allows users to delete values configured for metadata tags on the organization or form version levels.	Administrator Developer System Manager Operations
Notification View	Allows users to view the list of notifications configured for form spaces.	Administrator Developer System Manager Operations
Notification Edit	Allows users to create and edit notifications configured for form spaces.	Administrator Developer System Manager Operations
Notification Remove	Allows users to delete notifications configured for form spaces.	Administrator

Offline Sync Log View	Allows users to view the T.Field related offline sync and client logs.	Developer System Manager Operations Administrator
Operation Chart View	Allows users to view a number of Manager analytics charts.	Developer System Manager Operations Administrator
Operation Export Report View	Not used.	Developer System Manager Operations Administrator
Operation Form Events View	Not used.	Developer System Manager Operations Administrator
Organization View	Allows users to view organization information and part of its configuration. It also enables links to organizations.	Developer System Manager Operations Administrator Developer

Organization Edit	Allows users to create and edit organizations and associated configuration, such as categories and application packages.	System Manager Operations Administrator Developer
Organization Remove	Allows users to remove organizations. Because deleting organizations will also delete all associated forms and submissions, this is a permission that should be assigned sparingly.	System Manager Operations Administrator Developer
Organization Copy	Allows users to create a copy of an existing organization.	System Manager Operations Administrator Developer
Organization Portal Assign	Allows users to assign spaces to an organization for form hosting purposes.	System Manager Operations Administrator Developer
Organization Payment Edit	Allows users to view and modify an organization's payment gateway.	System Manager Operations Administrator Developer
Organization	Allows users to view the set of organization property values.	System Man-

Property View		ager Operations Administrator Developer System Manager
Organization Property Edit	Allows users to edit and delete organization property values.	Operations Administrator Developer System Manager
Organization Property Remove	Not used. The edit permission confers deletion privileges.	Operations Administrator Developer System Manager
Organization Storage Edit	Allows users to edit organization data storage and retention settings.	Operations Administrator Developer System Manager
Payment Log View	Allows users to view details of payments made by form users.	Operations Administrator Developer System Manager
Portal View	Allows users to view details of form spaces and modules.	

Portal Edit	Allows users to edit form spaces, modules and all associated configurations.	Operations Administrator Developer System Manager
Portal Content Edit	Allows users to edit form space content, such as pages and resources.	Operations Administrator Developer System Manager
Portal Style Edit	Allows users to edit some form space content related to styling, such as the index page, childportal.css and all resources of JPG type.	Operations Administrator Developer System Manager
Portal Remove	Allows users to remove form spaces and modules.	Operations Administrator Developer System Manager
Promotion View	Allows users to view the form version promotions and a tab on the form edit page.	Operations

Promotion 1 Development	Allows users to promote form versions to be in the Development status.	Administrator Developer System Manager Operations
Promotion 2 Ready For Test	Allows users to promote form versions to be in the Ready For Test status.	Administrator Developer System Manager Operations
Promotion 3 Test Failed	Allows users to promote form versions to be in the Test Failed status.	Administrator Developer System Manager Operations
Promotion 4 Test Passed	Allows users to promote form versions to be in the Test Passed status.	Administrator Developer System Manager Operations
Promotion 5 Ready For Production	Allows users to promote form versions to be in the Ready For Production status.	Administrator

Promotion 6 Not Ready For Production	Allows users to promote form versions to be in the Not Ready For Production status.	Developer System Manager Operations Administrator
Promotion 7 Production	Allows users to promote form versions to be in the Production status.	Developer System Manager Operations Administrator
Property Type View	Allows users to view property types.	Developer System Manager Operations Administrator
Property Type Edit	Allows users to create and edit property types	Developer System Manager Operations Administrator
Property Type Remove	Allows users to remove property types.	Developer System Manager Operations Administrator

Report View	Allows users to view reports, but it does not give read-only access to report configuration details, so you should add the Report Edit permission for that.	System Manager Operations Administrator Developer
Report Edit	Allows users to create and edit reports.	System Manager Operations Administrator Developer
Report Remove	Allows users to remove reports.	System Manager Operations Administrator Developer
Report Render	Allows users to render reports on screen and access the organization report schedules.	System Manager Operations Administrator Developer
Report Schedule View	Allows users to view report schedules on the organization level.	System Manager Operations Administrator Developer
Report	Allows users to create and edit report schedules on the	System Man-

Schedule Edit	organization level.	ager Operations Administrator Developer System Manager
Report Schedule Remove	Allows users to remove report schedules on the organization level.	Operations Administrator Developer System Manager
Report Schedule Trigger	Allows users to trigger report schedules on the organization level that send an email for the most recent reporting period.	Operations Administrator Developer System Manager
REST Application Package API	Allows users to access the Application Package REST service endpoint for listing application packages, and importing and exporting application packages in Manager.	Operations Administrator Developer System Manager
REST Archive API	Allows users to invoke the REST Archive API.	Operations Administrator Developer
REST Configuration Ser-	Allows users to invoke the REST Configuration Service API. For more information, see Configuration Service.	System Manager

vice API		Operations Administrator Developer System Manager
REST Delivery Service API	Allows users to invoke the REST Delivery API.	Operations Administrator Developer
REST Fluent API	Allows users to invoke Fluent API via the REST service. | 21.05 This feature was introduced in 21.05.	Administrator System Manager
REST Form Groups API	Allows users to access the Form Groups REST service endpoint for querying and creating Form Groups in Manager.	Operations Administrator Developer System Manager
REST Groovy Service Invoke	Allows users to access the Groovy Service REST service endpoint to invoke a Groovy service. This service endpoint can be used to perform operations such as uploading reference data for use in dynamic data or form prefill services. Enable the service parameter REST Invoke Enabled service parameter.	Operations Administrator Developer System Manager
REST JS App Deploy API	Allows users to invoke the REST JS App Deploy API.	Operations Administrator Developer
REST Ser-	Allows users to invoke the REST Service Definitions API.	System Man-

vice Definitions API		ager Operations Administrator Developer System Manager
REST Tasks API	Allows users to access the REST Tasks API service endpoint for creating and updating tasks in Manager in a secure and efficient manner.	Operations Administrator Developer System Manager
REST Test Center API	Allows users to access the REST TestCenter API service endpoint for performing test operations in Manager.	Operations Administrator Developer System Manager
REST TPac API	Allows users to access the REST TPac API service endpoint for deploying TPacs, and downloading or deleting TPac archive files in Manager.	Operations Administrator Developer System Manager
REST Transaction History API	Allows users to access the REST Transaction History API service endpoint for querying the OLAP Transaction History table in a secure and efficient manner.	Operations Administrator Developer System Manager
REST Transactions API	Allows users to invoke the REST Transactions API.	

Role View	Allows users to view roles, but it does not give read-only access to role configuration details, so you should add the Role Edit permission for that.	Operations Administrator Developer System Manager Operations Administrator Developer System Manager
Role Edit	Allows users to create, edit and view role details.	Operations Administrator Developer System Manager
Role Remove	Allows users to remove roles.	Operations Administrator Developer System Manager
Role Assign	Allows users to assign user accounts to a role. The set of roles that a user can assign is further restricted based on whether the user is a global administrator.	Operations Administrator Developer

Security Manager View	Allows users to view the set of security managers[1].	System Manager Operations Administrator Developer
Security Manager Edit	Allows users to create, edit and view all details of security managers, including managing authentication providers. The Service Edit permission currently controls access to some content, such as creating security managers and authentication providers and parameters.	System Manager Operations Administrator Developer
Security Manager Remove	Allows users to remove security managers.	System Manager Operations Administrator Developer
Service View	Allows users view the full set of service definitions, service connections, collaboration job definitions, server nodes, and scheduled jobs in a read-only mode. Displays the Recent Services card on the Journey Manager Home Dashboard.	System Manager Operations Administrator Developer

[1]Security managers are the Journey Manager's components that are responsible for User Authentication, User Authorization, User Self-Registration, and Session Management. Security mangers are designed to handle various sets of users requiring access to different form spaces and can be associated with one or more form spaces. Users can be external or internal, and they can require access to form spaces or a combination of form spaces and modules in different environments. You can find security managers under Security > Security Managers.

	NOTE If you need to view services configuration details, you need to add the Service Edit permission.	
Service Edit	Allows users to create, edit and view configuration details of service definitions, service connections, collaboration job definitions, server nodes, scheduled jobs, and system data retention settings. For a form developer to modify their own form related services, without giving them access to all service types, you should use the Form Service Edit permission instead.	System Manager Operations Administrator Developer
Service Remove	Allows users to remove service definitions, service connections, collaboration job definitions, server nodes, and scheduled jobs.	System Manager Operations Administrator Developer
Submission View	Allows users to view form transactions, transactions history, transaction delivery, saved transactions, abandoned transactions. It also enables links to submissions on other pages. Access to the actual content submitted by users is not granted by this permission, so you should add the Submission Data View and Submission XML Data View permissions.	System Manager Operations Administrator Developer
	Allows users to view form transactions, transactions history, transaction delivery, saved transactions, abandoned transactions. Displays the Latest Transactions card on the Journey Manager Home Dashboard. Enables links to submissions on other pages.	

	NOTE If you need to view the actual content submitted by users, you need to add the Submission Data View and Submission XML Data View permissions.	
Submission Edit	Allows users to modify submission statuses and retry delivery.	System Manager Operations Administrator Developer
Submission Data View	Allows users to view non-XML submission content, such as receipt and attachment content, but not the submitted XML, submitted by the user. It also provides access to the set of submission data extract values, including the form submission data. Allows access to user entered data, so you must handle it carefully.	System Manager Operations Administrator Developer
Submission XML Data View	Allows users to view XML submission content, such as the data entered into the form by the user. WARNING The data can contain Personal Identifiable Information[1] (PII), so you must handle it carefully.	System Manager Operations Administrator Developer
Submission XML Data Edit	Allows users to edit XML submission content, such as the data entered into the form by the user, and delete submission data extract values.	System Manager Operations Administrator

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

	WARNING The data can contain Personal Identifiable Information[1] (PII), so you must handle it carefully.	Developer
Submission Save Edit	Allows administrators to open and fill in saved transactions.	System Manager Operations Administrator Developer
System Health View	Allows users to view system health.	System Manager Operations Administrator Developer
System Info View	Allows users to access some pages that provide information on system info, server performance, and LiveCycle performance.	System Manager Operations Administrator Developer
System Report Schedule View	Allows users to view report schedules for global reports.	System Manager Operations Administrator Developer

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

System Report Schedule Edit	Allows users to create and edit report schedules for global reports.	System Manager Operations Administrator Developer
System Report Schedule Remove	Allows users to remove report schedules for global reports, but you should also add the Report Schedule Edit permission for this to work.	System Manager Operations Administrator Developer
System Report Schedule Trigger	Allows users to trigger report schedules for global reports that sends an email for the most recent reporting period.	System Manager Operations Administrator Developer
Task View	Allows users to view the list of task submissions.	System Manager Operations Administrator Developer
Task Edit	Allows users to create task submissions and edit related task.	System Manager Operations Administrator Developer
TPac	Allows users to remove imported TPacs.	System Man-

Remove		ager Operations Administrator Developer System Manager
TPac View	Allows users to view the imported TPacs.	Operations Administrator Developer System Manager
User View	Allows administrators to view user accounts, but it does not include a read-only view of user details, so you should add the User Edit permission for that.	Operations Administrator Developer System Manager
User Edit	Allows administrators to create, edit and view User Edit user properties.	Operations Administrator Developer System Manager
User Remove	Allows administrators to remove user accounts.	Operations Administrator Developer
User Bulk Import	Allows administrators to bulk import user accounts defined in an Excel spreadsheet.	System Manager

		Operations Administrator Developer System Manager
User Profile Edit	Allows administrators to create and edit user profiles, as well as mark a profile as the default one for a user.	Operations Administrator Developer System Manager
User Profile Remove	Allows administrators to remove user profiles.	Operations Administrator Developer System Manager
User Property View	Allows administrators to view user properties.	Operations Administrator Developer System Manager
User Property Edit	Allows administrators to create, edit and view user properties.	Operations Administrator Developer System Manager
User Property Remove	Allows administrators to remove user properties.	Operations

Web Service Invoke	Allows users to invoke Manager's web services.	Administrator Developer System Manager Operations Administrator Developer

Next, learn how to view roles.

Maestro Permissions

Maestro uses the following permissions to control user access to restricted content. The permissions are mapped to the standard roles, so you know what roles to assign to users to allow them to use certain product functionality. This is listed in the table below:

Permission for Product Functionality	Description	Role
Form Edit	Allows users to edit a form.	Maestro Administrator Maestro Developer
Form Publish	Allows users to build a form.	Maestro Administrator Maestro Developer
Form Remove	Allows users to delete a form.	Maestro Administrator Maestro Developer
Form View	Allows users to view a form.	Maestro Administrator Maestro Developer
Library Edit	Allows users to edit a library.	Maestro Administrator Maestro Developer
Library Remove	Allows users to delete a library.	Maestro Administrator Maestro

		Developer
Library Share	Allows users to share libraries. | 20.11 This feature was introduced in 20.11.	Maestro Administrator Maestro Developer
Library Share Global	Allows users to share libraries globally. | 20.11 This feature was introduced in 20.11.	Maestro Administrator Maestro Developer
Library View	Allows users to view a library.	Maestro Administrator Maestro Developer
Organization Edit	Allows users to edit an organization.	Maestro Administrator Maestro Developer
Project Edit	Allows users to edit a project.	Maestro Administrator Maestro Developer
Project Export	Allows users to export a project.	Maestro Administrator Maestro Developer
Project Import	Allows users to import a project.	Maestro Administrator Maestro Developer

Project Remove	Allows users to delete a project.	Maestro Administrator Maestro Developer
Project View	Allows users to view a project.	Maestro Administrator Maestro Developer
Release Edit	Allows a Maestro Administrator to edit releases.	Maestro Administrator
Release Remove	Allows a Maestro Administrator to delete releases.	Maestro Administrator
Release Version View	Allows a Maestro administrator to view release versions.	Maestro Administrator
SCM Design Delete	Allows staff who can use Maestro SCM to delete designs.	Maestro SCM Design
SCM Design Edit	Allows staff who can use Maestro SCM to edit designs.	Maestro SCM Design
SCM Design View	Allows staff who can use Maestro SCM to view designs.	Maestro SCM Design
SCM Org Library Add	Allows staff who can use Maestro SCM to add organization libraries.	Maestro SCM Org Library
SCM Org Library Edit	Allows staff who can use Maestro SCM to update organization libraries.	Maestro SCM Org Library
SCM Org Library View	Allows staff who can use Maestro SCM to view organization libraries.	Maestro SCM Org Library
SCM Project Delete	Allows staff who can use Maestro SCM to delete projects.	Maestro SCM Project
SCM Project Edit	Allows staff who can use Maestro SCM to update	Maestro SCM

	projects.	Project
SCM Project View	Allows staff who can use Maestro SCM to view projects.	Maestro SCM Project
SCM Project Library Add	Allows staff who can use Maestro SCM to delete project libraries.	Maestro SCM Project
SCM Project Library Edit	Allows staff who can use Maestro SCM to update project libraries.	Maestro SCM Project
SCM Project Library View	Allows staff who can use Maestro SCM to view project libraries.	Maestro SCM Project
Shared Domain Model Edit	Allows staff to edit shared domain model.	Maestro Administrator
Versions Admin	Allows staff to manage versions.	Maestro Administrator

Maestro doesn't have granular user access control to templates vs forms in the same project, so you can't allow users to modify form text in a form but refuse them to edit the form's styling.

As you have access to an organization, which is separate to your assigned permissions, you can't have read access to one organization or project and full write access to another organization or project.

However, you can have a separate organization for template developers and have the form in another organization and then utilize shared libraries that allows sharing of tagged libraries between organizations.

Next, learn how to view roles.

Exchange Permissions

Exchange uses the following permissions to control user access to restricted content. The permissions are mapped to the standard roles below, so you know what roles to assign to users to enable them to use certain product functionality.

Permission	Description	Role
TPac Install	Allows users to install and upgrade Exchange packages using TPacs.	Avoka Exchange
TPac Review	Allows users to add a review for the Exchange packages.	Avoka Exchange
TPac Uninstall	Allows users to uninstall Exchange packages.	Avoka Exchange

Next, learn how to view roles.

Journey Analytics Permissions

Journey Analytics uses the following permissions to control user access to restricted content. The permissions are mapped to the standard roles below, so you know what roles to assign to users to enable them to use certain product functionality.

Permission	Description	Role
Insights Admin	Allows administrators to access the Journey Analytics module, so they are able to change its global settings.	Transact Insights Administrator
Insights View	Allows users to access the Journey Analytics module, so they are able to view its analytics reports.	Transact Insights

Next, learn how to view roles.

Workspaces Permissions

Journey Workspaces uses the following permissions to control user access to restricted content. The permissions are mapped to the standard roles below, so you know what roles to assign to users to enable them to use certain product functionality.

Permission	Description	Role
Collaboration Job Completed View	Allows the reviewers and managers to monitor and manage completed tasks in the Workspaces form space. Compare it with the Manager's Collaboration Job View permission.	Workspaces Staff
Collaboration Job View	Allows the reviewers and managers to monitor and manage tasks in the Workspaces form space.	Workspaces Staff
Help Desk Authenticated Edit	Allows the help desk staff members to monitor submissions from authenticated customers in the Help Desk tab, which, by default, excludes the submissions from authenticated users to be viewed by the Workspaces staff and Maguire staff roles. Assign this permission to standard roles to include submissions from authenticated users. This permission is required to call the following API: • `txn/cancel` to cancel a Saved or Opened form transaction, which is not a task submission. NOTE You don't need this permission to call `txn/cancel` API to cancel an Opened form transaction, which is a task submission and is assigned to a user.	Workspaces Staff

Help Desk View	NOTE If you call `txn/cancel` API for Workspaces Operations Portal, you must have the Help Desk View permission configured instead. Allows the reviewers and managers to monitor and manage tasks in the Workspaces form space. It also enables you to see and use the Help Desk tab. The Workspaces Operations Portal requires this permission to call the following API: • `txn/status` | 22.04 This feature was introduced in 22.04. • `txn/task` | 22.04 This feature was introduced in 22.04. • `txn/purge` | 22.04 This feature was introduced in 22.04. • `txn/cancel` | 22.04 This feature was updated in 22.04. NOTE You don't need this permission to call `txn/cancel` API for other portals. In this case, you must have the Help Desk Authenticated Edit permission configured.	Workspaces Staff
Operations All Transactions	Allows the reviewers and managers to view all transactions from all form spaces. The Workspaces Operations Portal requires this permission to call the following API: • `txn/query`, with the query scope as `ALL` in a request, to list all transactions for a form space. | 22.04 This feature was updated in 22.04.	Workspaces Staff Workspaces API

NOTE

You don't need this permission to call `txn/query` API with other query scope options.

Next, learn how to view roles.

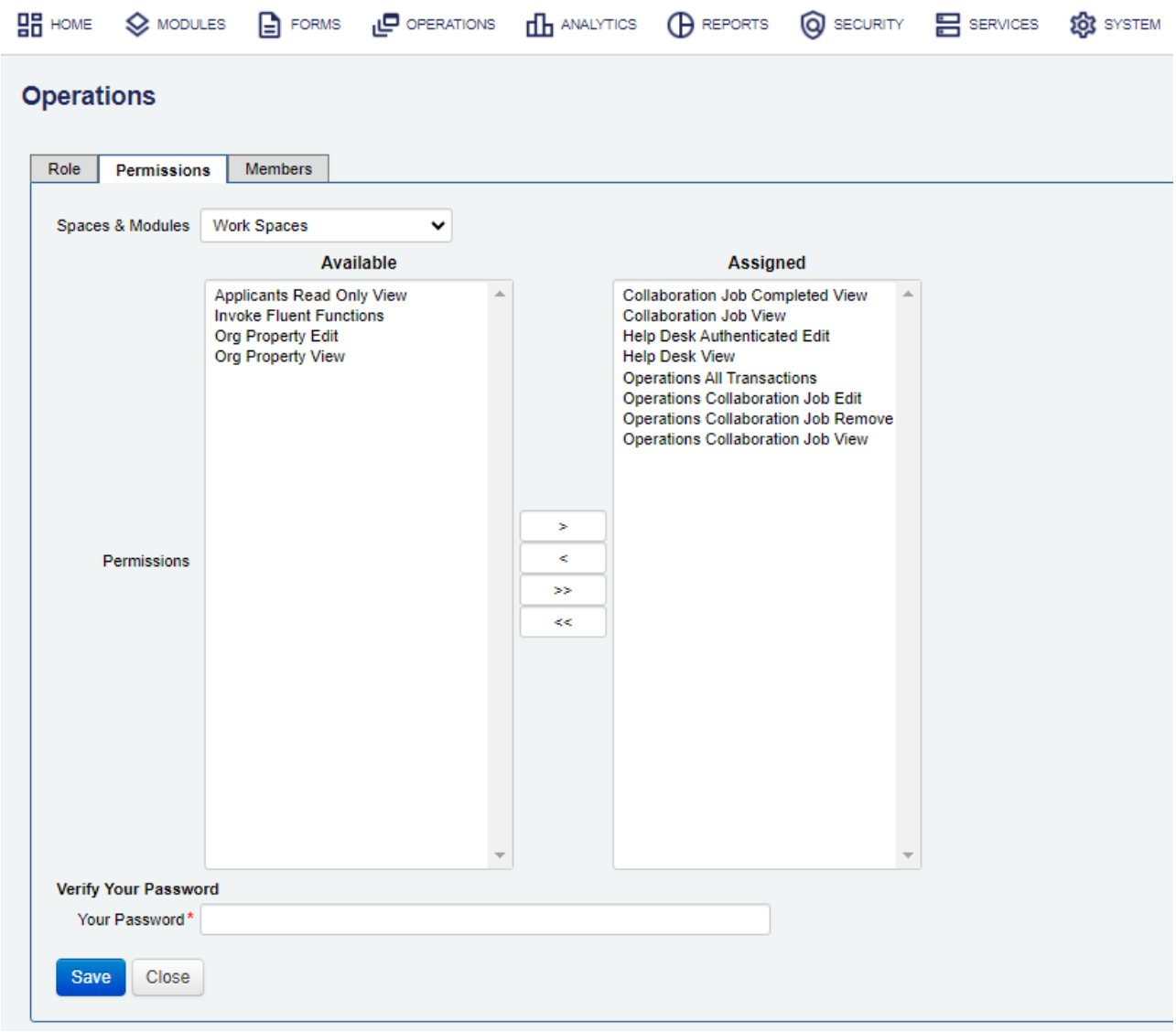
Assign Permissions to a Role

Manager allows you to assign permissions to a role.

> NOTE
>
> You must be a Journey Manager administrator to create or configure this functionality.

To assign permissions to a role:

1. Select Security > Roles.
2. Locate a role and click Edit.
3. Select the Permissions tab to add or remove permissions.



4. Select a form space of this role from the Spaces & Modules dropdown list. The set of permissions that are applicable to a form space are listed in the Available section, and if assigned to this role, the assigned permissions are listed in the Assigned section.

   NOTE

   Not all spaces and modules are included in this list. Only the spaces where access is managed by roles and their associated permissions are applicable and those missing are not applicable to have permissions assigned to roles.

5. Select one or more permissions from Available permissions and click > to move the selected permissions to Assigned permissions.

   NOTE

   You can use the `Shift` and `Ctrl` keys to select multiple groups on Windows PC.

6. To revoke any assigned permissions, select the permissions from Assigned permissions and click <.
7. To assign all permissions, click >>.
8. To revoke all permissions, click <<.
9. Provide your password in the Your Password field, for security reasons.
10. Click Save to update the changes.
11. Check this form space is assigned to the user.

NOTE

All permissions assigned to a role are loaded when a user logs into Manager, so the user has to log out and then log back in to see the changes to take effect.

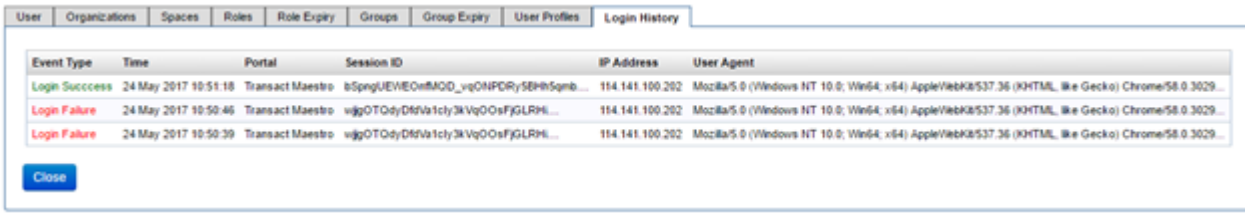
Next, learn how assign users to the role.

View User Login History

Manager allows you to view a list of all login attempts by each user or all users.

To view a user's login history:

1. Select Security > User Accounts.
2. Locate a user account and click Edit.
3. Select the Login History tab to see each login attempt and its details:



- Event Type is the status of the login attempt.

  A user login event type can be one of the following:

  - Account Locked Temp.
  - Login Failure.
  - Log Out.
  - Login Success.
  - Password Reset.
  - Session Expiry.
- Time is the date and time of the login attempt.
- Portal is the space or portal a user accessed or attempted to access.
- Session ID is a user's session ID[1] of the login attempt.

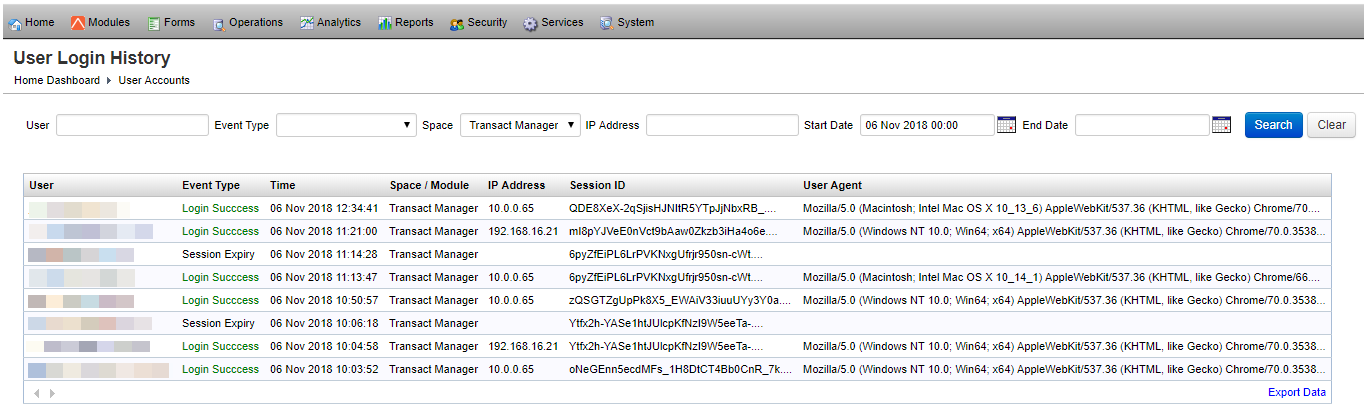
[1]Session ID is a unique number that a Web site's server assigns a specific user for the duration of that user's visit (session). The session ID can be stored as a cookie, form field, or URL (Uniform Resource Locator). Some Web servers generate session IDs by simply incrementing static numbers.

- IP Address is a user agent's IP address[1] used for this login attempt.
- User Agent is a user agent[2] used for this login attempt.

4. Click Close.

To view all users' login history, select Security > User Login History.

User login history records are shown page by page, with the maximum number of records per page configured in the users preferences.



To filter or search the user login history by one or more criteria, specify the following settings and click Search:

- User account's name or part of a name.
- Event Type.

  A user login event type can be one of the following:

  - Account Locked Temp.
  - Login Failure.
  - Log Out.
  - Login Success.
  - Password Reset.
  - Session Expiry.
- Space.

---

[1]IP address is a unique string of numbers separated by full stops that identifies each computer using the Internet Protocol to communicate over a network.
[2]User agent is software (a software agent) that is acting on behalf of a user. One common use of the term refers to a web browser that retrieves, renders and facilitates end user interaction with Web content.

- IP Address is a user agent's IP address[1] used for this login attempt.
- Start Date.
- End Date.

Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to view user accounts.

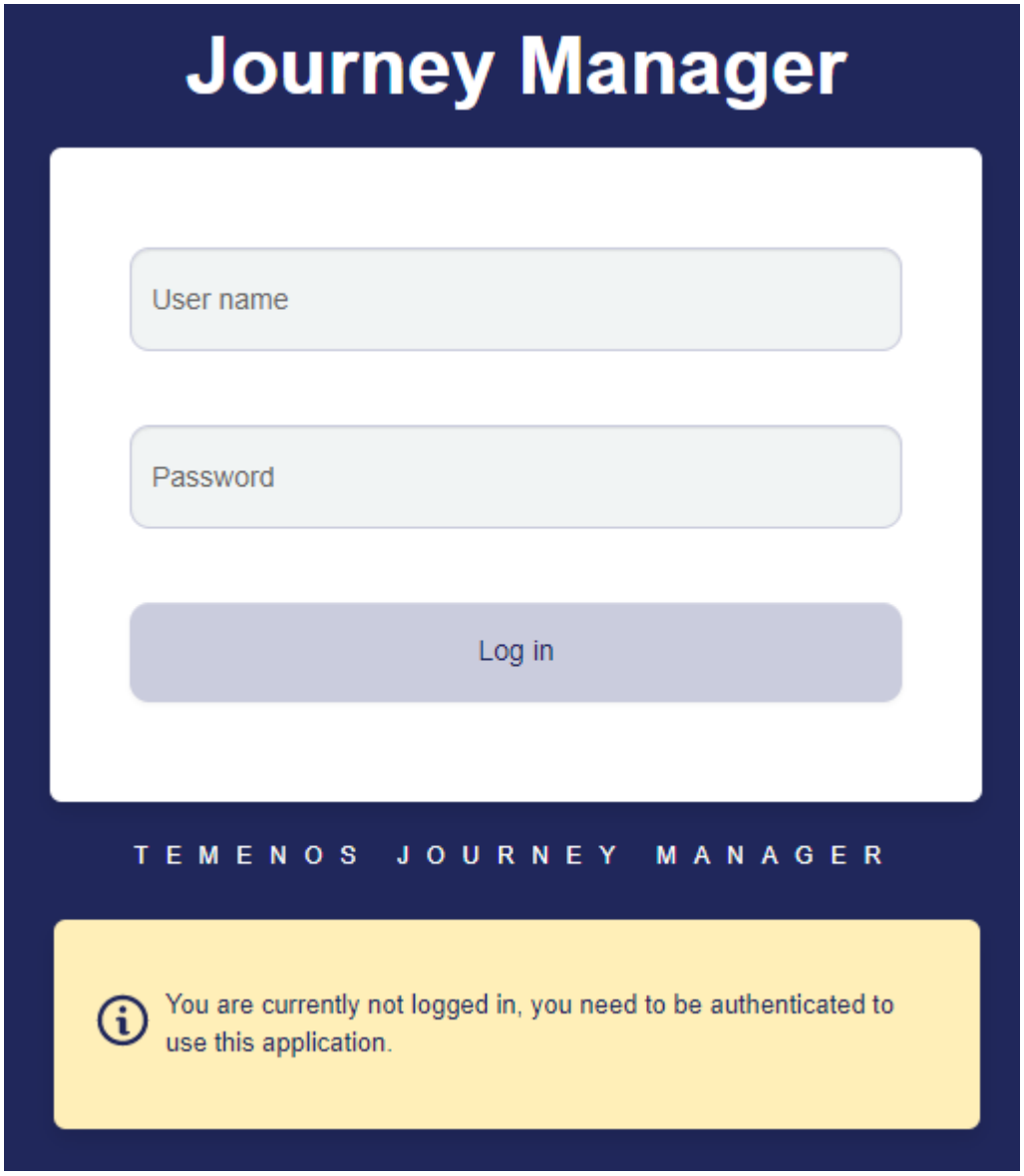
[1]IP address is a unique string of numbers separated by full stops that identifies each computer using the Internet Protocol to communicate over a network.

Log in to Manager

Journey Manager is a highly secure application that requires all users to sign in using their valid credentials, such as a user name and a password. You can also enable Two-factor Authentication[1] to harden the security of your applications, which is especially important in production environments.

When users want to log onto Manager, they must provide valid credentials on the login page.

[1]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.

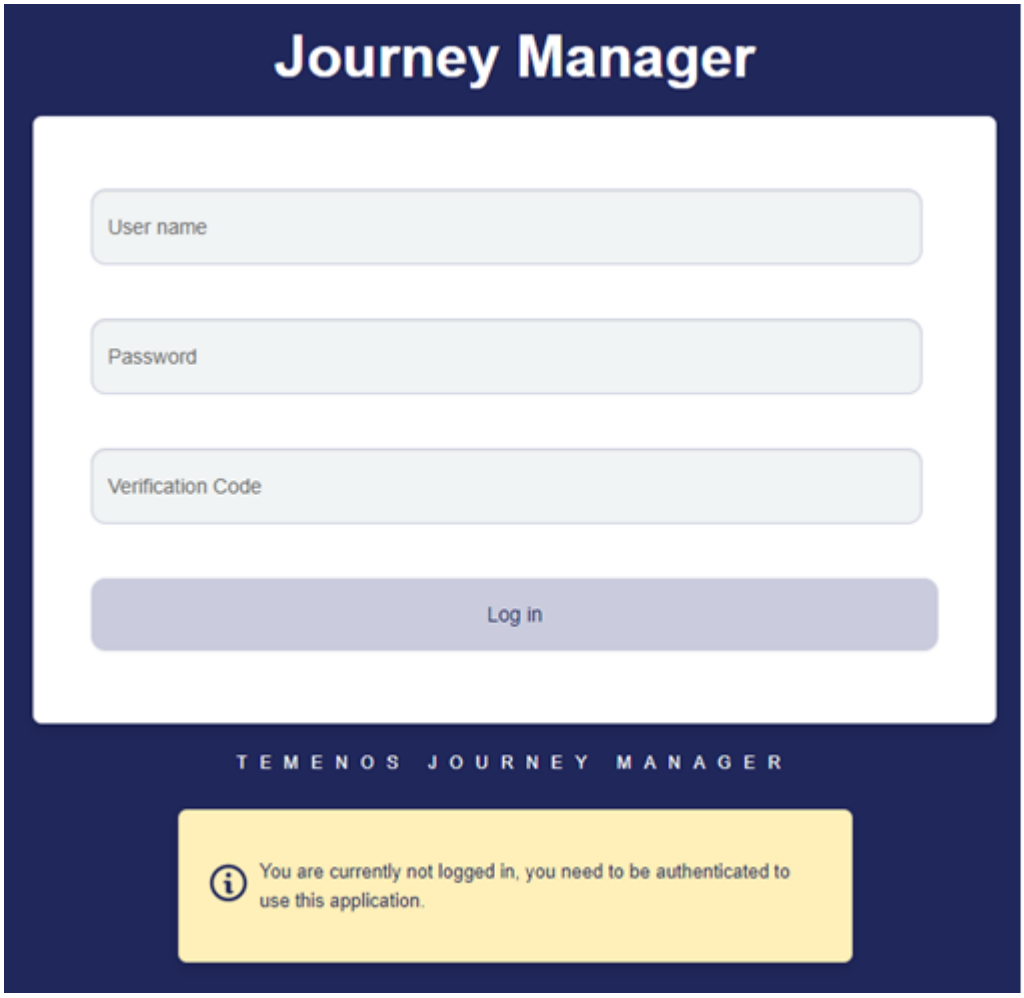


When you create new users, you can also force them to change their system generated passwords at first login by selecting the Change Password After Login checkbox.

A user can be forced to change password due to various reasons, such as, overall system security hardening. Check your security manager configuration as it defines how the login and password change work.

Two-Factor Authentication

To improve application security, you can enable Two-Factor Authentication (2FA)[1] to force a user to provide an extra piece of authentication into the additional Verification Code field during their login to a module or a form space where an app is running. This is shown below:



To enable 2FA:

---

[1]Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security process in which the user provides two different authentication factors to verify themselves to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of assurance than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically a password or passcode. Two-factor authentication methods rely on users providing a password as well as a second factor, usually either a security token or a biometric factor like a fingerprint or facial scan.

1. Configure a local security manager of a form space to use Google Authenticator. This will require users to install Google authenticator app on their mobile devices for 2FA to work.
2. Select the 2 Factor Authentication Required checkbox for each user who wants to use 2FA. | 23.10 This feature was introduced in 23.10
3. Optionally, specify a grace period - the date when 2FA will be enforced. Users can continue to sign in using their username and password during the grace period, but must configure 2FA in order to sign in after the grace period expires.

NOTE

You must have Administrator access and sufficient 2FA knowledge to configure a security manager. We recommend contacting our Customer Service team to configure 2FA for your particular needs.

Next, learn how to log out of Manager.

Change a Password

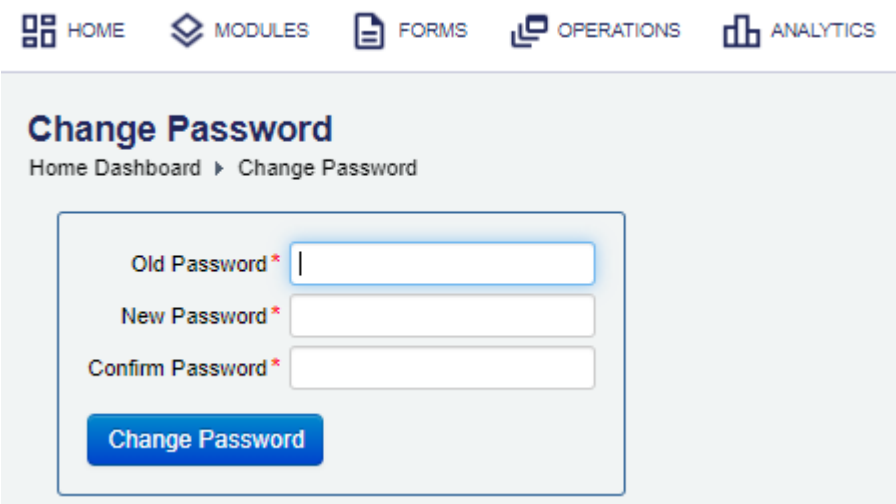
Journey Manager allows you to change a password for logging into your user account on a server. You perform this activity when:

- A password is about to expire or is already expired.
- A password is compromised.
- You are given access to Manager for the first time, so you must change a temporary password.

If Manager is configured to use a Local Security Manager for user authentication and authorization, it will monitor users' passwords expiration and enforce a password change when it has expired. A notification is displayed on the Home Dashboard when a password is about to expire in 10 days with a link to the Change Password UI.

To change a password:

1. Select Security > Change Password.



2. Enter your old password in the Old Password field.
3. Enter a new password in the New Password field.

   The standard Journey Manager password requirements are:

   - Must be at least 12 characters in length.
   - Must not exceed 40 characters. | 20.05 This feature was introduced in 20.05.
   - Must contain at least one or two letters and one digit.
   - Must contain at least one non-alphanumeric character, such as `%`, `&`.
   - Must contain mixed case, one upper case, and one lower case.

- Must not be a password existing in the Illegal Password Values list.

> NOTE
>
> Check specific password requirements of a security manager used to provide an access to Manager in your environment.

The Local Security Manager enforces the following password requirements:

- Must be different from the previously used passwords. The number of old passwords is defined in Number of old passwords. | 24.10 This feature was introduced in 24.10

4. Confirm the new password in the Confirm Password field.
5. Click Change Password.

Next, learn how to log out of Manager from the client.

Log out of Manager

The Journey Manager Home Dashboard provides you with the explicit option to log out of Manager, which is important in a few use cases. One example is using Single Sign-On[1] with a 3rd-party application. If the 3rd-party application user can logout, then login as a different user, the 3rd-party application user is now different than the Manager user. However, Manager will terminate all inactive user sessions after the specified time. You can change this time using the Session Timeout parameter of your security manager. For more information, see Configure a Security Manager.

> NOTE
>
> A user session is considered inactive, if none of the following actions has happened in the specified period of time:
>
> - Form Submit
> - User Save
> - Dynamic Data Service (DDS) or form function trigger call

Here is the actual example that drove the creation of this page: User A logs into Salesforce, gets an SSO account in Manager and passively logs in to Manager. User A then exits the Manager form and logs out of Salesforce. User B then logs in to Salesforce in the same browser. Because the User A, a login session is still running, User B's form is prefilled with User A's data.

The solution is to explicitly log out of Manager when the user exits the form. Here is a non-exhaustive list of possible places where this should be done:

- Cancel or Exit
- Form Submission
- The before unload or unload browser events.

To logout of Manager, make a `GET` request to the `/logout` path from the appropriate form space, which does the following:

- Logs the user out of Manager.
- Clears the Manager-related cookies.

---

[1]Single sign-on (SSO) is a property of access control of multiple related, yet independent, software systems. With this property, a user logs in with a single ID and password to gain access to any of several related systems.

A sample JavaScript code is shown below:

```
/* Logout of Journey Manager. */
function logoutTMInstance() {
    console.debug("Start Manager logout");
    var href = window.location.href;
    var arr = href.split("/");
    var url = arr[0] + '\/\/' + arr[2] + '\/' + arr[3] + '\/logout.htm';
    var xmlHttp = new XMLHttpRequest();
    /* Call service synchronously. Change 3rd parameter to "true" for an asynchronous
call. */
    xmlHttp.open( "GET", url, false);
    xmlHttp.send(null);
    console.debug("Manager logout complete");
};
```

Next, learn how to log in to Manager.

Operations Overview

IT operations refers to a set of processes and services provided by a business's IT department to support the business' IT hardware and software. Manager includes features that support IT operations for Temenos Journey Manager (TJM) solutions. These features allow you to manage and keep track of all transactions associated with user form submissions.

The IT operations activities fall into the following main categories:

- Transactions Search
- System Check
- Logs Check
- Data Retention Check
- Configurations Check
- Reporting

Let's look at each of these activities in more detail.

Transactions Search

Manager logs and stores all transactions pertinent to users' interaction with forms for a specified period of time after which the eligible transactions and their data are deleted from the system. It allows you to search for transactions to track specific details of user form submissions. This is the most common task that a Journey Support Team performs to troubleshoot any form-related issues.

When an issue is identified, you may want to resolve or mitigate it by performing some manual operations. This typically involves changing a transaction's status so an onboarding flow can continue.

> NOTE
>
> Each transaction goes through a well-defined sequence of status changes, so we recommend understanding Submission Life Cycle before changing its status.

Another common case is when a transaction get stuck for some reason, so you have to configure form abandonment or abandon it manually for the transaction to be purged. This is critical to ensure optimal system performance over time.

Most onboarding journey forms are complex single-page applications that rely on collaboration jobs to guide users through the form flows to the final step when they submit a form and form

data is delivered. There can be several collaboration jobs created for each form submission so you may need to view their configurations to troubleshoot them.

You will be performing the abovementioned maintenance activities on a regular basis, so Manager groups them together and they are available from the Operations menu. They are listed below:

- Transaction Support Search
- Form Transactions
- Abandoned Transactions
- Saved Transactions
- Transaction Delivery
- Assigned Tasks
- Payment Transactions
- Form Submission Data
- Collaboration Jobs
- Transaction History

System Check

Manager is equipped with the Home Dashboard allowing you to see the latest transactions and their details, and statuses in particular, to understand if anything needs to be done. From there, you can quickly drill down into the following transaction data:

- Form Sessions
- Form XML Data
- Form Data Extract
- Attachments
- Properties
- Transaction Timeline
- Transaction Events

Another important feature of the Home Dashboard is the System Issues card that lists 5 latest system events that are of importance to overall performance, stability and maintainability of the system. Click the relevant event ID to get more information and action accordingly. Some important services, which run in the background, log events that you may need to review and react to them. The important ones are Database Integrity Checker and Data Retention Management services.

Manager provides the following tools to keep an eye on the server performance:

- Form Request
- Form Rendering Metrics
- JM Server Performance

Manager constantly collects crucial system stats and metrics and you should review them periodically. They are grouped under the System menu:

- Event Listener Status
- System Health

Quite often, you may need to know a server version and a build number to see if any issues are caused by a particular Manager release version. If you require more details, you can find them in System Info. In some cases, you may want to know what 3rd-party libraries our software uses.

Logs Check

Manager generates a significant amount of log to make debugging issues easier. This is especially important to troubleshoot services, Transact functions and collaboration jobs. However, it's important to understand that excessive logging can have a detrimental impact on the system performance.

While investigating an issue, such as a service error or system fault, you may be requited to check the following logs:

- Event Log
- Error Log
- Error Log Trend
- Database Version Log
- Transaction Groovy Log
- Groovy Service Log
- Transaction Service Log

Manager logs all changes made to the system and its configuration into the Audit Log, so you may need to use it now and then. Another important log is the Security Manager Log where user authentication and authorization activities are recorded.

Data Retention Check

The Temenos Journey Manager platform is a System of Engagement[1] that enables you to create onboarding journeys for many types of products. It's strictly designed to store onboarding data for a short period of time, just long enough for a customer to complete their applications, and uses functionality, such as a configurable Data Retention policy to keep the amount of stored data low at any given time.

Temenos Journey Manager is not intended to be used as a System of Record[2] that stores large amount of data as this can affect system performance.

This means you must pay special attention to data retention configuration and check whether data has been purged as desired or not. It's instrumental in maintaining optimal system performance, so you need to act promptly if any abnormalities are found. Any changes to the data retention configurations must be noted and investigated, especially for the Data Retention Management the Data Retention Management v2 services.

Configurations Check

Manager lets you keep track of what is happening to the system and whether any changes have been made to its configuration and by whom. If you suspect the system performance and operation impacted by a recent configuration change, you should take a look at the Audit Log and use the Update operation filter to search for possible incorrect changes.

WARNING

We don't recommend deleting organizations, form spaces, core services, service connections, scheduled jobs, and security managers without proper analysis of possible impact to system data, especially transactions and form submissions in-flight.

---

[1]Systems of Engagement (SoE) is the technology used by an organization to help facilitate and orchestrate the customer journey via more personalized, seamless interactions across the various touchpoints. These include social media channels, email marketing platforms, mobile apps, and content management systems.

[2]System of Record (SOR) is an ISRS (information storage and retrieval system) that is the authoritative source for a particular data element in a system containing multiple sources of the same element. To ensure data integrity, there must be one -- and only one -- system of record for a given piece of information.

Reporting

Manager provides the following license reporting capabilities that you may need to use to verify billing transactions:

- Transaction Licensing Report
- Job Licensing Report

Next, learn how to use transaction support search.

Transactions Overview

A transaction is a fundamental element of Journey Manager, representing a user interaction with a form.

Manager distinguishes between the following transaction types: | 22.10 This feature was introduced in 22.10.

- Billable - transactions associated with customers and applicants
- Non-Billable -transactions associated with business users

NOTE

Prior to version 22.10, these transaction types were called:

- Internal - transactions associated with business users
- External - transactions associated with customers and applicants

Manager allows you to configure transaction types on the form space and form version levels. This gives you fine-grained control of transaction licensing.

A transaction created as a result of a user submitting a form is referred to as a submission.

Each transaction follows a pre-defined life cycle that can vary for different transaction types. Of all transaction life cycles, the submission life cycle is most important, covering the path a user navigates through a form. As such, the form's design is key to ensuring the user has an ideal customer journey.

Next, learn how to view transaction licensing.

Submission Life Cycle

Manager tracks user interaction with forms using submission objects, which are specialized transactions with a well-defined life cycle. A submission life cycle is the series of actions that a user initiates while navigating through a form.

Manager creates a submission object when a user opens a form or a task submission is assigned to a user. As the user progresses through the form and performs form-specific actions, such as save, submit, or payment, Manager knows what the next action is in the submission life cycle, allowing you to build complex form workflows. Manager updates the submission object with user actions and performs additional tasks, such as virus scanning or delivery, and eventually purges it from the system.

Manager provides several pre-defined triggers that you can use to invoke fluent functions that you use to implement a customer onboarding journey.

Submission life cycle consists of two flows:

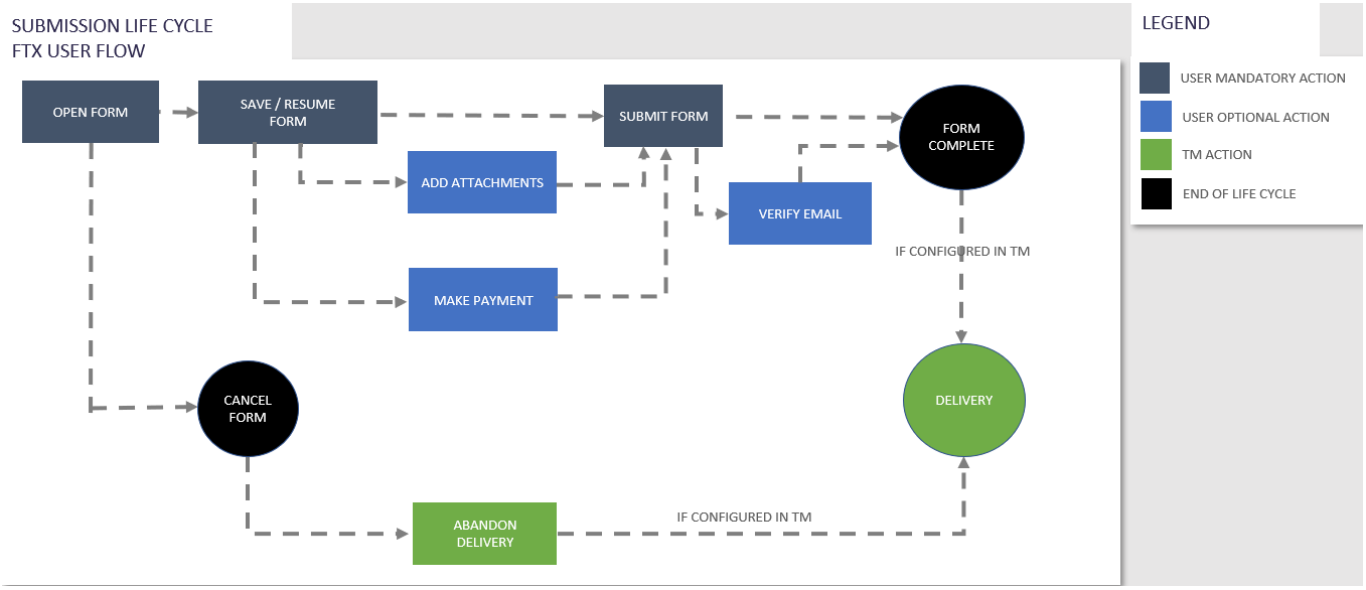
- A user flow, which defines actions a user performs to complete a form, so it can be delivered by Manager.
- A Manager back end flow, which defines actions Manager performs to deliver a form to a user.

Manager supports the following user flows:

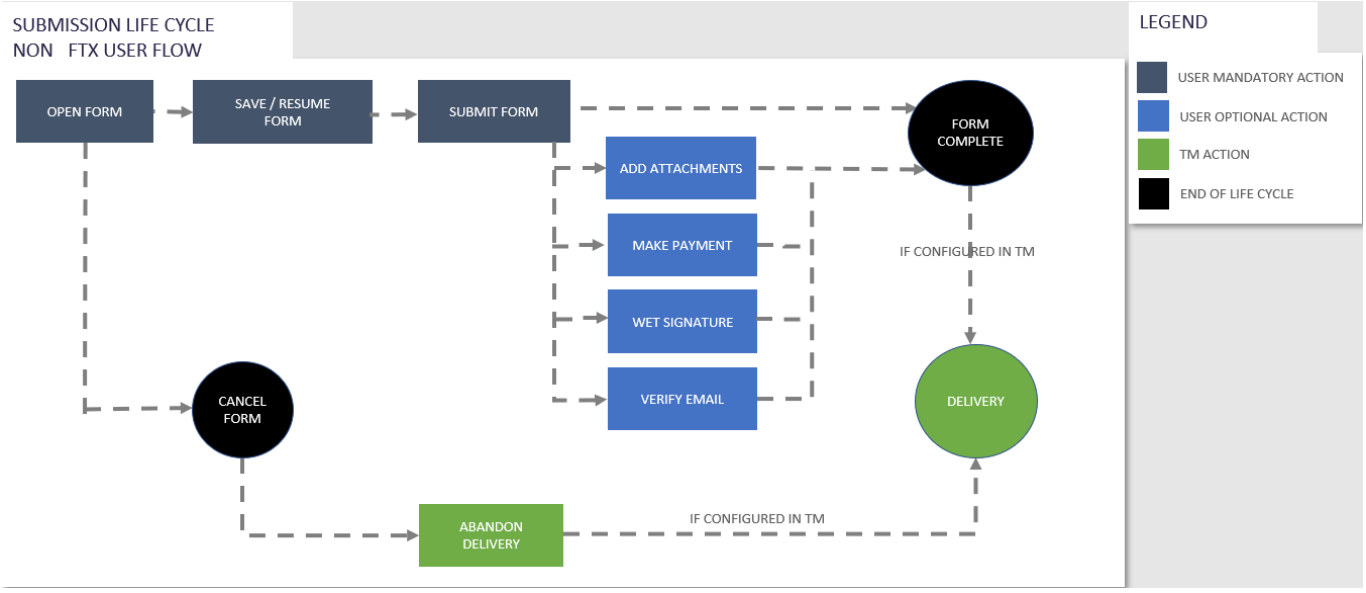
- Form Transaction Experiences (FTX) user flow when a transaction goes straight to Form Complete on form submission as all the attachments and payments are taken before the form is actually submitted. This Includes email verification.
- Traditional (non-FTX) user flow when attachments, signatures and payment are captured after the user submits the form.

These flows are illustrated in the diagrams below.

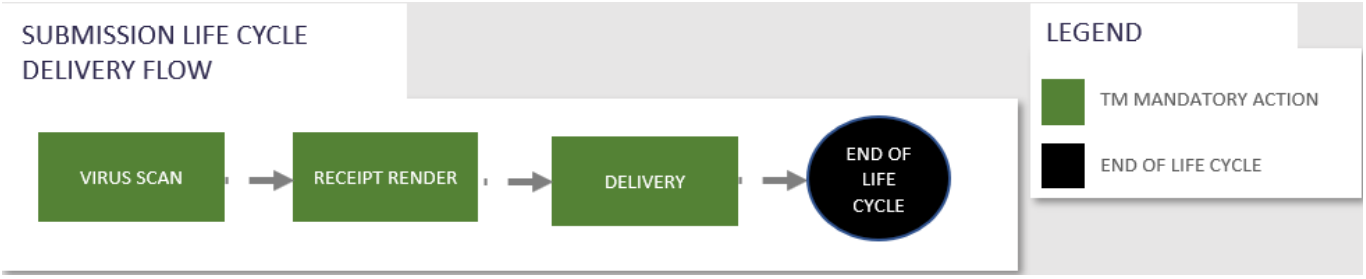
User flow - Form Transaction Experiences (FTX)



User flow -Traditional (Non-FTX)



Delivery flow



The detailed description of submission life cycle actions are listed in the table below.

User Actions

Action	Description	Next Actions
Add attachments	A user adds one or several attachments as a part of the application form. Manager marks the submission form status as Submitted or Completed, and the attachment status as Completed.	Make payment Sign a form
Authenticate a form	An unauthenticated user submits a form that requires authentication. This is often the case after a form has been saved offline. Manager marks the submission form status as Submitted and marks the authentication status as Completed after the user has acknowledged the submission.	Add attachments Make payment Sign a form Verify email address
Cancel a form	A user cancels the form. Manager marks the submission form status as Abandoned.	Abandonment delivery
Make payment	A user completes the payment as a part of the application form. Manager marks the submission form status as Submitted or Completed, and the payment status as Completed.	Sign a form
Open a form	A user opens the form in a browser. Manager creates a submission object, adds relevant entries in the request and submission log, marks the submission form status as Opened, and renders the form to the user.	Save a form Save a form offline Submit a form
Reject electronically signed submission	A user rejects to sign the form. Manager marks the submission form status as Submitted, the signature status as Cancelled, and the signature type as eSignature. The submitted data may be delivered as part of abandonment delivery, if it's configured in Manager.	End of life cycle
Reject submission	A user rejects the form. Manager marks the submission form status as Submitted and the authentication status as Cancelled.	End of life cycle

Save a form offline	A user saves the form offline (locally on the computer). The submission record is not updated in Manager. It's relevant for PDF forms.	Submit a form
Save a form	A user saves the form. Manager marks the submission form status as Saved.	Save a form offline Submit a form
Sign a form	A user signs the form and adds it as an attachment. Manager marks the submission form status as Submitted or Completed, the signature status as Completed, and the signature type as Wet.	Delivery Receipt render Virus scan Add attachments Authenticate a form Make payment Sign a form Verify email address
Submit a form	A user submits the form. Manager marks the submission form status as Submitted, if the user need to do any further tasks, otherwise, it's marked as Completed.	
Verify email address	Manager sends an email to users asking them to click on a link to verify their email address. This action happens in parallel with other actions, such as adding attachments and payment. Manager marks the submission form status as Submitted and the email verification status as Completed after the user has verified the email address. NOTE Manager must have submission email verification enabled.	Add attachments Make payment Sign a form

NOTE

These actions might not occur in every workflow, or they may occur in a different order. For example, attachments and payments can be completed in either order. This is subject to the form workflow implementation.

Manager Actions

Action	Description	Next Actions
Delivery	This is the final step. It occurs after the user has completed the submission and when all other Transaction Manager tasks have been completed successfully.	End of life cycle
Receipt render	This is done when a form is submitted, either when the user requests a receipt or as part of a routine job. If a form requires eSignatures, the generated receipt will contain the electronic signature of the organization hosting the form.	Virus scan
Virus scan	If a virus scanning service is configured, all submission attachments will be scanned for viruses and submissions will not be completed until the scans have finished.	Delivery

NOTE

At any stage of the submission life cycle, an error can be caused by a user or Manager. For more information about an error, check the submission events for a particular transaction.

Next, learn how to view form transactions.

Submission Categories

Manager categorizes transactions depending on which part of transaction life cycle they are currently in. There are three distinct submission categories:

- Completed submissions
- Abandoned submissions
- Submissions as part of collaboration jobs

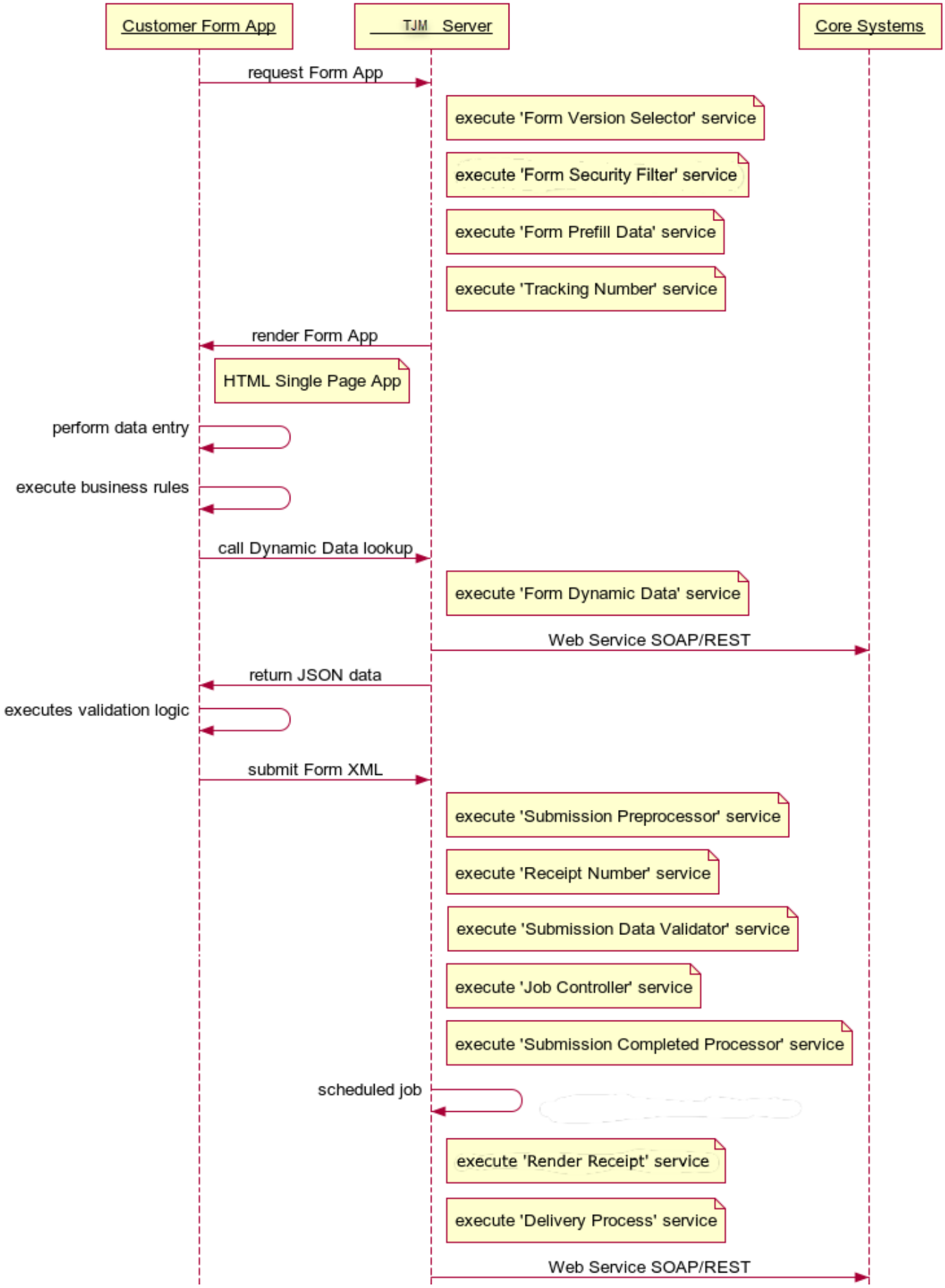
You need to understand the differences between these categories to configure Manager's data retention policies, which are used by the Transaction Processing scheduled job to perform the following operations:

- Abandons any transactions that are ready to be abandoned.
- Purges any data and transactions that are ready to be purged.

Next, learn about the transaction processing sequence.

Transaction Processing Sequence

Manager comes with the Transaction Processing scheduled job to perform various operations with transactions. The sequence diagram below illustrates key service extension points available during this transaction processing.


Customer Form App
TJM Server
Core Systems
request Form App
execute 'Form Version Selector' service
execute 'Form Security Filter' service
execute 'Form Prefill Data' service
execute 'Tracking Number' service
render Form App
HTML Single Page App
perform data entry
execute business rules
call Dynamic Data lookup
execute 'Form Dynamic Data' service
Web Service SOAP/REST
return JSON data
executes validation logic
submit Form XML
execute 'Submission Preprocessor' service
execute 'Receipt Number' service
execute 'Submission Data Validator' service
execute 'Job Controller' service
execute 'Submission Completed Processor' service
scheduled job
execute 'Render Receipt' service
execute 'Delivery Process' service
Web Service SOAP/REST

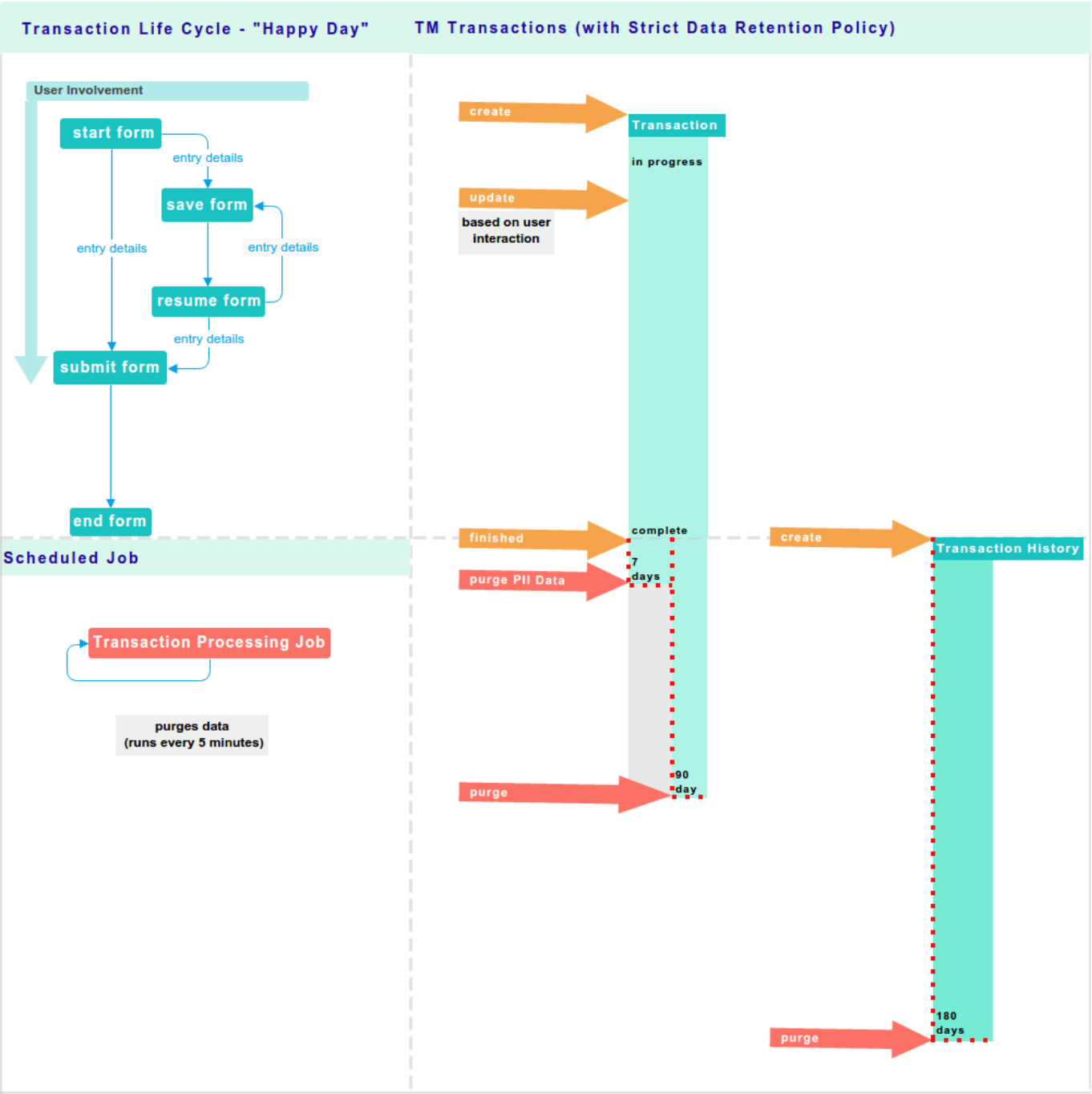
The diagram mentions the following services, which are typically involved in transaction processing:

- Form Version Selector
- Form Security Filter
- Form Prefill Data
- Tracking Number
- Form Dynamic Data
- Submission Preprocessor
- Receipt Number
- Submission Data Validator
- Job Controller
- Submission Completed Processor
- Render Receipt
- Delivery Process

Next, learn how to view transaction licensing.

Completed Submissions

Manager tracks all form submissions including their statuses. Each transaction goes through a life cycle. It's important to understand the transaction cycle so you know where each transaction is at and how to troubleshoot them. One of the most importance life cycles is the completed transaction cycle for a form submission. It is shown below.



This is a Happy Day scenario where the user completes and submits a form within the allowed time frame that is established by the saved transaction data retention setting, which is 30 days for strict policy.

The following steps describe this process:

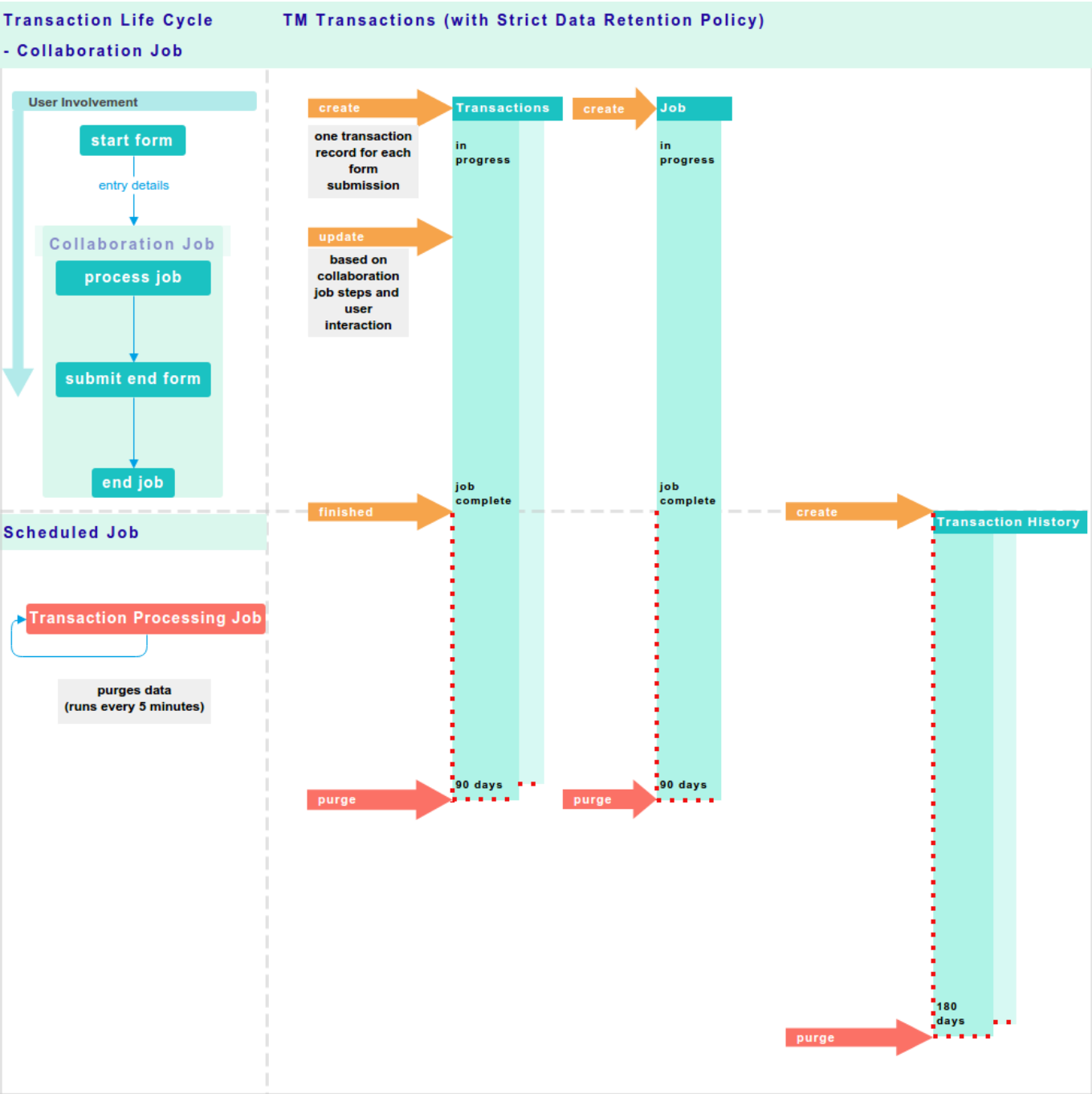
1. The form is started - opened - by the user and a transaction record is created. At this point the user may:
    - Complete and submit the form in one sitting or;
    - Save the form and return to it, subsequently completing and submitting it within the allowable timeframe. Before submitting the form, the user can save and resume the form as many times as needed. Each time the user resumes and saves the form, the timeframe allowed, configured by the saved transaction data retention setting, begins from this save. That is, the user can return and submit the form prior to the 30 days (for strict policy) having elapsed from when it was last saved.
2. On submission, the form is processed and delivered to the backend system.
3. Once delivered, the transaction is considered a finished transaction and a transaction history record is created.
4. From this point onwards, the transaction record and the transaction history record will be retained in Manager according to the data retention settings.
5. The Finished Transaction PII Data setting is used to determine when to remove PII data from the finished transaction. PII data makes up most of a submission's data footprint. Removing the PII data for a brief time (7 days for strict policy) after a transaction has been completed, has the following benefits:
6. The Manager database is leaner. The purge will remove submitted XML and attachment data.
7. All sensitive data is only available for a brief time.
8. When PII data has been deleted, the submission is still listed in form transactions, but the XML data can't be viewed, and attachments can't be downloaded anymore.
9. The Finished Transactions setting is used to determine when to purge the completed transaction from the transaction record. For the strict policy, the transaction will be deleted 90 days after the transaction has finished.
10. The Transaction History setting (in the System Logs section) is used to determine when to purge the related transaction history data. For the strict policy, the transaction history record will be deleted 180 days after the transaction has finished.

Next, learn about submission categories.

Collaboration Job Submission

If a submission is part of a collaboration job, it is not purged as part of standard submission purging. Instead, Manager uses the Finished Collaboration Jobs policy to purge the collaboration job and all its submissions, once the entire job has finished. It is important to emphasize here that, in contrast with a single form submit, a form submission that is part of a collaboration job is never automatically abandoned by Manager.

The image below illustrates the transaction life cycle for a collaboration job form submission that has been successfully completed.



The list below describes the process illustrated in the above diagram.

1. The form is started (opened) by the user and a transaction record is created. As this form involves a collaboration job, once the user submits the start form, it triggers the collaboration job processing which first creates a job record. This job record contains the job audit information and keeps track of all submissions related to the collaboration job.
2. One or more users who are involved interact as requested by the collaboration job. Typically, each user interaction is a separate submission reflected by creating a transaction record.

3. The last step is reached and the user responsible submits the end form.
4. On submission of the end form, the forms in the collaboration job are processed and delivered to the backend system.
5. Once delivered, it is considered a finished collaboration job reflected by a finished job record and a related set of one or more finished transactions. A transaction history record is now created for each of the finished transactions within the collaboration job.
6. From this point onwards, each transaction record and the corresponding transaction history record are retained in Manager according to the data retention settings.
7. The Finished Collaboration Jobs setting is used to determine when to purge the job record and all its related completed transactions. For the strict policy, these are deleted 90 days after the collaboration job has finished.
8. The Transaction History setting (in the System Logs section) is used to determine when to purge the related transaction history data. For the strict policy, the transaction history records are deleted 180 days after the collaboration job has finished.

However, a collaboration job may also be cancelled by a Manageroperations user. In this case, the incomplete job is marked as finished, and the same data retention procedure applies like a successfully completed job.

Next, learn about submission categories.

Abandoned Submissions

A transaction can be abandoned in several ways:

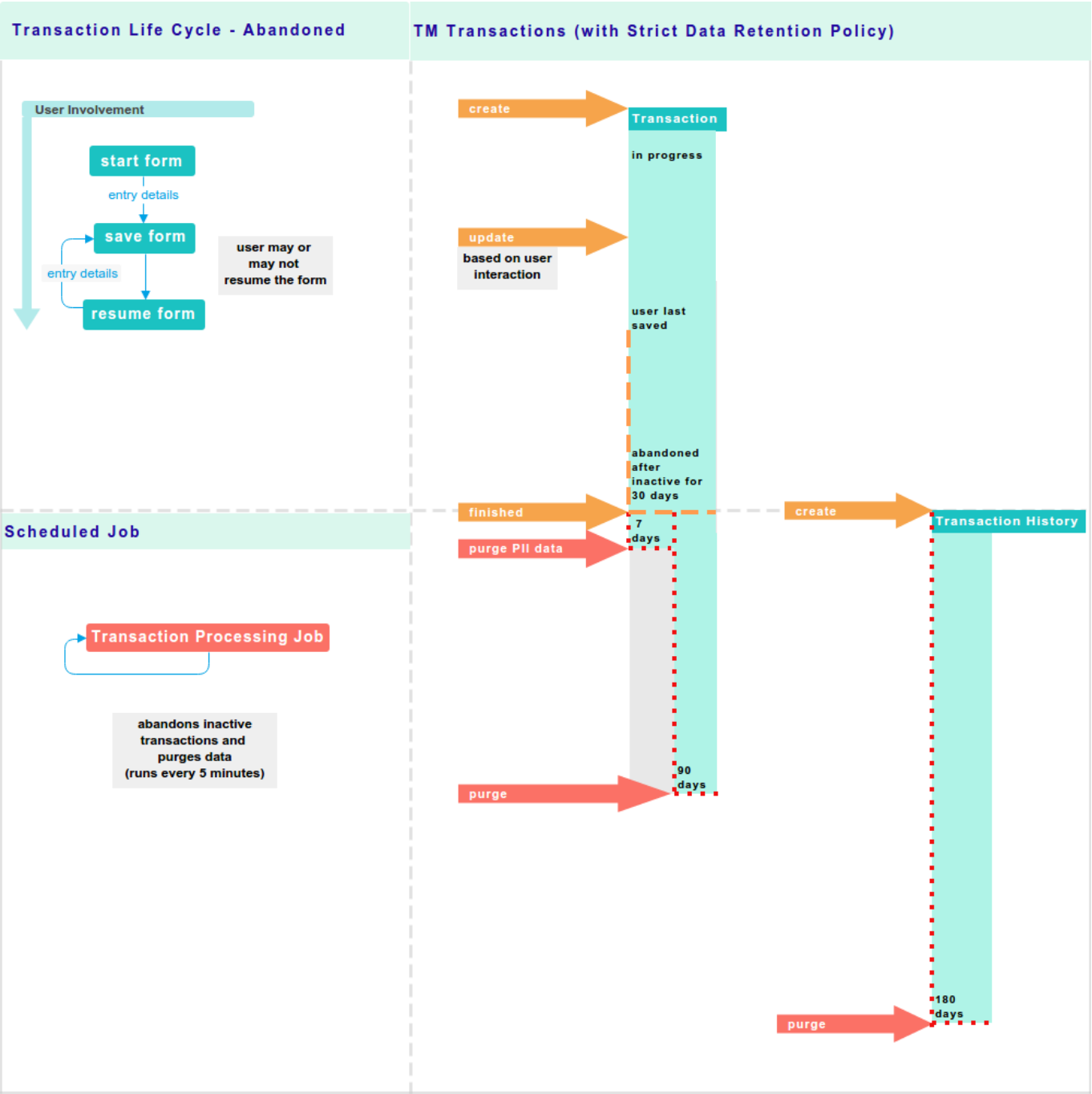
- Explicitly by the user clicking the Cancel button or closing the browser while still on the first page. Please note that some forms may automatically save the transaction as the user navigates between different pages. In this case, the form will temporarily be marked as Saved. If the user does not return to the form and save or submit, it will be marked as abandoned.
- Explicitly by an administrator canceling from the Managerconsole (only when necessary).
- Automatically by a Manager job, where a user:

- Saved the form but did not resume it in a timely fashion.
- Submitted the form but it remained incomplete, waiting for say missing attachments.

- Automatically by a form session timing out.
- Unintentionally by say some break in internet communication.

It is a balancing act to optimize conversion rates whilst keeping your database as lean as possible. As with the case of a saved form, it is hoped that the user will soon resume, complete, and submit it. We advise that sufficient time is allowed for all users to resume their forms before automatically abandoning them. According to the strict policy, the Saved Transactions setting gives the user 30 days to return to a previously saved form.

The image below illustrates an example where a transaction, that was previously saved, is abandoned automatically after a period of inactivity (based on the Saved Transactions setting).



The following steps describe this process:

1. The form is started (opened) by the user and a transaction record is created.
2. The user saves the form with the intention of completing it later.
3. However, the user does not resume the form within the allowed time frame (30 days as per the saved transactions setting of the strict policy).
4. After the 30 days, Manager automatically abandons the transaction.

5. An abandoned transaction is considered a Finished Transaction. At this stage, a transaction history record is created, and the same data retention procedure applies like a successfully submitted and delivered (Completed Transaction) submission.

In most cases, abandoned transactions are deleted according to the Finished Transactions setting. The only exception is where a user opens the form and leaves it open without having saved it (this includes any background saving). In this case, the transaction is automatically abandoned one day after first opened and then it's deleted 2 days after being abandoned.

Data for abandoned submissions is delivered if the abandoned delivery channel is configured in TM.

Next, learn about submission categories.

Authenticated vs Unauthenticated Transactions

Manager supports both authenticated or unauthenticated transactions.

When a transaction is authenticated, the user completing the transaction has logged into and been identified by Manager, usually by supplying a username and password. When a transaction is unauthenticated, the user completing the transaction has not logged in to and doesn't have an identity with Manager.

Manager provides similar features to both authenticated and unauthenticated users, but the features are implemented differently. There are different solutions available in both cases. It's important to understand that a transaction and a user are two different entities. There might be solutions where a SSO user account is created of a form transaction, for example, when you onboard a client and let them to work on other form tasks as authenticated users. In that case, you would create a user account record on the back of the onboarding or customer registration form transaction.

In other cases, unauthenticated clients access a public form available form a link on a customer's website, then submit the form, so no user account is involved at all and the forms are submitted as anonymous transactions.

There are many solutions where the relationship between a user and transactions are:

- 1 user : many transactions - an adviser onboards customers via Workspaces
- many users : many transactions - complex collaboration jobs with multiple parties participating in work
- no user : many transactions - public forms submitted by unauthenticated users

Form Pre-fill

Authenticated	Unauthenticated
When a user is authenticated, their identity, including the data stored about them in back-end systems, can be used to partially complete	When a user is unauthenticated, data can be pre-filled by using request parameter prefill mapping, which allows pre-fill data to be passed to a transaction through HTTP[1] request parameters, session attributes, or cookies. After that, any personal details from

[1]HTTP (HyperText Transfer Protocol) is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

forms with information that is already known about them.	the form can be used to call a dynamic data service to recover all missing anonymous user data from a back-end system so the form is prefilled.

Form Save & Resume

Authenticated	Unauthenticated
When a user is authenticated, they can save a partially completed transaction and associate it with their identity. Later, if they log back into their account, the incomplete transaction will be available to them.	When a user is unauthenticated, the save and resume functionality relies upon a reference code[1], similar to an airline booking. Optionally, a security question can be defined to provide a basic layer of security.

Transaction History

Authenticated	Unauthenticated
When a user is authenticated, they can log into Manager to see a history of their transactions. Generally, they only see a summary of the transactions, as data retention policies will often result in the details being purged.	When a user is unauthenticated, one method of providing a transaction history is sending a confirmation email to the user once the transaction is complete.

Tasks

Authenticated	Unauthenticated
When a user is authenticated, tasks can be assigned to them in Manager. These tasks will then appear in their task list.	When a user is unauthenticated, tasks can be assigned to them by directing them to a dynamically generated URL. When they visit this URL, they'll be prompted to perform the task.

Best Practices

Based on the content covered in this topic, you might be wondering if it's better to use authenticated or unauthenticated transactions. As with most decisions of this nature, it depends on exactly what your organization is trying to achieve.

Consider the following details:

---

[1] A reference number is a unique identifier assigned to each transaction.

- Many organizations already have their own methods of authentication integrated with their back-end systems and would rather not introduce a new mechanism of logging in via a username and password.
- Users are often reluctant to sign up with a username and password purely for the sake of saving and resuming a transaction. If they're forced to register, many users will abandon the transaction.
- Identity — and specifically, the identity of customers outside your organization's firewall — can be difficult to implement. Manager does support industry-standard technologies, such as LDAP[1] and Single Sign-On[2] (using SAML), but these do add complexity to the implementation process.

We generally recommend the following:

- External customers should be treated as unauthenticated users. This may have an impact on the ability to pre-fill data in their forms, but this approach simplifies the integration requirements, the security implications, and the time to market.
- Internal users, such as help desk and call center staff, should be treated as authenticated users. This is usually simpler to implement as internal users will be already be authenticated on the network, and implementing LDAP or Single Sign-On is relatively straight-forward.

In some cases, organizations will require external clients to be authenticated, and as we've covered in this topic, this is most certainly possible. it will, however, affect both the time and cost of implementation.

Next, learn how to change the access type for a form space.

---

[1]Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol network.
[2]Single sign-on (SSO) is a property of access control of multiple related, yet independent, software systems. With this property, a user logs in with a single ID and password to gain access to any of several related systems.

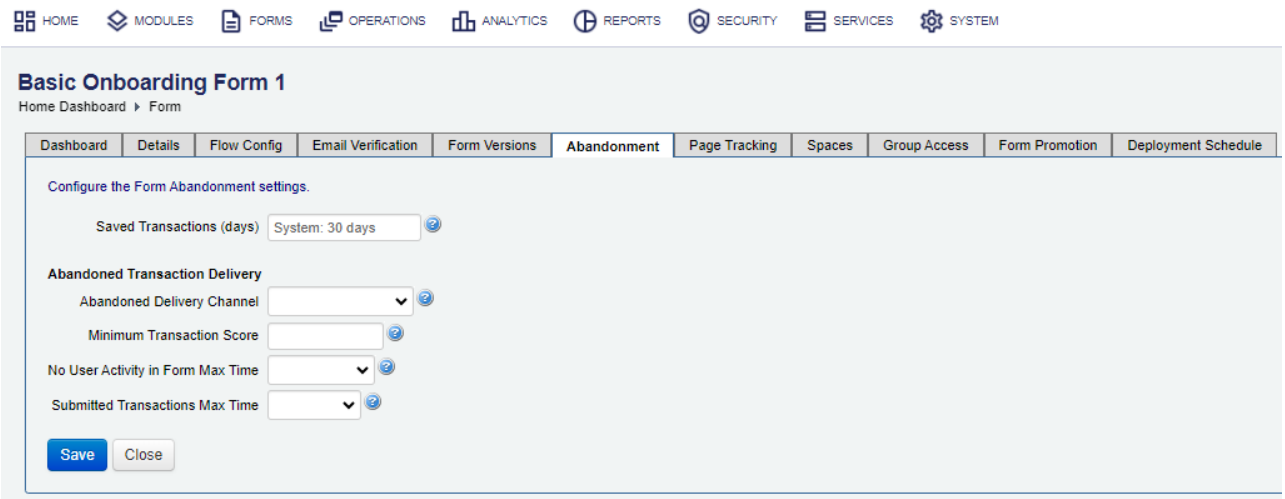
Configure Form Abandonment

Transactions can become abandoned either actively due to a user action, such as canceling a transaction, or passively based on the form's abandonment settings. You can use abandonment configuration to define when a form is considered as abandoned[1] after it has been saved, and what happens after a form transaction is abandoned. Manager allows you to configure abandonment rules of form's transactions at a global, organization, or form level. These configurations depend on:

- Customer requirements
- Server performance
- Specific data retention policies

To configure form abandonment:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Click the Abandonment tab.



4. Specify the maximum age of user saved transactions after which they will be automatically abandoned (marked as Abandoned) in the Saved Transaction (days) field. The default is 30 days, which is either the organization data retention setting, if exists, or the system data retention setting. When a saved transaction reaches this maximum age, it is marked as

[1]Any transaction that has been deemed as Abandoned in Manager. When a user leaves the application without explicitly canceling it, the transaction lives in Started status for a period defined in Manager, after which it is considered as an Abandoned transaction.

Abandoned by the system. The maximum age is based on the number of days a form has been untouched since it was last saved.

5. Select the organization delivery channel for delivering abandoned submissions from the Abandoned Delivery Channel dropdown list. The next three fields are enabled when a delivery channel is selected.

   NOTE

   Manager version 17.10 and higher has the following fields available for Composer forms only. However they are still displayed for Maestro forms.

6. Edit the Minimum Transaction Score to specify the transaction score that a form transaction must reach for it to be delivered to the Abandoned Delivery Channel. A transaction score is a numerical value calculated by the form, generally based on how much of the form was filled in. When left blank, all abandoned form transactions are delivered. Otherwise, only those that have been “adequately” filled in, meaning they have reached the Minimum Transaction Score, are delivered. Minimum Transaction Score specifies a minimum value representing how much of the form must be filled in for it to be delivered to the abandoned delivery channel. That is, a transaction score is a numerical value calculated by the form to indicate how much of the form was filled in. If you would like only submissions that exceed a certain transaction score to be delivered, enter the minimum transaction score into this field.
7. Select a time from the No User Activity in Form Max Time dropdown list to specify a time frame for allowing any form transaction by an anonymous user to remain untouched before marking it as abandoned. This is the period of time after which anonymous transactions are marked as abandoned. These transactions have been saved in the background and untouched by a user.

   NOTE

   A form transaction must also have been background saved for this setting to apply.

8. Select a time from the Submitted Transactions Max Time dropdown list to specify a time frame for allowing any form transaction that has been submitted by a user to remain incomplete before marking it as abandoned. This is the maximum age before a submitted but incomplete, for example missing attachments, transaction are marked as abandoned.

NOTE

A form transaction may be incomplete because it’s waiting for a post-submission requirement such as an attachment to be provided by the user.

9. Click Save to update the changes.

NOTE

To troubleshoot user form interactions, you may want to view abandoned transactions.

Next, learn hot to configure form details.

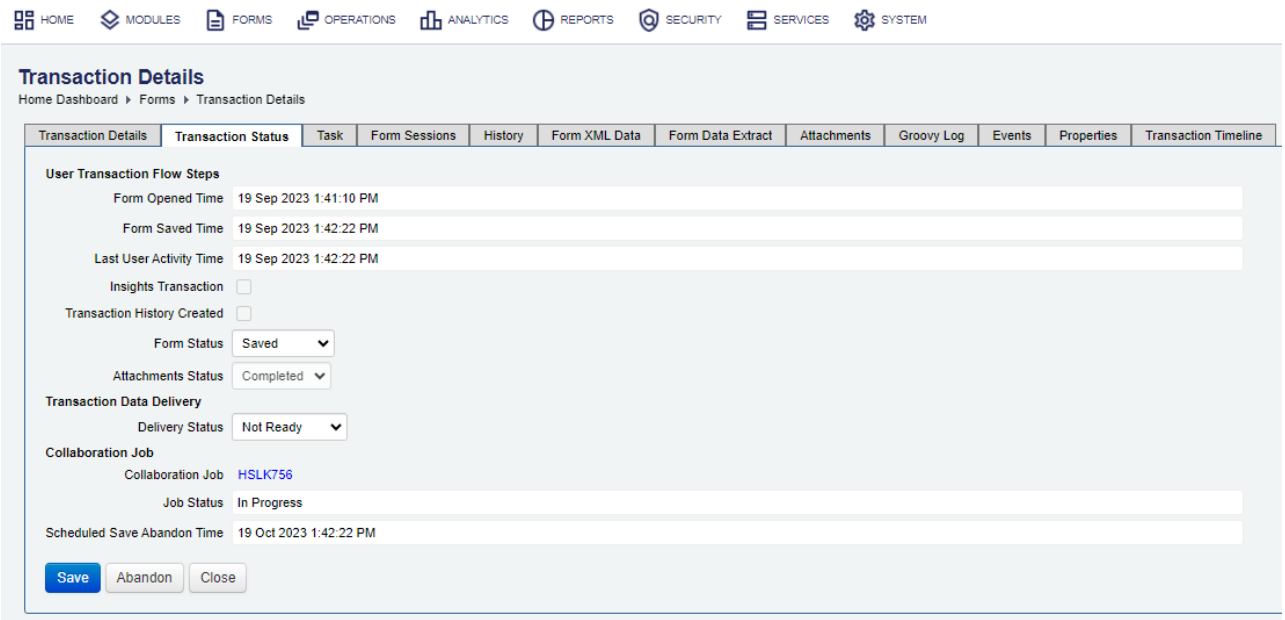
Abandon a Transaction

Journey Manager allows you to view abandoned transactions so you can inspect them and make some corrective actions, such as changing their status.

One of the actions is to abandon a transaction manually, which you may want to use to purge the transaction.

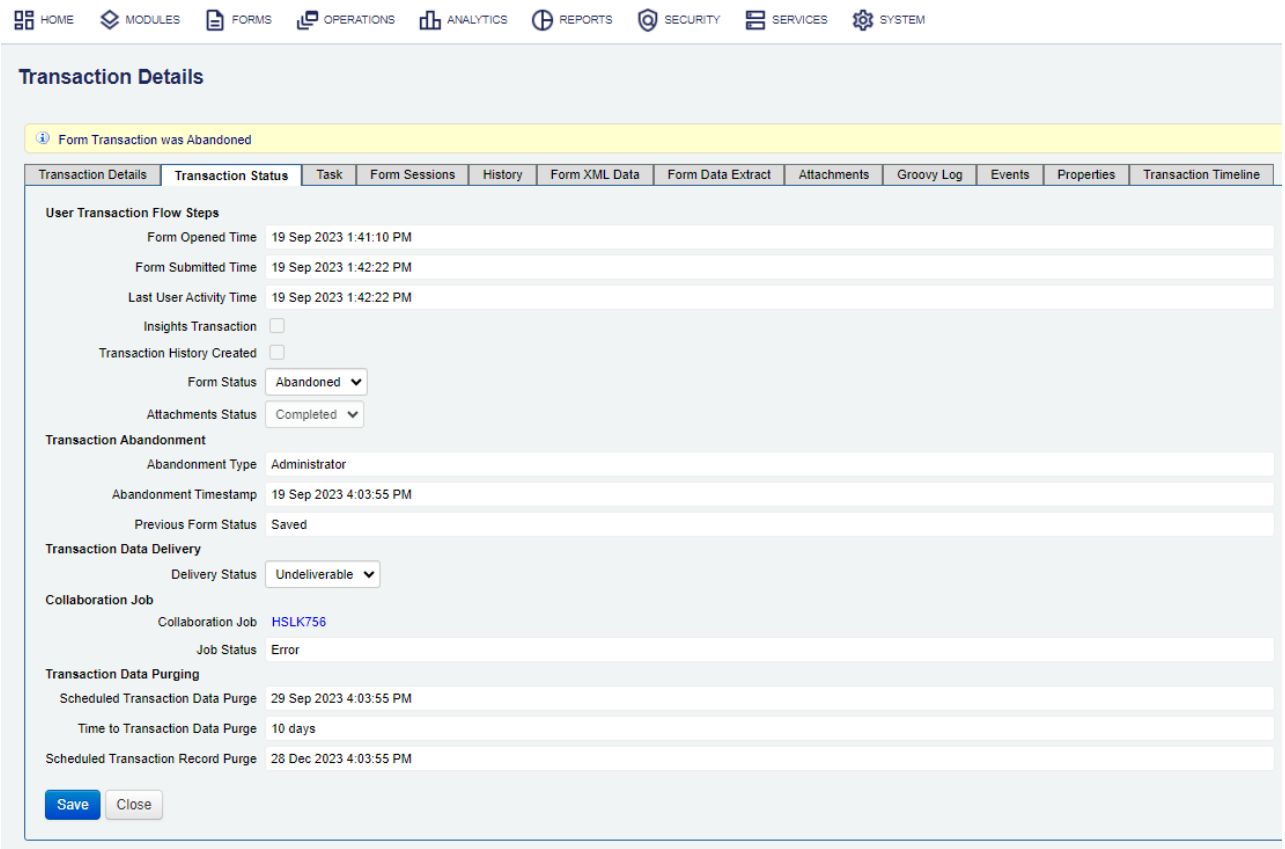
To abandon a transaction:

1. Select Operations > Form Transactions or Operations > Saved Transactions or Forms > Form.
2. Locate a form and click ID in the Latest Transactions section or locate a transaction and click View.
3. Select the Transaction Status tab to update a form status.

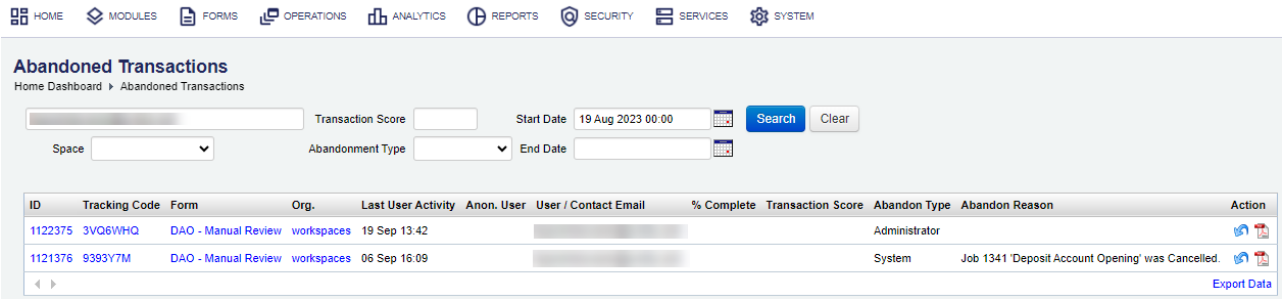


4. Select Abandoned from the Form Status dropdown list or click Abandon and confirm your action.
5. Click Save to update the changes. The transaction is moved to the abandoned transactions.

6. Check the additional transaction details on the Transaction Status tab.



7. View Abandoned Transaction to confirm the transaction is in the list.



NOTE

If you have abandoned a transaction by mistake, you can revert this action and set the transaction status to Saved before it purged. To do this, click Make Abandoned Transaction Saved and check it disappeared from the list.

Next, learn how to view saved transactions.

Transactions Abandonment

When a user submits a form, Manager creates a transaction, associated with this form. As a result of transaction processing and user inputs, it can become completed or abandoned. Manager considers abandoned transactions as completed transactions, so they can be purged. Abandoned transactions can also be configured to deliver data depending on a threshold value, which determines how incomplete form data was before the transaction became abandoned. Manager stores abandoned transactions along with all other transactions that are part of the transaction life cycle.

In addition to transactions, tasks and collaboration jobs can be abandoned. Also, it all can be done automatically or manually.

Automatic Abandonment

Automatic abandonment can happen under varying conditions depending on your Manager configuration. Manager applies the following rules to govern when draft submissions are automatically abandoned. The rules use a set of data retention parameters that you may need to configure for your particular requirements.

Transaction Abandonment

Data retention policy determines how long a saved transaction is maintained in Manager before it is automatically abandoned. Forms with abandonment delivery can have a more aggressive abandonment policy than the standard 1 day (24 hours) policy.

Form Status	Standard	Abandoned Delivery
Open / Started	Abandoned after 1 day	Not applicable
		Abandon as per the form level No User Activity in Form Max Time setting. Transaction will then be delivered if the transaction score threshold has been exceeded.
Background Saved	Abandoned after 1 day [1]	Transactions which are rendered but never actually captured any data or transactions which are never saved by a user, even though they are background saved by Maestro Editor or using the `Form.backgroundSave()` Maestro API.
User Saved	Abandoned	Abandon after the data retention setting for Saved

	after data retention setting for Saved Transactions (days)[2]	Transactions (days)[2]
Submitted	Is not abandoned automatically [3]	Abandon as per the form level Submitted Transactions Max Time setting. Transaction will then be delivered if the transaction score threshold has been exceeded.
Expired	Abandoned 7 Days after expiry	Abandon 7 Days after expiry

NOTE

[1] - You can't change this 1 day abandonment policy time interval.
[2] - Saved Transactions (days) can be set at the global, organization and form levels.
[3] - Transactions in this state are mainly legacy. Normally, with a FTX flow, transactions will be completes before the transaction submits. When a transaction is left in the submitted state, it will be waiting for attachments or payments and the expectation is that help desk staff or administrators will be reviewing these transactions and either helping the users complete the form or manually abandoning them.

Task Abandonment

Tasks with configured expiration time are abandoned after this time has elapsed. If no expiration time is set, then the save abandonment time is used.

Form Status	Standard	Abandoned Delivery
Assigned / Open / Started	Abandoned after data retention setting for Saved Transactions (days) [1]	Not applicable
Background Saved	Abandoned after data retention setting for Saved Transactions (days) [1]	Abandon as per the form level No User Activity in Form Max Time configuration. Transaction will then be delivered if the transaction score threshold has been exceeded.

User Saved	Abandoned after data retention setting for Saved Transactions (days) [1]	Abandon after data retention setting for Saved Transactions (days) [1]
Submitted	Is not abandoned automatically [2]	Abandon as per the form level Submitted Transactions Max Time setting. Transaction will then be delivered if the transaction score threshold has been exceeded.
Expired	Expiry time overrides the above for Assigned, Open, and Started statuses and the tasks will abandon as per the expiry setting.	Expiry time overrides the above for Assigned, Open, and Started and the tasks will abandon as per the expiry setting.

NOTE

[1] - Saved Transactions (days) can be set at the global, organization and form levels.
[2] - Transactions in this state are mainly legacy. Normally, with a FTX flow, transactions will be completes before the transaction submits. When a transaction is left in the submitted state, it will be waiting for attachments or payments and the expectation is that help desk staff or administrators will be reviewing these transactions and either helping the users complete the form or manually abandoning them.

Collaboration Job Abandonment

The Transaction Processor service automatically abandons a collaboration job when either its Last Processed Time or Creation Time parameter expires, that is, when it is greater than a specified period of time, which you define as Collaboration Jobs Abandonment in the Data Retention configuration on both the system and organization levels. You can choose which submission attribute - the Last Processed Time or Creation Time - to be used for abandonment by configuring the Abandon Job based on Time Created deployment property.

If you have the Abandon Jobs based on Creation Time property set to `true`, make sure that the Collaboration Jobs Abandonment time is greater or equal of a total expiry time of all steps and tasks of a longest collaboration job flow, so In Progress collaboration jobs are not abandoned before they are actually completed. For example, if a collaboration job has 3 sequential steps or

tasks, and we allow 20 days for each task to complete, the Collaboration Jobs Abandonment time should be set to at least 60 days.

When the collaboration job is abandoned, all its transactions are also abandoned. A purge date of the collaboration job, the subscription data, including PII data, are set according to the data retention configuration.

| 22.04 This feature was introduced in 22.04.

Collaboration Job Status	Standard	Abandoned Delivery
In Progress Error Paused	Abandoned after a period of time specified in the Collaboration Jobs Abandonment of data retention configuration	Transactions, associated with a collaboration job, will be abandoned and their delivery status will set to Not Required
Completed Canceled Expired	Purged after a period of time specified in the Finished Collaboration Jobs of data retention configuration. The collaboration jobs with these statuses can't be abandoned.	A delivery status of completed transactions, associated with a collaboration job, will be updated as Not Required, the delivery date will be set to the current date and submission's delivery message will be set to `Job xyx id 123 was abandoned`

For more information, see collaboration job's statuses.

Transaction and Task Purging

Form Status	Date	Record
Open / Started	Not applicable	Transactions which were abandoned from this state will be purged in a fixed 2 day period
Abandoned / Completed	Purged as per the Finished Transaction PII Data (days)	It'll be purged after delivery (if required) as per the Finished Transactions (days)

NOTE

For Collaboration Jobs transaction data purging is a per standard policy but transaction record purging is different and the record will be purged when the collaboration job is purged. Collaboration Job purging is controlled by the 'Finished Collaboration Jobs (days) ' data retention setting.

Manual Abandonment

There are situations when manual abandonment is required. It can come from:

- A user via Workspaces
- A system administrator via the Manager dashboard
- A Fluent API call using the `new TxnUpdater(txn).setFormStatus(Txn.FORM_ABANDONED).update()` method. For more information, see Fluent API documentation for your specific Manager version.

Next, learn how to view submissions history.

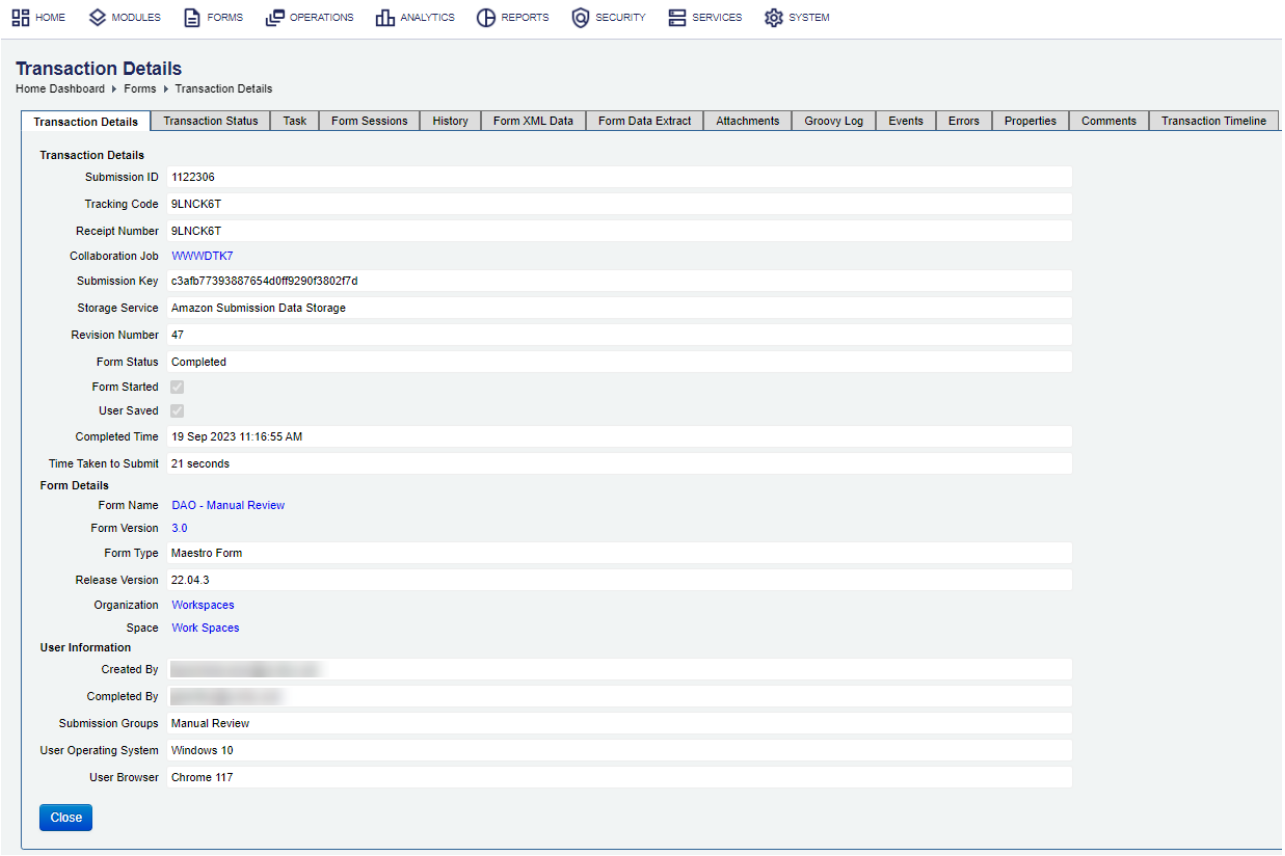
Transaction Details

Manager allows you view detailed information of a selected transaction associated with a form. This is handy when you troubleshoot form submissions.

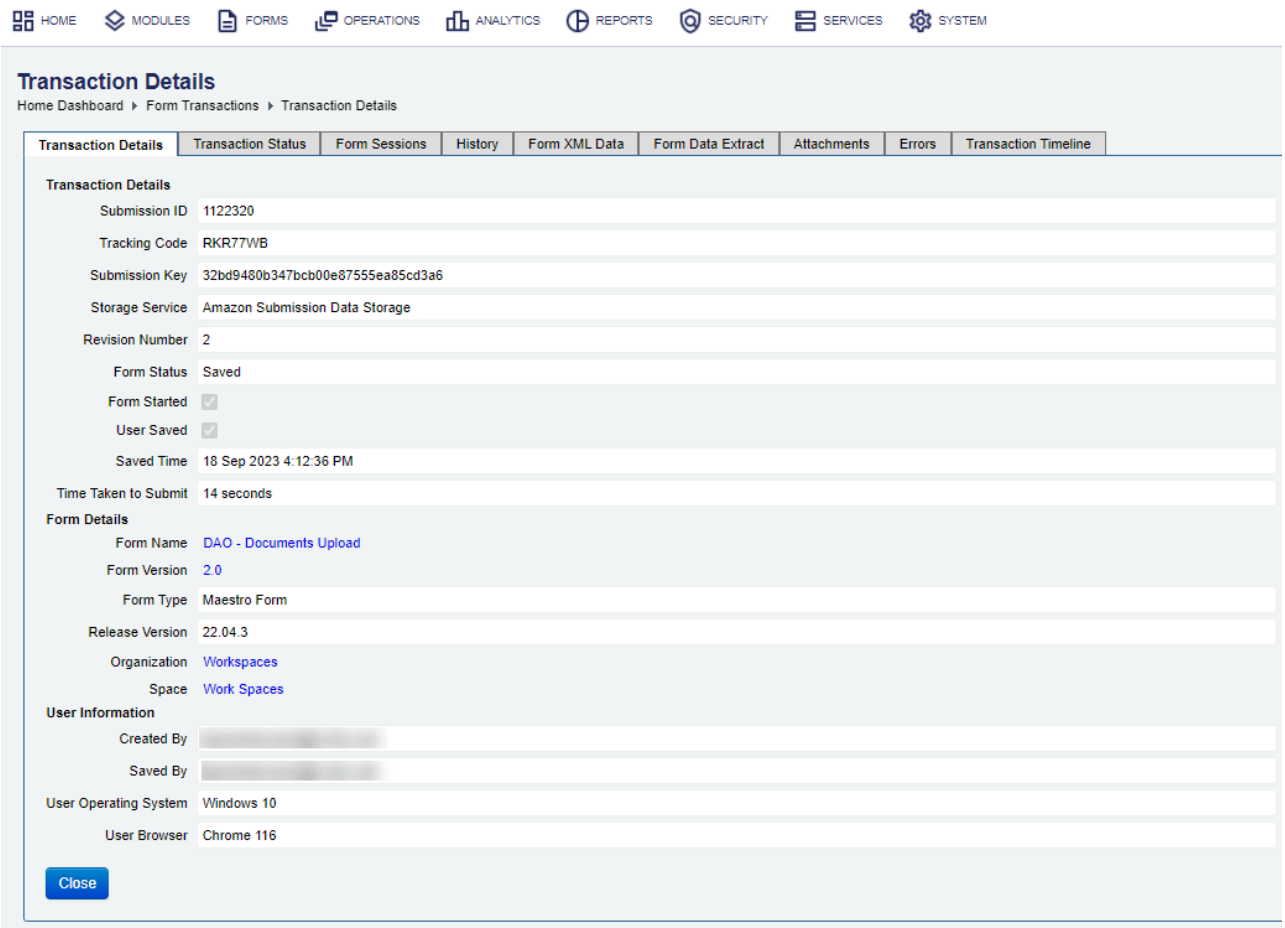
To view transaction details:

1. Select Operations > Form Transactions or Operations > Saved Transactions or Forms > Form.
2. Locate a transaction and click View to see transaction details. Depending on the status and content of a selected transaction, you can see different transaction details as well as additional tabs. For example, if the transaction encountered any errors, the Errors tab is displayed where you can check these errors. This is illustrated below for 3 transactions with different form statuses:

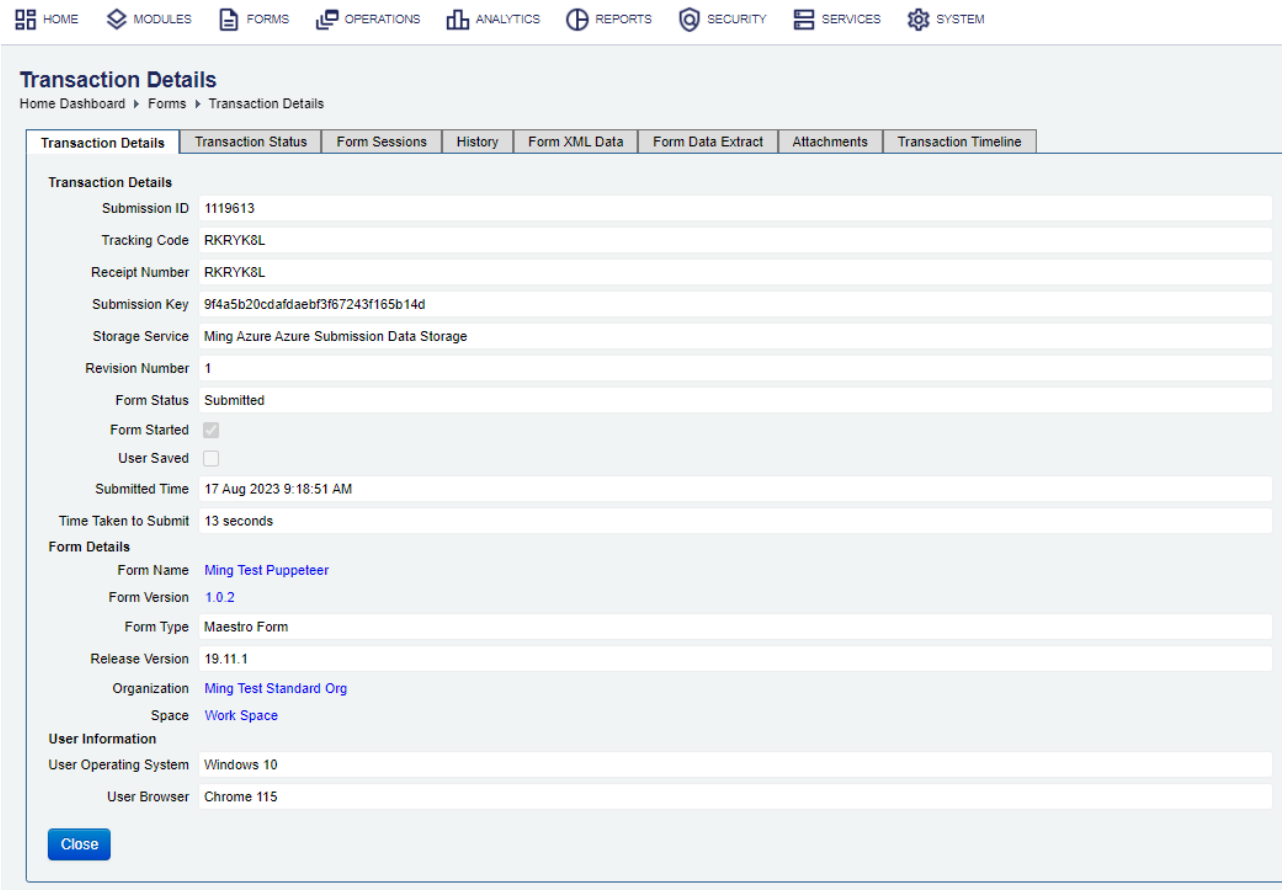
   Completed



Saved


HOME
MODULES
FORMS
OPERATIONS
ANALYTICS
REPORTS
SECURITY
SERVICES
SYSTEM
Transaction Details
Home Dashboard ▸ Form Transactions ▸ Transaction Details
Transaction Details
Transaction Status
Form Sessions
History
Form XML Data
Form Data Extract
Attachments
Errors
Transaction Timeline
Transaction Details
Submission ID 1122320
Tracking Code RKR77WB
Submission Key 32bd9480b347bcb00e87555ea85cd3a6
Storage Service Amazon Submission Data Storage
Revision Number 2
Form Status Saved
Form Started
User Saved
Saved Time 18 Sep 2023 4:12:36 PM
Time Taken to Submit 14 seconds
Form Details
Form Name DAO - Documents Upload
Form Version 2.0
Form Type Maestro Form
Release Version 22.04.3
Organization Workspaces
Space Work Spaces
User Information
Created By
Saved By
User Operating System Windows 10
User Browser Chrome 116
Close

Submitted



3. Check the Transaction Details section, including:
    - Submission ID
    - Tracking code
    - Receipt number
    - The collaboration job number associated with this transaction
    - Storage service used to persist this transaction
    - Revision number
    - Form status
    - Form started
    - Used saved - shows whether a user saved a form or it was background saved
    - Saved time - a time stamp when the form was saved
    - Completed Time - a time stamp when the form was completed
    - Time Taken to Submit
    - Submission Expiry Time - a date, defined in form details, which was used to ensure a form can't be submitted on or after this date
4. Check the Form Details section, including:

- Form Name
- Form Version
- Form Type
- Release Version
- Organization - an organization associated with the form
- Space - a form space associated with the form

5. Check the User Information section, which documents the following details of the user who completed the transaction:
    - Created by - a contact email address of a user that created the transaction
    - Completed by - a contact email address of a user that created the transaction
    - Submission Group
    - User Operating System - an OS that was used to create the transaction
    - User browser - a browser that was used to create the transaction

Next, learn how to change a transaction status.

Transaction Status

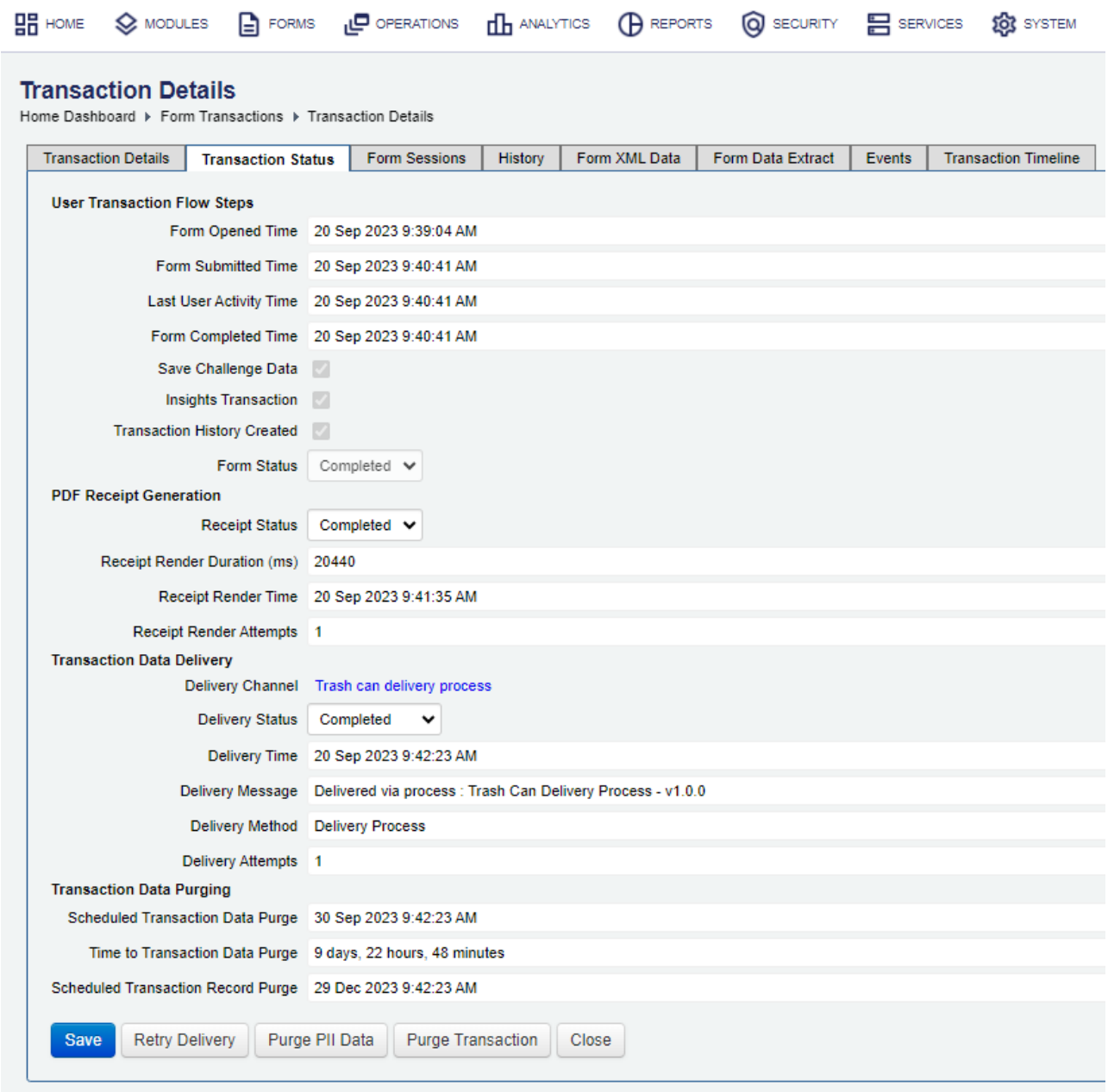
Manager allows you to check and modify a status of a transaction, so you can correct a transaction flow. When Manager creates a transaction, it assigns it a status based on the current progress of the transaction.

To check a transaction status:

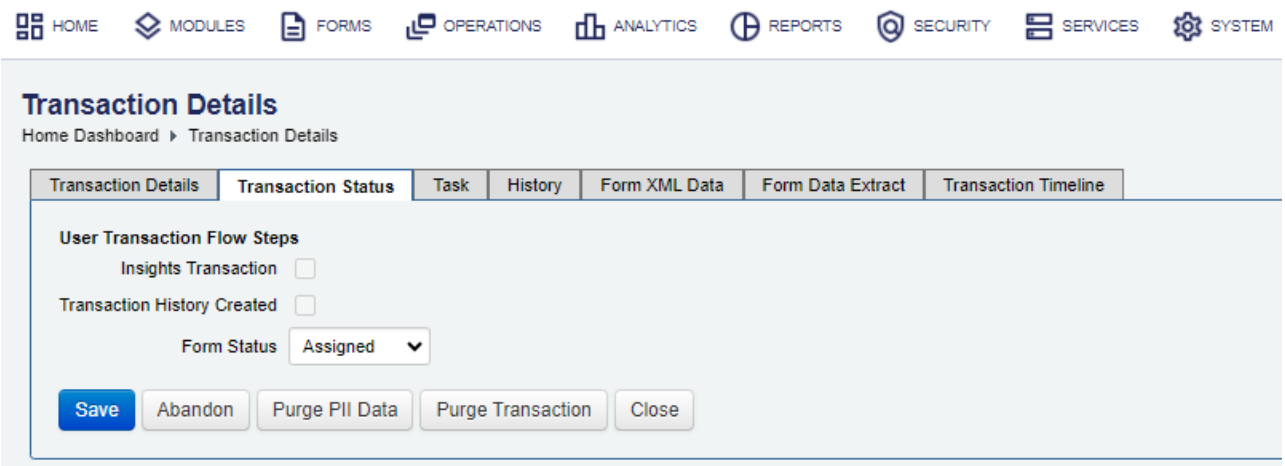
1. Select Operations > Form Transactions.
2. Locate a transaction and click View.
3. Select the Transaction Status tab to view relevant form's, receipt's, and delivery's properties that depend on a current status of a form.

   A different set of properties is shown for the same transaction as its status changes. This is illustrated below:

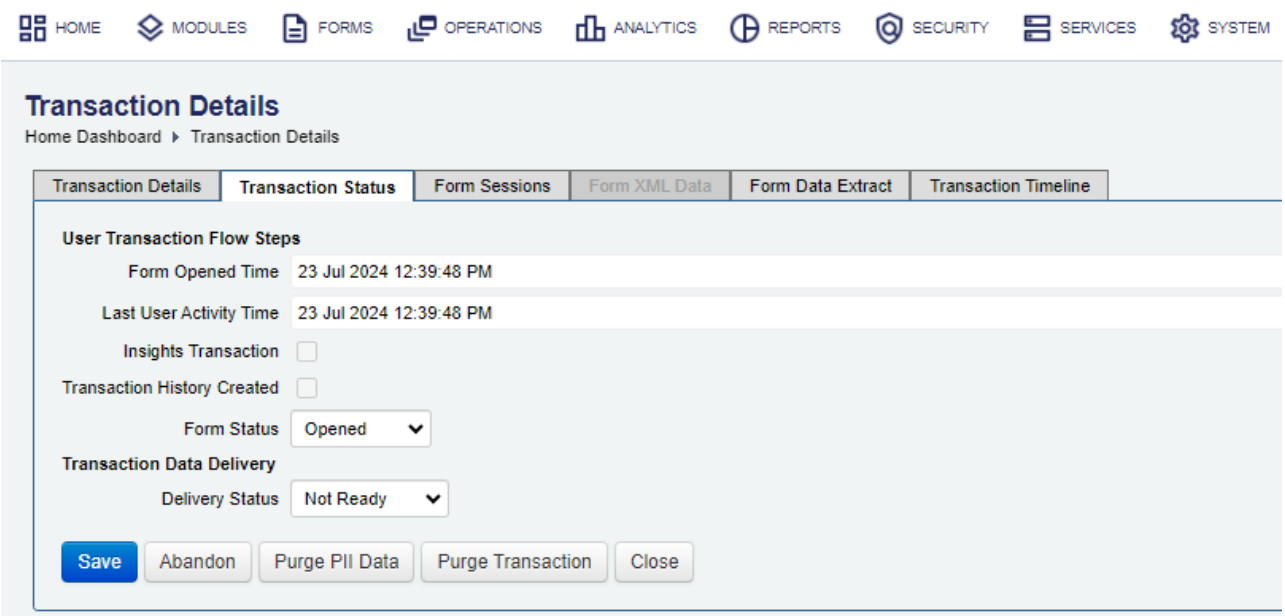
   - Assigned
   - Opened
   - Saved
   - Submitted
   - Completed
   - Completed with a Collaboration Job
   - Abandoned
   - Expired



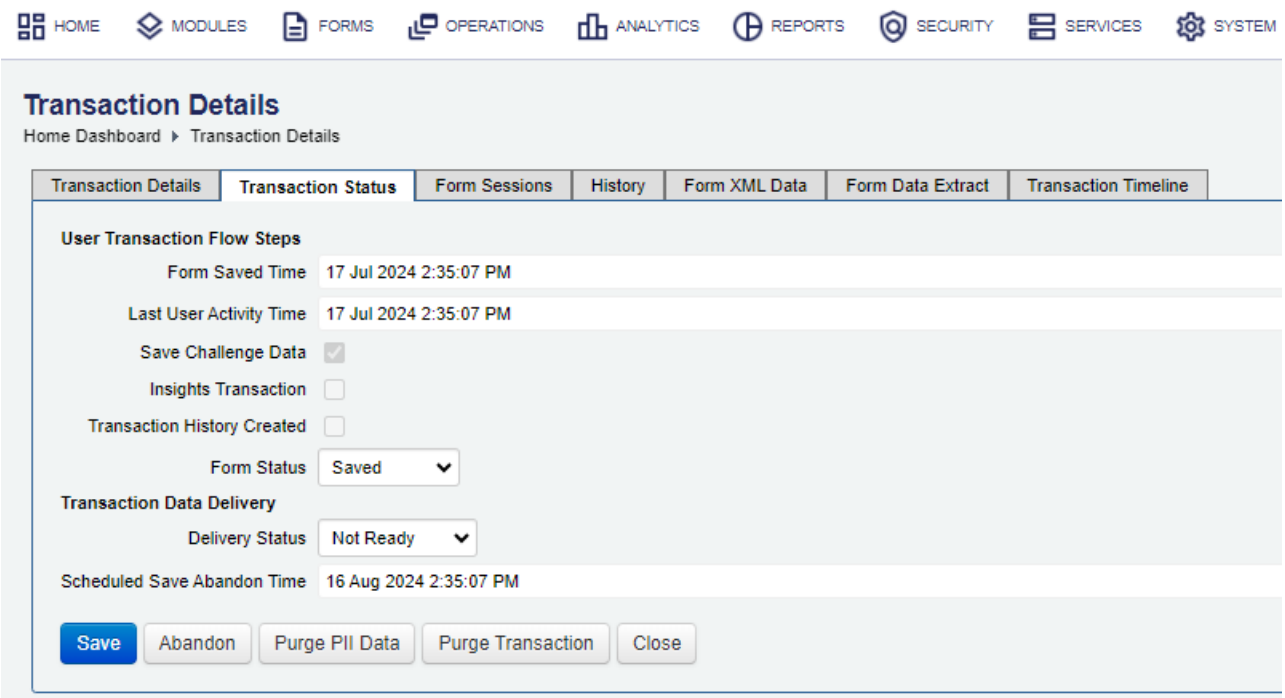
Completed without collaboration jobs



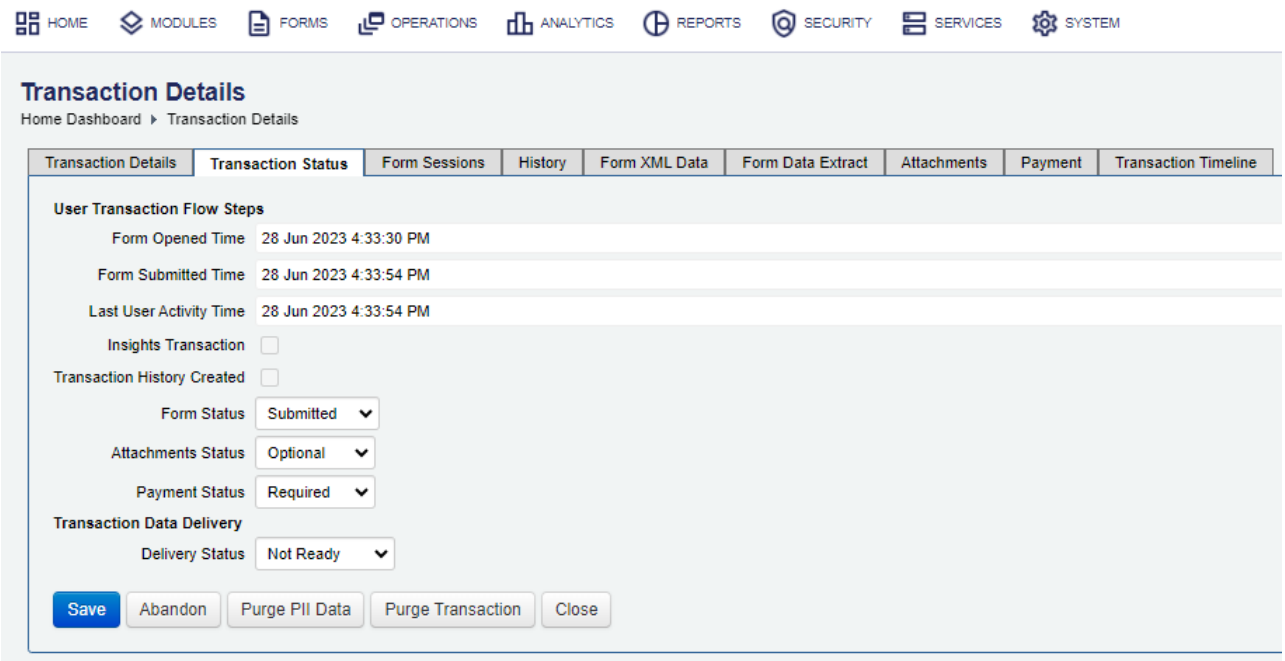
Assigned



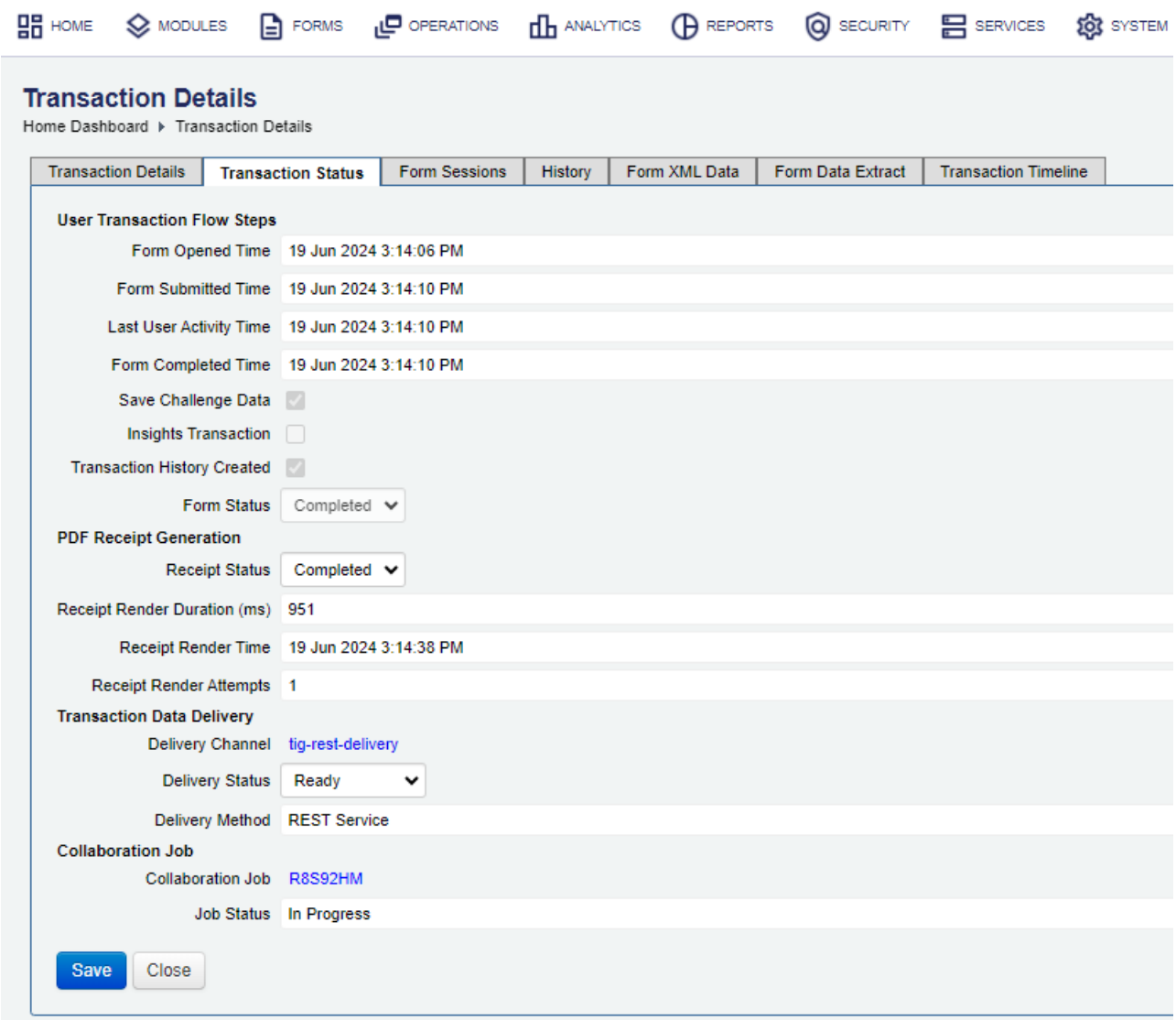
Opened



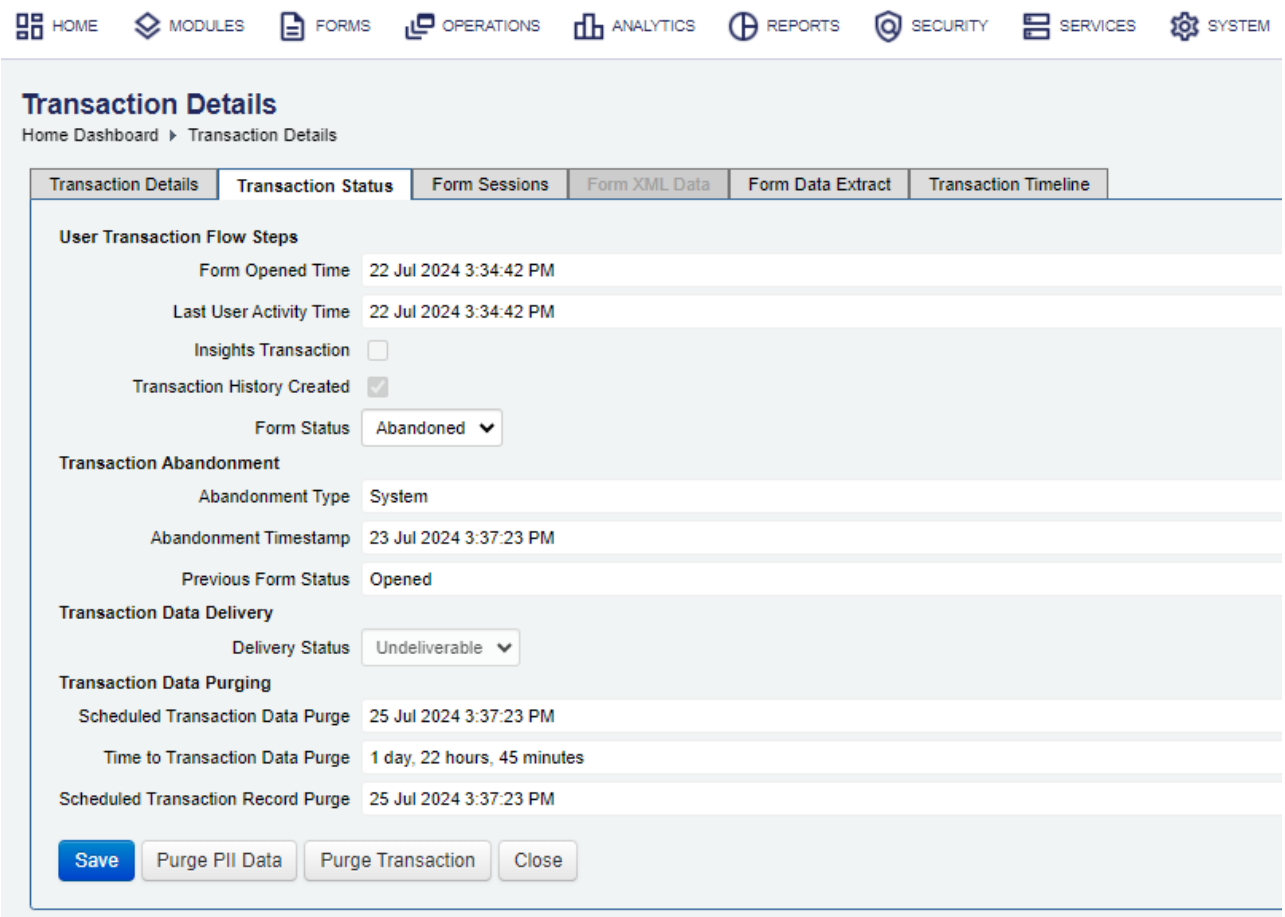
Saved



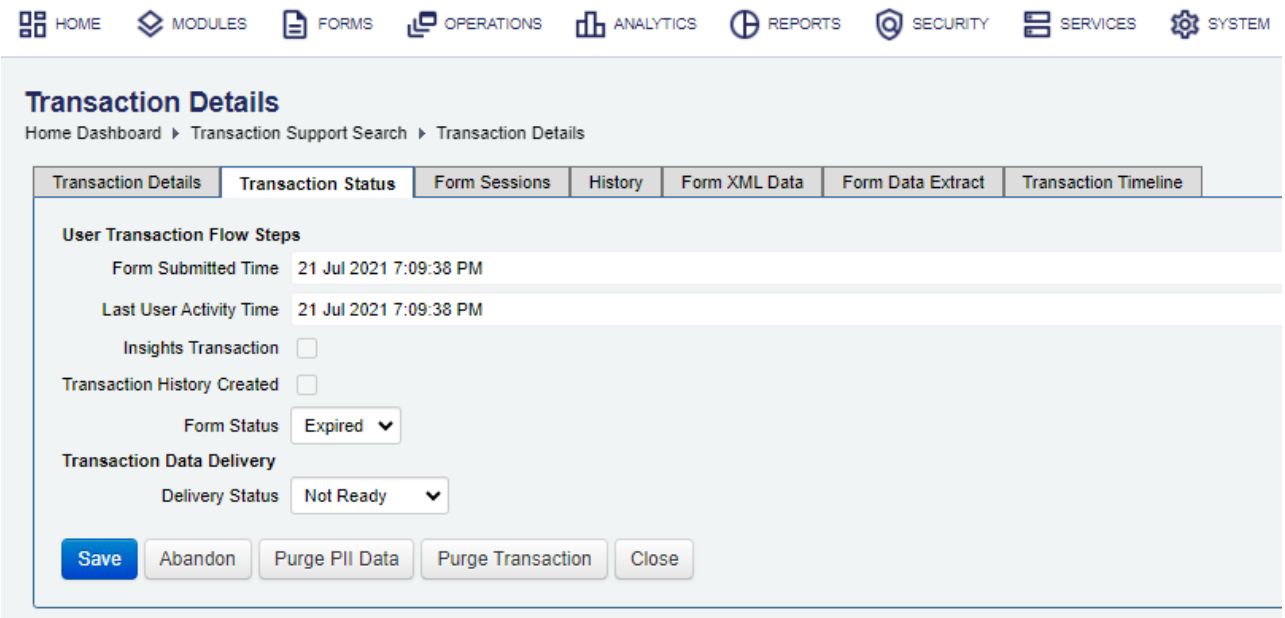
Submitted


HOME
MODULES
FORMS
OPERATIONS
ANALYTICS
REPORTS
SECURITY
SERVICES
SYSTEM
Transaction Details
Home Dashboard ▸ Transaction Details
Transaction Details
Transaction Status
Form Sessions
History
Form XML Data
Form Data Extract
Transaction Timeline
User Transaction Flow Steps
Form Opened Time 19 Jun 2024 3:14:06 PM
Form Submitted Time 19 Jun 2024 3:14:10 PM
Last User Activity Time 19 Jun 2024 3:14:10 PM
Form Completed Time 19 Jun 2024 3:14:10 PM
Save Challenge Data
Insights Transaction
Transaction History Created
Form Status Completed
PDF Receipt Generation
Receipt Status Completed
Receipt Render Duration (ms) 951
Receipt Render Time 19 Jun 2024 3:14:38 PM
Receipt Render Attempts 1
Transaction Data Delivery
Delivery Channel tig-rest-delivery
Delivery Status Ready
Delivery Method REST Service
Collaboration Job
Collaboration Job R8S92HM
Job Status In Progress
Save
Close

Completed



Abandoned



Expired

Transaction properties are grouped into the following sections:

User Transaction Flow Steps

This section lists the essential steps of a form's submission life cycle, including a form status that you can adjust depending on a transaction status as per the form transactions list.

PDF Receipt Generation

This section shows information relevant to a receipt generated for the form. If the receipt has not been generated, this section is not shown.

Transaction Data Delivery

This section shows details of a delivery process that is used to send transaction data obtained during form's submission.

Collaboration Job

This section shows collaboration job's details if a transaction is a part of a collaboration job. If the transaction is not part of a collaboration job, this section is not shown.

Transaction Data Purging

This section shows whether a transaction and its data have been purged or date when it will happen.

User Transaction Flow Steps

Check the following properties:

- The date and time when a form was opened in the Form Opened Time field.
- The date and time when the form was submitted or initially saved by the user in the Form Submitted Time field.
- The date and time of the form last activity, such as viewing or rendering, in the Last User Activity Time field.
- The date and time when the form was completed in the Form Completed Time field.
- The Save Challenge Data is selected for this form so users can resume form applications using the Save Challenge.
- Whether this is an Insights transaction in the Insights Transaction field.
- Whether the transaction history was created for this transaction in the Transaction History Created checkbox.

4. Update the transaction status in the Form Status field with one of the following option available depending on the current transaction status:

Current Form Status	Available Form Status
Assigned	Abandoned

Opened	Abandoned
Saved	Abandoned
Submitted	Abandoned
Completed	You can not change the Form Status
Abandoned	Saved
Expired	You can not change the Form Status

PDF Receipt Generation

Check the following properties:

- The PDF receipt status under the PDF Receipt Generator section, which can be one of the following:
  - Ready: a receipt is waiting to be allocated to a receipt generator.
  - In Progress: a receipt is in a receipt generator's queue.
  - Completed: a receipt has been generated.
  - Error: there is an error during receipt generation. A retry may fix it.
  - Error - No Data: there is no XML data available to generate a receipt.

Transaction Data Delivery

Check the following properties:

- Click a link in the Delivery Channel field to view the configuration of a delivery channel used by this transaction.
- The Delivery Status of the transaction under the Transaction Data Delivery section. You can manually update the delivery status of the transaction depending on the transaction type and its current status.

  A delivery status can be one of the following:

  - Completed: a transaction has been successfully delivered, based on the configured delivery channel.
  - Undeliverable: a delivery is not possible. Check the error logs for further details of the error.
  - Not Ready: a transaction is not ready for delivery yet. This is the start state before a transaction has been completed by the user. Generally, the receipt render service will kick the transaction from this state to the Ready state when the receipt is rendered.

    - Not Required: a delivery is not required. It may be that this form is part of a collaboration job and the delivery is to be handled separately say as a consolidation of the transaction details from all the associated forms.
    - Ready: a transaction is ready to be delivered. A transaction will be marked as Ready until the delivery channel is processed, or it may remain marked as Ready if a delivery channel is not configured.
    - In Progress: a delivery has been started but is waiting to be completed.
    - Paused: a transaction delivery is paused. This transaction's status can be changed by some other means, for example by a Groovy Scheduled Task.
    - Error: a transaction delivery has failed and is waiting to be retried.
    - Sent Email: a secure email has been sent and now a transaction is waiting for the user to log in and process it.
    - Pending: a transaction delivery is pending.

- The form's delivery time in the Delivery Time field.
- The Delivery Message field to see an additional information on how the form was delivered.
- What delivery method was used for this form in the Delivery Method field, for example, Delivery Process.
- How many delivery attempts it has taken to deliver a form in the Delivery Attempts field.

Collaboration Job

Check the following properties:

- Click a collaboration job number in the Collaboration Job field to view the collaboration job details. This is only applicable if this form is configured to be used with collaboration jobs.
- The collaboration job status in the Job Status field.

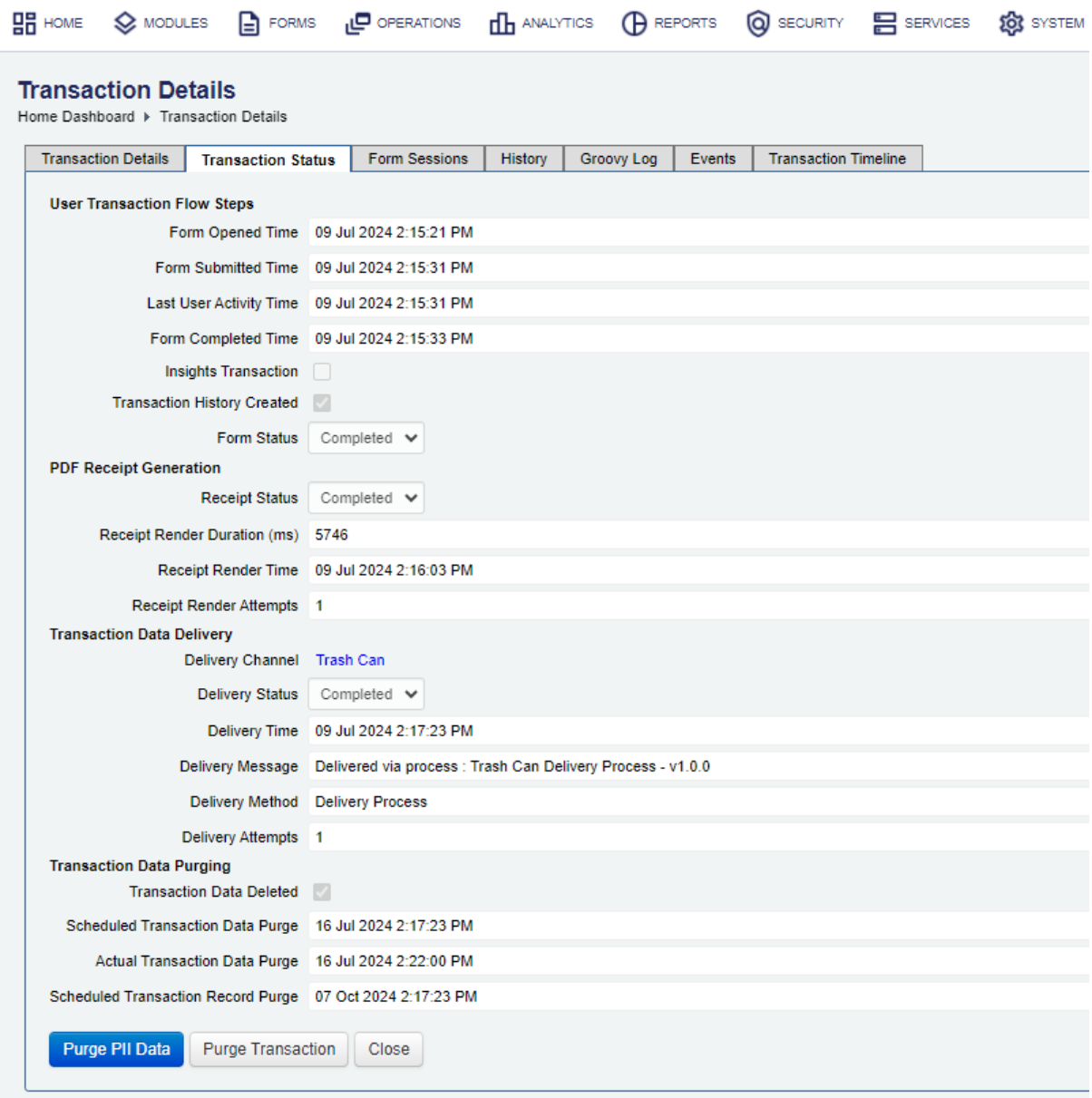
Transaction Data Purging

Check the following properties:

- If the Transaction Data Deleted checkbox is selected, then the PII data was purged.
- The Actual Transaction Data Purge and Scheduled Transaction Data Purge times tell you when the transaction was purged.
- The Scheduled Transaction Record Purge time to see when the transaction data record is due to be deleted.

For example, from the transaction details below we can tell that the PII data was purged (the Transaction Data Deleted is selected) a few minutes after it had been scheduled to be purged (the Actual Transaction Data Purge and Scheduled Transaction Data Purge). Also, the transaction data record is due to be deleted on 7 October 2024 (the Scheduled Transaction Record Purge).

Note that after 7 October 2024, this transaction along with its submission details will no longer exist. Only a transaction history record remains, which contains some details about the transaction.



5. Click Abandon to mark a transaction to be purged. For more information, see Abandon a Transaction.

6. Click Retry Delivery to retry a form delivery for completed forms where delivery has not been completed and submission data is available. Attempts to deliver the submission to the current delivery details set up in the form. This can be used to retry delivery, for example, after changing delivery details. For submissions with email delivery, this action can also be used after delivery has been completed.
7. Click Purge PII Data to purge only PII[1] data of this transactions. Active transactions will be abandoned, if they are not already abandoned. | 18.11 This feature was introduced in 18.11.

   NOTE

   This action is not available if the transaction is a part of a collaboration job, such as a review task, and the transaction PII data has been purged from the system.

8. Click Purge Transaction to purge this transaction immediately. Active transactions will be abandoned, if they are not already abandoned. | 18.11 This feature was introduced in 18.11.

   NOTE

   This action is not available if the transaction is a part of a collaboration job, such as a review task, and the transaction PII data has been purged from the system.

   TIP

   You can purge PII data and transactions in bulk.

9. Click Save to update the changes.

   NOTE

   This action is not available if the transaction PII data has been purged from the system.

You can use Fluent API to retrieve transaction status details, for example, a transaction's Last User Activity Time.

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

```
import com.avoka.tm.query.*

def txn = new TxnQuery()
    .setTrackingCode("1234567") // change for your transactior
    .firstValue()

txn.timeUserLastModified
```

Next, learn how to track submissions.

Change Transaction Status

Manager allows you to change a status of a form transaction manually, for example, change its status from an Abandoned to a Saved status.

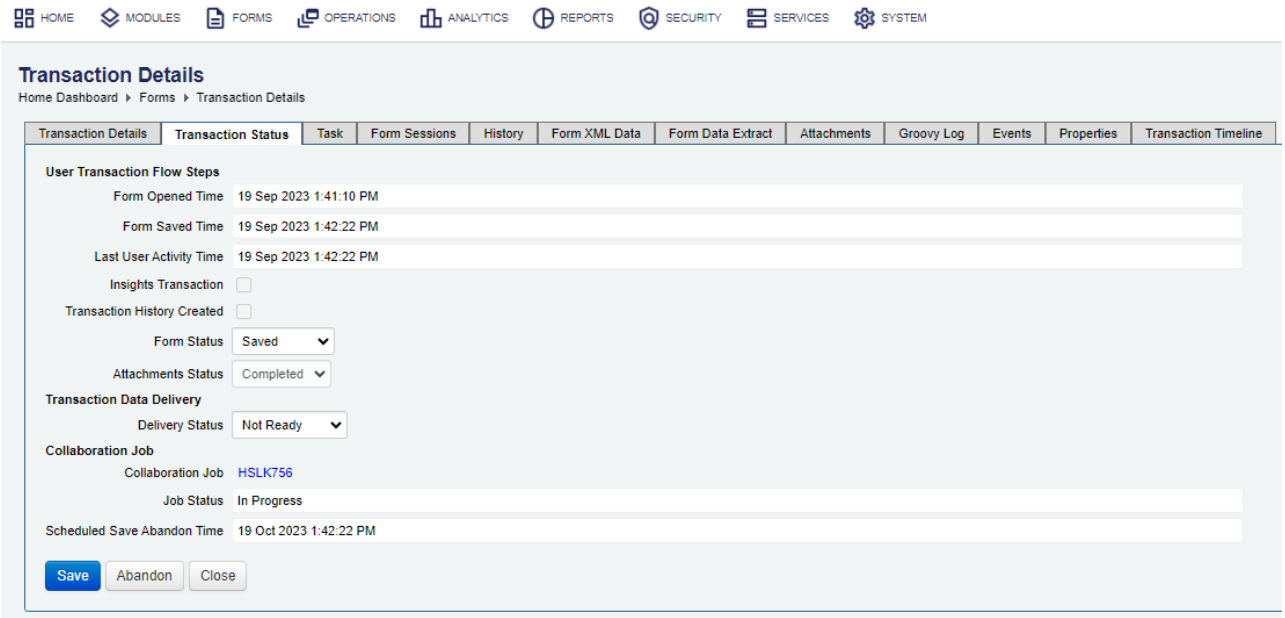
> NOTE
>
> Only a Manager administrator can change an abandoned transaction status to Saved.

Manager provides you with the following methods to change a transaction status:

Method 1

To change a transaction status :

1. Select Operations > Form Transactions, or select Operations > Saved Transactions, or select Forms > Form.
2. Locate a form and click ID in the Latest Transactions section.
3. Find a transaction you want to update and click View.
4. Select the Transaction Status tab.

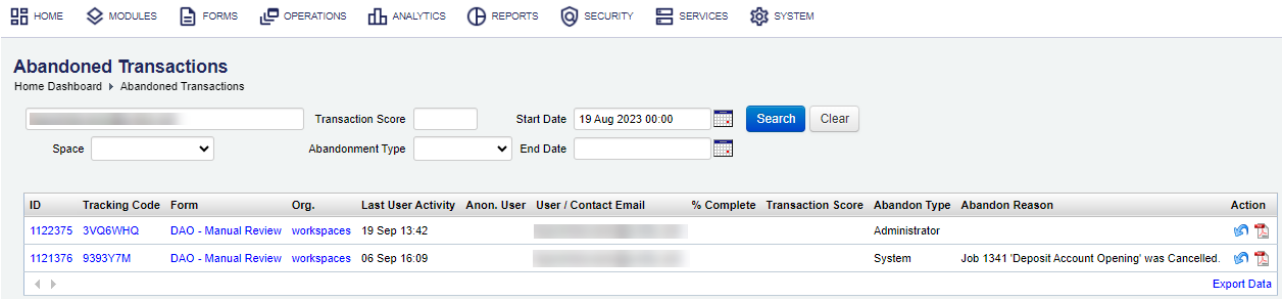


5. Select a desired form status from the Form Status dropdown list, which can be: Saved or Abandoned.
6. Click Save to update the changes.
7. Check the updated transaction status in either abandoned transactions or saved transactions.

Method 2

To change a transaction status:

1. Select Operations > Abandoned Transactions.
2. Locate a transaction and click Make Abandoned Transaction Saved.



3. Click OK to confirm. The transaction is moved from the abandoned transactions to the saved transactions.

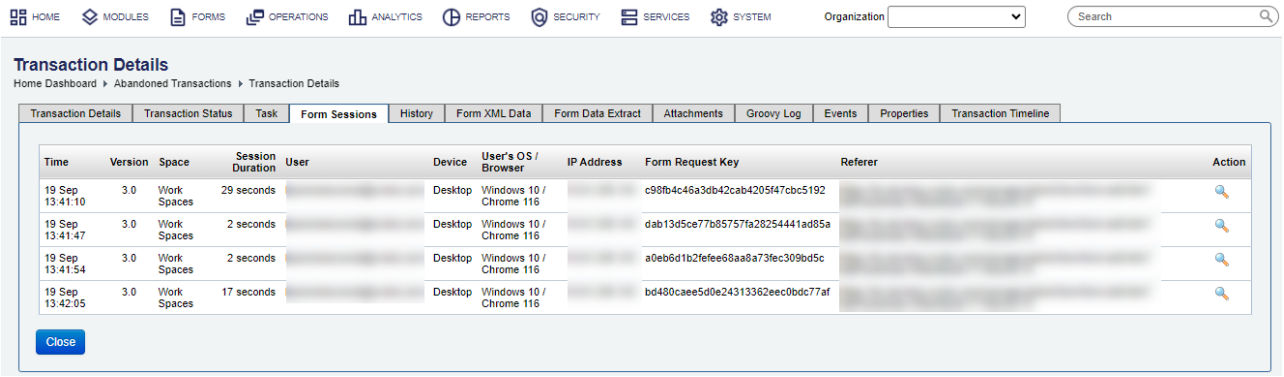
Next, learn how to configure form abandonment.

Form Sessions

Manager creates a form session when a user opens a form and ends it as soon as the user leaves the form. Manager allows you to view and analyze each of the form sessions that were created for a transaction.

To view form session information:

1. Locate a transaction.
2. Select the Form Sessions tab.



3. Check the following information for each session:
   - Date and time of the session.
   - The version of the session.
   - The form space associated with the selected transaction.
   - The session duration, which is the amount of time (in minutes and seconds) that the user spent on the transaction in this session.
   - User.
   - The device used to access a form from.
   - User OS / Browser used to create a session.
   - The IP Address of the user that created a session.
   - Form Request Key, which is a unique identifier of a session.
   - The referrer URL that the user used to access the form that was used to create a session, for example, `https://my-domain.com/manager/admin/form/form-edit.htm?parameter1=0&parameter2=1234`
4. Click Action to check Form Request Details.

NOTE

This data can be used to analyze the efficiency of a form.

Next, learn how to view transaction history.

Transaction History

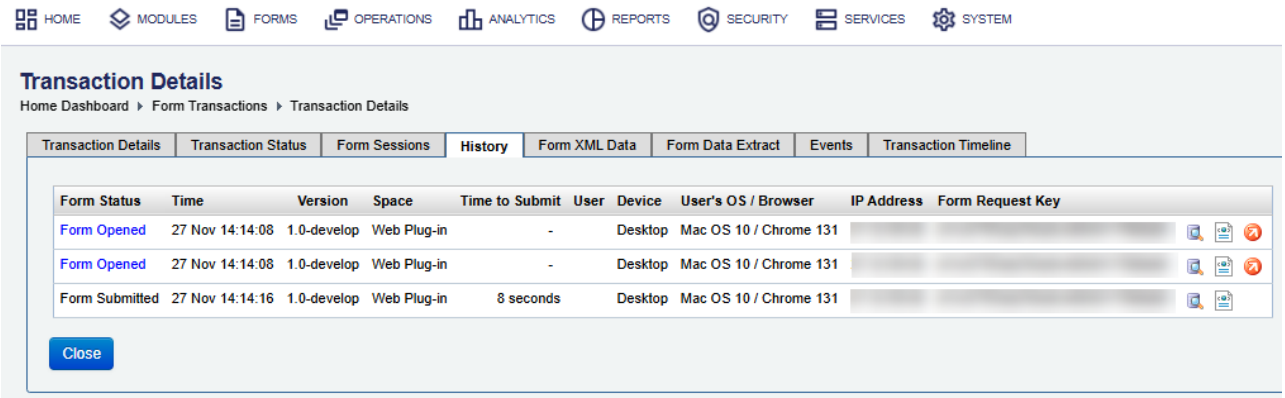
Manager allows you to view transaction history of user's interaction with a form. You can use this functionality for transaction's troubleshooting, analysis, and audit.

> NOTE
>
> We recommend using our Journey Analytics product to check transaction's trends and analyze user's behavior.

When a transaction is finished, Manager writes all relevant submission information, such as final status, form name, time to submit, and browser information into a transaction history table. This table does not contain any PII[1] data, aside from an email address, which is used for security auditing purposes.

To view a transaction's history:

1. Select Operations > Form Transactions.
2. Locate a transaction and click View.
3. Select the History tab to see versions of a selected transaction:



4. Check the following information for each transaction:
    - Form Status, which can be one of the following:
        - Form Opened:
        - Form Submitted:
    - Time: Date and time of a transaction.

---

[1]Personally identifiable information (PII) is any data that could potentially identify a specific individual. Any information that can be used to distinguish one person from another and can be used for de-anonymizing anonymous data can be considered PII.

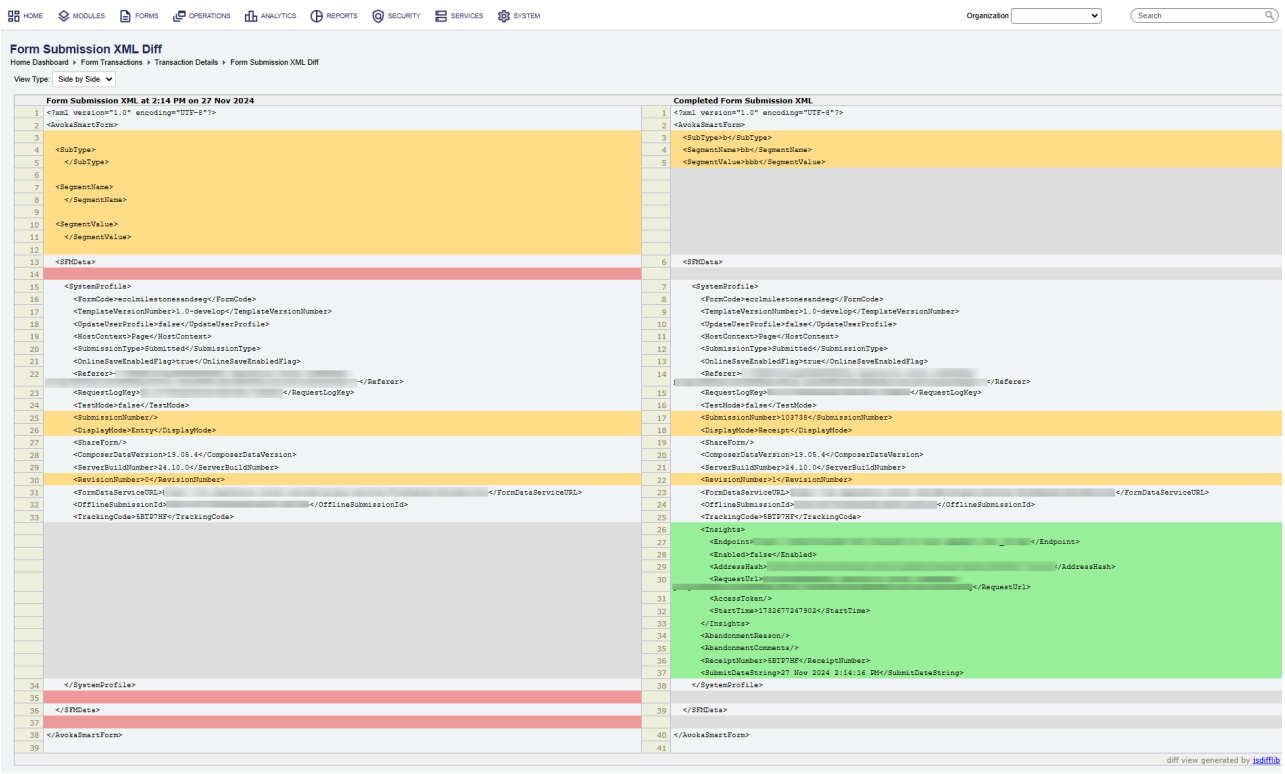
- Version: a version of a transaction, which is created each time a user opens and saves a form until the form is submitted.
- Space: a form space associated with the selected transaction .
- Time To Submit: a time it took for a user to fill in a form and submit it.
- User: a user name or blank for an anonymous user.
- Device: a device used to access a form from.
- User's OS / Browser: a user's OS and a browser used to create a transaction. This is used for auditing purposes.
- IP Address: an IP address of the user that created a transaction. This is used for auditing purposes.
- Form Request Key: a unique identifier of a transaction

5. Click Form Session to view details about the transaction in Form Request Details.
6. Click View XML History to download a copy of the submitted XML data. This option is commonly available to Composer Forms.

> NOTE
>
> The View XML History option is available only if submission XML data is logged. Select the Log Submission History XML checkbox in the form details configuration to start logging submission XML data.

7. Click View XML Diff to view the difference between the submitted XML data and the XML data of the blank form with no input from the user. This option is commonly available to Composer Forms.

> NOTE
>
> The View XML Diff option is available only if submission XML data is logged. Select the Log Submission History XML checkbox in the form details configuration to start logging submission XML data.



8. Click Close.

Next, learn how to track submissions.

Transaction History Details

Manager allows you view detailed information of a selected transaction associated with a form. This is handy when you troubleshoot form submissions.

To view transaction history details:

1. Select Operations > Transactions History.
2. Locate a transaction and click View to see the transaction details.

Home Modules Forms Operations Analytics Reports Security Services System Organization

Transaction History Details

Home Dashboard ▸ Transactions History ▸ Transaction Details ▸ Transaction History Details

Transaction Details

Transaction History ID	4420
Submission ID	4663
Submit Key	[redacted]
Tracking Number	9VG8SYQ
Assigned Task	☐

Form Details

Form Name	testConfirmKeyworkWithoutTF
Form Version	1.0.0
Form Type	Maestro Form
Release Version	21.11.0
Organization	[redacted]
Space	Work Space
Form Code	testconfirmwithoutTF
Client Code	[redacted]

Request Details

Time Request	22 Mar 2022 13:48:40
URL	https://[redacted]
Cookies	JSESSIONID=[redacted]
IP Address	[redacted]
Referer	https://[redacted]
Insights Enabled	☐

Transaction Status

Authenticated	☐
Form Status	Abandoned
Abandonment Type	System
Form Abandonment Type	Started
Abandonment Form Status	Opened
Abandonment Timestamp	23 Mar 2022 13:57:23
Delivery Configured	☐

User Information

Anonymous Participants	1
User Saved	☐
Form Sessions	1
User Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/[redacted] Safari/537.36
Device Type	Desktop
Browser Type	Chrome
Browser Version	99
OS Type	Windows
OS Version	10

Performance Information

Form Render Time (ms)	20
Browser Load Time (ms)	1
Total Render Time (ms)	37

Close

For more information, see transactions details.

3. Click Close.

Next, learn how to change a transaction status.

Form XML Data

Journey Manager allows you to view the XML[1] data of a submitted transaction. An administrator can access and edit the submitted data to re-attempt delivery, thereby addressing issues like these:

- A user completed a form, but form data has not yet been purged, and an error is preventing receipt generation or delivery.
- A user entered invalid or unexpected characters that have caused back-end processing issues. Generally, forms are designed to capture invalid data before submission. However, if invalid form input is accepted, this can cause receipt generation or delivery to be blocked.
- Clean the data before it reaches the user or back-end processing.

Whenever an Administrator views or changes user XML data, Manager logs the operation to the audit log. When an administrator changes the data and saves the changes, the resulting XML is verified and stored as a new submission history entry, preserving the data previously submitted by the user.

You must have the certain permissions set by an Administrator to see form XML data. For more information, see Administrator's documentation.

To view form's XML data of a transaction:

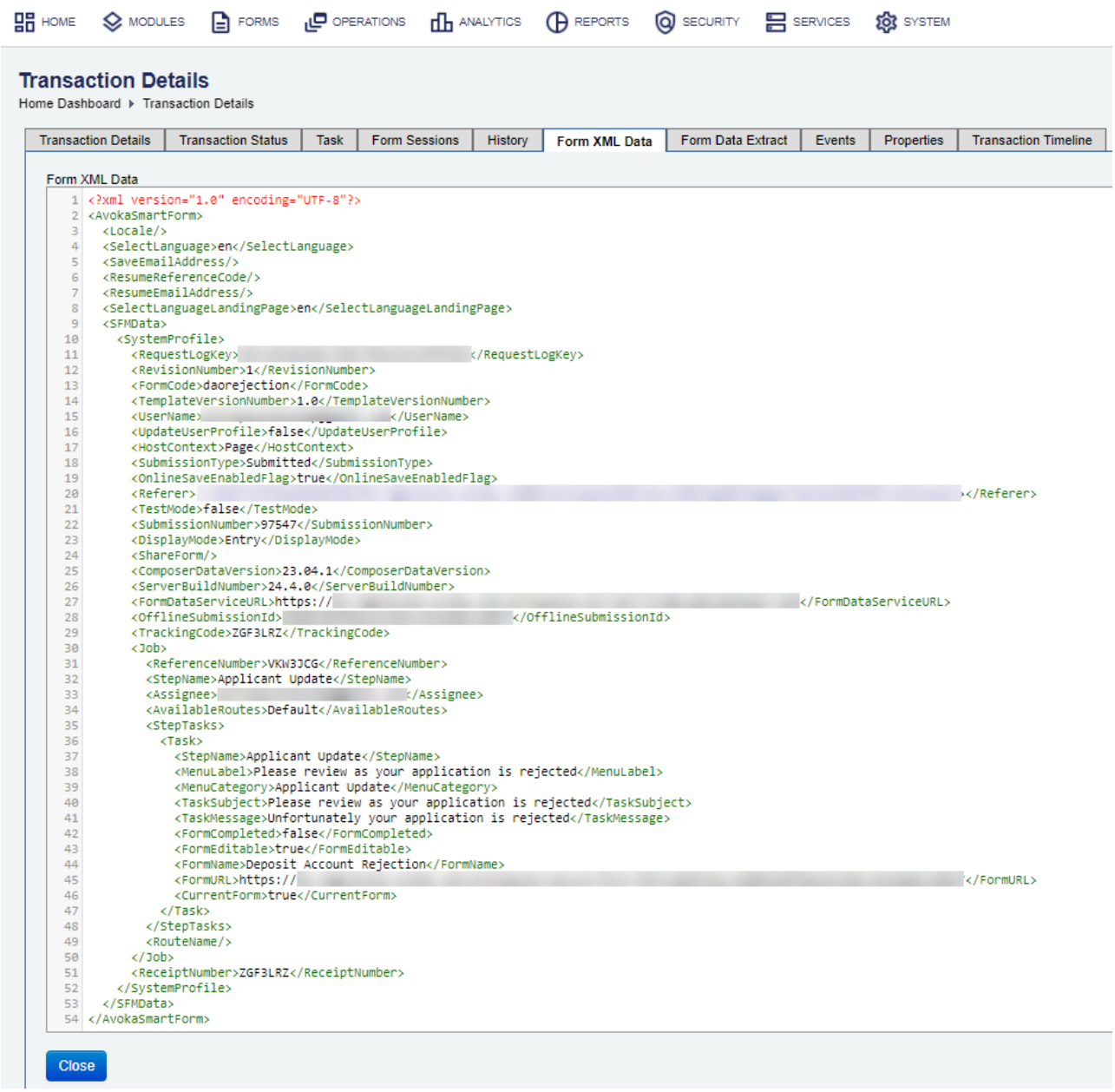
1. Select Operations > Form Transactions.
2. Locate a transaction and click View.
3. Select the Form XML Data tab to view the transaction XML data.

   NOTE

   This tab is not shown if transaction data is purged.

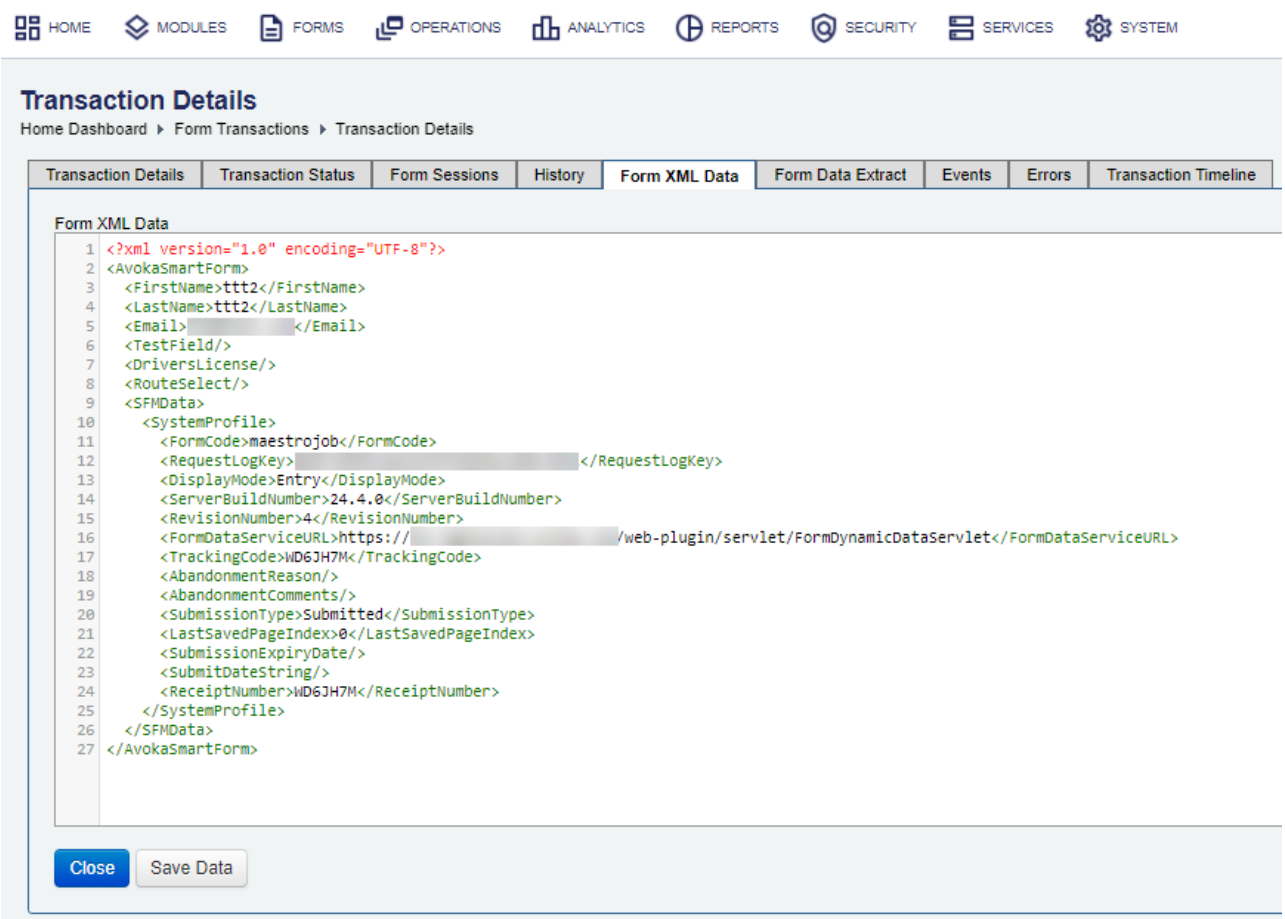
   - Unpurged
   - Delivery Error

[1]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.




```xml
<?xml version="1.0" encoding="UTF-8"?>
<AvokaSmartForm>
  <Locale/>
  <SelectLanguage>en</SelectLanguage>
  <SaveEmailAddress/>
  <ResumeReferenceCode/>
  <ResumeEmailAddress/>
  <SelectLanguageLandingPage>en</SelectLanguageLandingPage>
  <SFMData>
    <SystemProfile>
      <RequestLogKey></RequestLogKey>
      <RevisionNumber>1</RevisionNumber>
      <FormCode>daorejection</FormCode>
      <TemplateVersionNumber>1.0</TemplateVersionNumber>
      <UserName></UserName>
      <UpdateUserProfile>false</UpdateUserProfile>
      <HostContext>Page</HostContext>
      <SubmissionType>Submitted</SubmissionType>
      <OnlineSaveEnabledFlag>true</OnlineSaveEnabledFlag>
      <Referer></Referer>
      <TestMode>false</TestMode>
      <SubmissionNumber>97547</SubmissionNumber>
      <DisplayMode>Entry</DisplayMode>
      <ShareForm/>
      <ComposerDataVersion>23.04.1</ComposerDataVersion>
      <ServerBuildNumber>24.4.0</ServerBuildNumber>
      <FormDataServiceURL>https://</FormDataServiceURL>
      <OfflineSubmissionId></OfflineSubmissionId>
      <TrackingCode>ZGF3LRZ</TrackingCode>
      <Job>
        <ReferenceNumber>VKW3JCG</ReferenceNumber>
        <StepName>Applicant Update</StepName>
        <Assignee></Assignee>
        <AvailableRoutes>Default</AvailableRoutes>
        <StepTasks>
          <Task>
            <StepName>Applicant Update</StepName>
            <MenuLabel>Please review as your application is rejected</MenuLabel>
            <MenuCategory>Applicant Update</MenuCategory>
            <TaskSubject>Please review as your application is rejected</TaskSubject>
            <TaskMessage>Unfortunately your application is rejected</TaskMessage>
            <FormCompleted>false</FormCompleted>
            <FormEditable>true</FormEditable>
            <FormName>Deposit Account Rejection</FormName>
            <FormURL>https://</FormURL>
            <CurrentForm>true</CurrentForm>
          </Task>
        </StepTasks>
        <RouteName/>
      </Job>
      <ReceiptNumber>ZGF3LRZ</ReceiptNumber>
    </SystemProfile>
  </SFMData>
</AvokaSmartForm>
```


Unpurged



Delivery Error

4. Review and edit the XML data of a transaction. For more information on the structure of the XML file, see Form Data Overview.

> NOTE
>
> You are only allowed to edit the XML data if the Delivery status of a transaction is in the Error state. For more information, see View Form Transactions.

5. Click Close.
6. Click Save to update the changes.

> NOTE
>
> This button is shown only if the Delivery status of a transaction is in the Error state.

> TIP
>
> You can change some formatting of the XML data, such as the submit date string.

Next, learn how to track submissions.

Form Data Extract

Manager allows you to view all items and their values that were extracted from the submitted XML data of a form.

WARNING

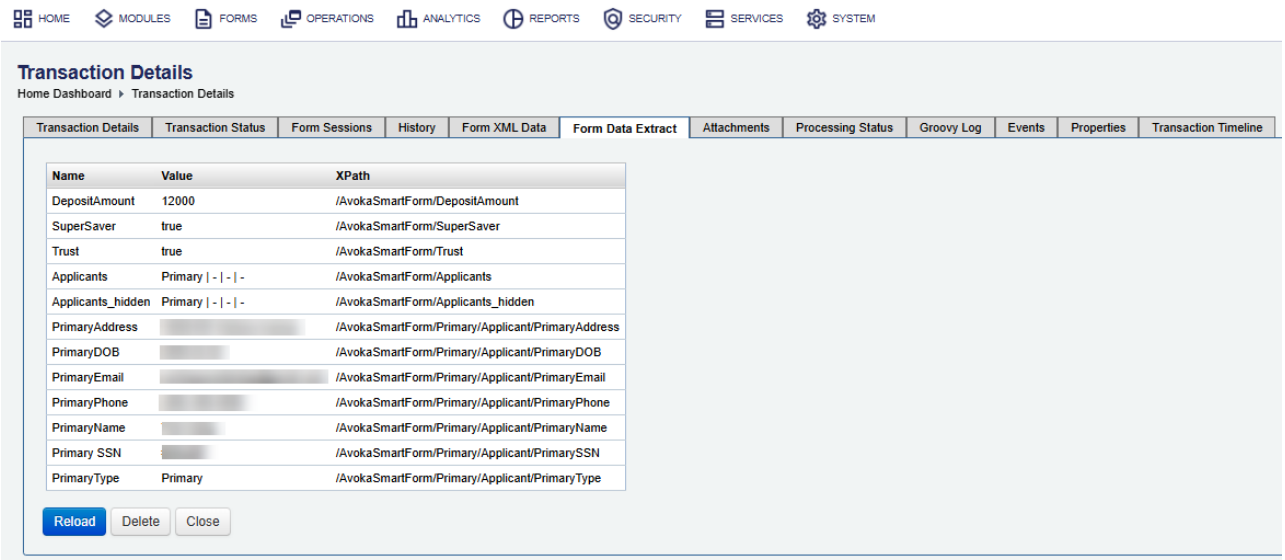
Form data extracts may contain PII[1] data, so you must adhere to the best practices of handling such data as well as global and local regulations.
We don't recommend extracting PII data.

NOTE

The Form Data Extract tab is only available if form data extracts are configured.

To view a form data extract:

1. Select Operations > Form Transactions.
2. Locate a transaction and click View.
3. Select the Form Data Extract tab to see a list of values for each configured data extract. For more information, see view form submission data extract.



[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

The list displays the following details:

- Name:
- Value:
- XPath:

4. Click Reload to refresh form XML data extract values.
5. Click Delete to remove the data extracts.
6. Click Close.

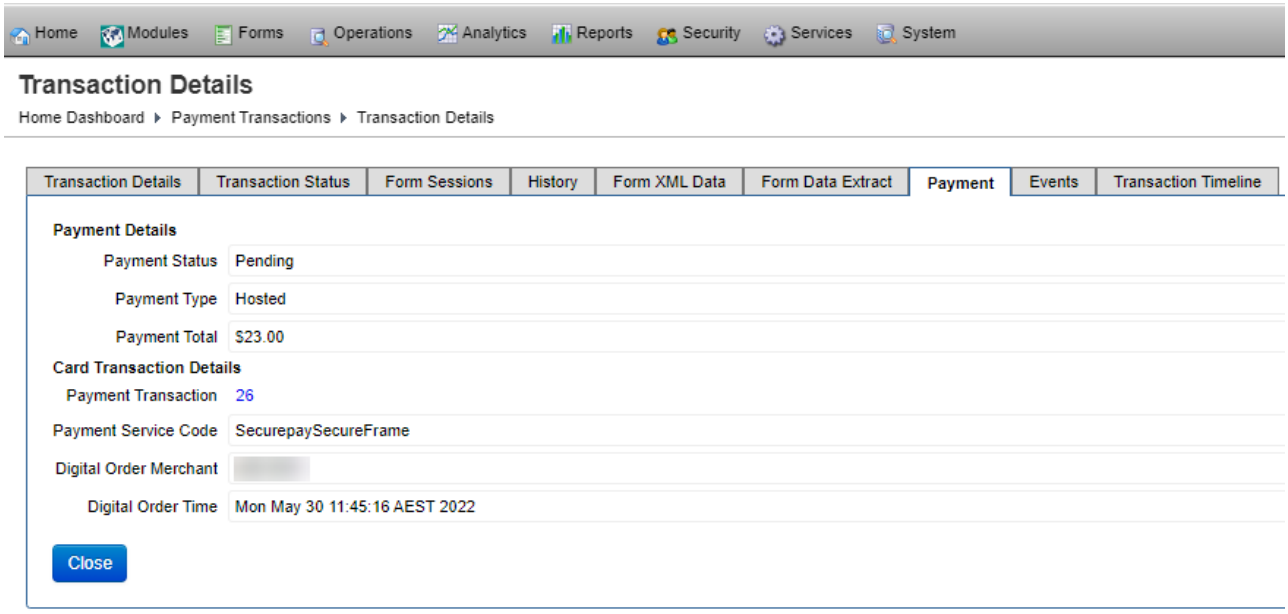
Next, learn how to view form submission data.

Payment

Manager allows you to view payment details of a selected transaction associated with a form. This is handy when you troubleshoot form submissions and track down problems that occurred during payments.

To view payment details:

1. Select Operations > Form Transactions.
2. Locate a transaction and click View or ID.
3. Select the Payment tab to see payment transaction details.



4. Check the Payment Details section, including:
    - Payment Status
        - Cancelled:
        - Completed:
        - Error:
        - Pending:
    - Payment Type
    - Payment Total - the amount that was paid
5. Check the Card Transaction Details section, including:
    - Payment Transaction - an ID of the payment transaction. Click the ID to view payment transaction details.
    - Payment Service Code

- Digital Order Merchant
- Digital Order Time

6. Click Close.

Next, learn how to view attachments.

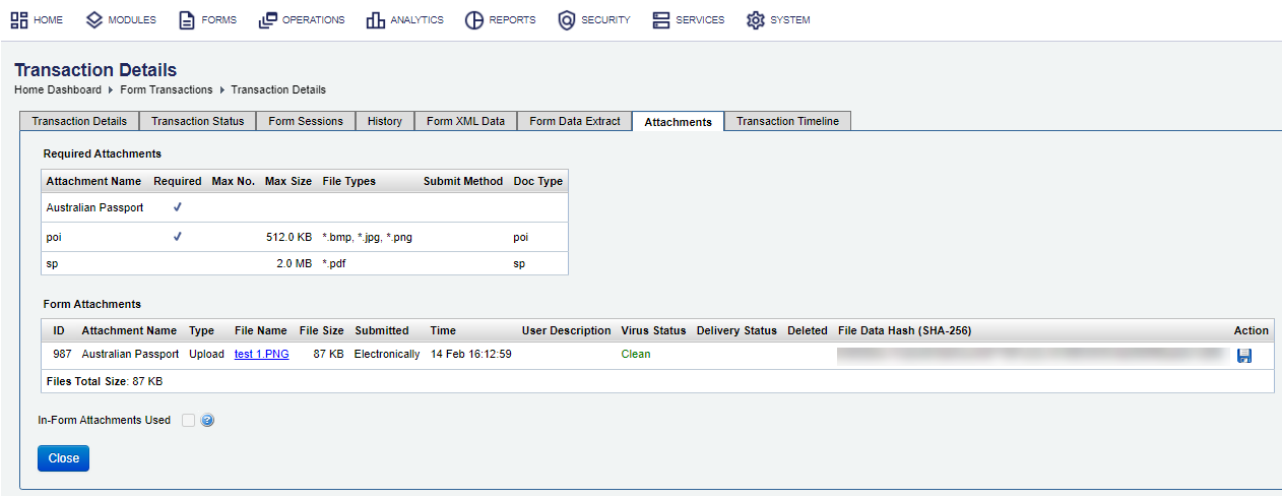
Attachments

Manager allows you to view all attachments pertinent to a form, after a form user submits the form. You can use this information to see that all required attachments are uploaded as per a form flow. You can also check the virus scan status for each attachment.

Manager receives all required information about attachment files from the Attachment Field or Attachment Field v2 components, which are Maestro components allowing a form user to attach documents to a form as per each onboarding journey. For example, a name of attachment is populated with data from the Attachment Name property. However, Manager can have its own definition of attachments with max size, number, and so on, which takes precedence over what comes in from Maestro.

To view submitted attachments associated with a transaction:

1. Select Operations > Form Transactions.
2. Locate a transaction and click View.
3. Select the Attachments tab.



4. Check the Required Attachments section that shows the following information of all attachments required for successful submission of a form:
   - Attachment Name - a name of an attachment
   - Required - whether the attachment is required or not
   - Max No. - a maximum number of attachments
   - Max Size - a maximum size of attachments
   - File Types - file types of attachments
   - Submit Method - a method used to submit an attachment
   - Doc Type - a type of a document

5. Check the Form Attachments section that shows the following information of all required and additional attachments that have been submitted with the form:
    - ID - an attachment ID
    - Attachment Name
    - Type - the attachment action
    - File Name - a name of the attachment
    - File Size - a file size of the attachment
    - Submitted - a way the attachment is submitted, for example, Electronically
    - Time - date and time when the file is attached to the form
    - User description - a browser used to attach the file
    - Virus Status - a status of a virus scan of the attachment, which can be one of the following:
        - Clean - an attachment is free of viruses or malicious software
        - Virus - an attachment is infected
        - Scan Error - an error happened during the scan
        - No Data - an attachment is missing actual data and doesn't need to be scanned anymore
    - Delivery Status - a status of a form submission delivery
    - Deleted- whether a form's attachment is deleted or not. If the Virus Status is Virus, the attachment will be deleted so the Deleted column will have a mark.
    - File data hash (SHA-256) - a unique identifier of the attachment
    - In-Form Attachments Used - specifies whether a user uploaded the attachments inside the form
6. Click Download File or a file name link to download the attachment to your PC.

    NOTE

    You must be a Manager administrator to be able to download form's attachments.

7. Click Close.

Next, learn how to view saved transactions.

Transaction Groovy Log

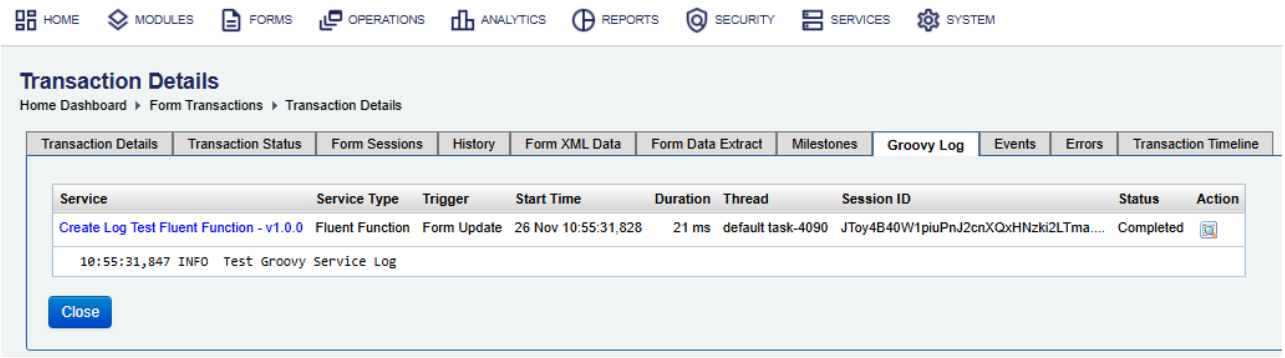
Manager allows you to view Groovy service output messages for a selected transaction, which you can use for troubleshooting purposes.

> NOTE
>
> Do not log PII[1] data to the service log, for example, to debug a service, as there's a risk of exposing customer's details through the log file.

To view transaction's Groovy log:

1. Select Operations > Form Transactions.
2. Locate a transaction and click View.
3. Select the Groovy Log tab to see the log details.

> NOTE
>
> This tab is only available if there are log details pertaining to the selected transaction.

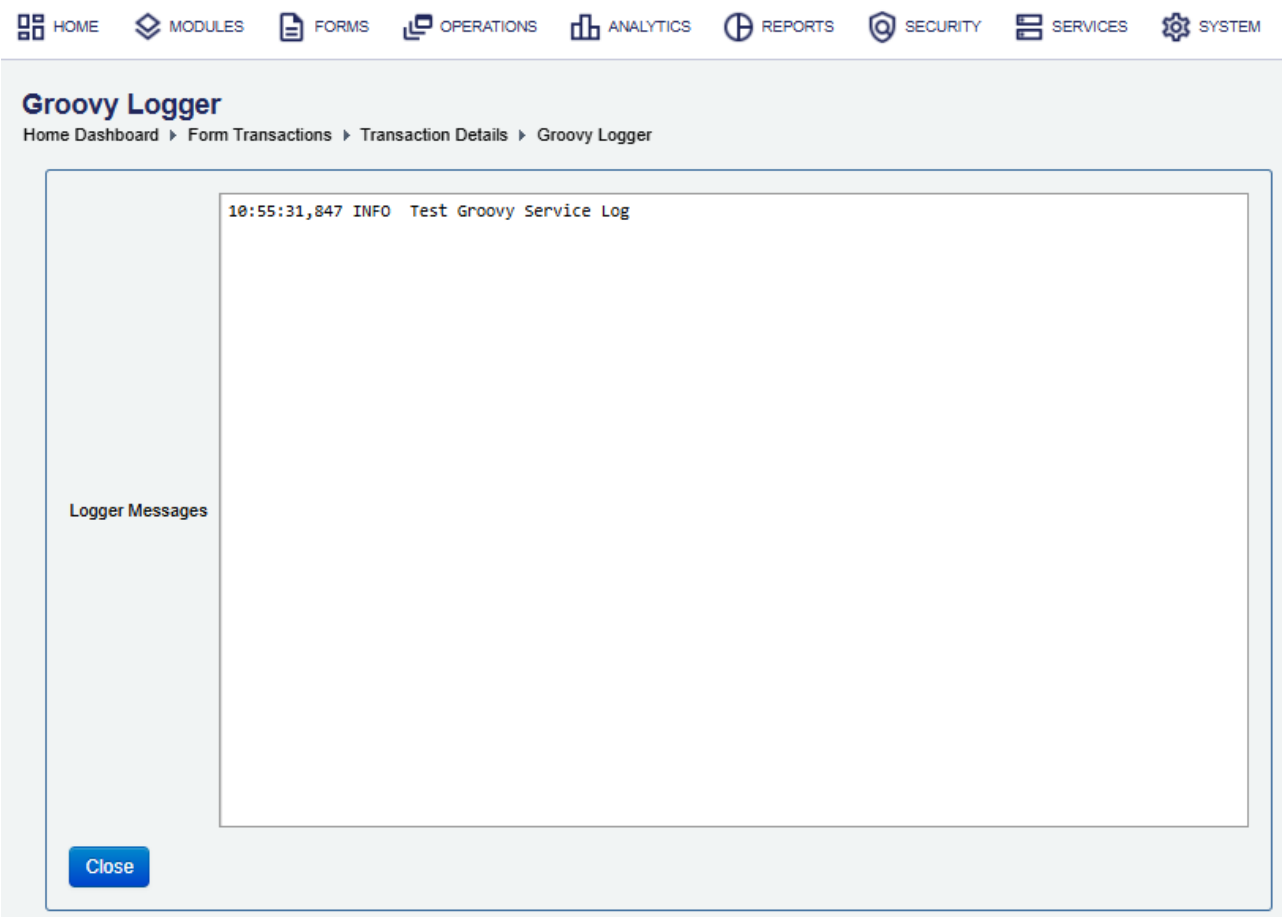


[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

4. Click a service name to view the service configuration.

> NOTE
>
> Select the service's Groovy Service Log tab to view the same logged messages.

5. Click Groovy Logger to see the detailed messages generated during the service invocation.



6. Click Close.

Next, learn about the transaction details.

Transaction Events

Manager allows you to see all events that were logged against a selected transaction.

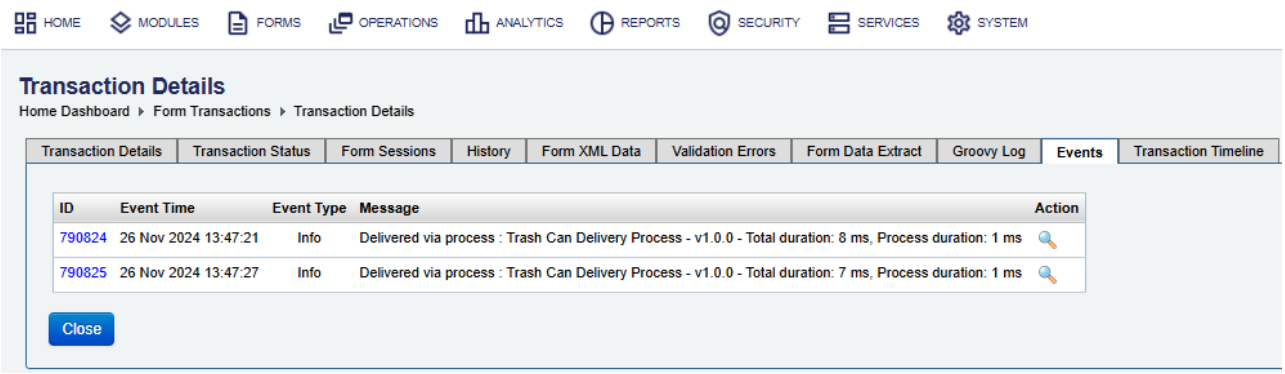
Events include:

- Error transactions
- Saved transactions

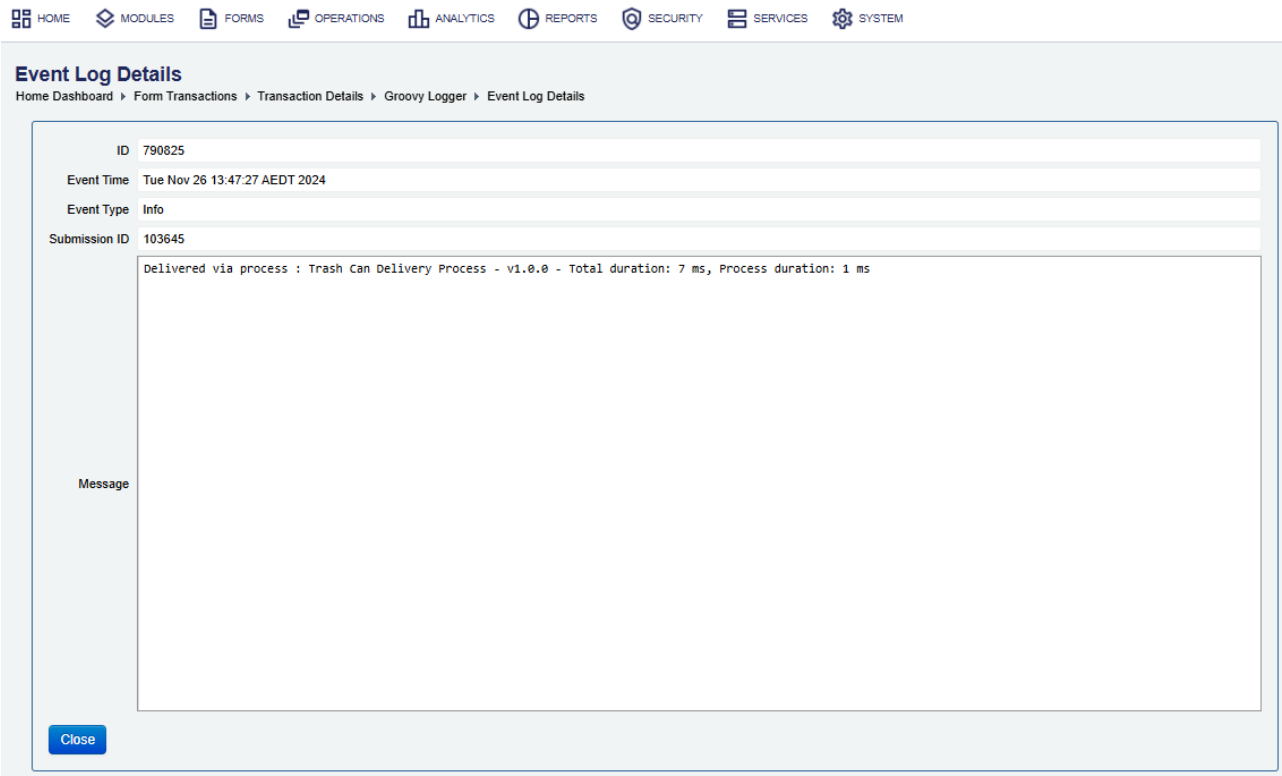
To view all events pertaining to a transaction:

1. Select Operations > Form Transactions.
2. Locate a transaction and click View.
3. Select the Events tab to see all events pertaining to the selected transaction.

> NOTE
>
> This tab is only available if events have occurred within the selected transaction.



4. Click View to see the event's details.



For more information, see Event Log.

5. Click Close.

Next, learn how to track transactions.

Transaction Properties

Journey Manager allows you to view and create transaction properties, which represent arbitrary data as string objects. The main persistence model in Manager is an XML[1] file, which is stored on the server and sent to a browser when a form is rendered. It's then sent back to Manager after the form is submitted. It is also saved to the server in background save events.

There are situations when it is useful to store some information, which is not part of the XML file, on the server only. For example, you may want to record a user's actual credit score, but not show it to the user. In this case, you can store results from the back-end server calls and send only a pass or fail result to the form, so form users won't see their actual score or the reasons why they failed. You can also store in-progress data that has been used in the back-end service calls temporarily to ensure that form users don't "game" the system by changing the data in the form at different points of their journey.

Manager comes with the Unified Application Data Document Model that allows you to store integration data separately from a form data. This means that data, which should not be visible or accessible in a browser, is not stored in the `AvokaSmartform` tag in the XML. Instead, data is stored in the `server xml` node which remains accessible only on server-side.To achieve this, you need to enable the Unified App Data feature for a form version, which you can easily do while configuring the form version.

> NOTE
>
> It's strongly recommended to store integration data using the Unified Application Data Document Model.

To view transaction's properties:

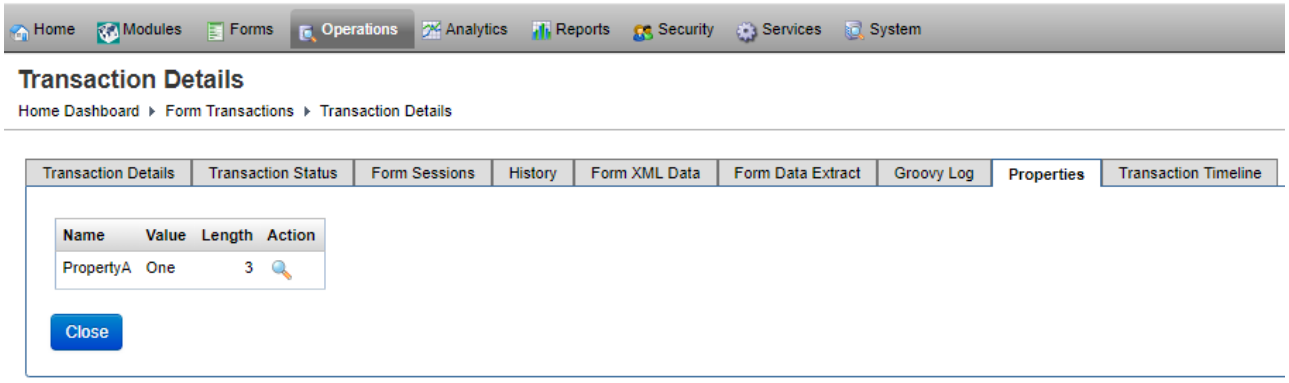
1. Select Operations > Form Transactions.
2. Locate a transaction and click View.
3. Select the Properties tab to see all properties pertaining to the selected transaction.

---

[1]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

NOTE

This tab is only available if there are properties associated with the selected transaction.



4. Select a property and click View to see the property's value in a new browser tab.
5. Click Close.

Sometimes, you may want to add a new property to a transaction so it will be stored together. To add and store a transaction property, you should use the following Fluent Groovy API:

```
package exchange;

import com.avoka.tm.func.*
import com.avoka.tm.svc.*
import com.avoka.tm.util.*
import com.avoka.tm.vo.*
import javax.servlet.http.*
import com.avoka.tm.query.*

public class GetExchangeRate {

	// Injected at runtime
	public Logger logger

	/*
	* Perform Fluent Function call.
	*
	* returns: FuncResult
	*/
	FuncResult invoke(FuncParam param) {

		logger.info(param.txn.formStatus)

		new TxnUpdater(param.txn)
			.setProperty("PropertyA", "One")
			.update()

		FormFuncResult result = new FormFuncResult()

		logger.info("test log event")
		logger.debug("Debug log event")

		return result
	}
}
```

To retrieve transaction properties, you can use the Fluent Groovy code:

```
Txn txn = new TxnQuery()
       .setSubmitKey(…) // or similar to find the correct transactior
       .withPropertyMap() // to retrieve the properties
       .firstValue() //to find the first matching transactior
// The txn contains an attribute propertyMap of type Map<String,String>
```

When working with transaction properties, you should take into account the following:

- Data stored in the Transaction Properties are encrypted and subject to data retention policies, just like XML.
- In some cases, it may be necessary to store data against a collaboration job rather than an individual transaction.
- The PropertyQuery service is used for locating configuration properties for form versions, Workspaces, and so on. This is for static configuration properties, not for per-transaction or runtime properties.
- It is also possible to use the standard HTTP session to store temporary data. However, this data will not be persisted beyond an individual session. A transaction property is persisted in the database.

Next, learn about the transaction timeline.

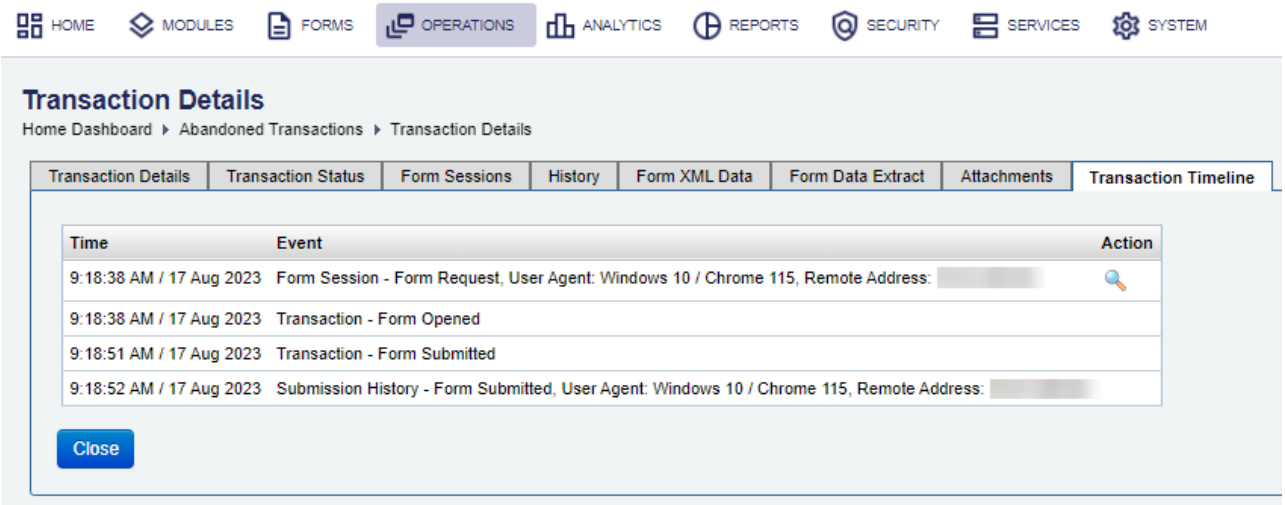
Transaction Timeline

Manager allows you to view transaction timeline listing the important events that happened to a selected transaction. A transaction timeline is not a transaction audit trail, meaning that some of the entries derived from timestamps of the transaction, and, over time, some of the values may change to represent the latest state of the transaction.

To view a transaction's time line:

1. Select Operations > Form Transactions.
2. Locate a transaction and click View.
3. Select the Transaction Timeline tab to see a time line of the selected transaction.

> NOTE
>
> This tab is only available if there are events pertaining to the selected transaction.



4. Click View to see the transaction's details.
5. Click Close.

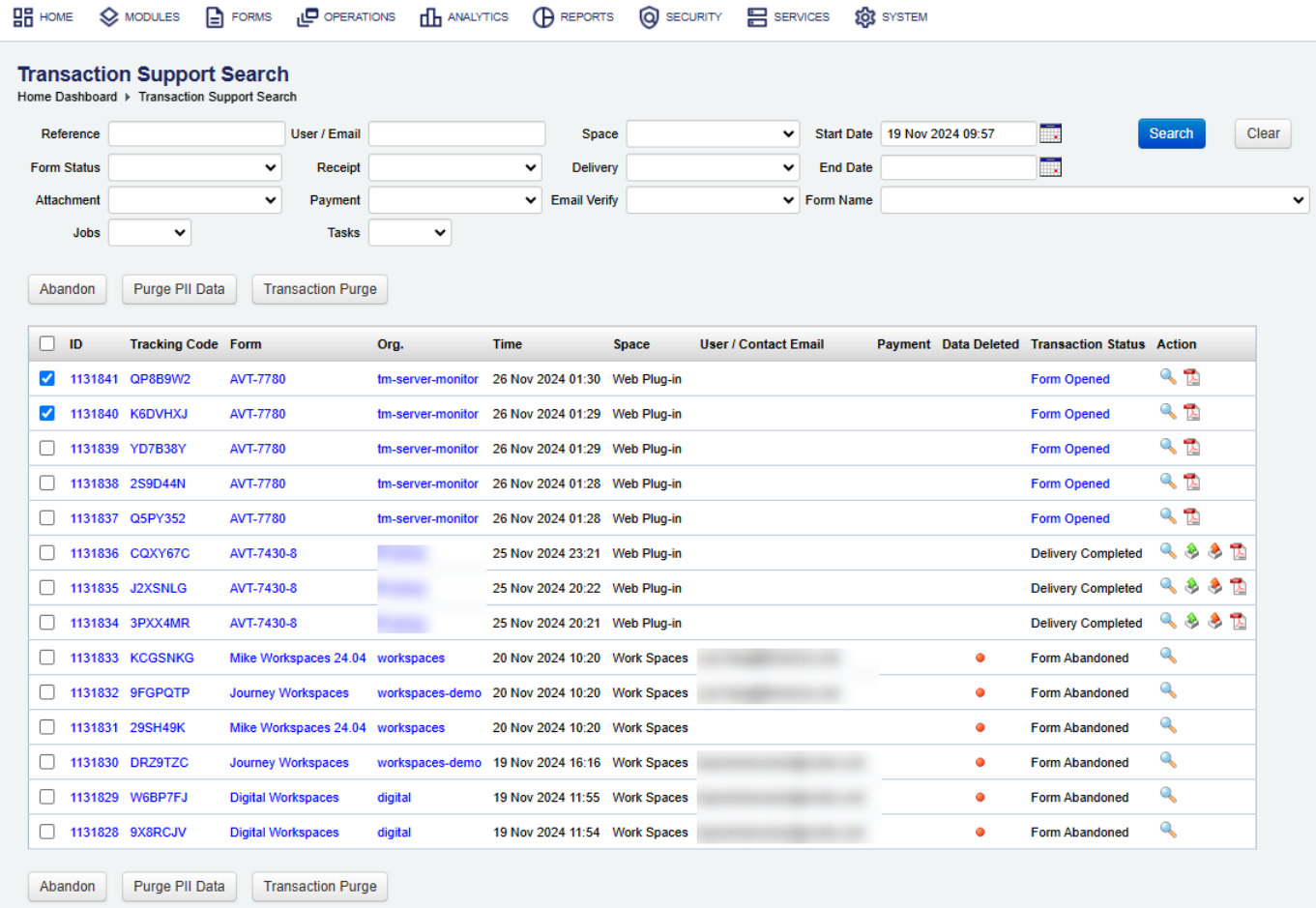
Next, learn about the transaction details.

Transaction Support Search

Journey Manager allows you to search for transactions based on a wide range of criteria which can be useful when troubleshooting transactions.

To navigate to transaction support search, select Operations > Txn Support Search.

Transactions are shown page by page, with the maximum number of records per page configured via the user's preferences.



The list displays the following details:

- ID: a transaction ID.
- Tracking Code: a transaction's tracking code.
- Form: a name of the form.
- Org.: an organization associated with the form.
- Time: time and date of a transaction, which can be Form Opened Time, Form Saved Time, Form Submitted Time, Last User Activity Time, Form Completed Time. This time is stored in `submission.time_request`.

- Space: a form space where the form is hosted.
- User/ Contact Email: a user name and an email address, if available.
- Payment: a payment amount, for example, `$10`, which is displayed in black for Required payments and in blue for Completed payments.
- Data Deleted: a status showing with whether the transaction data, including PII[1], was deleted from Manager.
- Transaction Status: a transaction status, which can be one of the following:
  - Form Opened.
  - Form Abandoned.
  - Form Saved.
  - Delivery Completed.
  - Delivery Ready, Method Not Configured.
  - Job Delivery Not Ready.

To filter or search the transactions by one or more criteria, specify the following settings and click Search:

- Reference: a transaction ID or tracking code.
- Form status, which can be one of the following:
  - Abandoned: a transaction that has been abandoned by the user, or based on the data retention policy
  - Assigned: a task which is waiting to be completed by a user
  - Completed: a transaction that has been successfully submitted and contain all required data
  - Opened: a transaction that has been started by a user. When a transaction is created by a user (by starting a new form), the initial form status is set to Opened. This status will be changed once the user saves, cancels, or submits the transaction.
  - Saved: a transaction that has been saved by the user or automatically by the system
  - Submitted: a transaction that has been submitted but may require additional information, such as attachments, payment, and so on. Once the required data is collected, the transaction status will change to Completed. This status is not common for Maestro forms as all information, such as attachments and payments, are collected prior to the user submitting the form. Most Maestro forms will skip the Submitted status and

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

will be marked as Completed after the form is submitted by the user. This is a legacy feature.
    - Expired: a form associated with this transaction has expired and is no longer available
- Attachments.

  An attachment type can be one of the following:

    - Completed.
    - Optional.
    - Required.
- Form Name - select a form name from the dropdown list.
- User / Email - provide a user name or an email address.
- Receipt.
    - Ready: a receipt is waiting to be allocated to a receipt generator.
    - In Progress: a receipt is in a receipt generator's queue.
    - Completed: a receipt has been generated.
    - Error: there is an error during receipt generation. A retry may fix it.
    - Error - No Data: there is no XML data available to generate a receipt.
- Payment, which can be one of the following:
    - Required:
    - Completed:
    - Error:
    - Pending:
- Space.
- Delivery.

  A delivery status can be one of the following:

    - Completed: a transaction has been successfully delivered, based on the configured delivery channel.
    - Undeliverable: a delivery is not possible. Check the error logs for further details of the error.
    - Not Ready: a transaction is not ready for delivery yet. This is the start state before a transaction has been completed by the user. Generally, the receipt render service will kick the transaction from this state to the Ready state when the receipt is rendered.
    - Not Required: a delivery is not required. It may be that this form is part of a collaboration job and the delivery is to be handled separately say as a consolidation of the transaction details from all the associated forms.

    - Ready: a transaction is ready to be delivered. A transaction will be marked as Ready until the delivery channel is processed, or it may remain marked as Ready if a delivery channel is not configured.
    - In Progress: a delivery has been started but is waiting to be completed.
    - Paused: a transaction delivery is paused. This transaction's status can be changed by some other means, for example by a Groovy Scheduled Task.
    - Error: a transaction delivery has failed and is waiting to be retried.
    - Sent Email: a secure email has been sent and now a transaction is waiting for the user to log in and process it.
    - Pending: a transaction delivery is pending.
- Signature, which can be one of the following:
    - Required:
    - Completed:
    - Completed - Deliver Manually:
    - Completed - Upload Scanned Copy:
- Jobs.

  A job type can be one of the following:

    - Include - transactions associated with a job are included.
    - Exclude - transactions associated with a job are excluded.
- Start Date: the date and time you want to search from. Manager uses timestamps of several distinct transaction's life cycle states to search for transactions, for example, Form Opened Time.
- End Date: the date and time you want to search until. Manager uses timestamps of several distinct transaction's life cycle states to search for transactions, for example, Form Opened Time.

  NOTE

  The time interval is inclusive.

- Email Verify.

  An email verification type can be one of the following:

    - Required.
    - Pending.
    - Completed.
    - Abandoned.

    - Error.
    - Error - No Data.
- Tasks: to filter transactions related to tasks.

  A task type can be one of the following:

    - Include - transactions associated with a task are included.
    - Exclude - transactions associated with a task are excluded.

Click Clear to reset the search criteria.

You can perform one of the following actions for each selected transaction:

- Click View to view transaction details.
- Click View Receipt to view a PDF receipt of the selected transaction.
- Click Retry Delivery to retry delivery of the transaction. This is only available if delivery of a transaction has not been completed, but submission data is still available.
- Click Retry Delivery From Scratch to attempts to deliver the submission using the current delivery details set up in the form after deleting all existing delivery checkpoints for the submission. If no delivery checkpoints exist, this is equivalent to the retry delivery option above. This is only available if delivery of a transaction has not been completed, but submission data is still available.
- Click Open Saved Form to open the saved form to complete the form on behalf of the user.

  > NOTE
  >
  > To use this option, you must have the Submission Save Edit permission.

Bulk Operations

You can also perform several bulk operations on multiple selected transactions. | 18.11 This feature was introduced in 18.11.

> NOTE
>
> To perform bulk operations, you must have the Submission Data Purge or Submission Edit permissions.

To perform a bulk operation:

1. Select or filter transactions you need, as described above.
2. Select all these transaction ID checkboxes or select the header checkbox to bulk select all filtered transactions.
3. Apply one of the following actions:
   - Click Abandon to abandon the selected transactions.
   - Click Purge PII Data to purge only PII[1] data of the selected transactions. Active transactions will be abandoned, if they are not already abandoned.
   - Click Transaction Purge to purge the selected transactions immediately. Active transactions will be abandoned, if they are not already abandoned.
4. Click OK to accept the warning message that the transactions and all the data will be purged.

   NOTE

   Transactions relating to collaboration jobs are rejected and this is recorded in the event log entry. The user is warned this has happened.

5. Check the event log entry to see if the purging was successful. An example of a successful message is: `Submission 3090 was marked for immediate PII data purging by user john.smith@email.com`.

NOTE

You can purge each transaction separately from the Transaction Status tab. | 18.11 This feature was introduced in 18.11.

NOTE

The System Event Publisher publishes submission PII purging events. | 18.11 This feature was introduced in 18.11.

Next, learn how to view form transactions.

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

Track Transactions

One of the key features of Manager is to keep track of all transactions made by form users. Manager captures this information in form transactions, so operations staff can search for a particular transaction and view its details.

You can also use the following Manager functionality to track user transactions:

- View abandoned transactions
- View form submission data
- View saved transactions
- View submitted transactions

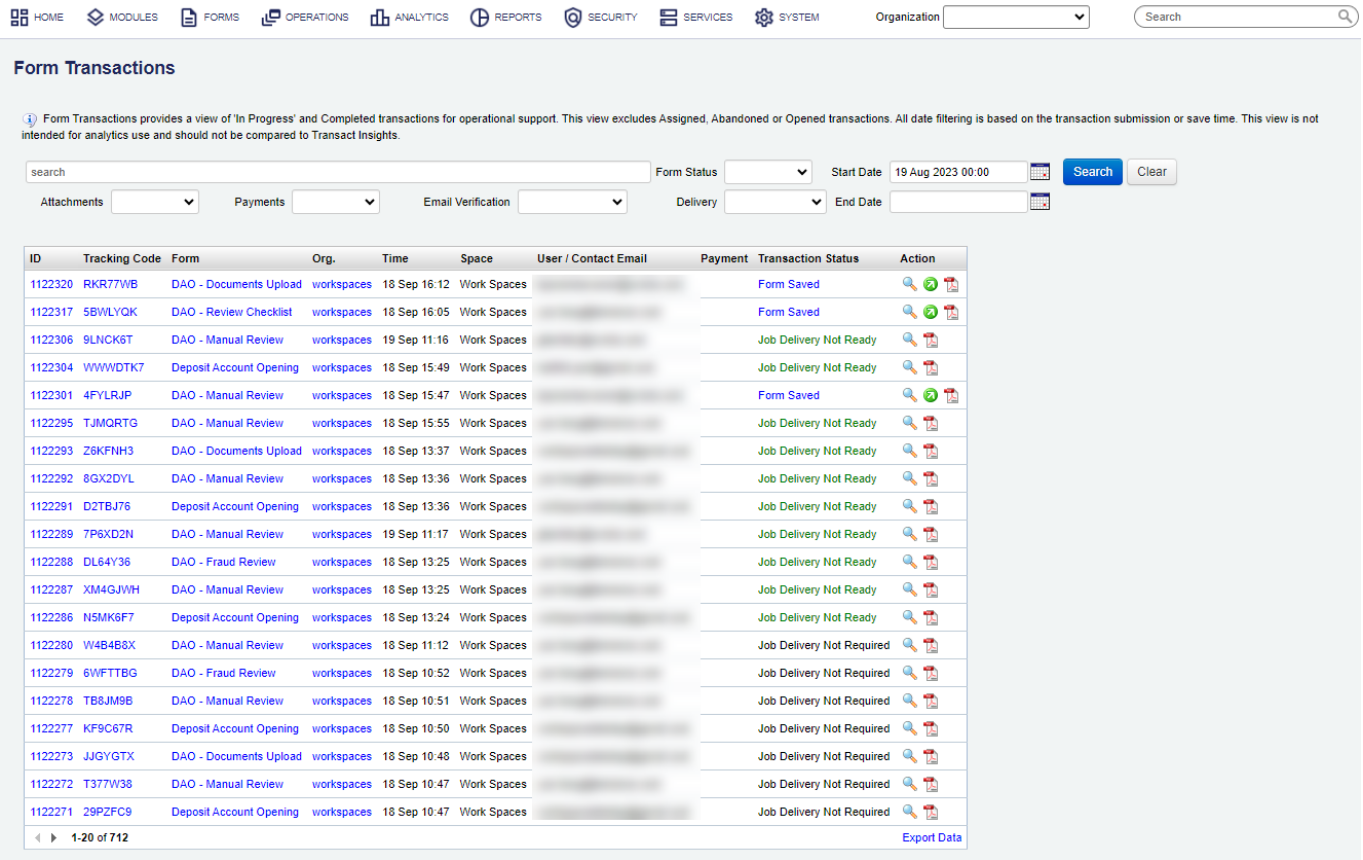
Next, learn how to view form transactions.

View Form Transactions

Manager keeps track of all transactions that represent form submissions made by application form users.

To view form transactions, select Operations > Form Transactions.

Transactions are shown page by page, with the maximum number of records per page configured via the user's preferences.



To filter or search the transactions by one or more criteria, specify the following settings and click Search:

- Transaction's name or part of a name.
- Form status: a status of a form, which can be one of the following:
  - Completed - a transaction that has been successfully submitted and contain all required data
  - Saved - a transaction that has been saved by the user or automatically by the system
  - Submitted - If transactions have post submission processing or elaborated delivery methods, they can remain in a submitted state. You can use this status to locate

submissions, which are potentially having problems in delivery and so on, so they have not moved to completed state.

- Attachments:
  - Required:
  - Optional:
  - Completed:
- Payments:
  - Required:
  - Completed:
  - Error:
  - Pending:
- Email Verification:
  - Required:
  - Pending:
  - Completed:
  - Abandoned:
  - Error:
  - Error - No Data:
- Delivery: a delivery status of a transaction, which can be one of the following:
  - Not Ready:
  - Ready:
  - Sent Email:
  - In Progress:
  - Pending:
  - Completed:
  - Error:
  - Undeliverable:
  - Not Required:
- Start Date:
- End Date:

Click Clear to reset the search criteria.

The key column is the Transaction Status, which indicates the status of the selected transaction and various actions associated with it.

You can perform the following actions:

- Click View to view transaction details.
- Click View Receipt to view the PDF receipt of the selected transaction.

- Click Retry Delivery to retry delivery of the transaction. This is only available if delivery of a transaction has not been completed, but submission data is still available.
- Click Retry Delivery From Scratch to attempts to deliver the submission using the current delivery details set up in the form after deleting all existing delivery checkpoints for the submission. If no delivery checkpoints exist, this is equivalent to the retry delivery option above. This is only available if delivery of a transaction has not been completed, but submission data is still available.
- Click Open Saved Form to open the saved form to allow an administrator to complete the form on behalf of the user.

  NOTE

  This action is only available to users with the Submission Save Edit permission.

- Click Payment Page to open a payment page where you can submit a payment using a valid credit card.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to view form requests.

Form Data Extract

Manager allows you to view all items and their values that were extracted from the submitted XML data of a form.

WARNING

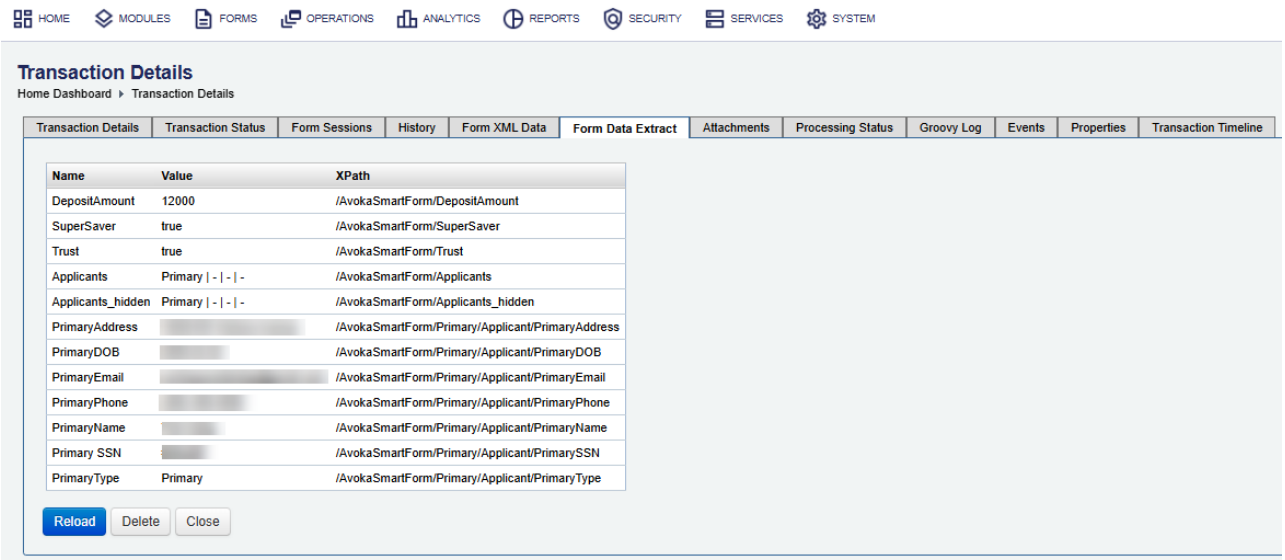
Form data extracts may contain PII[1] data, so you must adhere to the best practices of handling such data as well as global and local regulations.
We don't recommend extracting PII data.

NOTE

The Form Data Extract tab is only available if form data extracts are configured.

To view a form data extract:

1. Select Operations > Form Transactions.
2. Locate a transaction and click View.
3. Select the Form Data Extract tab to see a list of values for each configured data extract. For more information, see view form submission data extract.



[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

The list displays the following details:

- Name:
- Value:
- XPath:

4. Click Reload to refresh form XML data extract values.
5. Click Delete to remove the data extracts.
6. Click Close.

Next, learn how to view form submission data.

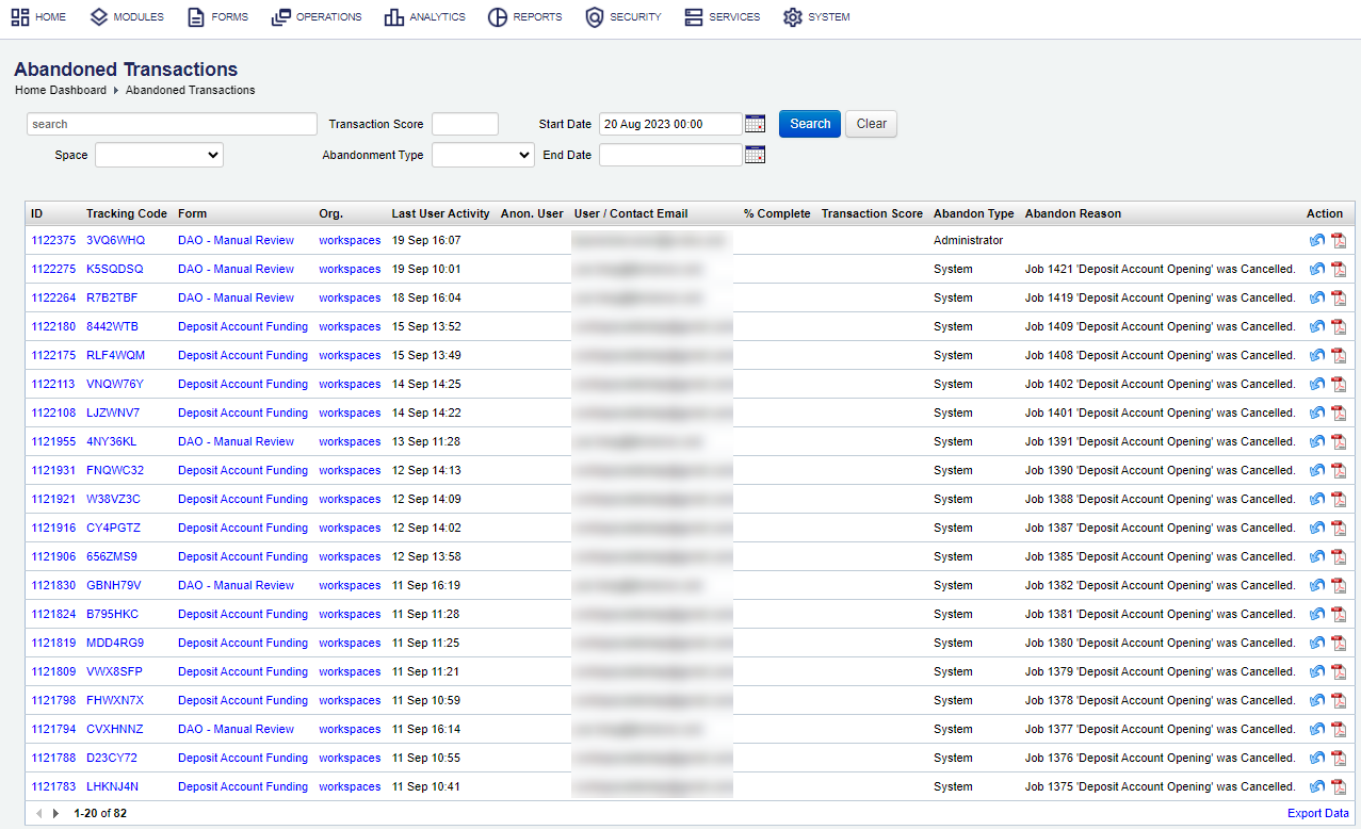
View Abandoned Transactions

Journey Manager allows you to view abandoned transactions so you can inspect them and make some corrective actions, such as changing their status.

> NOTE
>
> You must be an administrator to change an abandoned transaction's status.

To view the abandoned transactions and submissions, select Operations > Abandoned Transactions.

Transactions are shown page by page, with the maximum number of records per page configured via the user's preferences.



ID	Tracking Code	Form	Org.	Last User Activity	Anon. User	User / Contact Email	% Complete	Transaction Score	Abandon Type	Abandon Reason	Action
1122375	3VQ6WHQ	DAO - Manual Review	workspaces	19 Sep 16:07					Administrator		
1122275	K5SQDSQ	DAO - Manual Review	workspaces	19 Sep 10:01					System	Job 1421 'Deposit Account Opening' was Cancelled.	
1122264	R7B2TBF	DAO - Manual Review	workspaces	18 Sep 16:04					System	Job 1419 'Deposit Account Opening' was Cancelled.	
1122180	8442WTB	Deposit Account Funding	workspaces	15 Sep 13:52					System	Job 1409 'Deposit Account Opening' was Cancelled.	
1122175	RLF4WQM	Deposit Account Funding	workspaces	15 Sep 13:49					System	Job 1408 'Deposit Account Opening' was Cancelled.	
1122113	VNQW76Y	Deposit Account Funding	workspaces	14 Sep 14:25					System	Job 1402 'Deposit Account Opening' was Cancelled.	
1122108	LJZWNV7	Deposit Account Funding	workspaces	14 Sep 14:22					System	Job 1401 'Deposit Account Opening' was Cancelled.	
1121955	4NY36KL	DAO - Manual Review	workspaces	13 Sep 11:28					System	Job 1391 'Deposit Account Opening' was Cancelled.	
1121931	FNQWC32	Deposit Account Funding	workspaces	12 Sep 14:13					System	Job 1390 'Deposit Account Opening' was Cancelled.	
1121921	W38VZ3C	Deposit Account Funding	workspaces	12 Sep 14:09					System	Job 1388 'Deposit Account Opening' was Cancelled.	
1121916	CY4PGTZ	Deposit Account Funding	workspaces	12 Sep 14:02					System	Job 1387 'Deposit Account Opening' was Cancelled.	
1121906	656ZMS9	Deposit Account Funding	workspaces	12 Sep 13:58					System	Job 1385 'Deposit Account Opening' was Cancelled.	
1121830	GBNH79V	DAO - Manual Review	workspaces	11 Sep 16:19					System	Job 1382 'Deposit Account Opening' was Cancelled.	
1121824	B795HKC	Deposit Account Funding	workspaces	11 Sep 11:28					System	Job 1381 'Deposit Account Opening' was Cancelled.	
1121819	MDD4RG9	Deposit Account Funding	workspaces	11 Sep 11:25					System	Job 1380 'Deposit Account Opening' was Cancelled.	
1121809	VWX8SFP	Deposit Account Funding	workspaces	11 Sep 11:21					System	Job 1379 'Deposit Account Opening' was Cancelled.	
1121798	FHWXN7X	Deposit Account Funding	workspaces	11 Sep 10:59					System	Job 1378 'Deposit Account Opening' was Cancelled.	
1121794	CVXHNNZ	DAO - Manual Review	workspaces	11 Sep 16:14					System	Job 1377 'Deposit Account Opening' was Cancelled.	
1121788	D23CY72	Deposit Account Funding	workspaces	11 Sep 10:55					System	Job 1376 'Deposit Account Opening' was Cancelled.	
1121783	LHKNJ4N	Deposit Account Funding	workspaces	11 Sep 10:41					System	Job 1375 'Deposit Account Opening' was Cancelled.	

The list displays the following details for each transaction:

- The Transaction ID
- The Tracking Code
- The Name of the form
- The Organization associated with the form
- The time and date of the user’s last activity with the transaction

- The name of the user and contact email.
- The identification if the form user is anonymous represented by a tick ( )
- The Abandonment Type
- The Abandonment reason (if known)

To search the abandoned transactions list by one or more criteria, specify the following settings and click Search:

- Spaces
- Transaction Score
- Abandonment Type: User, System, Administrator, or Ineligible.
- Transaction text, or part of a transaction text.
- Name
- Start Date and End Date.

Click Clear to reset the search criteria.

You can perform the following actions on a selected transaction:

- Click Make Abandoned Transaction Saved to revert the submission to the Saved status, if submission data is available. Use this if an original submitter (user) would like to complete the submission now.
- Click View Receipt to view the receipt as a PDF file.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

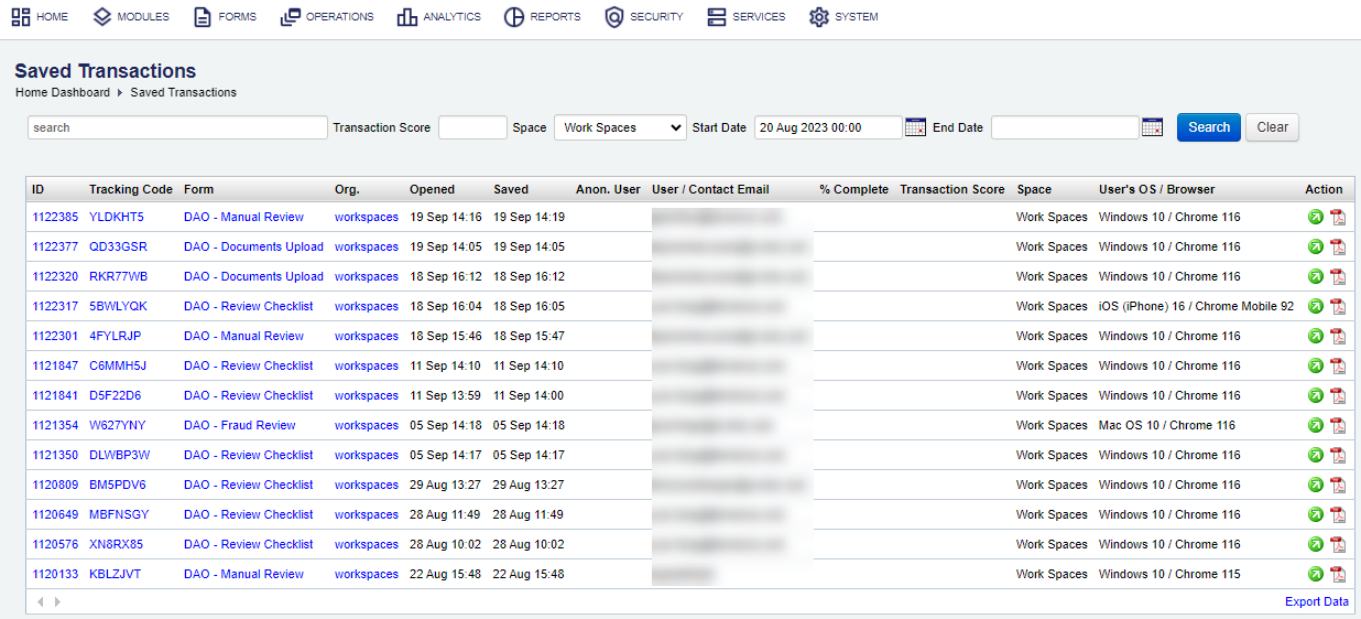
Next, learn how to configure form abandonment.

View Saved Transactions

Journey Manager allows an administrator to view and update saved transactions.

To view the saved transactions, select Operations > Saved Transactions.

Transactions are shown page by page, with the maximum number of records per page configured via the user's preferences.



ID	Tracking Code	Form	Org.	Opened	Saved	Anon. User	User / Contact Email	% Complete	Transaction Score	Space	User's OS / Browser	Action
1122385	YLDKHT5	DAO - Manual Review	workspaces	19 Sep 14:16	19 Sep 14:19					Work Spaces	Windows 10 / Chrome 116	
1122377	QD33GSR	DAO - Documents Upload	workspaces	19 Sep 14:05	19 Sep 14:05					Work Spaces	Windows 10 / Chrome 116	
1122320	RKR77WB	DAO - Documents Upload	workspaces	18 Sep 16:12	18 Sep 16:12					Work Spaces	Windows 10 / Chrome 116	
1122317	5BWLYQK	DAO - Review Checklist	workspaces	18 Sep 16:04	18 Sep 16:05					Work Spaces	iOS (iPhone) 16 / Chrome Mobile 92	
1122301	4FYLRJP	DAO - Manual Review	workspaces	18 Sep 15:46	18 Sep 15:47					Work Spaces	Windows 10 / Chrome 116	
1121847	C6MMH5J	DAO - Review Checklist	workspaces	11 Sep 14:10	11 Sep 14:10					Work Spaces	Windows 10 / Chrome 116	
1121841	D5F22D6	DAO - Review Checklist	workspaces	11 Sep 13:59	11 Sep 14:00					Work Spaces	Windows 10 / Chrome 116	
1121354	W627YNY	DAO - Fraud Review	workspaces	05 Sep 14:18	05 Sep 14:18					Work Spaces	Mac OS 10 / Chrome 116	
1121350	DLWBP3W	DAO - Review Checklist	workspaces	05 Sep 14:17	05 Sep 14:17					Work Spaces	Windows 10 / Chrome 116	
1120809	BM5PDV6	DAO - Review Checklist	workspaces	29 Aug 13:27	29 Aug 13:27					Work Spaces	Windows 10 / Chrome 116	
1120649	MBFNSGY	DAO - Review Checklist	workspaces	28 Aug 11:49	28 Aug 11:49					Work Spaces	Windows 10 / Chrome 116	
1120576	XN8RX85	DAO - Review Checklist	workspaces	28 Aug 10:02	28 Aug 10:02					Work Spaces	Windows 10 / Chrome 116	
1120133	KBLZJVT	DAO - Manual Review	workspaces	22 Aug 15:48	22 Aug 15:48					Work Spaces	Windows 10 / Chrome 115	

The transaction list displays the following details:

- Transaction ID
- Tracking Code
- Name of the form
- Organization associated with the form
- Time and date the form was opened
- Time and date the form was saved
- Name of the user and contact email
- % Complete
- Transaction Score
- Space
- User's OS / Browser

To search the saved transactions by one or more criteria, specify the following settings and click Search:

- Transaction text, or part of a transaction text.
- Transaction Score.
- Space
- Start Date
- End Date

Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

You can perform the following actions on a selected transaction:

- Click Open Saved Form to open the saved form to complete the form on behalf of the user.

  NOTE

  To use this option, you must have the Submission Save Edit permission.

- Click View Receipt to view the receipt as a PDF file.

Next, learn how to view submitted transactions.

View Payment Transactions

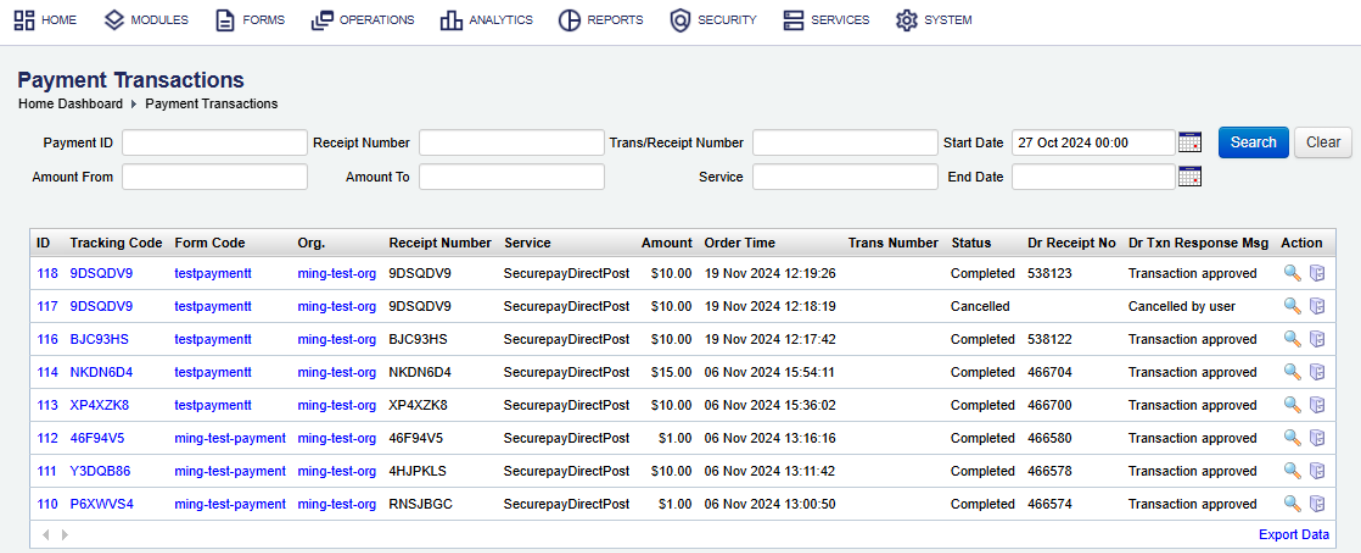
Manager allows you to view all credit card payments and payment attempts. This is useful when monitoring and troubleshooting problems that occurred during payments.

> NOTE
>
> To view details of payments made by form users, you must have the Payment Log View permission.

> NOTE
>
> Manager doesn't store the user's credit card details.

To view payment transactions, select Operations > Payment Transactions.

Transactions are shown page by page, with the maximum number of records per page configured via the user's preferences.



ID	Tracking Code	Form Code	Org.	Receipt Number	Service	Amount	Order Time	Trans Number	Status	Dr Receipt No	Dr Txn Response Msg	Action
118	9DSQDV9	testpaymentt	ming-test-org	9DSQDV9	SecurepayDirectPost	$10.00	19 Nov 2024 12:19:26		Completed	538123	Transaction approved	
117	9DSQDV9	testpaymentt	ming-test-org	9DSQDV9	SecurepayDirectPost	$10.00	19 Nov 2024 12:18:19		Cancelled		Cancelled by user	
116	BJC93HS	testpaymentt	ming-test-org	BJC93HS	SecurepayDirectPost	$10.00	19 Nov 2024 12:17:42		Completed	538122	Transaction approved	
114	NKDN6D4	testpaymentt	ming-test-org	NKDN6D4	SecurepayDirectPost	$15.00	06 Nov 2024 15:54:11		Completed	466704	Transaction approved	
113	XP4XZK8	testpaymentt	ming-test-org	XP4XZK8	SecurepayDirectPost	$10.00	06 Nov 2024 15:36:02		Completed	466700	Transaction approved	
112	46F94V5	ming-test-payment	ming-test-org	46F94V5	SecurepayDirectPost	$1.00	06 Nov 2024 13:16:16		Completed	466580	Transaction approved	
111	Y3DQB86	ming-test-payment	ming-test-org	4HJPKLS	SecurepayDirectPost	$10.00	06 Nov 2024 13:11:42		Completed	466578	Transaction approved	
110	P6XWVS4	ming-test-payment	ming-test-org	RNSJBGC	SecurepayDirectPost	$1.00	06 Nov 2024 13:00:50		Completed	466574	Transaction approved	

The list displays the following details:

- ID: a payment transaction ID. Click it to view payment transaction details.
- Tracking Code: a unique transaction code associated with a payment. Click it to view transaction details.
- Form Code: a unique form id identifying a form used for payment.
- Org.: the organization where the payment gateway is configured and the payment form is hosted.

- Receipt Number: a number identifying a rendered receipt.
- Service: a name of a service implementing the payment mechanism.
- Amount: the amount that was paid.
- Order Time: the date and time of the payment.
- Trans Number: a transaction number
- Status: a payment status, which can be one of the following:
    - Cancelled:
    - Completed:
    - Error:
    - Pending:
- DR Receipt No: a digital receipt number of a payment transaction.
- DR Txn Response Message: a payment description.

To filter or search the payment transactions by one or more criteria, specify the following settings and click Search:

- Payment ID
- Amount From
- Receipt Number
- Amount To
- Trans/Receipt Number: a transaction or a receipt number assigned by the payment gateway.
- Service
- Start Date
- End Date

Click Clear to reset the search criteria.

You can also perform the following actions:

- Click View or ID to view payment transaction details.
- Click View Submission or Tracking Code to view transaction details.
- Click Payment Query to re-query the payment gateway about the status of the payment, if the payment gateway supports this, which, in some cases, can also cancel the payment. The example of a status of the payment is shown below:

```
Payment Query: {Payment Log Status=PaymentLog has been cancelled, Submission Pay-
ment Status=Required}
Please note the Payment Gateway only maintains payment records for 3 days
```

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

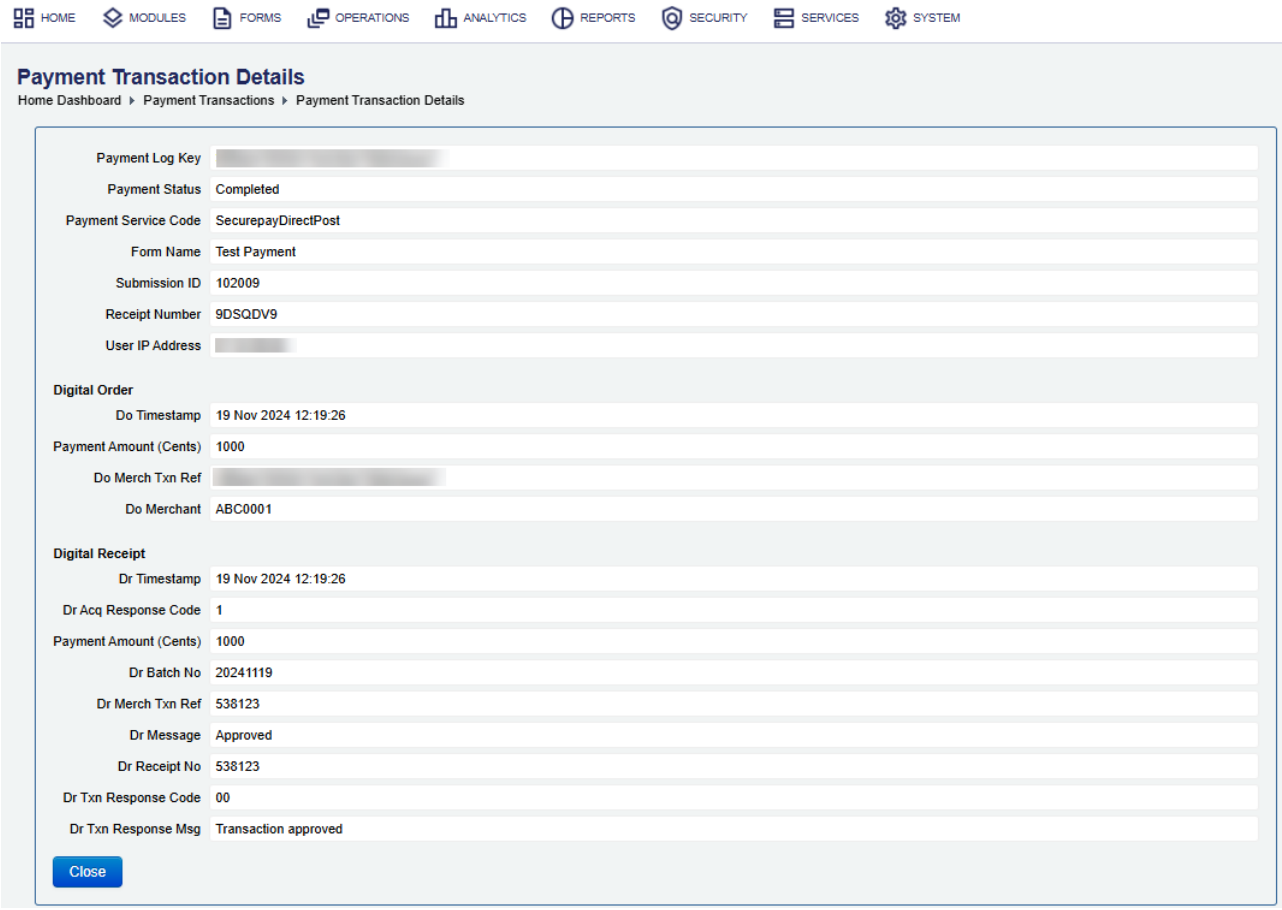
Next, learn how to view form requests.

Payment Transaction Details

Manager allows you to view payment information (a payment log entry) of a selected payment transaction. This is useful when monitoring and troubleshooting problems that occurred during payments.

To view payment transaction details:

1. Select Operations > Payment Transactions.
2. Locate a transaction and click View or ID to see payment transaction details.



> NOTE
>
> The payment transaction details may vary between payment gateways, so not all fields may be populated.

3. Check the Payment Transaction Details section, including:

- Payment Log Key:
- Payment Status:
    - Cancelled:
    - Completed:
    - Error:
    - Pending:
- Payment Service Code: the payment service that was used to make this payment.
- Form Name:
- Submission ID:
- Receipt Number:
- User IP Address: the IP address of the user making the payment.

4. Check the Digital Order (DO) section, including:
    - DO Timestamp: the day and time of a digital order.
    - Payment Amount (Cents): the payable amount, which is usually in cents.
    - DO Merch Txn Ref:
    - DO Merchant: the Merchant ID.
5. Check the Digital Receipt (DR) section, including:
    - DR Timestamp: the day and time of the response from the payment gateway.
    - DR Acr Response Code:
    - Payment Amount (Cents):
    - DR Batch Number: the batch in which the payment was processed, as obtained by the payment gateway.
    - DR Merch Txn Ref:
    - DR Message: a message containing some additional information, such as Payment Approved.
    - DR Receipt Number: a receipt number.
    - DR Txn Response Code: a response type, such as Success or Error.
    - DR Txn Response Message: a message containing diagnostic information.
6. Click Cancel Payment Transaction to cancel the selected payment transaction.

Next, learn how to view form submission data.

View Form Submission Data

Manager allows you to view data extracts of form submissions, which is useful for analyzing and troubleshooting user interaction with forms. For example, you can find out a value of a radio button that a user selected for each transaction for a specific form.

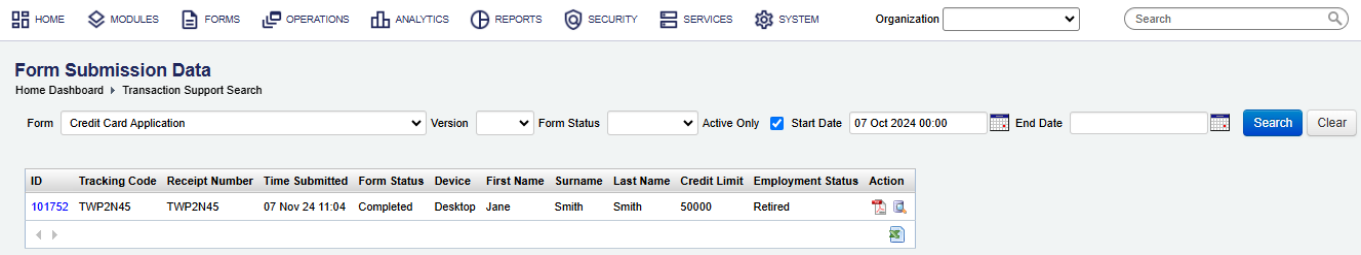
WARNING

Form data extracts may contain PII[1] data, so you must adhere to the best practices of handling such data as well as global and local regulations.
We don't recommend extracting PII data.

NOTE

Data extracts are purged frequently and may not be available by the time you perform a search.

To view form submission data, select Operations > Form Submission Data.



The form submission data list displays

The list displays the following details:

- ID: a transaction ID.
- Tracking Code: a tracking code.
- Receipt Number: a receipt number.
- Time Submitted: time stamp when a form was submitted.
- Form Status: a form status.

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

- Device: a device used to submit a form, for example, Desktop.
- Form XML data for each transaction. The number of columns is different for each form and depends on the form XML data mapping.

To search the form submission data by one or more criteria, specify the following settings and click Search:

- Form: a form you want to analyze.
- Version: a version of a form, or blank if not applicable.
- Form Status:
  - Saved
  - Submitted
  - Completed
  - Abandoned
- Active Only: to show the Active forms only
- Start Date:
- End Date:

Click Clear to reset the search criteria.

Click ID or View Submission Details to see the transaction details with data extract information.

Click View PDF Receipt to view and download a selected form submission data as a PDF file.

> NOTE
>
> You can't export multiple forms as a PDF file.

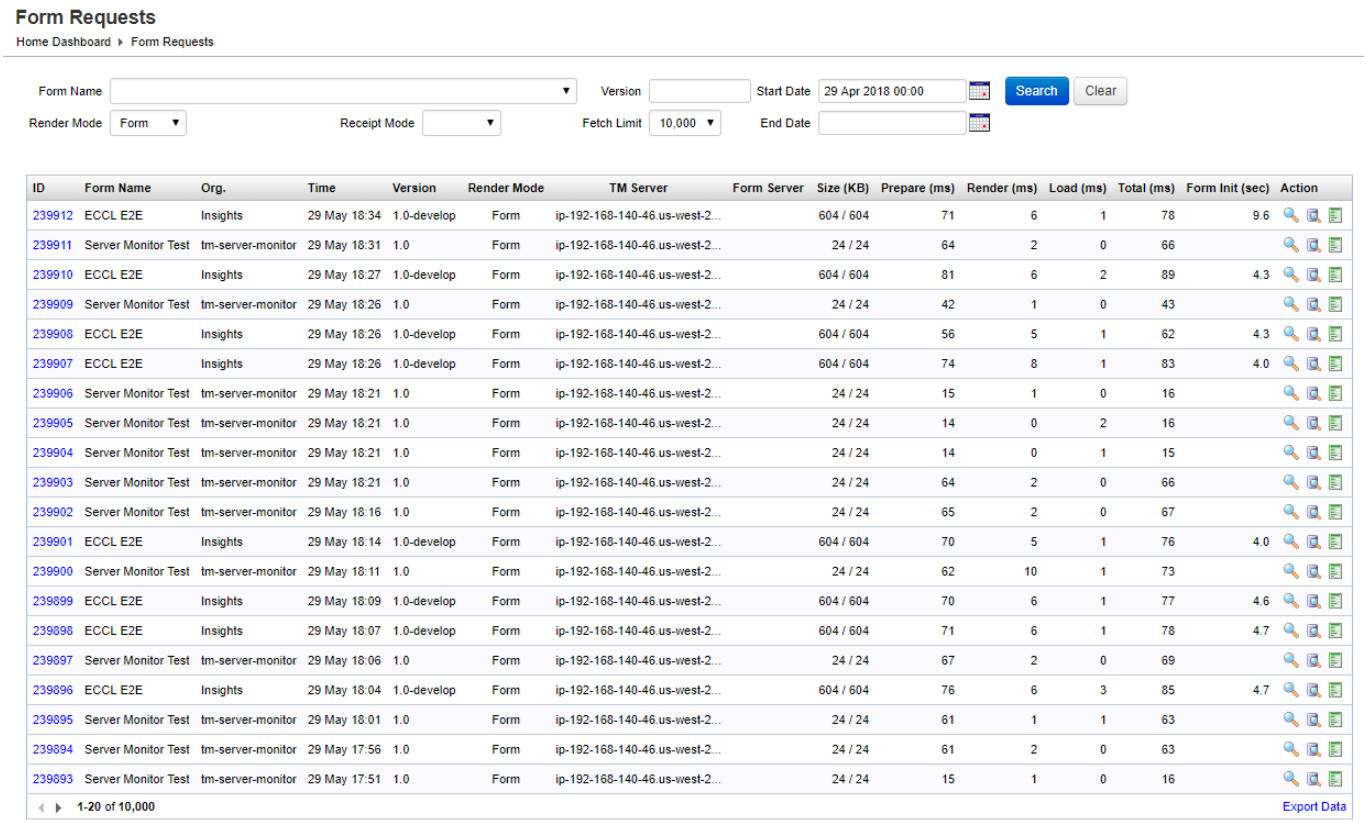
Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to view form transactions.

View Form Requests

Manager allows you to view all form requests initiated by users. When a user selects a form, this action is logged as a form request, which can even include a generated receipt. You can use this information to troubleshoot transaction issues and analyze form request processing performance.

To view form requests, select Operations > Form Requests.



ID	Form Name	Org.	Time	Version	Render Mode	TM Server	Form Server	Size (KB)	Prepare (ms)	Render (ms)	Load (ms)	Total (ms)	Form Init (sec)	Action
239912	ECCL E2E	Insights	29 May 18:34	1.0-develop	Form	ip-192-168-140-46.us-west-2...		604 / 604	71	6	1	78	9.6	
239911	Server Monitor Test	tm-server-monitor	29 May 18:31	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	64	2	0	66		
239910	ECCL E2E	Insights	29 May 18:27	1.0-develop	Form	ip-192-168-140-46.us-west-2...		604 / 604	81	6	2	89	4.3	
239909	Server Monitor Test	tm-server-monitor	29 May 18:26	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	42	1	0	43		
239908	ECCL E2E	Insights	29 May 18:26	1.0-develop	Form	ip-192-168-140-46.us-west-2...		604 / 604	56	5	1	62	4.3	
239907	ECCL E2E	Insights	29 May 18:26	1.0-develop	Form	ip-192-168-140-46.us-west-2...		604 / 604	74	8	1	83	4.0	
239906	Server Monitor Test	tm-server-monitor	29 May 18:21	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	15	1	0	16		
239905	Server Monitor Test	tm-server-monitor	29 May 18:21	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	14	0	2	16		
239904	Server Monitor Test	tm-server-monitor	29 May 18:21	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	14	0	1	15		
239903	Server Monitor Test	tm-server-monitor	29 May 18:21	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	64	2	0	66		
239902	Server Monitor Test	tm-server-monitor	29 May 18:16	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	65	2	0	67		
239901	ECCL E2E	Insights	29 May 18:14	1.0-develop	Form	ip-192-168-140-46.us-west-2...		604 / 604	70	5	1	76	4.0	
239900	Server Monitor Test	tm-server-monitor	29 May 18:11	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	62	10	1	73		
239899	ECCL E2E	Insights	29 May 18:09	1.0-develop	Form	ip-192-168-140-46.us-west-2...		604 / 604	70	6	1	77	4.6	
239898	ECCL E2E	Insights	29 May 18:07	1.0-develop	Form	ip-192-168-140-46.us-west-2...		604 / 604	71	6	1	78	4.7	
239897	Server Monitor Test	tm-server-monitor	29 May 18:06	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	67	2	0	69		
239896	ECCL E2E	Insights	29 May 18:04	1.0-develop	Form	ip-192-168-140-46.us-west-2...		604 / 604	76	6	3	85	4.7	
239895	Server Monitor Test	tm-server-monitor	29 May 18:01	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	61	1	1	63		
239894	Server Monitor Test	tm-server-monitor	29 May 17:56	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	61	2	0	63		
239893	Server Monitor Test	tm-server-monitor	29 May 17:51	1.0	Form	ip-192-168-140-46.us-west-2...		24 / 24	15	1	0	16		

The form requests list displays the following details for each transaction:

- ID - the transaction request identifier.
- Form Name - the name of the form.
- Org - the organization associated with the form.
- Time - the date and time that the request was made.
- Version - a form version.
- Render Mode - refers to the task requested by the user, which either a form request or receipt request.
- TM Server - the host name of a Manager server.
- Size (KB) - the size of the request (in kilobytes).

- Prepare (ms) - the amount of time (in milliseconds) it took for the server to prepare the request.
- Render (ms) - the amount of time (in milliseconds) it took for the server to render the request.
- Load (ms) - the amount of time (in milliseconds) it took for the server to load the request.
- Total (ms) - the total amount of time (in milliseconds) it took from the request being made to the request being rendered to the user.
- Form Init (sec) - the amount of time (in seconds) it took to initialize the form. This is only relevant for forms using form functions.

To search the form requests by one or more criteria, specify the following settings and click Search:

- A form name.
- A form version.
- Render Mode – Form or Receipt.
- Receipt Mode

  A receipt modes can be one of the following:

  - Web: the receipt was requested by a user using a web browser.
  - Delivery: the receipt was generated during the submission delivery process.
  - Email: a form user had requested an email copy of the receipt.
- Fetch limit to constrain the number of records to select and display. | 18.05 This feature was introduced in 18.05.

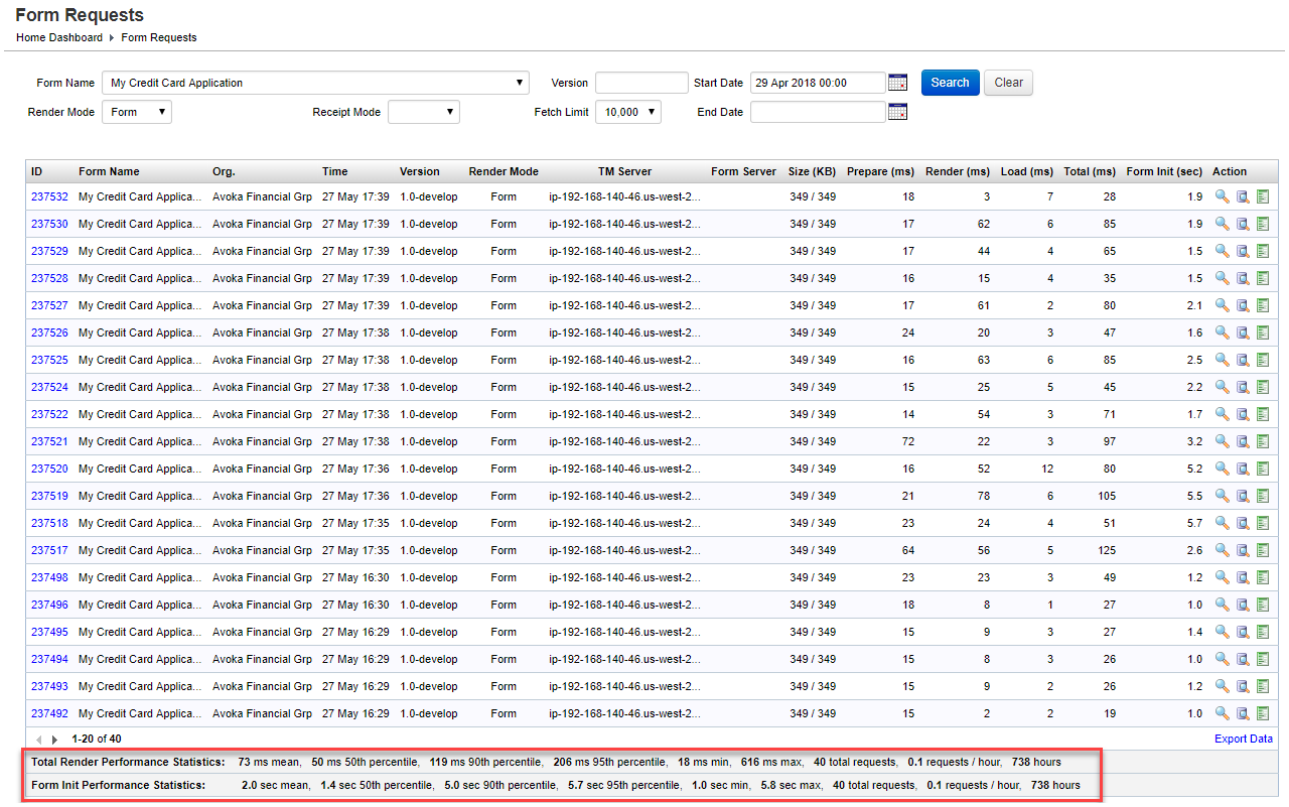
  > NOTE
  >
  > This page is automatically limited to display 10,000 log entries to avoid performance issues. If you wish for more or fewer log entries to be displayed, select a number from the Fetch Limit dropdown list.

- Start Date and End Date – By default, this is set to a date range starting one month before, up to and including today.

To view a form performance statistics summary for each form:

1. Select the Show Performance Stats checkbox in system preferences
2. Select the form from the Form Name dropdown list.
3. Click Search to generate the following statistics. | 18.05 This feature was introduced in 18.05.

- mean
- 50th percentile
- 90th percentile
- 95th percentile
- minimum time
- maximum time
- total requests
- number of requests per hour



You can also perform the following actions:

- Click View to see the additional details of a selected form request.
- Click View Submission Details to see the transaction details related to a form submission.
- Click View Form Details to go to the dashboard of a form related to this form request.

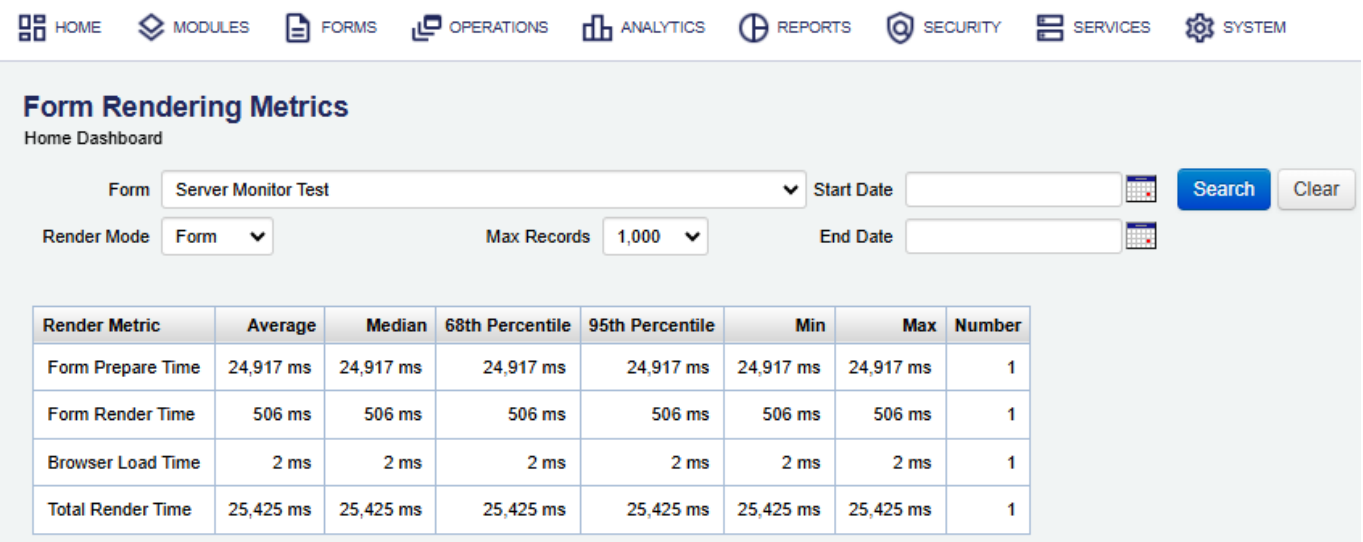
Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to view form sessions.

View Form Rendering Metrics

Manager allows you to see how fast forms are rendered to users, so you can analyze their rendering performance. The metrics include the total render time (in milliseconds) and its components for all form requests.

To view the form rendering metrics, select Operations > Form Rendering Metrics:



Render Metric	Average	Median	68th Percentile	95th Percentile	Min	Max	Number
Form Prepare Time	24,917 ms	24,917 ms	24,917 ms	24,917 ms	24,917 ms	24,917 ms	1
Form Render Time	506 ms	506 ms	506 ms	506 ms	506 ms	506 ms	1
Browser Load Time	2 ms	2 ms	2 ms	2 ms	2 ms	2 ms	1
Total Render Time	25,425 ms	25,425 ms	25,425 ms	25,425 ms	25,425 ms	25,425 ms	1

The list displays the following details:

- Form Prepare Time: the amount of time it took to prepare the render request.
- Form Render Time: the amount of time it took to render the form request on to the server.
- Browser Load Time: the time it took to stream the rendered form to the user's browser.
- Total Render Time: the total time it took from the moment the request was made to when the form was displayed in the browser. This time is the sum of the form prepare time, form render time, and browser load time.

Each of the render metrics are broken down into the following groups:

- Average:
- Median:
- 68th Percentile: the 68th percentile[1].

[1]The 68th percentile is often used because it roughly corresponds to the "typical" range within a normal distribution, where approximately 68% of data points fall within one standard deviation from the mean

- 95th Percentile: the 95th percentile[1].
- Min:
- Max:
- Number:

To filter or search the form rendering metrics by one or more criteria, specify the following settings and click Search:

- Form
- Render Mode: form, receipt
- Max Records
- Start Date and End Date

Click Clear to reset the search criteria.

> NOTE
>
> If you select a large number of records in the Max Records field, the metrics calculation may take longer.

Next, learn how to view form requests.

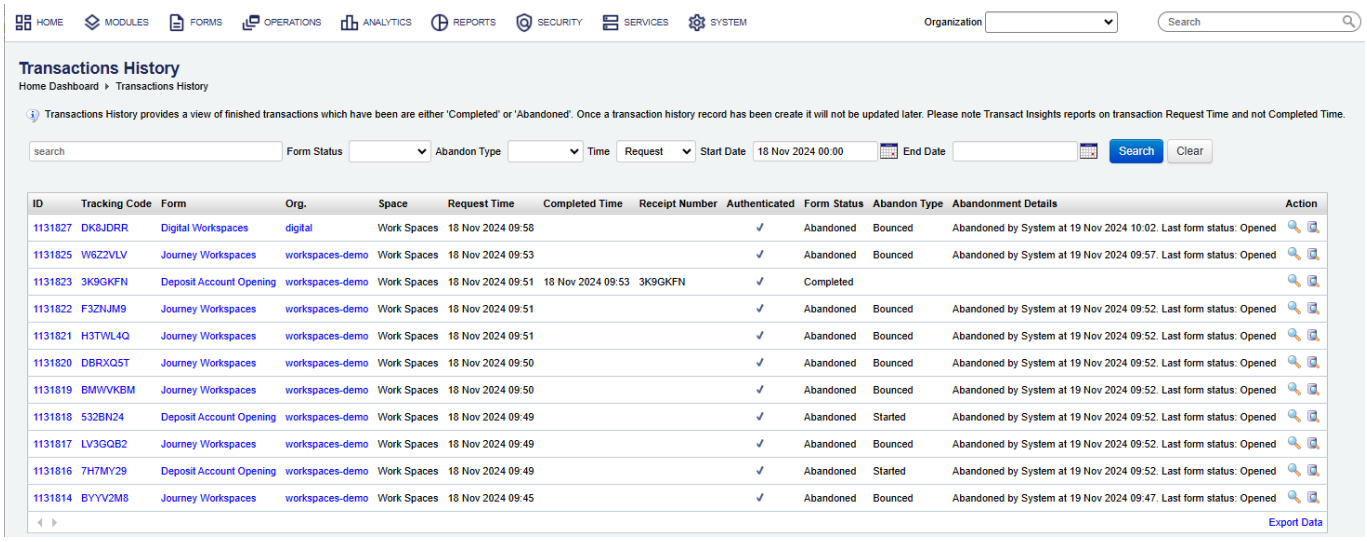
[1]The 95th percentile is commonly used because it provides a reliable picture of performance by focusing on the experience of the vast majority of users, excluding extreme outliers, making it a valuable metric for understanding typical performance levels while mitigating the impact of rare, large spikes in data, especially in situations like network traffic, application response times, or system resource usage.

View Transactions History

Manager allows you to view transactions history of form submissions made by users. It provides a view of finished transactions which have been are either Completed or Abandoned. Once a transaction history record has been created, it will not be updated later. Journey Analytics reports on transaction Request Time and not Completed Time.

To view transactions history, select Operations > Transactions History.

Transactions are shown page by page, with the maximum number of records per page configured via the user's preferences.



The list displays the following details:

- ID: a unique transaction ID.
- Tracking Code: a tracking code of a transaction.
- Form: a form name.
- Org.: an organization that a form belongs to.
- Space: a form space a form is hosted on.
- Request Time:
- Completed Time:
- Receipt Number:
- Authenticated:
- Form Status: a status of a form, which can be one of the following:
    - Completed
    - Abandoned

- Abandon Type: a type of the transaction abandonment, which can be one of the following:
  - Bounced: a transaction is bounced
  - Started: a transaction is started
  - Saved: a transaction is saved
  - Cancelled: a transaction is canceled
  - Ineligible: a transaction is ineligible
  - Submitted: a transaction is submitted
- Abandonment Details: a reason why the transaction was abandoned.

To filter or search the transactions by one or more criteria, specify the following settings and click Search:

- Transaction's name or part of a name
- Form status
- Abandon Type
- Time, which can be one of the following:
  - Request:
  - Completed:
- Start Date: the date and time of a transaction.
- End Date: the date and time of a transaction.

Click Clear to reset the search criteria.

You can perform the following actions:

- Click View to the transaction history or transaction history details.
- Click View Submission Details to view transaction's details.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to view form requests.

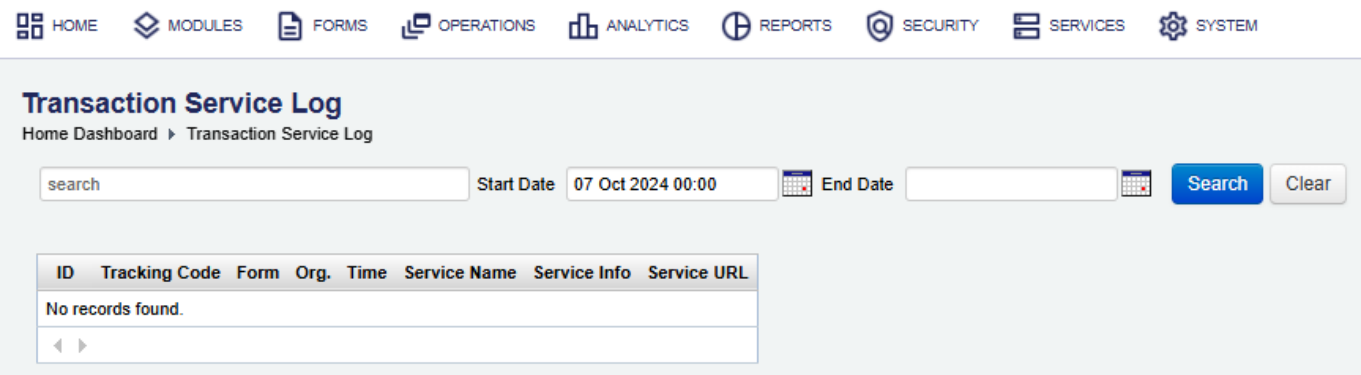
View Transaction Service Log

Manager allows you to view logging records made by services in regards to user transactions.

> NOTE
>
> We recommend using Transaction Support Search instead.

To view transaction service log, select Operations > Transaction Service Log.

Transactions are shown page by page, with the maximum number of records per page configured via the user's preferences.



The list displays the following details:

- ID: a unique transaction ID.
- Tracking Code: a tracking code of a transaction.
- Form: a form name.
- Org.: an organization that a form belongs to.
- Time: a time of a transaction.
- Service Name: a name of a service.
- Service Info: a service info.
- Service URL: a service URL.

To filter or search the transactions by one or more criteria, specify the following settings and click Search:

- Transaction's name or part of a name
- Start Date: the date and time of a transaction.
- End Date: the date and time of a transaction.

Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to view form requests.

Debug Maestro Forms

If a Maestro form doesn't behave as expected, it might be related to one of the business rules you've created using the Maestro script editor. It's very important to understand the form architecture before you delve into form debugging, as there are certain assumptions of how the 3 form elements are used in interact within the form.

Tracking down problems in business rules can be tricky, so we've summarized several useful techniques for finding and fixing problems using Chrome DevTools. which are available when you open a form in a Google Chrome browser. Most modern browsers come with good developer tools these days. We prefer to use Google Chrome, but Firefox and Microsoft Edge can also be used.

Maestro forms are built on the AngularJS framework and plugins are available to provide specific support for AngularJS web applications. For example, in Chrome, you can use AngularJS Batarang.

Debugging application forms in the browser is a technical task and requires some understanding of web application development in general, specifically the web standards of HTML, CSS and JavaScript. We recommend getting familiar with HTML, CSS, and JavaScript technologies to be able to debug forms. Also, as Maestro runs on the AngularJS JavaScript framework, AngularJS experience is useful if you'll be debugging on a regular basis, but it is not required. If you want to know more about AngularJS, read the official tutorial.

The debugging process includes the following steps:

1. Compare form versions
2. Build a form
3. Open Chrome DevTools
4. Inspect the Form object
5. Debug JavaScript Rules
6. Set Breakpoints

Compare a Form Version with a Previous Version

Check what changes have been done in a form version you are troubleshooting as some changes can cause unexpected behaviors. You can use the Compare Designs functionality to view the differences between 2 form versions. You should review and understand whether these changes caused any issues and correct them if needed.

Build a Form

Before you can start debugging a Maestro, you need to build it with the Minify Code checkbox unselected, so unnecessary characters, such as white space, new lines, and comments, are not removed.

> NOTE
>
> If the Minify Code is selected, the result of minification is a single line of code that is very long and difficult to read and debug.

After that, you render the form so it's opened in a new Google Chrome browser. If a new browser tab doesn't open, check if your browser is blocking pop-ups and, if so, allow them from the Maestro URL.

Whenever you make a change to the form, you'll need to rebuild it.

> NOTE
>
> You can't debug a Maestro form in the Design, Code, or Preview modes.

Open Chrome DevTools

Chrome DevTools provide access to a range of features for diving into the look and feel, functionality, and performance of web applications. Once the published form is loaded in the browser you can activate the browser's developer tools.

To open Chrome DevTools:

1. Select the browser tab that contains the rendered Maestro form.
2. Click `F12` or `Control+Shift+I` to open the DevTools pane, as shown below:

```
Elements  Console  Sources  Network  Performance  Memory  Application  Security  »   3 1 1
top ▾   app.js   Default levels ▾   1 Issue: 1   3 hidden
Bootstrapping                                                    app.js:726
Form loading                                                     app.js:2384
Bootstrapped in 48 ms                                            app.js:729
Form loaded in 29ms                                              app.js:2448
> maestro.Form
<· ▾ Form {maestroVersion: '23.10.1', view: {…}, data: {…}, items: {…}, pageName: 'landingPage', …}
    ▸ $Pages: Pages {viewIndex: 0, isPageDisplayed: true, policy: {…}, index: 0, isLeft: undefined, …}
    ▸ Rules: Rules {load_AvokaSmartForm: ƒ, onExpiry_sessionTimeoutCountdown: ƒ, eq_sessionTimeoutShowDialog: ƒ, sh_header: ƒ, chang
    ▸ Transact: Transact {userSave: ƒ, userSubmit: ƒ, userCancel: ƒ, formLoad: ƒ, formStart: ƒ, …}
    ▸ Translation: Translation {T: {…}, fetch: ƒ, setValues: ƒ, set: ƒ, updateDict: ƒ, …}
    ▸ addPreviewDDSData: ƒ (serviceName, dataOrHandler)
    ▸ addRepeatInstance: ƒ (item, repeatData, transferFocus)
    ▸ availableRoutes: ƒ ()
    ▸ backgroundSave: ƒ ()
    ▸ cacheData: ƒ (itemOrId, data)
    ▸ clearValidationErrors: ƒ (itemOrId, data)
    ▸ completeSectionsOnSubmit: [{…}]
    ▸ convertToFieldDataMap: ƒ (fieldRefMap, data)
    ▸ copyData: ƒ (fromItem, fromData, toItem, toData)
    ▸ data: {locale: '', selectLanguage: 'en', addressLine1: '', addressLine2: '', city: '', …}
      dialogName: ""
      dialogTemplate: ""
      disableSave: false
    ▸ emailOperation: ƒ (operationName, opItemId, emailAddressId, senderNameId, emailBodyId)
    ▸ fireRule: ƒ (ruleType, itemOrItemId, data, responseData)
    ▸ fixReceiptLayout: ƒ (formContainer)
    ▸ focusOnFirstHeading: ƒ (element, failHandler)
    ▸ getBlockData: ƒ (item, data)
    ▸ getCurrentPageId: ƒ ()
    ▸ getCurrentPageNumber: ƒ ()
    ▸ getDataXml: ƒ (prettify, excludeSfmData, dataObj)
    ▸ getDisplayValue: ƒ (elementOrId, data)
    ▸ getDynamicPageTitle: ƒ ()
    ▸ getElementItemId: ƒ ($element)
    ▸ getEntityData: ƒ (path, onlySubmittedData)
    ▸ getFieldRefPropValue: ƒ (itemOrPath, prop)
    ▸ getImagesByName: ƒ (imageName)
    ▸ getItem: ƒ ($scope, $element)
    ▸ getItemFromPath: ƒ (path)
    ▸ getItemProperty: ƒ (pathOrItem, propName)
    ▸ getItemsOfType: ƒ (type)
    ▸ getNavPageCount: ƒ ()
    ▸ getNavPages: ƒ ()
    ▸ getPathData: ƒ (path, source)
    ▸ getRepeatData: ƒ (itemOrId, data, filterMethod)
    ▸ getRepeatedLayout: ƒ (col, threshold)
    ▸ getRevisionNumber: ƒ ()
    ▸ getSelectionLabel: ƒ (dataPath, data)
    ▸ getSystemData: ƒ (name)
    ▸ getUniqueId: ƒ (id, data)
    ▸ getXmlPath: ƒ (item, includeRepeatParents)
    ▸ goToNextError: ƒ ()
Console   What's New ×
```

For more information, see Google's Chrome DevTools.

Inspect the Form Object

When you load a Maestro form in your browser, a Form object, which contains the data required to render the form, is initialized. In the Maestro framework, the Form object contains everything else you might need to access for debugging including the View, Items and Data, so to access any of these elements you need to scope the Form object. There are a number of ways to do this from the Console tab in the browser developer tools.

To inspect the Form object:

1. Selecting the Console tab and executing one of the following queries to retrieve a Form object:
   - Using JQuery:

     ```
     $("Form").scope().Form
     ```

   - Using the Maestro root element included with the framework:

     ```
     maestro.Form
     ```

   - Using AngularJS Batarang, if installed:

     ```
     $scope.Form
     ```

     This may also allow you to access the Form object by way of the `$scope` object. However, you need to select an element in the Elements tab first, for example, a body or form.

2. Expand the Form object to see all elements. Usually, the object is quite large, so you need to know that there are three objects within the Form object that are worth exploring: `data`, `items`, and `view`. For more information, see Maestro Form Architecture.
3. Check these 3 main objects. If you have a JavaScript debugging session active, as described below, you will have access to the Form Data object in the function scope and can readily inspect its content.

Debug JavaScript Rules

All business rules in Maestro are implemented in JavaScript. As a result, a straightforward way to debug these rules is by using `console.log` statements in the script editor: `console.log("Hello world...");`.

To distinguish between different types of output, you can also use the `console.warn("This message will have a yellow background");` and `console.error("This message will have a red background");` methods:

Below are some useful snippets for outputting relevant data inside a business rule:

```
// Inside a business rule, "value" refers to the data attached to the current field:
console.log(value);

// The "data" object, explained above, can be output from inside a business rule:
console.log(data)

// You can access any field value via the data object by referring to its ID:
console.log(data.lastName)

// The "item" object, explained above, can be output from inside a business rule:
console.log(item)

// Some components, such as the Page Controller component, output extra information via
the "info" object:
console.log(info)
```

You can run various commands in the console to print form's data. For example, you can use `console.log(maestro.Form.getDataXml(true))` to list the `/SFMData/SystemProfile` content as an XML file, as shown below:

```
<AvokaSmartForm>
       <AboutYou>
               <FirstName>
               </FirstName>
       </AboutYou>
       <SFMData>
               <SystemProfile>
                   <FormCode>Test 1</FormCode>
                       <RequestLogKey>12345</RequestLogKey>
                       <DisplayMode>Entry</DisplayMode>
                       <ServerBuildNumber>22.4.0</ServerBuildNumber>
                       <RevisionNumber>0</RevisionNumber>
                       <FormDataServiceURL>https:/[/YOUR-SERVER-HOST-NAME]/web-plu-
gin/servlet/FormDynamicDataServlet</FormDataServiceURL>
                       <TrackingCode>ABCD1234</TrackingCode>
                       <AbandonmentReason>
                       </AbandonmentReason>
                       <AbandonmentComments>
                       </AbandonmentComments>
               </SystemProfile>
       </SFMData>
</AvokaSmartForm>
```

There is a handy Util function to generate a printable representation of an item and then log it to the console: `$scope.Util.logItem($scope.Form.items.personalDetailsBlock)`. The output is shown below:

```
"personalDetailsBlock
┌ firstName
└ middleNames
- lastName
- dateOfBirth
 ┌ day
 │ month
 └ year
 - date
 - daysData
- emailAddress
- mobileNumber
"
```

While `console.log` statements are easy to use, you'll need to rebuild and reload the form every time you make a change. This makes it a fairly slow method of debugging. For more productive debugging, we recommend using breakpoints.

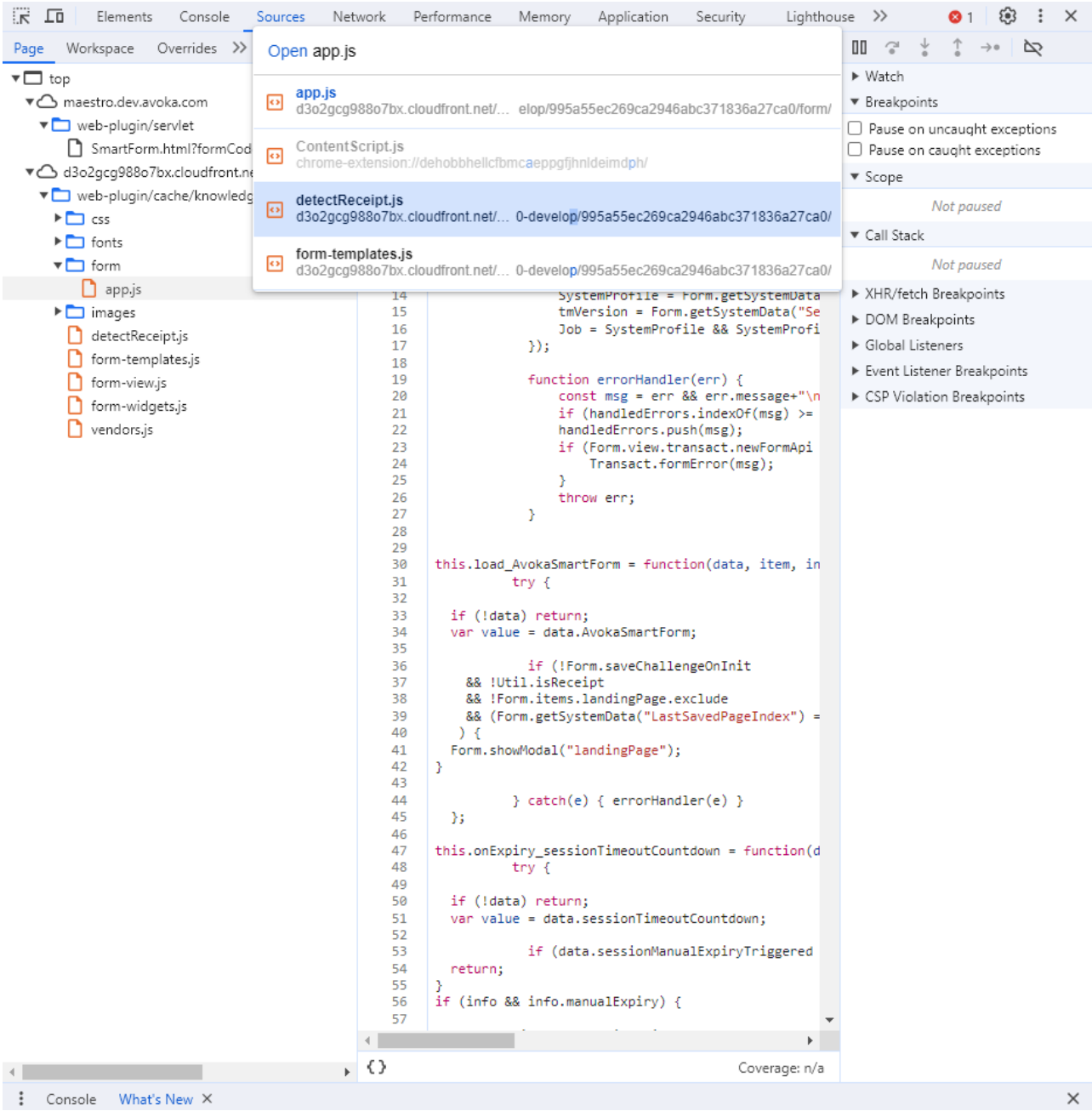
Set Breakpoints

Breakpoints allow you to define places in your JavaScript where the script execution must pause, so you can investigate where exactly your code breaks without having to write a bunch of `console.log` statements or continually rebuild and refresh the form.

All JavaScript code used by form's business rules are stored in the `app.js` file, so you must locate this file and set breakpoints there.

To set breakpoints:

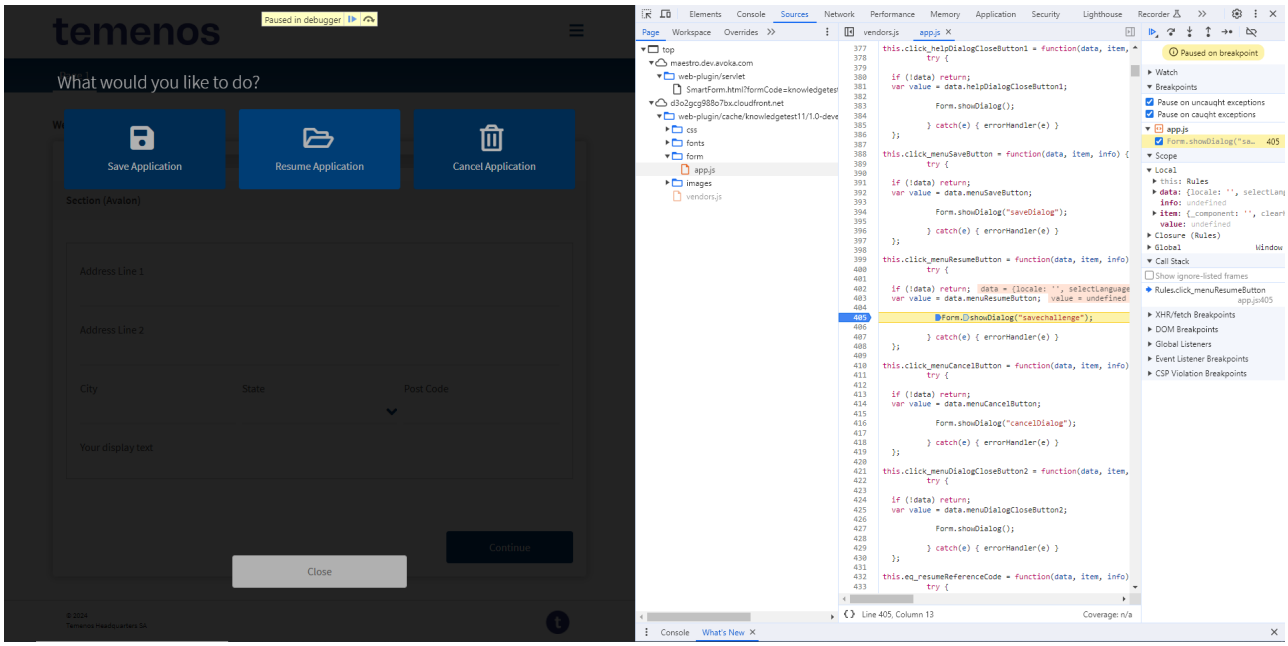
1. Select the Sources tab in Chrome DevTools.
2. Click `Control+O` (`Command+O` on a Mac) and type `app.js` to find all references to the `app.js` file. Alternatively, navigate through the Page hierarchy on the left until you see the `app.js` file.



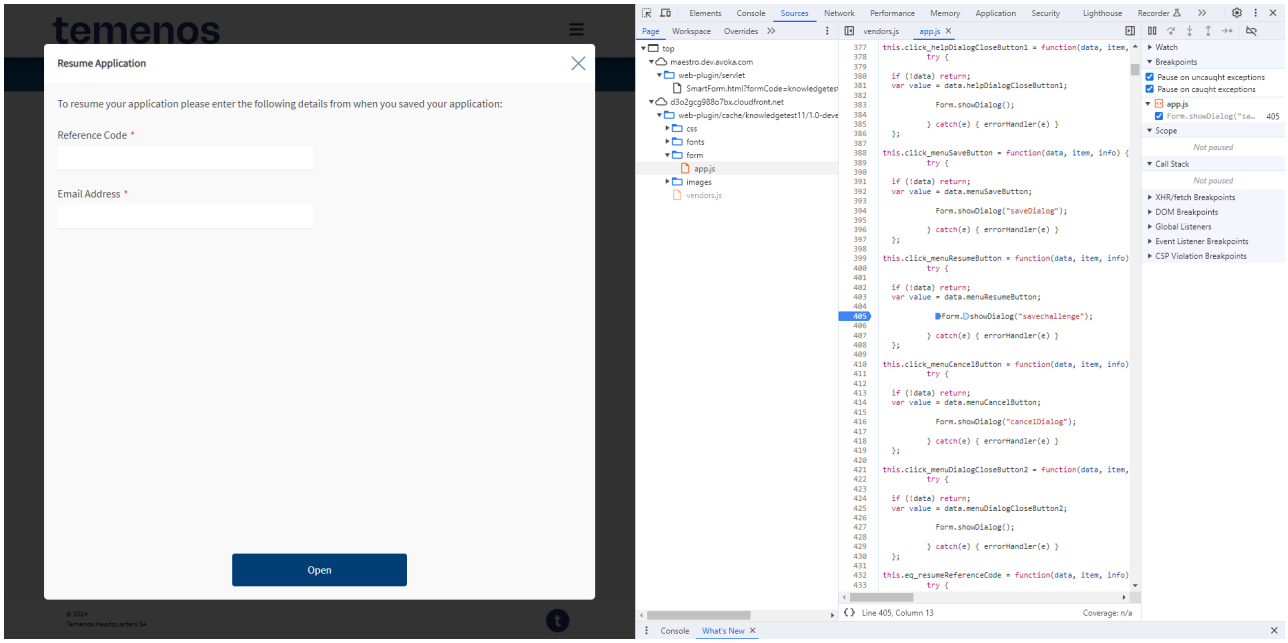
3. Select the `app.js` and click it.
4. Click `Control+F` on Windows or `Command+F` on Mac to find the `Form.showDialog ("savechallenge")` line of code and set a breakpoint there. In this example, the code opens the modal window allowing form users to re-open a previously saved form to resume the application. If you're debugging more complicated logic, you can set multiple breakpoints. For more information, see [Pause your code with breakpoints](#).
5. Select Pause on uncaught exceptions or Pause on caught exceptions to stop a script execution depending on exceptions. Now you are ready to debug the rule. You can return to

your form window and perform the action that will trigger the rule to execute and step through your rule script using the standard JavaScript debug functions.

6. Click Resume Application to execute the `Form.showDialog("savechallenge")` method. However, the modal window doesn't appear. Instead, the line of code where the breakpoint is set is highlighted and the Paused in debugger message is displayed to indicate that the execution of the JavaScript has paused on that line. This is illustrated below:

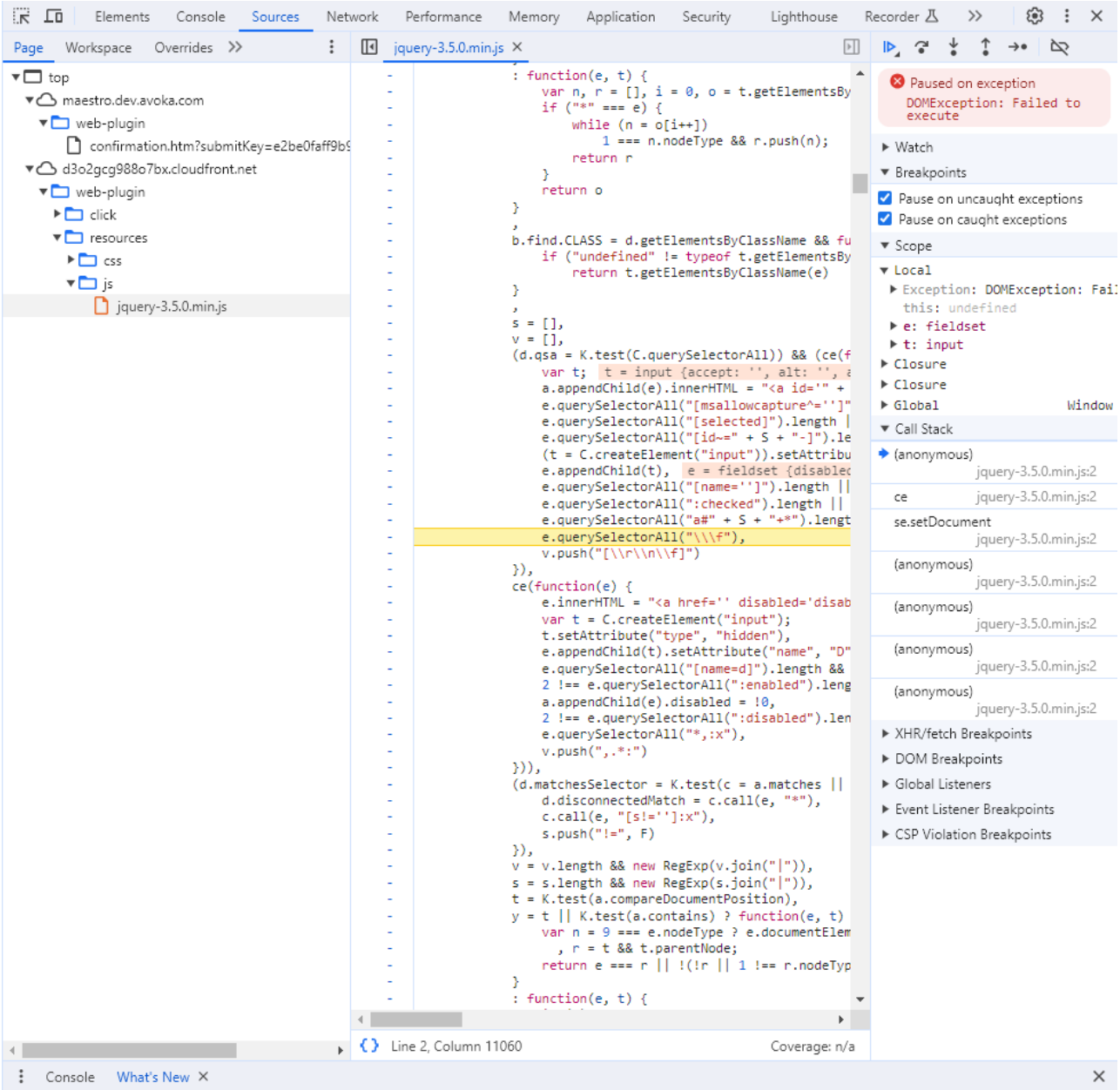


7. Click Play to continue executing the script, thereby allowing the modal window to open.



When you continue after each breakpoint, execution of the script will stop at the next breakpoint, allowing you to move step-by-step through a script.

While moving between breakpoints, pay attention to the Scope section, as it shows the current state of the form's data that may provide insight into why something isn't behaving as expected.



As you saw above, it's not difficult to find the `app.js` file. In an actual form though, the file may contain hundreds of rules, which can make it difficult to find and debug the code for a specific rule. One way is to use the `Control+F` shortcut to search for a statement from one of the rules. For example, if you know there is a `console.log` statement in the rule, you can search for `console.log`.

But if you have many rules, which contain similar statements, this maks this approach less convenient. Fortunately, every rule in Maestro has a corresponding JavaScript function that follows a strict naming convention. The functions are named using a rule type code and the ID of the field, with an underscore between them - `<rule-type-code>_<field-id>`. For more information, see List of Maestro Rules.

You may need to review your rule script in the Maestro Code editor and select a snippet that is likely to be unique. For example, if you search for the snippet `Form.getItemFromPath('data.buttonGroup')` and located the function `eq_ddLabel` that contains the business rule script.

> NOTE
>
> We don't recommend modifying JavaScript in-line in the browser debug tools because results are often unreliable. Instead, you should rebuild and republish a form each time you want to change a business rule.

Debug Form Styles

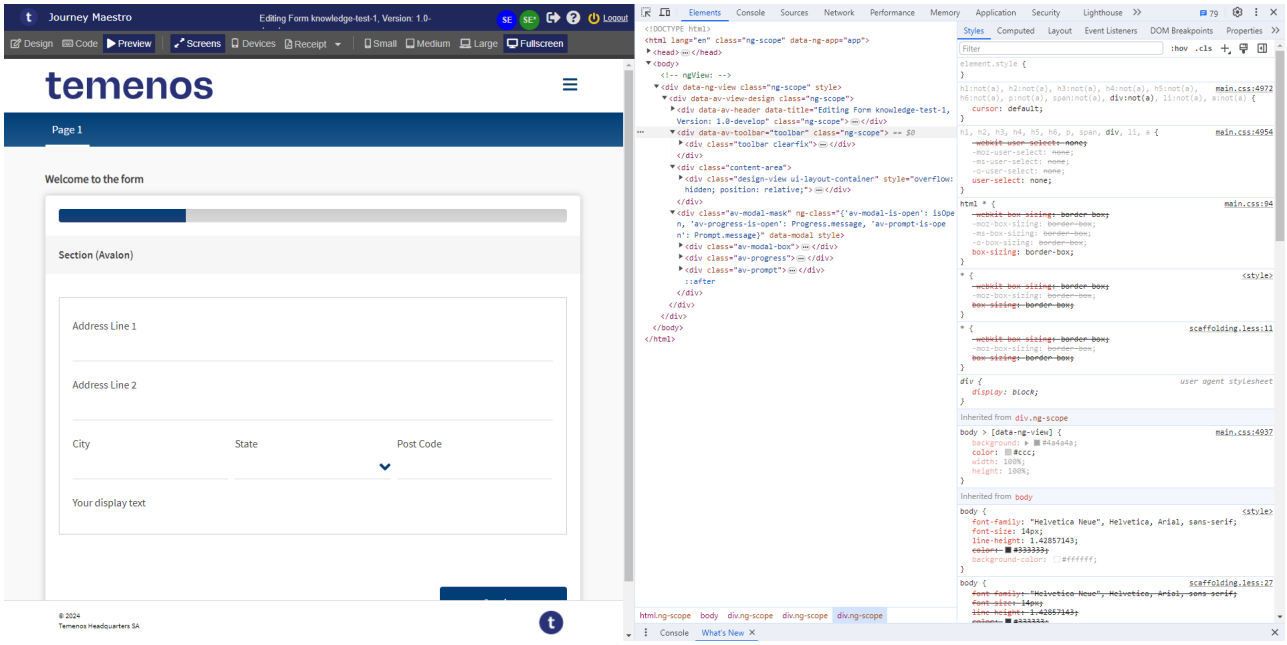
Maestro forms and templates use CSS extensively. CSS styling can become complicated for big form application. In particular, if two different CSS statements provide conflicting instructions about how to style a particular item, it can be very confusing to debug why your changes do not appear to be working.

To debug styling problems, we recommend using the native debugging features built into your browser of choice, rather than attempting to understand this entirely within Maestro. You need to have a moderate to advanced understanding of CSS to debug styles.

To debug form's styles:

1. Select Preview and open Chrome DevTools.
2. Select Elements > Styles to explore a style of a selected element of the HTML form. This is

shown below:



When there are style conflicts, the usual CSS rules apply:

- If two identical instructions are processed, the one that is later in the CSS definition will override the earlier one.
- The more specific the CSS selector, the higher it's precedence.
- Precedence can be fine-tuned by adding the `!Important` declaration.

Next, learn about [UI test automation](#).

Change Data Format in Form XML Data

Manager allows you to change data format in the XML[1] data of a selected transaction submitted by the user.

Starting from Manager version 18.11.0 the product now uses Open JDK 11, therefore the XML-based locale data is the Unicode Common Locale Data Repository (CLDR), older version of JDK used JRE(COMPAT), the new XML based locale introduced a stop character after the month Abbreviation for some locales like en_AU.

Below is an extract of the Form XML Data, where the Submit Date String before Transact Manager 18.11 is displayed:

```
<OfflineSubmissionId/>
<ReceiptNumber>XXXXXX</ReceiptNumber>
<SubmitDateString>17 Sep 2019 23:35:55 pm</SubmitDateString>
<UpdateUserProfile>false</UpdateUserProfile>
<ShareForm/>
<ServerBuildNumber>18.11.5</ServerBuildNumber>
<RevisionNumber>1</RevisionNumber>
<TrackingCode>XXXXXX</TrackingCode>
```

Below is an example of the Submit Date String from Manager 18.11 onwards:

```
<OfflineSubmissionId/>
<ReceiptNumber>XXXXXX</ReceiptNumber>
<SubmitDateString>17 Sep. 2019 23:35:55 pm</SubmitDateString>
<UpdateUserProfile>false</UpdateUserProfile>
<ShareForm/>
<ServerBuildNumber>18.11.5</ServerBuildNumber>
<RevisionNumber>1</RevisionNumber>
<TrackingCode>XXXXXX</TrackingCode>
```

To be able to get the date stamp without the extra full stop after the month, you will need to update your JBoss virtual machine variables.

---

[1]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

You have to update the `JAVA_OPTS` variable to append the property string `-Djava.locale.providers=COMPAT`.

This action needs to be performed at `manager/server/bin/standalone.conf` on the Manager instance of the server.

> NOTE
>
> You will need to restart the Manager server to pull the new change.

Below is how the change in the configuration file should looks like in the different operating systems:

Linux

File: `standalone.conf`

Config: `JAVA_OPTS="$JAVA_OPTS -Djava.locale.providers=COMPAT"`

Windows

File: `standalone.conf.bat`

Config: `set "JAVA_OPTS=%JAVA_OPTS% -Djava.locale.providers=COMPAT"`

Next, learn how to track submissions.

Operations Portal Overview

Journey Manager (JM) The transaction engine for the platform. | System Manager / DevOps | 22.04 This feature was introduced in 22.04.

IT operations refers to a set of processes and services provided by a business's IT department to support the business' IT hardware and software. Manager includes features that support IT operations for Temenos Journey Manager (TJM) solutions. These features allow you to manage and keep track of all transactions associated with user form submissions.

The Operations portal is a new Manager application focused on TJM operations features, and targeted at System Managers and DevOps users responsible for providing operations services. You can use theOperations portal to find transactions and then drill down into transaction details.

Features available in the Operations portal include:

- Find transactions based on data such as tracking code, form name, and date created.
- Targeted views for several transaction states including form transactions, saved transactions, and abandoned transactions, as well as assigned tasks.
- View a transaction's details, grouped by tabs containing custom information cards.
- View the receipt for a selected transaction.
- View form XML for a submitted transaction.
- Purge all transaction data or PII data only for selected transactions.
- Abandon a transaction, and recover an abandoned transaction.
- Resume a saved transaction.
- Update the status of several transaction fields including Attachments Status, Form Status, Receipt Status, and Delivery Status, as well as a task's Subject, Message, and Assigned User.
- View attachments, comments, event logs, error logs and other transactional metadata that are part of a selected transaction.

Concepts

Several Manager concepts are key to understanding how Operations portal works. More terminology is defined in the Glossary.

Collaboration Job

A collaboration job controls a workflow of transactions that represents user interaction with a form. A collaboration job can be considered as a series of tasks that Manager creates for users to complete, review or reject a form. A collaboration job consists of several steps with

each step containing actions or endpoints. Steps are not ordered, but actions are sequential within each step.

To learn more about collaboration jobs, see Collaboration Jobs Overview.

Transaction

A transaction is a fundamental element of Journey Manager, representing a user interaction with a form.

To learn more about transactions, see Transactions Overview.

Task

In Manager, a task is a combination of:

- A form or form-code.
- Some XML data, used to pre-populate the form.
- The ID of a user or group that the task is assigned to.

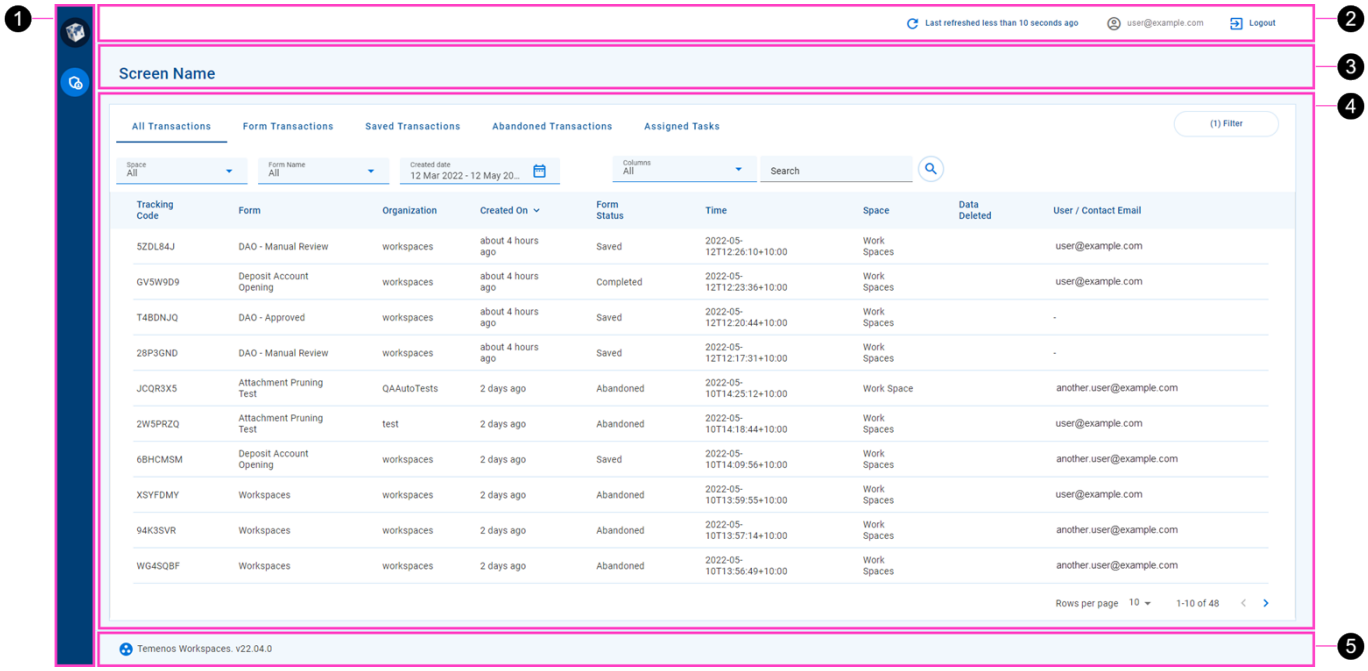
To learn more about tasks, see Tasks Overview

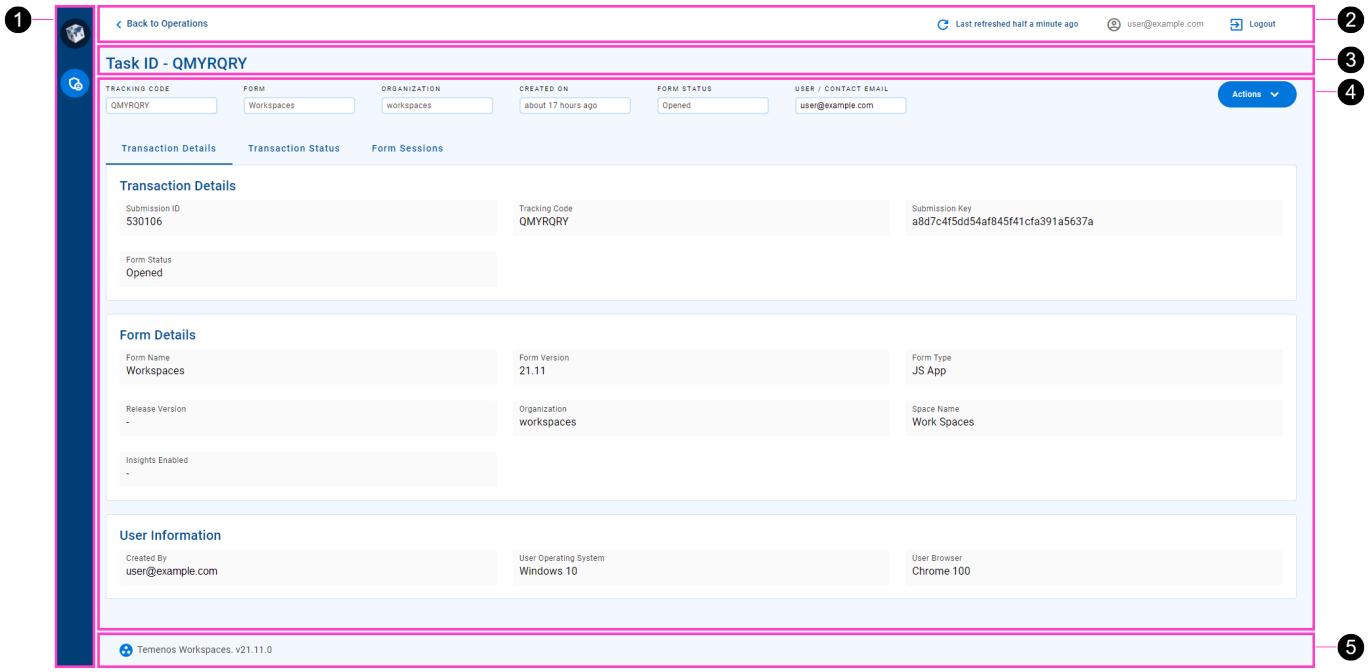
UI Tour

With a consistent layout and common user interface elements, Journey Operations is designed to give you a great user experience right from the start. Let's take a look at some of the common elements that make up the Journey Operations user interface (UI).

Each screen is based on a common layout that can be modified as required through configuration. All screens have a header and screen title at the top, a navigation bar on the left, and a footer at the bottom that provide common functionality. The remainder of each screen contains content and features that are specific to the selected space and the screen type (List, Search or Details).

- List screen
- Details screen





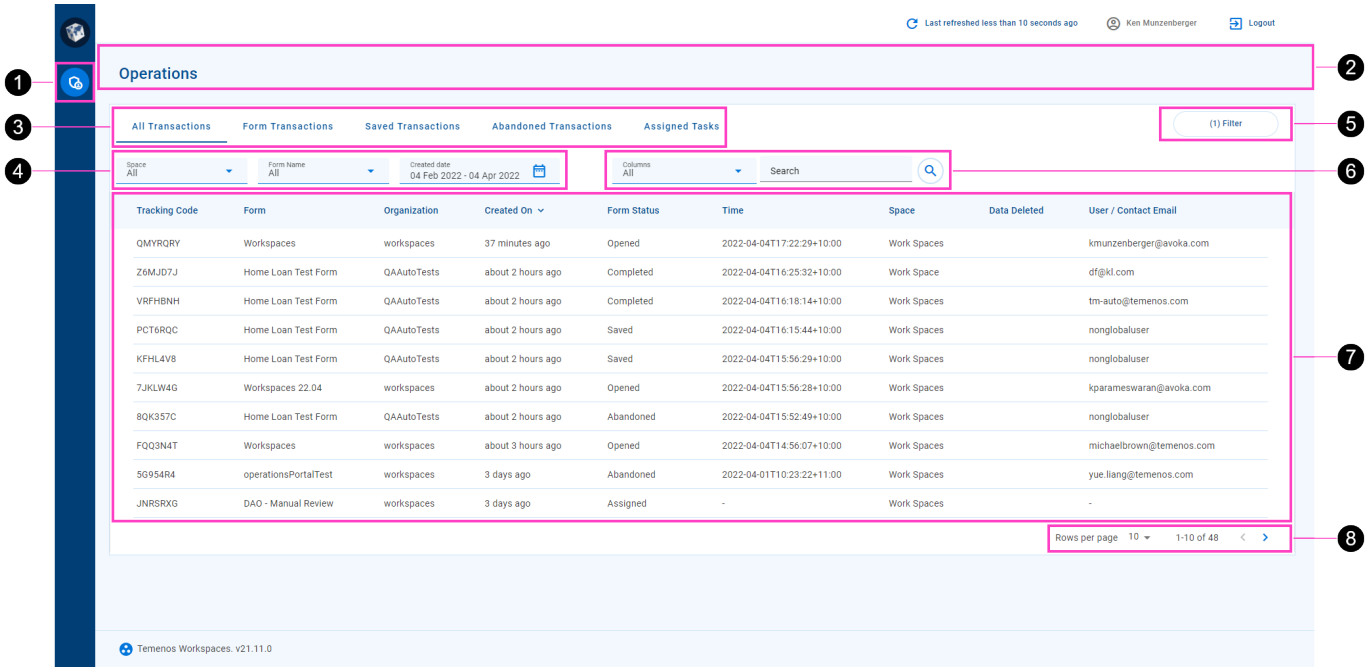
1. **Spaces navigation bar (navbar):** The Spaces navbar lets you select a space to work in. For example, select the **Operations** navbar item to switch to the Operations space.
2. **Screen header:** The screen header displays when the content pane was last refreshed and includes an option to refresh it. The name of the logged in user is also displayed, and there's an option to logout.
3. **Screen name:** The screen name changes depending upon the screen currently displayed. For a List screen, the screen name is the same as the name of the selected space. For a Details screen, the screen name corresponds to an item selected on a List screen.

4. **Content pane:** The majority of each screen displays content specific to the screen type (List or Details).
5. **Screen footer:** At the bottom of the screen is the screen footer which displays the application name and version.

Manage Transactions

The Operations portal is designed to assist System Managers and DevOps users in meeting their responsibility for providing operations services. One such responsibility relates to managing transactions at various stages of their life cycle. To support transaction management, the Operations portal includes an Operations space which has features that help you to find transactions, drill down into the transaction details, and perform appropriate actions such as recover an abandoned transaction or view the receipt for a submitted transaction.

Selecting the Operations space displays a List screen configured with a set of features common to all Journey UI applications. For an introduction to these common features, see UI Tour. Here, we'll focus on the additional features focused on transaction management available on the List screen.



1. The **Operations** space navbar item.
2. This screen's name, **Operations**, corresponding to the selected space.
3. A tabset presenting the views available on this screen.
4. A set of global filters that restrict the data fetched from Journey Manager.
5. A **Filter** button that provides access to view filters.
6. Components supporting the Search feature.
7. A sortable item list populated with transaction items matching the selected view.
8. Item list paging tools.

The main element on the List screen is the item list which displays brief information for a particular set of transactions. The view, filter, and search selectors at the top of the screen are the main way you control which transactions are displayed in the item list. There are two kinds of filter: global filters that are available for all views, and view filters that are specific to each view. If you can't see the transaction you're looking for, you can sort the item list to change the order items appear. Finally, paging tools let you select how many items are displayed on an item list page and browse through all the items one page at a time.

Once you've found the transaction you're looking for, you can interact with it in different ways. On the List screen, you can point your cursor at an item to reveal one or two shortcut action buttons on the right, then click one of these buttons to perform the corresponding action. Alternatively, you can click anywhere else on a list item to display more information about the transaction on the Details screen where you can interact with it further, including to perform actions.

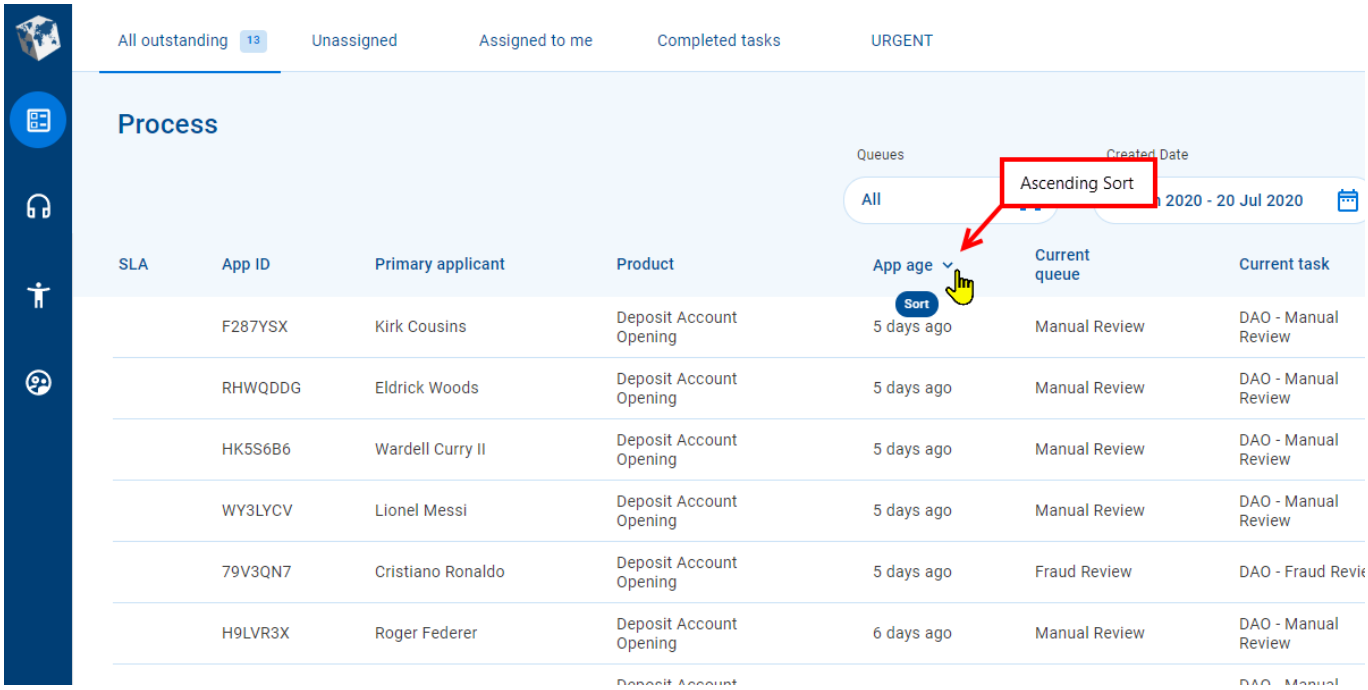
The available actions for a transaction change over the course of it's life cycle. While some actions are always available, other actions are only available when, for example, the transaction is in a particular state. Further, some actions are available from the Details screen only. To learn about actions, see Workspaces Actions.

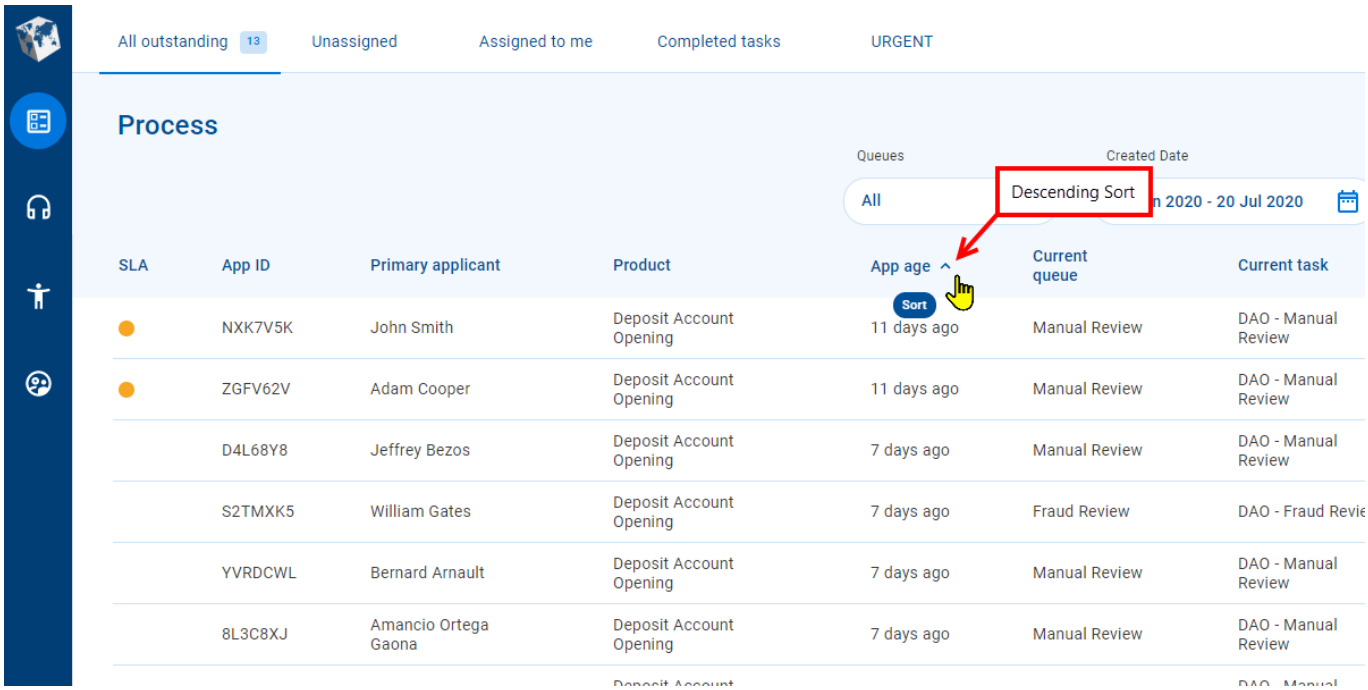
Item List

The main element on the List screen is the item list which displays brief information for a particular set of transactions. The items displayed in the item list satisfy a set of criteria selected by the user starting with a view then refined by searching and filtering. Each row of the item list represents a single transaction, and displays brief information that helps you to identify the transactions you're interested in. The fields (columns) displayed in the item list and the available filters are dependent on which view is selected. To learn more about views in general and the specific views available in the Operations portal, see Views below.

Items in the item list are displayed in an order determined by the current sort selection. This is indicated by an icon on the right-hand side of a column name. The expand_moreSort Ascending icon indicates the item list is sorted by the column values in ascending order, while expand_lessSort Descending indicates items are sorted by the column values in descending order.

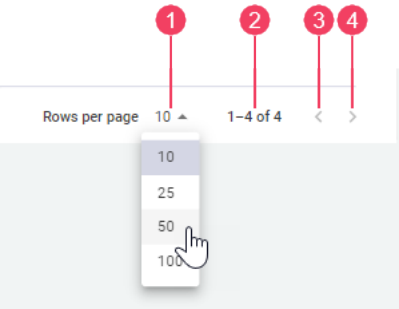
- Sort – Ascending order
- Sort – Descending order





You can determine whether a column is available to use for sorting by pointing your cursor at a column heading. If your cursor changes to the hand pointer and a "Sort" tooltip is displayed, you can sort by the column. To change the sort order, click the column name. If you click the column name for the current sort selection, sorting will alternate between ascending and descending order.

Sometimes, there are too many items to display all at once on the screen. Paging allows you to see some of the items now then browse the item list one page at a time to see more. Paging tools let you choose the number of items to display on an item list page, and navigate (forward or backward) through the available pages of items.



1. Rows per page
2. Items in list / Total matching items
3. Previous Page
4. Next Page

The navigate_beforePrevious Page and navigate_nextNext Page paging tools let you browse through item list pages to find the items you need. Click these tools to display the previous or next page of items in the item list. The navigate_beforePrevious Page tool is disabled when the first item list page is displayed; similarly, the navigate_nextNext Page tool is disabled when the last item list page is displayed.

Rows per page shows the maximum number of items displayed on an item list page. You can select a different page size from the Rows per page dropdown. Selecting a different page size, resets the item list to show the first page again.

Views

A view is a predefined configuration of item list fields, view filters and sort options. Selecting a view applies it's configuration to the item list on a List screen, changing the fields displayed and determining which view filters are available. These settings are selected automatically when a viewis displayed causing various screen elements to be updated, and reflecting the transactions that are now available for the selected view.

The Operations portal provides the following views:

- **All Transactions:** All transactions.
- **Form Transactions:** Transactions representing form submissions made by application form users.

- **Saved Transactions:** Transactions for forms that have been saved by application form users but not yet submitted.
- **Abandoned Transactions:** Transactions that have become abandoned actively by user actions, such as canceling an application, or passively based on the form's abandonment settings.
- **Assigned Tasks:** Transaction-related tasks that are assigned to a single user or a user group.

Click the links above for more detailed information about each of these views.

Global Filters

At the top of the screen, below the view tabs, are a set of global filters that let you control which tasks are displayed in the item list.

The global filters are:

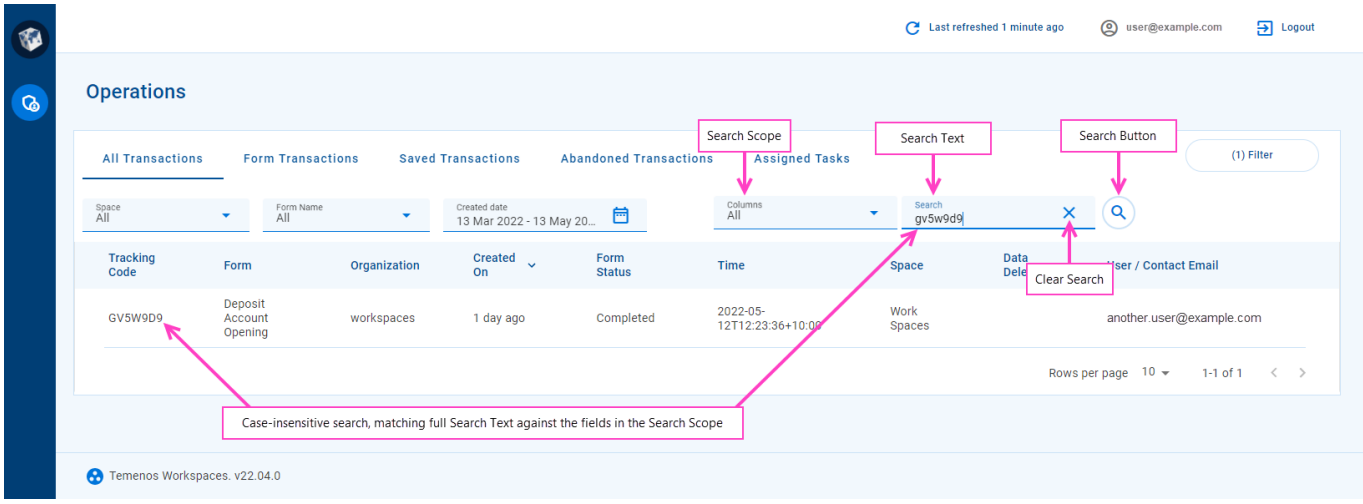
- **Space**: Shows the name of the active Space filter, and lets you select another space.
- **Form Name**: Shows the name of the active Form Name filter, and lets you select another form.
- **Created Date**: Lets you hone in on transactions created in a specific period, and displays the date or date range that is currently selected.

The values available in the dropdown lists for the Space and Form Name filters are specific to each Manager environment. Each of these dropdown lists also includes an All item. Select a filter value to restrict the items displayed in the item list to just those that match the selected value. If All is selected, the item list is populated with items for all values for that global filter.

Finding transactions

Search

If you know which transaction you're looking for and you can't see it in the item list, you can try to search for it. Searching allows you to refine the list of items displayed to show just those relevant to what you are looking for.



To search for a transaction, select a search scope from the Columns selector, enter your search text in the Search field, and press `Enter` or click searchSearch. The search text is compared against the searchable fields in the search scope, and only matching items are displayed in the item list. If the search scope is All, the search text is compared against all searchable fields.

Here are some things to know about how searching works:

- Searching is case-insensitive; that is, it treats uppercase and lowercase characters as the same.
- Partial match is not currently supported, so searching only finds items that match the search text exactly (ignoring letter case).
- When a search is active, the items displayed in the item list satisfy the current view and filter selections as well as the search criteria.

A search remains active until you clear it. Deleting all of the search text does not clear an active search. To clear a search, click the closeClear Search button.

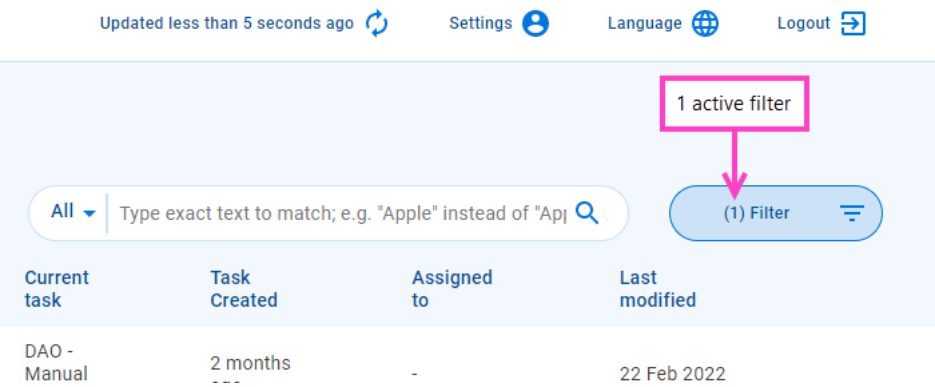
If you select an item in the item list while a search is active, when you subsequently return from the Details screen to the List screen, the search will still be active. This is indicated by the text and closeClear Search in the **Search** field.

View Filters

If you're looking for something specific and there are too many items to scroll through, filters might help you to find what you're after. A filter restricts what appears in the item list so that items more relevant to what you are looking for right now are shown. Filters are applied to the item list using the Filter dialog, however, a View can also include filters.

A filter is a requirement that must be satisfied for an item to appear in the item list. A filter requirement specifies one or more values to compare against a column in the item list, and all matching items satisfy the filter requirement. Multiple filters can be defined at the same time. Only items that satisfy all the active filters will appear in the item list.

If you select an item in the item list while a filter is active, when you subsequently return from the Details screen to the List screen, the filter is still active. This is indicated on the Filter button by a number in parentheses identifying how many filters are active.

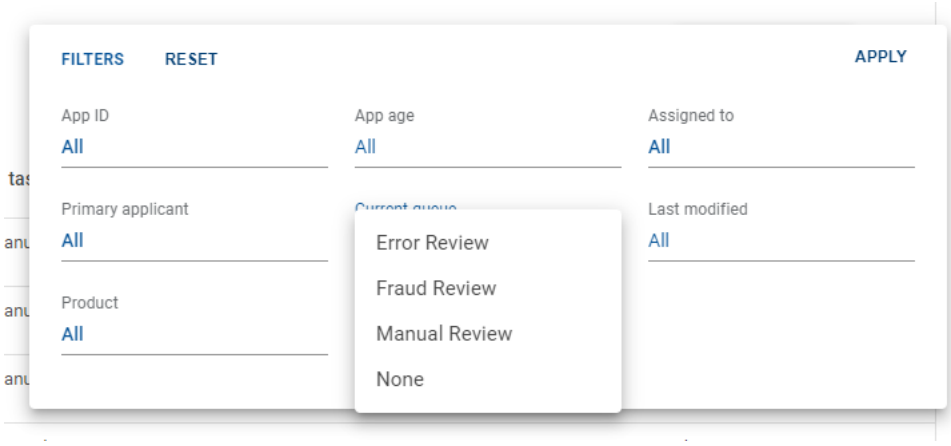


To view the current filters, click **Filter**. The Filter dialog is displayed showing all the filters that are currently applied to the item list. Each filter allows you to select a value or set of values to compare against the column.

Several different filter types are supported:

- **Single-select:** A single value selected from a predefined list of values.
- **Multi-select:** Multiple values selected from a predefined list of values.
- **Date:** A single date or a date range.
- **Text:** Similar to searching but applied to a single column.

When you click a single-select or multi-select filter field, a drop-down list appears populated with a predefined list of values. For a single-select filter, select a value from the dropdown list to filter on. For a multi-select filter, you can select one or more values from the dropdown list.

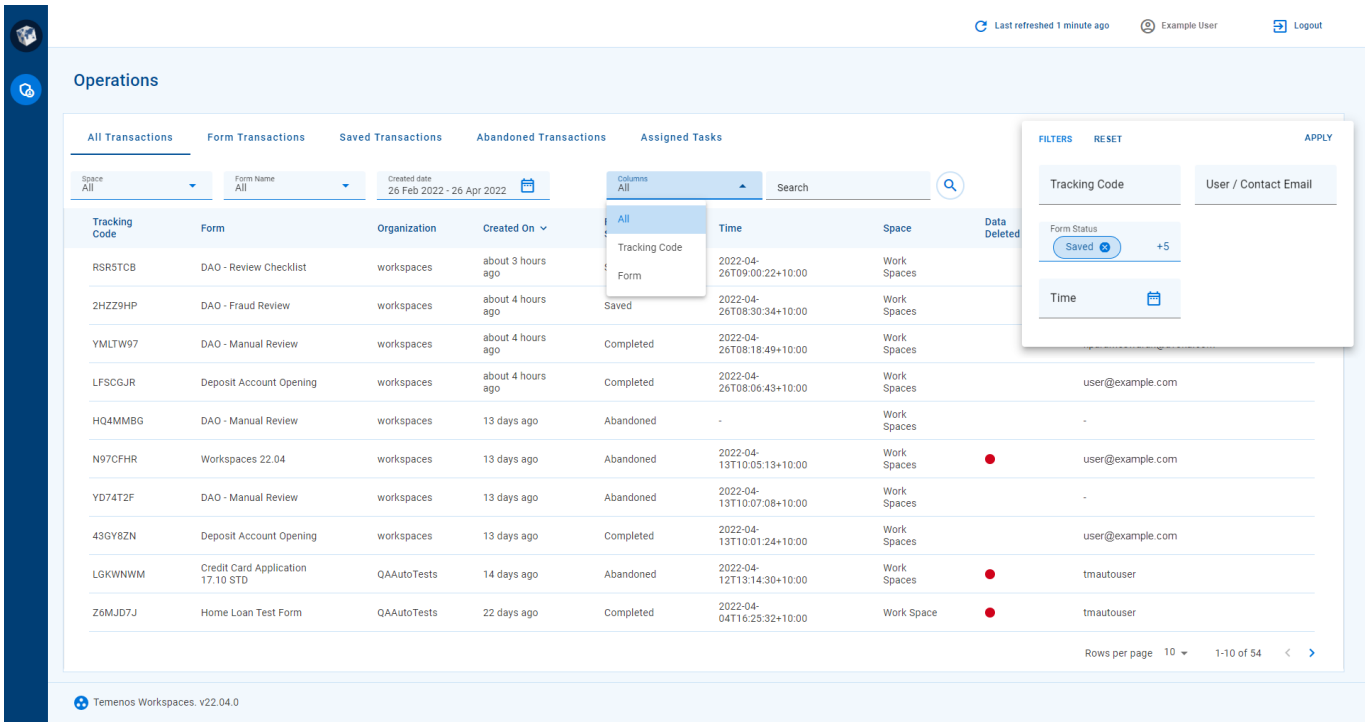


To apply a filter to the item list, click **Filter** to display the Filter dialog then select the filter values you want to apply from the various filter items and click Apply. Click outside the Filter dialog to close it. The number of active filter items is displayed on the Filter button.

All Transactions View

Journey Manager allows you to search for transactions based on a wide range of criteria which can be useful when troubleshooting transactions.You can interact with these transactions on the All Transactions view. This view appears on the Operations portal's List screen. For general information about the List screen, see Manage Transactions.

> NOTE
>
> The All Transactions view is the same as the Operations > Txn Support Search menu item in the Journey Manager console. For more information about this menu item, see Transaction Support Search.



By default, the All Transactions view shows transactions that satisfy the following conditions.

- `formStatus` is one of the following: **Abandoned**, **Assigned**, **Completed**, **Expired**, **Opened**, **Saved**, **Submitted**.
- `timeRequest` is in the inclusive date range [1 month ago, Now].

Item List

The item list in this view includes the following fields.

Tracking Code

A transaction identifier, globally unique in your Journey Manager database. For more information about tracking codes, see Configure Receipt Numbers and Tracking Codes.

Form

The name of a Maestro form. For more information about forms, see Maestro Forms.

Organization

A business entity or part thereof, such as a company, government department, or product group. For more information about organizations, see Introduction to Organizations and Organizations Overview.

Created On

When the transaction was created.

Form Status

The status of a form transaction. For more information about form status, see Transaction Support Search.

Time

The transaction's request time; that is, when it was created.

Space

The transaction's form space. For more information about form spaces, see Form Spaces Overview.

Data Deleted

The transaction's user PII data has been deleted.

User / Contact Email

The user who created the transaction or their contact email address.

View Filters

View filters targeting the following item list fields are available for this view. To learn more about the targeted fields, see the previous section. For each view filter, the component type and valid values are described.

Tracking Code

A text field expecting an alphanumeric tracking code value. Input is not validated.

Form Status

A multi-select chip group containing one or more form status values. Valid values are: **Abandoned***, **Assigned***, **Completed***, **Expired***, **Opened**, **Saved***, **Submitted***. Form Status filter values marked with an asterisk (*) are selected by default in this view.

Time

A date range composed of a **From Date** and a **To Date** in chronological order. Valid values for **From Date** and **To Date** are in the range 1 Jan 1900 to 31 Dec 2100.

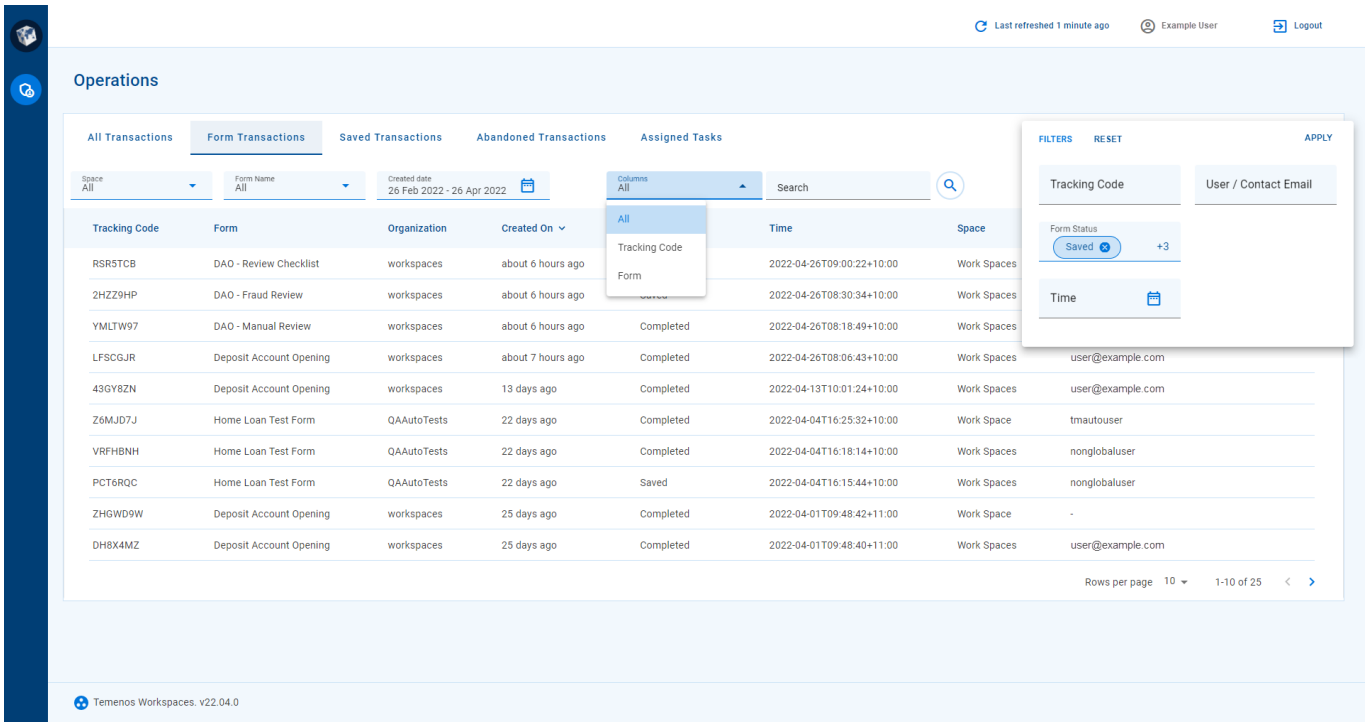
User / Contact Email

A text field expecting a user name or an email address. Input is not validated.

Form Transactions View

Manager keeps track of all transactions that represent form submissions made by application form users.You can interact with these transactions on the Form Transactions view. This view appears on the Operations portal's List screen. For general information about the List screen, see Manage Transactions.

> NOTE
>
> The Form Transactions view is the same as the Operations > Form Transactions menu item in the Journey Manager console. For more information about this menu item, see View Form Transactions.



By default, the Form Transactions view shows transactions that satisfy the following conditions.

- `formStatus` is one of the following: **Completed**, **Expired**, **Saved**, **Submitted**.

Item List

The item list in this view includes the following fields.

Tracking Code

A transaction identifier, globally unique in your Journey Manager database. For more information about tracking codes, see Configure Receipt Numbers and Tracking Codes.

Form

The name of a Maestro form. For more information about forms, see Maestro Forms.

Organization

A business entity or part thereof, such as a company, government department, or product group. For more information about organizations, see Introduction to Organizations and Organizations Overview.

Form Status

The status of a form transaction. For more information about form status, see Transaction Support Search.

Filter

View filters targeting the following item list fields are available for this view. To learn more about the targeted fields, see the previous section. For each view filter, the component type and valid values are described.

Tracking Code

A text field expecting an alphanumeric tracking code value. Input is not validated.

Form Status

A multi-select chip group containing one or more form status values. Valid values are: **Abandoned***, **Assigned***, **Completed***, **Expired***, **Opened**, **Saved***, **Submitted***. Form Status filter values marked with an asterisk (*) are selected by default in this view.

Time

A date range composed of a **From Date** and a **To Date** in chronological order. Valid values for **From Date** and **To Date** are in the range 1 Jan 1900 to 31 Dec 2100.

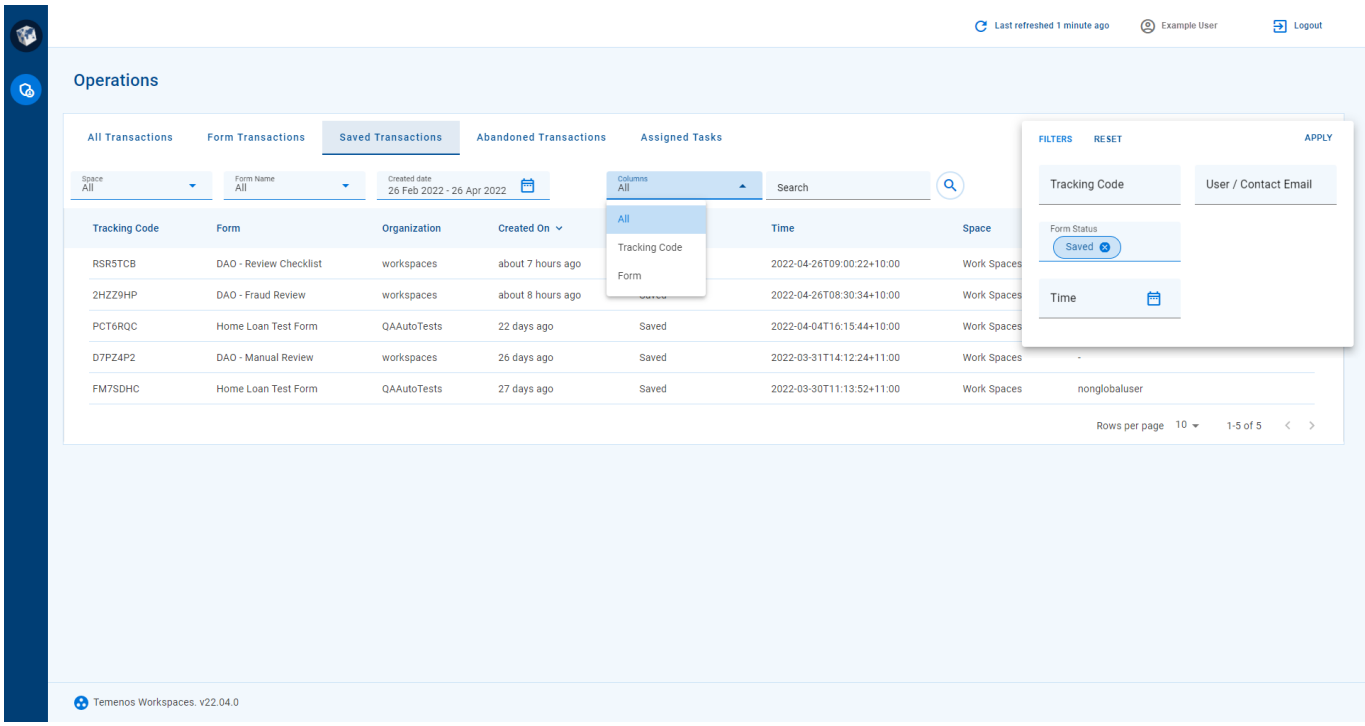
User / Contact Email

A text field expecting a user name or an email address. Input is not validated.

Saved Transactions View

Journey Manager allows an administrator to view and update saved transactions.You can interact with these transactions on the Saved Transactions view. This view appears on the Operations portal's List screen. For general information about the List screen, see Manage Transactions.

> NOTE
>
> The Saved Transactions view is the same as the Operations > Saved Transactions menu item in the Journey Manager console. For more information about this menu item, see View Saved Transactions.



By default, the Saved Transactions view shows transactions that satisfy the following conditions.

- `formStatus` is Saved.

Item List

The item list in this view includes the following fields.

Tracking Code

A transaction identifier, globally unique in your Journey Manager database. For more information about tracking codes, see Configure Receipt Numbers and Tracking Codes.

Form

The name of a Maestro form. For more information about forms, see Maestro Forms.

Organization

A business entity or part thereof, such as a company, government department, or product group. For more information about organizations, see Introduction to Organizations and Organizations Overview.

Created On

When the transaction was created.

Form Status

The status of a form transaction. For more information about form status, see Transaction Support Search.

Time

The transaction's request time; that is, when it was created.

Space

The transaction's form space. For more information about form spaces, see Form Spaces Overview.

User / Contact Email

The user who created the transaction or their contact email address.

Filter

View filters targeting the following item list fields are available for this view. To learn more about the targeted fields, see the previous section. For each view filter, the component type and valid values are described.

Tracking Code

A text field expecting an alphanumeric tracking code value. Input is not validated.

Form Status

A multi-select chip group containing one or more form status values. Valid values are: **Abandoned***, **Assigned***, **Completed***, **Expired***, **Opened**, **Saved***, **Submitted***. Form Status filter values marked with an asterisk (*) are selected by default in this view.

Time

A date range composed of a **From Date** and a **To Date** in chronological order. Valid values for **From Date** and **To Date** are in the range 1 Jan 1900 to 31 Dec 2100.

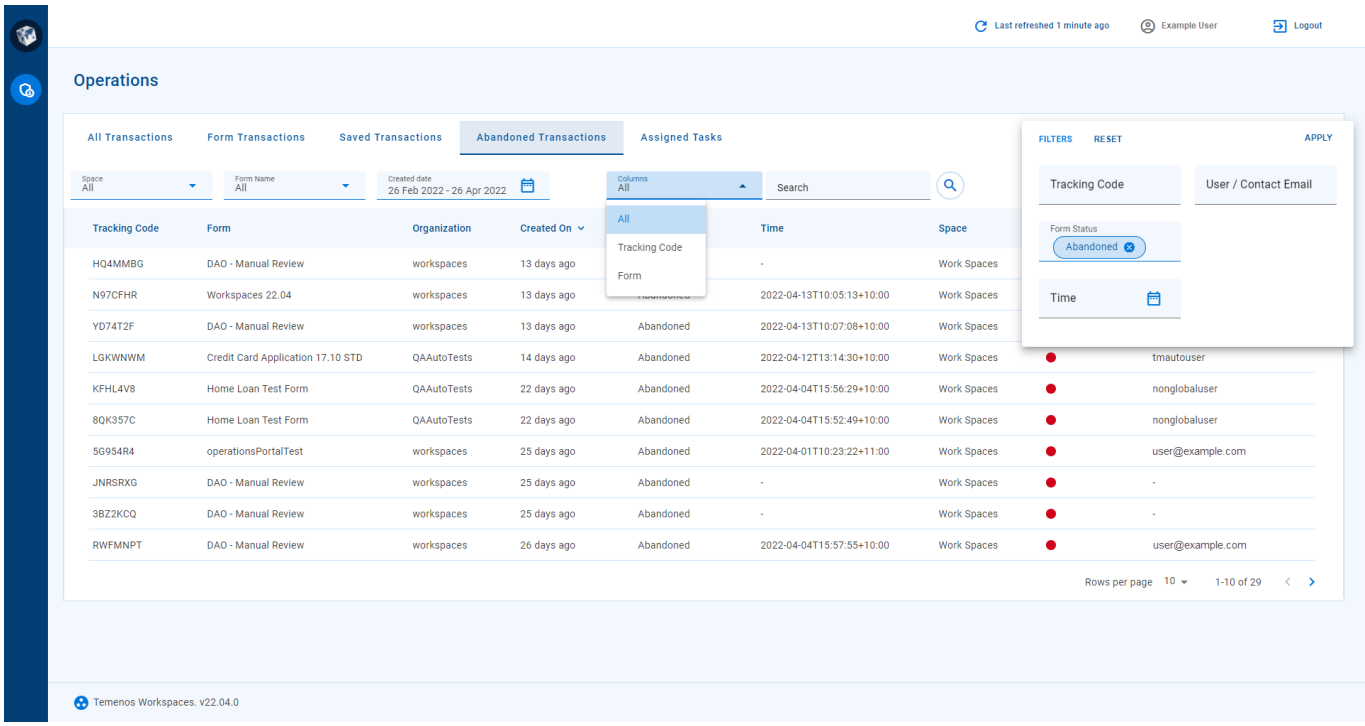
User / Contact Email

A text field expecting a user name or an email address. Input is not validated.

Abandoned Transactions View

Journey Manager allows you to view abandoned transactions so you can inspect them and make some corrective actions, such as changing their status.You can interact with these transactions on the Abandoned Transactions view. This view appears on the Operations portal's List screen. For general information about the List screen, see Manage Transactions.

> NOTE
>
> The Abandoned Transactions view is the same as the Operations > Abandoned Transactions menu item in the Journey Manager console. For more information about this menu item, see View Abandoned Transactions.



By default, the Abandoned Transactions view shows transactions that satisfy the following conditions.

- `formStatus` is Abandoned.

Item List

The item list in this view includes the following fields.

Tracking Code

A transaction identifier, globally unique in your Journey Manager database. For more information about tracking codes, see Configure Receipt Numbers and Tracking Codes.

Form

The name of a Maestro form. For more information about forms, see Maestro Forms.

Organization

A business entity or part thereof, such as a company, government department, or product group. For more information about organizations, see Introduction to Organizations and Organizations Overview.

Created On

When the transaction was created.

Form Status

The status of a form transaction. For more information about form status, see Transaction Support Search.

Time

The transaction's request time; that is, when it was created.

Space

The transaction's form space. For more information about form spaces, see Form Spaces Overview.

Data Deleted

The transaction's user PII data has been deleted.

User / Contact Email

The user who created the transaction or their contact email address.

Filter

View filters targeting the following item list fields are available for this view. To learn more about the targeted fields, see the previous section. For each view filter, the component type and valid values are described.

Tracking Code

A text field expecting an alphanumeric tracking code value. Input is not validated.

Form Status

A multi-select chip group containing one or more form status values. Valid values are: **Abandoned***, **Assigned***, **Completed***, **Expired***, **Opened**, **Saved***, **Submitted***. Form Status filter values marked with an asterisk (*) are selected by default in this view.

Time

A date range composed of a **From Date** and a **To Date** in chronological order. Valid values for **From Date** and **To Date** are in the range 1 Jan 1900 to 31 Dec 2100.

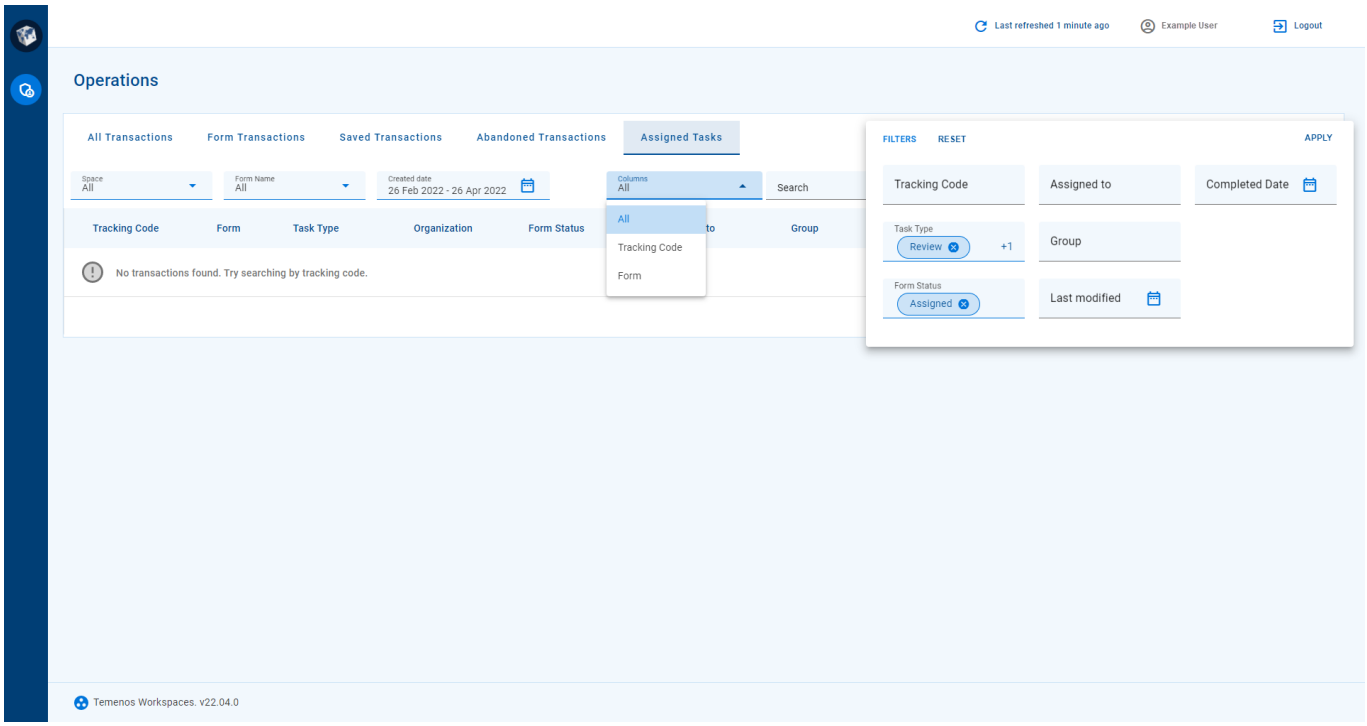
User / Contact Email

A text field expecting a user name or an email address. Input is not validated.

Assigned Tasks View

Journey Manager allows you to create and view tasks, including an option to search for associated transactions which helps you to identify and manage tasks.You can interact with these transactions on the Assigned Tasks view. This view appears on the Operations portal's List screen. For general information about the List screen, see Manage Transactions.

> NOTE
>
> The Assigned Tasks view is the same as the Operations > Assigned Tasks menu item in the Journey Manager console. For more information about this menu item, see View Assigned Tasks.



By default, the Assigned Tasks view shows transactions that satisfy the following conditions.

- `formStatus` is Assigned.

Item List

The item list in this view includes the following fields.

Tracking Code

A transaction identifier, globally unique in your Journey Manager database. For more information about tracking codes, see Configure Receipt Numbers and Tracking Codes.

Form

The name of a Maestro form. For more information about forms, see Maestro Forms.

Task Type

A transaction task type. For more information about task types, see Tasks Overview.

Organization

A business entity or part thereof, such as a company, government department, or product group. For more information about organizations, see Introduction to Organizations and Organizations Overview.

Form Status

The status of a form transaction. For more information about form status, see Transaction Support Search.

Assigned To

The user or user group that a task is assigned to. For more information about task assignment, see Tasks Overview.

Group

A form group, controlling access to the form. For more information about form groups, see Groups Overview.

Created On

When the transaction was created.

Last Modified

When the transaction was last modified.

Completed Date

When the transaction was completed.

Filter

View filters targeting the following item list fields are available for this view. To learn more about the targeted fields, see the previous section. For each view filter, the component type and valid values are described.

Tracking Code

A text field expecting an alphanumeric tracking code value. Input is not validated.

Task Type

A multi-select chip group containing one or more task type values. Valid values are: **Form***, **Review***. Task Type filter values marked with an asterisk (*) are selected by default in this view.

Form Status

A multi-select chip group containing one or more form status values. Valid values are: **Abandoned***, **Assigned***, **Completed***, **Expired***, **Opened**, **Saved***, **Submitted***. Form Status filter values marked with an asterisk (*) are selected by default in this view.

Assigned To

A text field expecting the name of a user or user group. Input is not validated.

Group

A text field expecting the name of a form group. Input is not validated.

Last Modified

A date range composed of a **From Date** and a **To Date** in chronological order. Valid values for **From Date** and **To Date** are in the range 1 Jan 1900 to 31 Dec 2100.

Completed Date

A date range composed of a **From Date** and a **To Date** in chronological order. Valid values for **From Date** and **To Date** are in the range 1 Jan 1900 to 31 Dec 2100.

Transaction Actions Overview

Journey Manager (JM) The transaction engine for the platform. | System Manager / DevOps | 22.04 This feature was introduced in 22.04.

The Operations portal provides a set of standard actions that expose core functionality for working with transactions and jobs.

The transaction-related actions available in the Operations portal are:

- **Abandon:** Abandon the transaction for a submitted application.
- **Purge PII Data:** Purge PII data from selected transactions.
- **Purge Transaction:** Purge all data for selected transactions. For more information about purging transaction data, see Transaction Support Search > Transaction Purge.
- **Receipt:** View (read-only) the receipt for a submitted transaction. For more information, see Receipts Overview and Receipt - View a Receipt.
- **Recover:** Recover the transaction for an abandoned or withdrawn application. For more information about recovering transactions, see **Make Abandoned Transaction Saved** in Change Transaction Status, and Recover an abandoned application.
- **Resume:** Re-open an application in the same form used to start the application. For more information, see Resume: Re-open an application form.
- **View Form XML:** View (read-only) the transaction for an application that has been saved but not yet submitted. For more information, see Form XML Data.

Action Availability

Whether or not an action is available for a transaction is dependent on factors such as a transaction's form status and whether it is a review task. While all available actions can be performed from the Details screen, some actions can also be performed from the Listscreen as Shortcut Actions.

Action	Availability
Abandon	The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following condition. • Form Status is not one of the following: **Abandoned**, **Completed**, **Expired**, or **Submitted**. When this action is available, you can perform it as a shortcut action on the List

Action	Availability
	screen.
Purge PII Data	The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following conditions. • Form Status is not one of the following: **Abandoned**, **Completed**, **Expired**, or **Submitted**. • The transaction is not a review task.
Purge Transaction	The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following conditions. • Form Status is not one of the following: **Abandoned**, **Completed**, **Expired**, or **Submitted**. • The transaction is not a review task.
Receipt	The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following condition. • The transaction has a receipt URL. When this action is available, you can perform it as a shortcut action on the List screen.
Recover	The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following conditions. • Form Status is **Abandoned**. • The transaction is not a review task. When this action is available, you can perform it as a shortcut action on the List screen.
Resume	The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy one or more of the following conditions.

Action	Availability
View Form XML	• Form Status is not one of the following: **Abandoned**, **Assigned**, **Completed**, **Expired**, or **Submitted**. • The transaction is assigned to a user. When this action is available, you can perform it as a shortcut action on the List screen. The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following condition. • The transaction has submission XML.

Abandon a Transaction

Journey Manager allows an administrator to abandon a transaction manually. You can use this functionality to prevent a user from accessing certain transactions. For more information about abandoned transactions, see Transactions and Tasks Abandonment.

The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following condition.

- Form Status is not one of the following: **Abandoned**, **Completed**, **Expired**, or **Submitted**.

When this action is available, you can perform it from the Details screen or as a shortcut action on the List screen.

To abandon a transaction, follow these instructions.

1. Login to the Operations portal and select the Operations space.
2. On the List screen, select an appropriate view (not the Abandoned Transactions view), and locate the transaction to be abandoned. You can use search and filters to refine the transactions in the item list, and paging tools to navigate through the item list if there are multiple pages.
3. To abandon the transaction from the List screen:
   a. Point your cursor at the transaction item to reveal the available actions at the right-hand side.
   b. Click (Abandon) .
4. To abandon the transaction from the Details screen:
   a. Select the transaction item to display its transaction details.
   b. Click Actions and select Abandon from the dropdown list.
5. The transaction's form status is changed to **Abandoned**.

NOTE

This action is the same as the Abandon option in the JM console. For more information about this option, see Abandon a Transaction.

Purge PII Data

Form transactions can include sensitive business or personal data, such as a form user's Personally Identifiable Information[1] (PII), depending on how transactions have been designed. Purging this data after delivery provides a stronger security story. To maintain optimal system performance over time, Journey Manager provides data storage, retention and purging capabilities that you can adjust to your company's data requirements. For more information about PII data and the automated purging process, see Data Retention Management Overview. In addition to the automated purging of data, the Operations portal allows you to purge PII data for a specific transaction.

The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following conditions.

- Form Status is not one of the following: **Abandoned**, **Completed**, **Expired**, or **Submitted**.
- The transaction is not a review task.

When this action is available, you can perform it from the Details screen.

To purge a transaction's PII data, follow these instructions.

1. Login to the Operations portal and select the Operations space.
2. On the List screen, select an appropriate view, and locate the transaction to be purged of PII data. You can use search and filters to refine the transactions in the item list, and paging tools to navigate through the item list if there are multiple pages.
3. Select the transaction item to display it on the Details screen.
4. Click Actions and select Purge PII Data from the dropdown list.
5. The transaction's PII data is purged.

NOTE

This action is the same as the Purge PII Data option in the JM console. For more information about this option, see Transaction Status (Step 22).

[1]Personally identifiable information (PII) is any data that could potentially identify a specific individual. Any information that can be used to distinguish one person from another and can be used for de-anonymizing anonymous data can be considered PII.

Purge a Transaction

Journey Manager (JM) The transaction engine for the platform. | System Manager / DevOps | 22.04 This feature was introduced in 22.04.

Journey Manager has a high volume of user form transactions and associated resources that can accumulate a significant amount of data, even during a short period of operation. This can have a detrimental effect on overall system performance which, in turn, may negatively impact user experience and third party integrations. To maintain optimal system performance over time, Manager provides data storage, retention and purging capabilities that can be adjusted to meet your company's data requirements. While data retention management is an automated process, you can also purge transactions manually when required.

The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following conditions.

- Form Status is not one of the following: **Abandoned**, **Completed**, **Expired**, or **Submitted**.
- The transaction is not a review task.

When this action is available, you can perform it from the Details screen.

To purge a transaction, follow these instructions.

1. Login to the Operations portal and select the Operations space.
2. On the List screen, select an appropriate view, and locate the transaction to be purged. You can use search and filters to refine the transactions in the item list, and paging tools to navigate through the item list if there are multiple pages.
3. Select the transaction item to display it on the Details screen.
4. Click Actions and select Purge Transaction from the dropdown list.
5. The transaction is purged.

NOTE

This action is the same as the Purge Transaction option in the JM console. For more information about this option, see Transaction Status (Step 23).

Recover a Transaction

Transactions can become abandoned either actively due to a user action, such as canceling a transaction, or passively based on the form's abandonment settings. Abandoned transactions can be recovered so that they can continue in the transaction life cycle.

> NOTE
>
> Any Personally Identifiable Information (PII) in a recovered application is not available if the data retention period has expired.

The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following conditions.

- Form Status is **Abandoned**.
- The transaction is not a review task.

When this action is available, you can perform it from the Details screen or as a shortcut action on the List screen.

To recover a transaction, follow these instructions.

1. Login to the Operations portal and select the Operations space.
2. On the List screen, select an appropriate view, and locate the transaction to be recovered. You can use search and filters to refine the transactions in the item list, and paging tools to navigate through the item list if there are multiple pages.
3. To recover the transaction from the List screen:
   a. Point your cursor at the transaction item to reveal the available actions at the right-hand side.
   b. Click (Recover) .
4. To recover the transaction from the Details screen:
   a. Select the transaction item to display its transaction details.
   b. Click Actions and select Recover from the dropdown list.
5. The transaction is recovered.

> NOTE
>
> This action is the same as the Make Abandoned Transaction Saved option in the JM console. For more information about this option, see Change Transaction Status.

Resume a Transaction

Transactions can become abandoned either actively due to a user action, such as canceling a transaction, or passively based on the form's abandonment settings. Abandoned transactions can be recovered so that they can continue in the transaction life cycle.

The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy one or more of the following conditions.

- Form Status is not one of the following: **Abandoned**, **Assigned**, **Completed**, **Expired**, or **Submitted**.
- The transaction is assigned to a user.

When this action is available, you can perform it from the Details screen or as a shortcut action on the List screen.

To resume a transaction, follow these instructions.

1. Login to the Operations portal and select the Operations space.
2. On the List screen, select an appropriate view, and locate the transaction to be resumed. You can use search and filters to refine the transactions in the item list, and paging tools to navigate through the item list if there are multiple pages.
3. To resume the transaction from the List screen:
   a. Point your cursor at the transaction item to reveal the available actions at the right-hand side.
   b. Click (Resume) .
4. To resume the transaction from the Details screen:
   a. Select the transaction item to display its transaction details.
   b. Click Actions and select Resume from the dropdown list.
5. The transaction's form is displayed in a modal dialog.

NOTE

This action is the same as the Open Saved Form option in the JM console. For more information about this option, see View Form Transactions or View Saved Transactions.

View Form XML

Journey Manager allows you to view the XML[1] data of a submitted transaction. An administrator can access and edit the submitted data to re-attempt delivery, thereby addressing issues like these:

- A user completed a form, but form data has not yet been purged, and an error is preventing receipt generation or delivery.
- A user entered invalid or unexpected characters that have caused back-end processing issues. Generally, forms are designed to capture invalid data before submission. However, if invalid form input is accepted, this can cause receipt generation or delivery to be blocked.
- Clean the data before it reaches the user or back-end processing.

The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following condition.

- The transaction has submission XML.

When this action is available, you can perform it from the Details screen.

To view form XML for a form transaction, follow these instructions.

1. Login to the Operations portal and select the Operations space.
2. On the List screen, select an appropriate view, and locate the transaction to view form XML for. You can use search and filters to refine the transactions in the item list, and paging tools to navigate through the item list if there are multiple pages.
3. Select the transaction item to display it on the Details screen.
4. Click Actions and select View Form XML from the dropdown list.
5. The transaction's form XML is displayed in a modal dialog.

NOTE

This action is the same as the View Form XML option in the JM console. For more information about this option, see View Form XML.

[1]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

View a Transaction Receipt

Journey Manager provides a form user with a receipt that is read-only statement acknowledging the data submitted by the user. Receipts are an essential part of the form submission process. Receipts are PDF forms that are generated, based on form content, for all completed transactions. Each receipt contains a unique receipt number that you can use to identify the corresponding transaction. For more information about transaction receipts, see Receipts Overview.

The availability of Operations portal actions is dependent on transaction data, not the selected view. This action is available for transactions that satisfy the following condition.

- The transaction has a receipt URL.

When this action is available, you can perform it from the Details screen or as a shortcut action on the List screen.

To view a transaction receipt, follow these instructions.

1. Login to the Operations portal and select the Operations space.
2. On the List screen, select an appropriate view, and locate the transaction to view the receipt for. You can use search and filters to refine the transactions in the item list, and paging tools to navigate through the item list if there are multiple pages.
3. To view the transaction receipt from the List screen:
   a. Point your cursor at the transaction item to reveal the available actions at the right-hand side.
   b. Click (Receipt) .
4. To view the transaction receipt from the Details screen:
   a. Select the transaction item to display its transaction details.
   b. Click Actions and select Receipt from the dropdown list.
5. The transaction's receipt is displayed in a modal dialog.

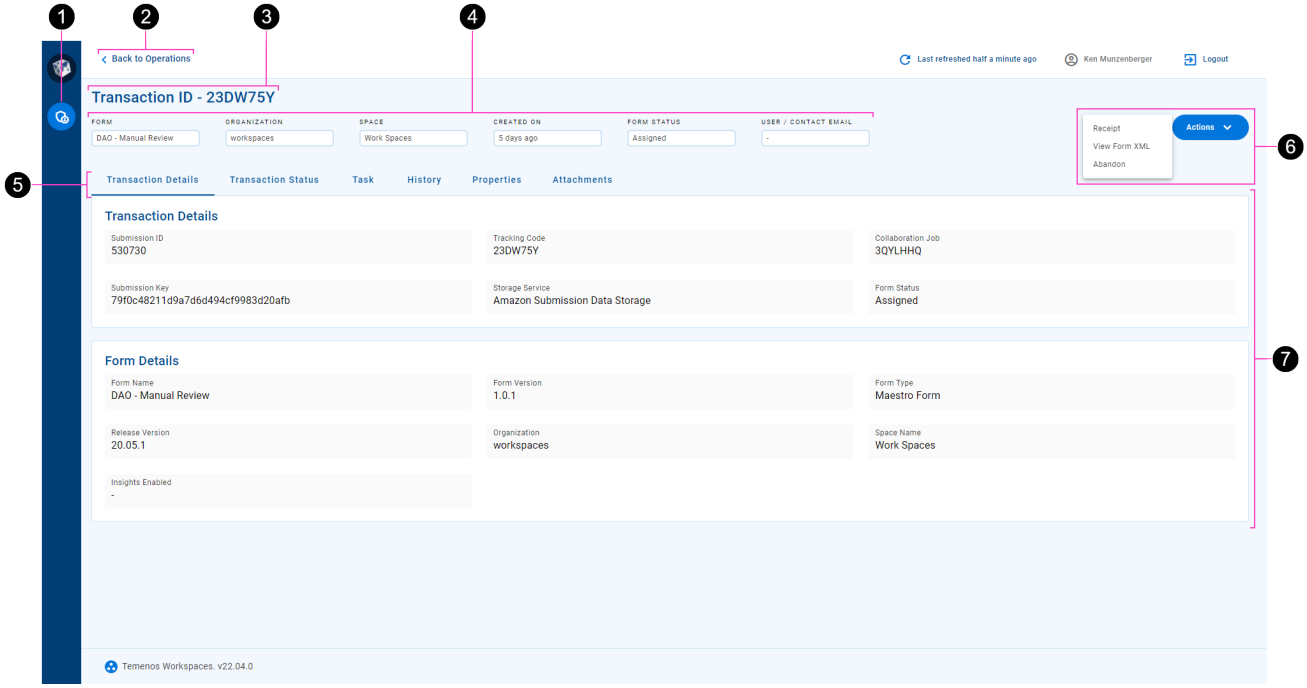
NOTE

This action is the same as the View Receipt option in the JM console. For more information about this option, see one of the transaction search topics, such as Transaction Support Search.

Transaction Details Screen

The Transaction Details screen, as the name suggests, displays detailed information pertaining to a specific transaction for a saved or submitted application, and provides options for you to work with these transactions. Select an item from the item list on the List screen to display the Details screen. The information displayed on this screen is similar to what is displayed in the Journey Manager console.

The Details screen is configured with a set of features common to all Journey UI applications. For an introduction to these common features, see UI Tour. Here, we'll focus on the additional transaction management features available on this screen.



1. The **Operations** space navbar item.
2. A breadcrumb leading back to the List screen.
3. This screen's name, identifying the selected transaction.
4. Key transaction information (key info).
5. A tabset presenting different kinds of transaction information on each tab.
6. An action button with optional dropdown list.
7. Information cards grouping similar information together.

The primary purpose of the Details screen is to display transaction information, which can be seen in the screen name, key info, and the information tabset. The screen name is a prominent display of the transaction's ID which is the primary transaction identifier and is unique in each Journey Manager environment. The key info section displays between four and six items

depending on the width of your browser window. At full screen width, six items are visible but as the width of your browser becomes smaller, items are removed progressively until only four items are displayed. Each key info item has a label and a value. The key info section displays the following information:

- Form
- Organization
- Space
- Created On
- Form Status
- User / Contact Email

The majority of transaction information is contained in information cards under various tabs that group related information. These tabs correspond to similarly named tabs in the Journey Manager console. The following tabs are available on the Details screen.

- **Transaction Details**: A read-only display of key transaction information, including details of an associated form and the user who completed it.
- **Transaction Status**: Information relating to a transaction's progress such as workflow and status details. On this tab, you can update the following transaction fields: **Attachments Status**, **Form Status**, **Receipt Status**, and **Delivery Status**.
- **Task**: Task detail and status information. On this tab, you can update the following task fields: **Subject**, **Message**, and **Assigned User**.
- **Form Sessions**: The list of form sessions related to the transaction, and detailed information for each form session.
- **History**: Transaction history let's you see a user's interaction with a form.
- **Form Data Extracts**: Information extracted from a submitted form's XML data, displayed as name/value pairs.
- **Properties**: A read-only display of transaction properties.
- **Notes**: A read-only display of the comments made against a selected transaction.
- **Attachments**: A list of files attached to the transaction, with attachment meta-data and an option to view the attachments.
- **Groovy Log**: Groovy service output messages for a selected transaction, useful for troubleshooting.
- **Events**: The list of events logged against a selected transaction.
- **Errors**: A read-only view of any errors encountered during a transaction's life cycle.

There's a separate topic for each tab, each of which identifies the information cards included on the tab. To learn more about each tab, click the tab names in the list above.

While most tabs are read-only displays of transaction-related information, a few also provide access to some additional transaction maintenance features. At this stage, these features are limited to changing the value of various transaction data fields.

Transaction field	Where to change this field
Attachments Status	**Transaction Status** tab, **User Transaction Flow Steps** card
Form Status	**Transaction Status** tab, **User Transaction Flow Steps** card
Receipt Status	**Transaction Status** tab, **PDF Receipt Generation** card
Delivery Status	**Transaction Status** tab, **Transaction Data Delivery** card
Subject	**Task** tab, **Task Details** card
Message	**Task** tab, **Task Details** card
Assigned User	**Task** tab, **Task Status** card

Transaction-related actions can be performed using the action button in the top-right corner of the screen. One or more actions are available at any time depending on several factors including the transaction's current state. If only one action is available, the action button's label corresponds to the action name, and clicking the button performs the action. If multiple actions are available, the action button's label is **Action**, and clicking the button displays a dropdown list containing all the available actions; select an action item to perform that action. To learn more about Operations portal actions and when they're available, see Transaction Actions Overview.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- An information card is hidden if all of the fields it contains are empty.
- A Details tab is hidden if all of the information cards it contains are hidden.

When you've finished with this transaction, you can return to the List screen using the breadcrumb at the top of the screen.

Transaction Details Tab

The Transaction Details tab shows a read-only display of key transaction, form, and user details. Similar information is available in the Journey Manager console; for details, see Transaction Details.

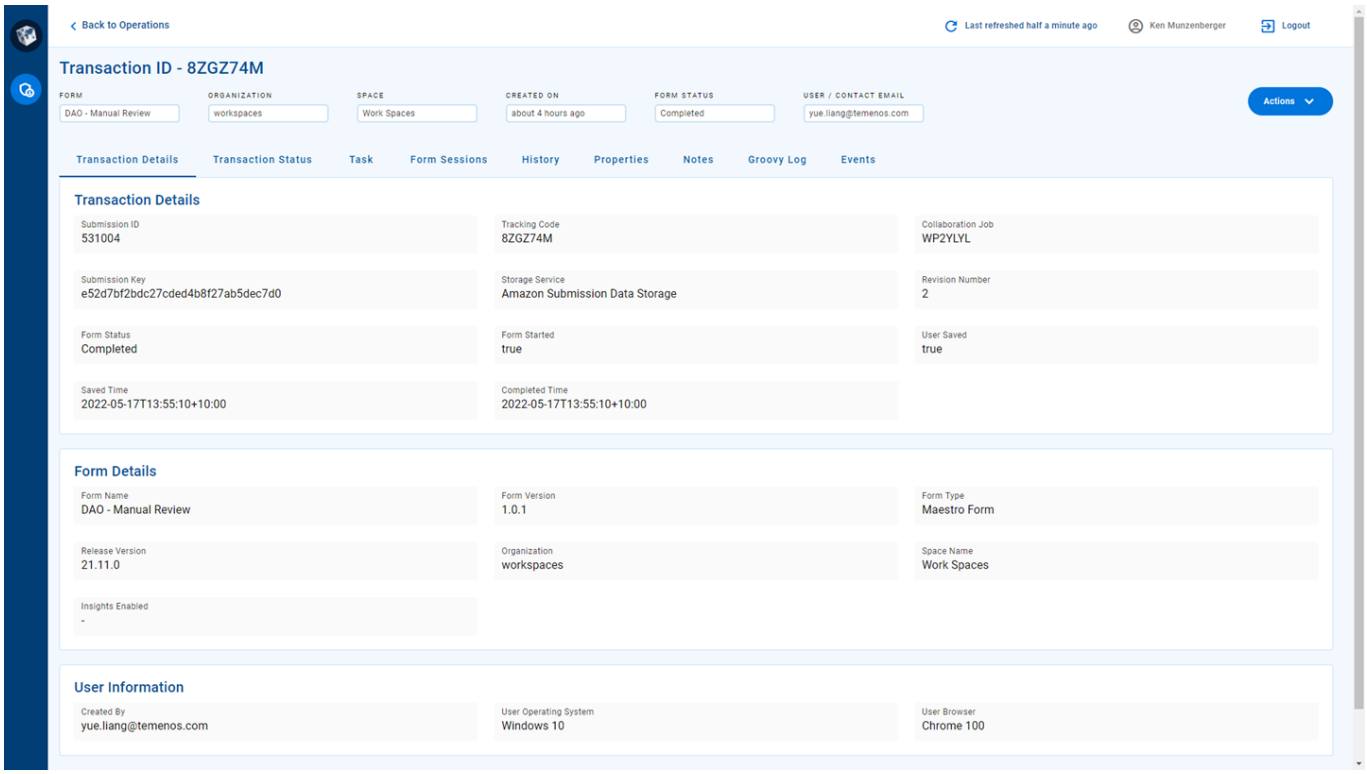
This tab contains information separated into the following sections.

- Transaction Details: Key transaction details, including Submission ID, Tracking Code, and Form Started.
- Form Details: Form-related transaction details such as the form name, and the associated organization and space.
- User Information: Details of the user who completed the transaction.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Transaction Details** tab is hidden if all of its cards are hidden.



Transaction Details

The **Transaction Details** section displays information for the following fields. in this section.

- Submission ID
- Tracking Code
- Collaboration Job
- Submission Key
- Storage Service
- Revision Number
- Form Status
- Form Started
- User Saved
- Saved Time
- Completed Time

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Transaction Details** tab, **Transaction Details** section.

Form Details

The **Form Details** section displays information for the following fields. in this section.

- Form Name
- Form Version
- Form Type
- Release Version
- Organization
- Space Name
- Insights Enabled

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Transaction Details** tab, **Form Details** section.

User Information

The **User Information** section displays information for the following fields. in this section.

- Created By
- Contact Phone
- Contact Email Address

- User Operating System
- User Browser

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Transaction Details** tab, **User Information** section.

Transaction Status Tab

The Transaction Status tab shows information relating to a transaction's progress. Similar information is available in the Journey Manager console; for details, see Transaction Status.

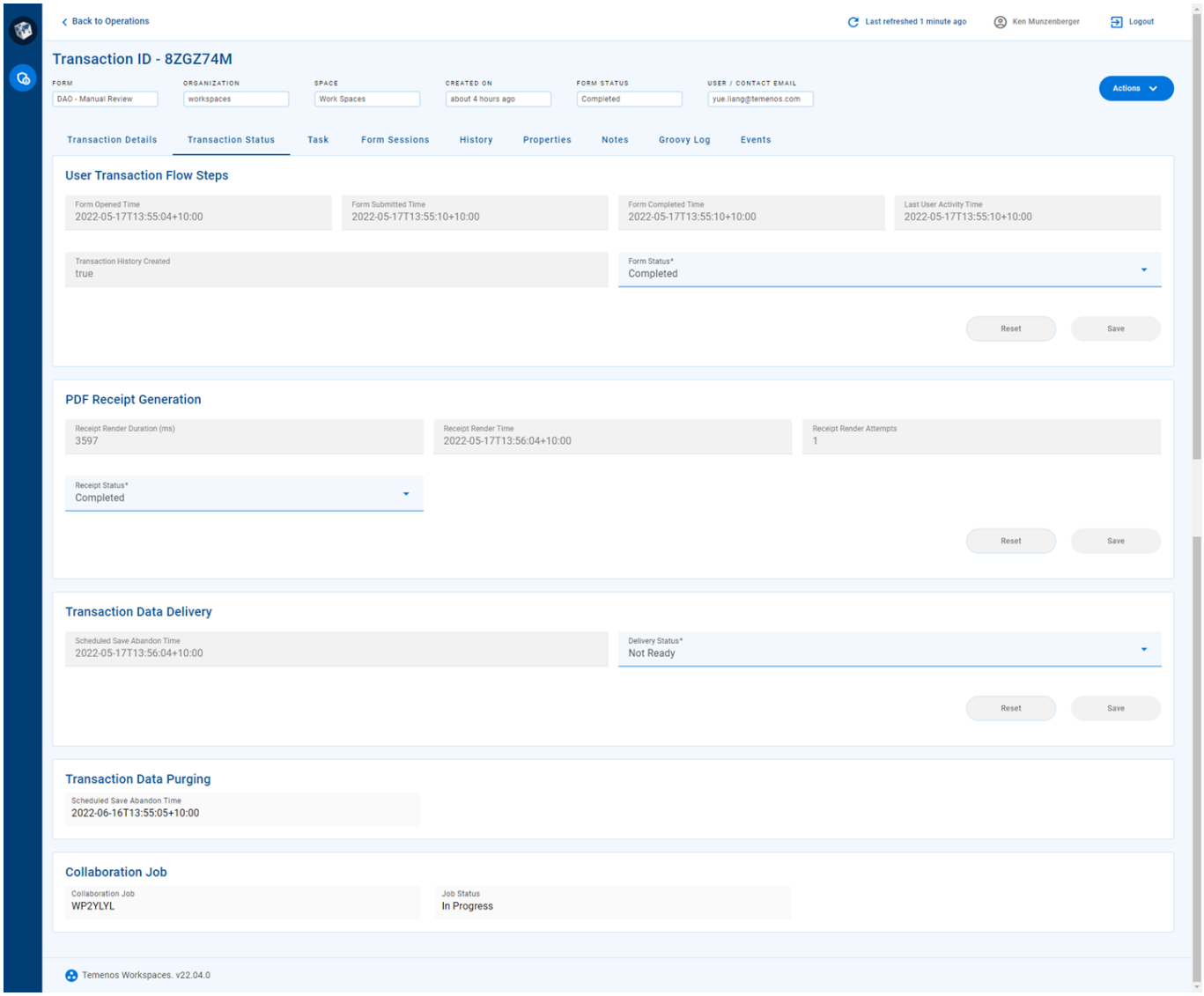
This tab contains information separated into the following sections.

- User Transaction Flow Steps: Transaction workflow information such as when the transaction was created, submitted and completed.
- User Processing Status Message: A processing status message for the user and when it was updated.
- PDF Receipt Generation: Information about the generation of PDF receipts.
- Transaction Abandonment: Transaction abandonment information.
- Transaction Data Delivery: The transaction's data delivery status.
- Transaction Data Purging: Information about scheduled purging of transaction data.
- Collaboration Job: Collaboration job information.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Transaction Status** tab is hidden if all of its cards are hidden.



User Transaction Flow Steps

The **User Transaction Flow Steps** section displays information for the following fields. Fields marked with an asterisk (*) can be updated in this section.

- Form Opened Time
- Form Submitted Time
- Form Completed Time
- Last User Activity Time
- Insights Transaction
- Transaction History Created
- Attachments Status*: To update this field, select a value from the dropdown list and click **Save**. Valid values are: **Completed**, **Optional**, **Required**.

- Form Status*: To update this field, select a value from the dropdown list and click **Save**. Valid values are: **Abandoned**, **Assigned**, **Completed**, **Expired**, **Opened**, **Saved**, **Submitted**.
- Submit Key

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Transaction Status** tab, **User Transaction Flow Steps** section.

User Processing Status Message

The **User Processing Status Message** section displays information for the following fields. in this section.

- Processing Status
- Processing Status Time

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Transaction Status** tab, **User Processing Status Message** section.

PDF Receipt Generation

The **PDF Receipt Generation** section displays information for the following fields. Fields marked with an asterisk (*) can be updated in this section.

- Submit Key
- Receipt Render Duration (ms)
- Receipt Render Time
- Receipt Render Attempts
- Next Receipt Render Time
- Receipt Status*: To update this field, select a value from the dropdown list and click **Save**. Valid values are: **Completed**, **Error**, **In Progress**, **Ready**.

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Transaction Status** tab, **PDF Receipt Generation** section.

Transaction Abandonment

The **Transaction Abandonment** section displays information for the following fields. in this section.

- Abandonment Type
- Abandonment Timestamp

- Previous Form Status

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Transaction Status** tab, **Transaction Abandonment** section.

Transaction Data Delivery

The **Transaction Data Delivery** section displays information for the following fields. Fields marked with an asterisk (*) can be updated in this section.

- Submit Key
- Scheduled Save Abandon Time
- Delivery Status*: To update this field, select a value from the dropdown list and click **Save**. Valid values are: **Completed**, **Error**, **In Progress**, **Not Ready**, **Not Required**, **Pending**, **Ready**, **Sent Email**, **Undeliverable**.

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Transaction Status** tab, **Transaction Data Delivery** section.

Transaction Data Purging

The **Transaction Data Purging** section displays information for the following fields. in this section.

- Transaction Data Deleted
- Scheduled Save Abandon Time
- Scheduled Transaction Data Purge
- Actual Transaction Data Purge
- Scheduled Transaction Record Purge

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Transaction Status** tab, **Transaction Data Purging** section.

Collaboration Job

The **Collaboration Job** section displays information for the following fields. in this section.

- Collaboration Job
- Job Status

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Transaction Status** tab, **Collaboration Job** section.

Task Tab

The Task tab shows information relating to a task transaction. Similar information is available in the Journey Manager console; for details, see Transaction Status.

This tab contains information separated into the following sections.

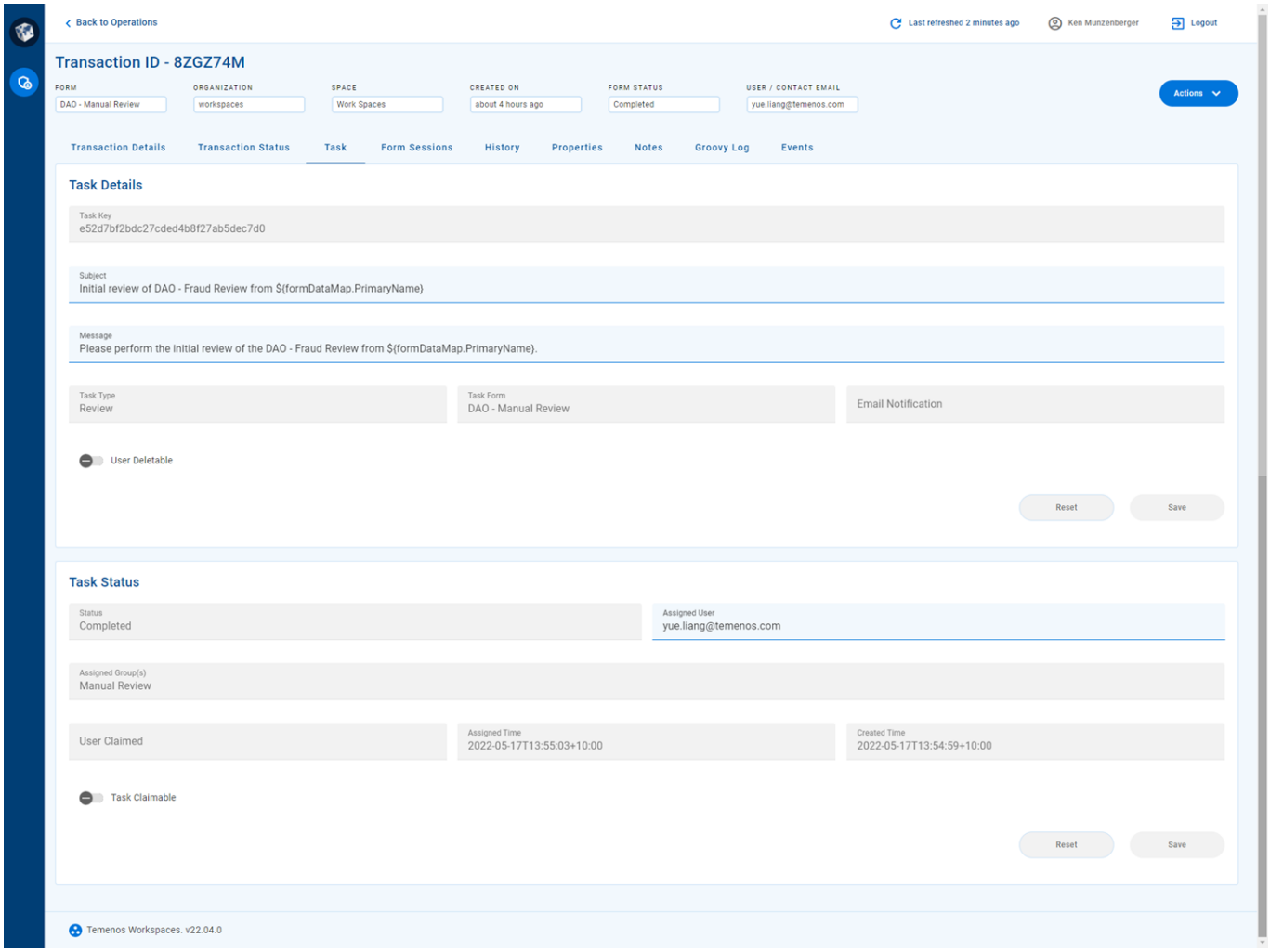
- Task Details: View and manage task details.
- Task Status: View and manage the task status.

This tab is displayed if the value of the **Task Type** field (on the **Task Details** card) is "Form" or "Review".

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Task** tab is hidden if all of its cards are hidden.



Task Details

The **Task Details** section displays information for the following fields. Fields marked with an asterisk (*) can be updated in this section.

- Submit Key
- Task Key
- Subject*: To update this field, enter a value and click **Save**.
- Message*: To update this field, enter a value and click **Save**.
- Task Type
- Task Form
- Email Notification
- User Deletable

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Task** tab, **Task Details** section.

Task Status

The **Task Status** section displays information for the following fields. Fields marked with an asterisk (*) can be updated in this section.

- Submit Key
- Status
- Assigned User*: To update this field, enter a value and click **Save**.
- Assigned Group(s)
- User Claimed
- Assigned Time
- Created Time
- Task Claimable

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Task** tab, **Task Status** section.

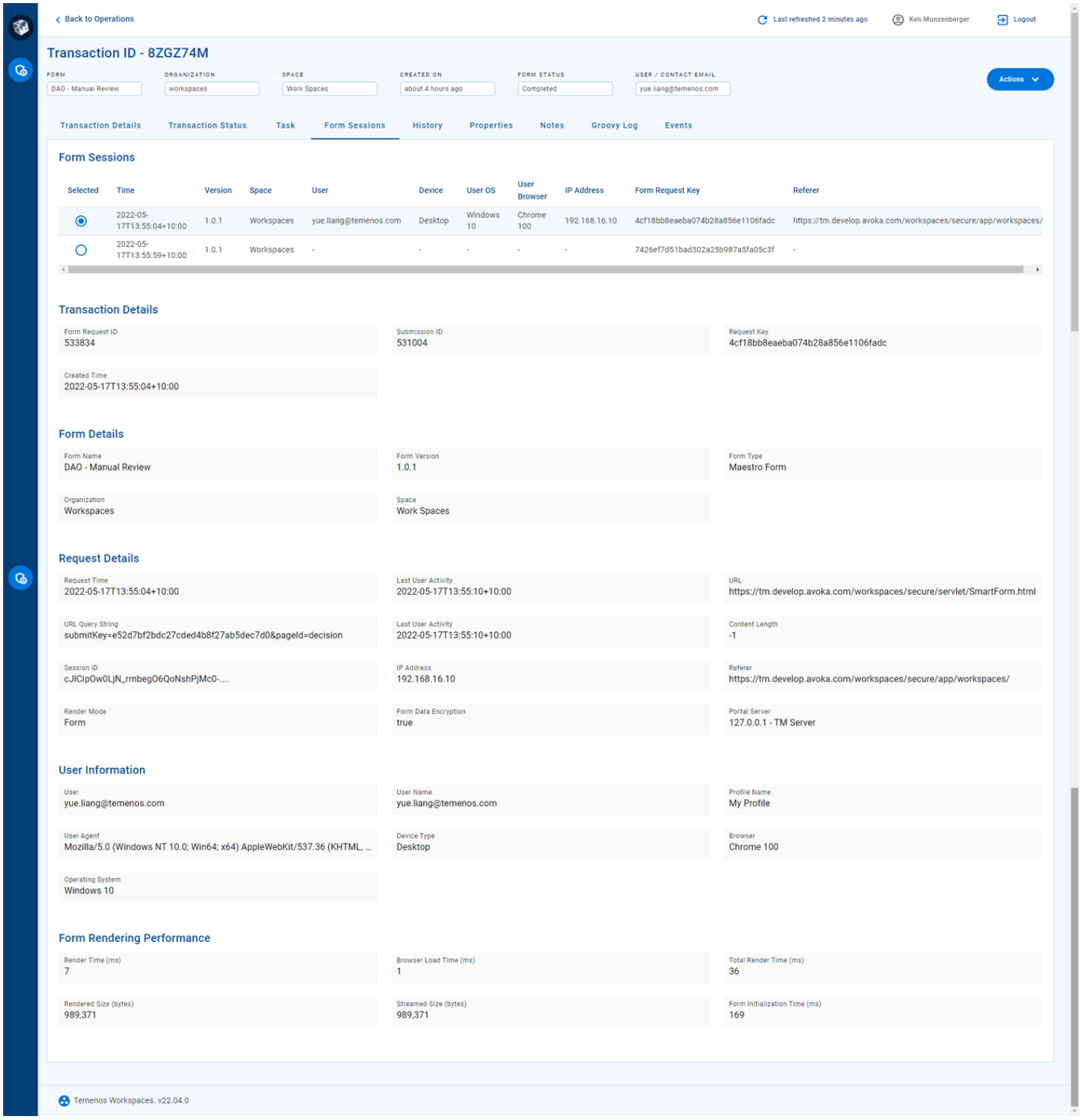
Form Sessions Tab

Journey Manager creates a form session when a user opens a form and ends it as soon as the user leaves the form.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Form Sessions** tab is hidden if all of its cards are hidden.



The **Form Sessions** tab contains a list of form session items, displaying the following information.

- Device
- Form Request Key
- IP Address
- Referer
- Space

- Time
- User
- User Browser
- User OS
- Version

Selecting an item from the list displays information for the selected item in the following sections.

- Transaction Details
  - Form Request ID
  - Submission ID
  - Request Key
  - Created Time
- Form Details
  - Form Name
  - Form Version
  - Form Type
  - Organization
  - Space Name
- Request Details
  - Request Time
  - Last User Activity
  - URL
  - URL Query String
  - Content Length
  - Content type
  - Session ID
  - IP Address
  - Referer
  - Render Mode
  - Receipt Mode
  - Form Data Encryption
  - Portal Server
  - Form Server
- User Information
  - User
  - User Name
  - Profile Name
  - User Agent

    - Device Type
    - Browser
    - Operation System
- Form Rendering Performance
    - Render Time (ms)
    - Browser Load Time (ms)
    - Total Render Time (ms)
    - Rendered Size (bytes)
    - Streamed Size (bytes)
    - Form Initialization Time (ms)

Similar information is available in the Journey Manager console on the **Form Request Details** page, accessed via the **Transaction Details** page, **Form Sessions** tab; for details, see Form Sessions.

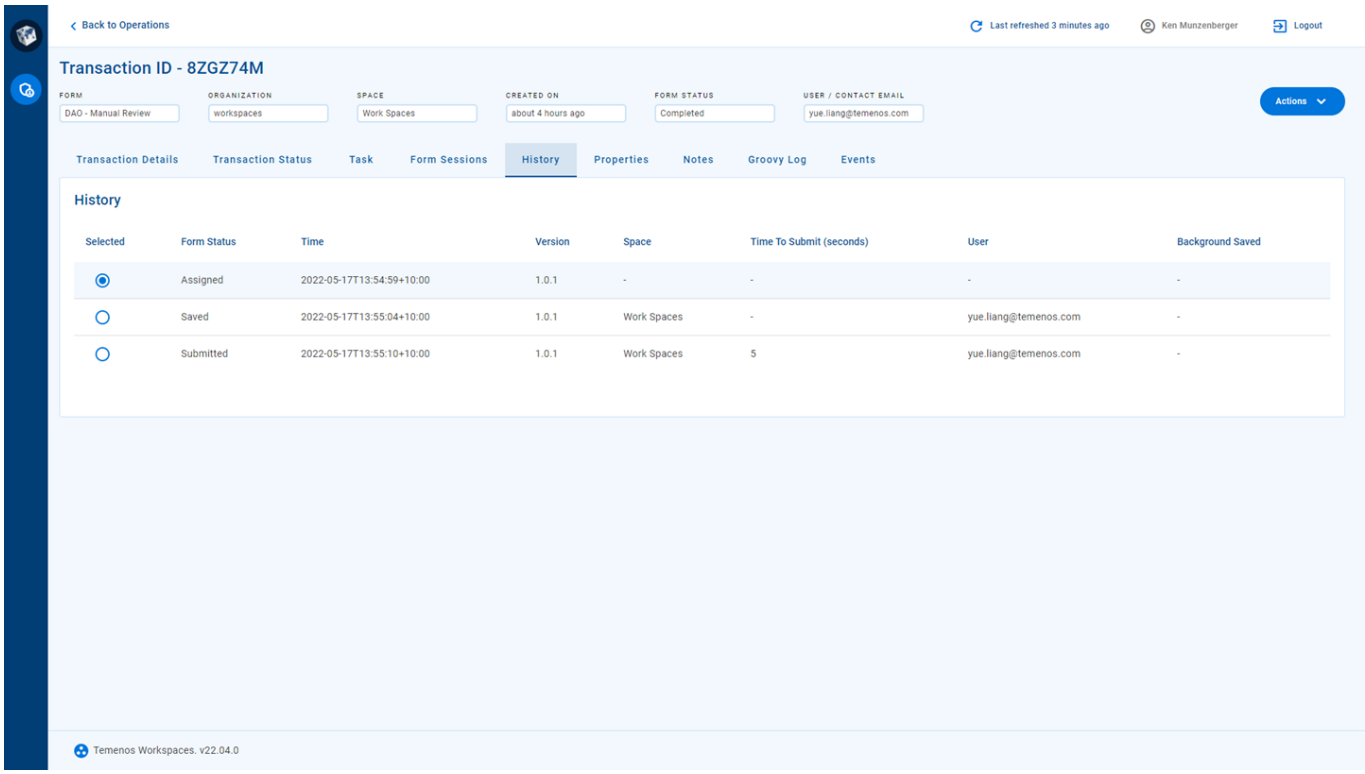
Transaction History Tab

Transaction history let's you see a user's interaction with a form.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **History** tab is hidden if all of its cards are hidden.



The History tab contains a list of form interaction items, displaying the following information for each item.

- Form Status
- Time
- Version
- Space
- Time To Submit (seconds)
- User
- Background Saved

Selecting an item from the list displays the following information for the selected item.

- Device
- IP Address
- Form Request Key
- Form Shared
- User OS / Browser

Similar information is available in the Journey Manager console on the **Transaction Details** page, **History** tab; for details, see Transaction History.

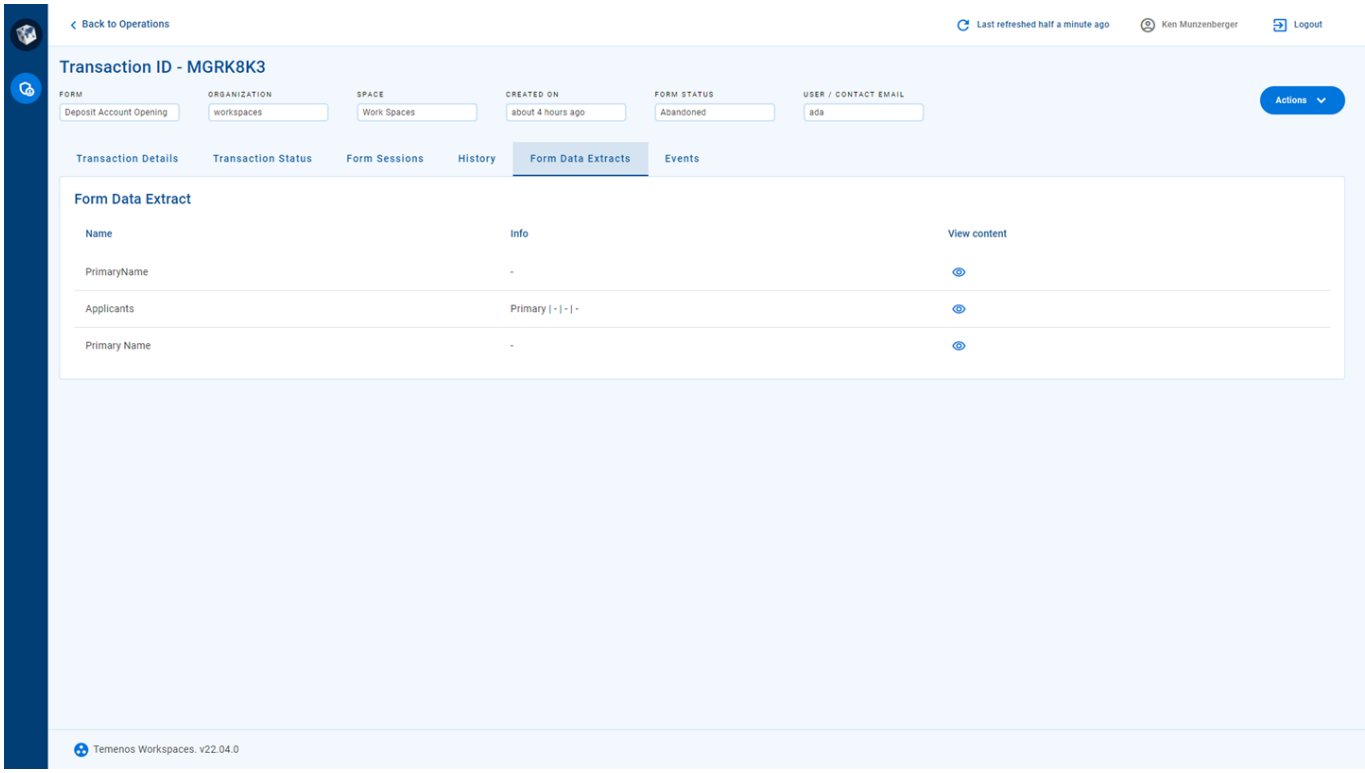
Form Data Extracts Tab

Journey Manager supports a data extract feature to extract form data and store it in the external data elements against form submissions. Whenever a user submits a form with data extracts, this data is stored against each form's submission as a convenient way of viewing and accessing essential data from within Manager. Data extracts may assist in transaction monitoring and reporting, especially for operational support teams, and are required for supporting collaboration jobs.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Form Data Extracts** tab is hidden if all of its cards are hidden.



The Form Data Extracts tab contains a table with the list of data items extracted from the form XML. The following information is displayed for each item.

- Name
- Info: This value may be displayed truncated in the list. You can view the full value using the (View content) option.
- View content: Click the (View content) icon to display the full value of the **Info** field in a modal dialog.

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Form Data Extract** tab.

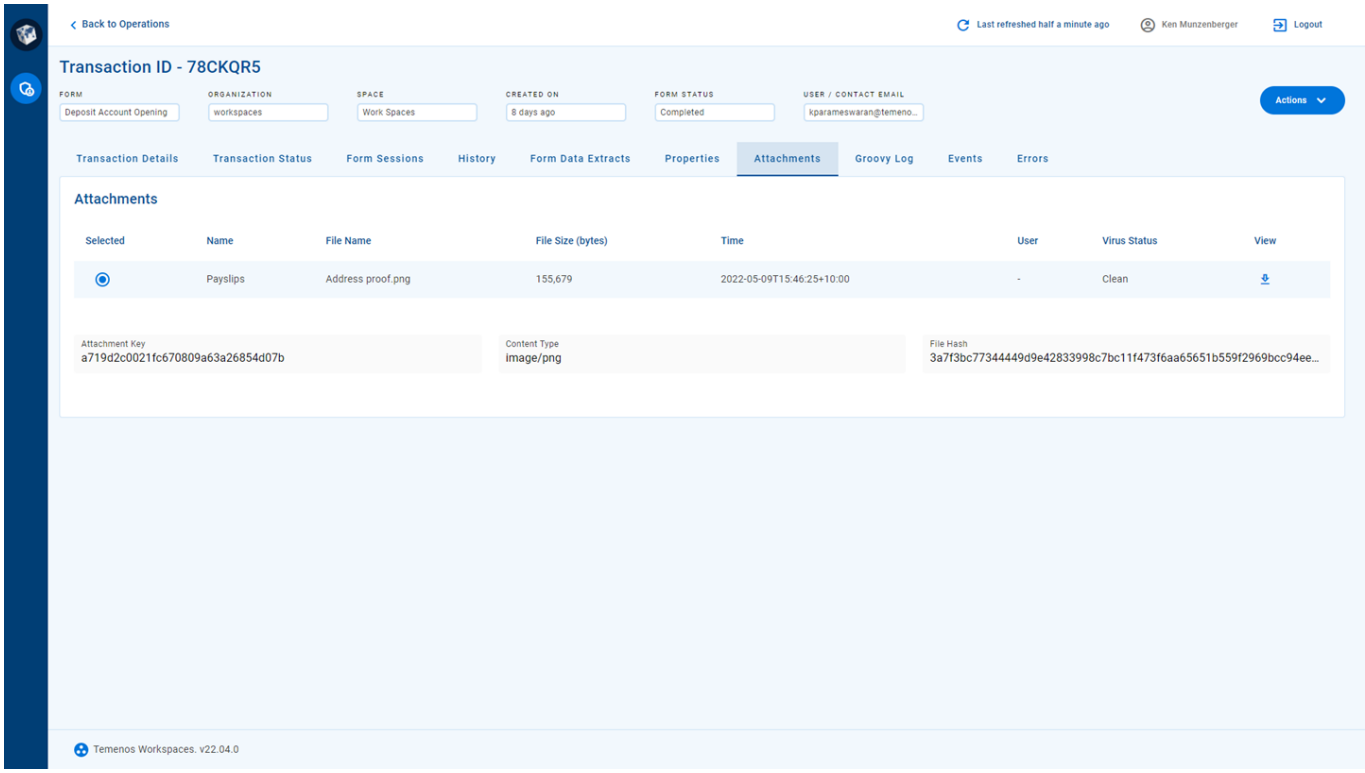
Transaction Attachments Tab

Manager forms often require the user to provide documentation supporting the information entered in the application. These supporting documents are attached to the form transaction when uploaded.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Attachments** tab is hidden if all of its cards are hidden.



The Transaction Attachments tab provides a read-only view of document files attached to the transaction, with attachment meta-data and an option to view the attachments. At the top of the Attachments tab is a list of attachments displaying the following information.

- Name
- File Name
- File Size (bytes)
- Time

- User
- Virus Status
- View: Click the (View) icon to download the attached file so that you can view it.

Selecting an item from the list of attachments displays the following information for the selected item.

- Attachment Key
- Content Type
- File Hash

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Attachments** tab.

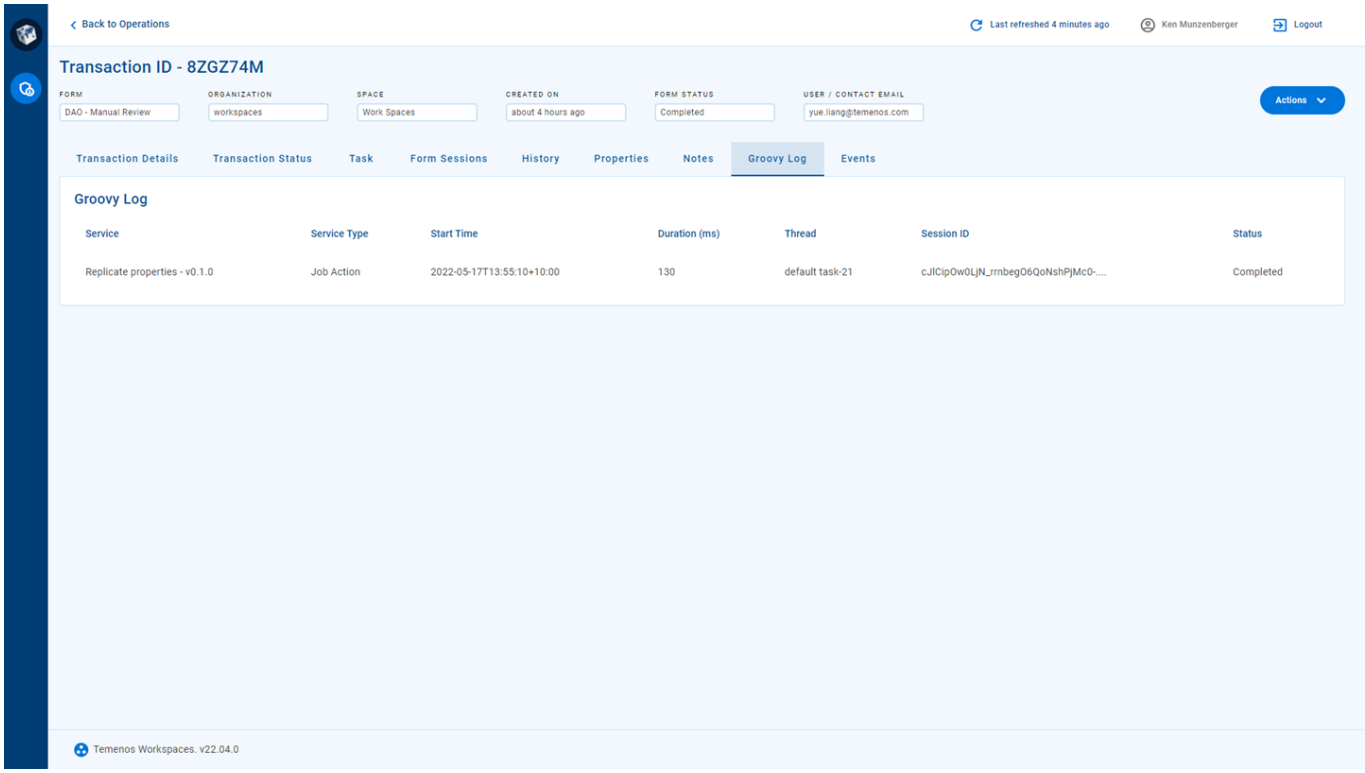
Groovy Log Tab

The Groovy Log tab presents Groovy service output messages for a selected transaction which can be useful for troubleshooting.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Groovy Log** tab is hidden if all of its cards are hidden.



The Groovy Log tab contains a list of items where each item represents the execution of a Groovy script that implements a service associated with a form. The following information is displayed for each item.

- Service
- Service Type
- Start Time
- Duration (ms)
- Thread

- Session ID
- Status

A Groovy script may be executed for a form more than once. If this happens, a separate item appears in the list for each time the script has been executed.

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Groovy Log** tab.

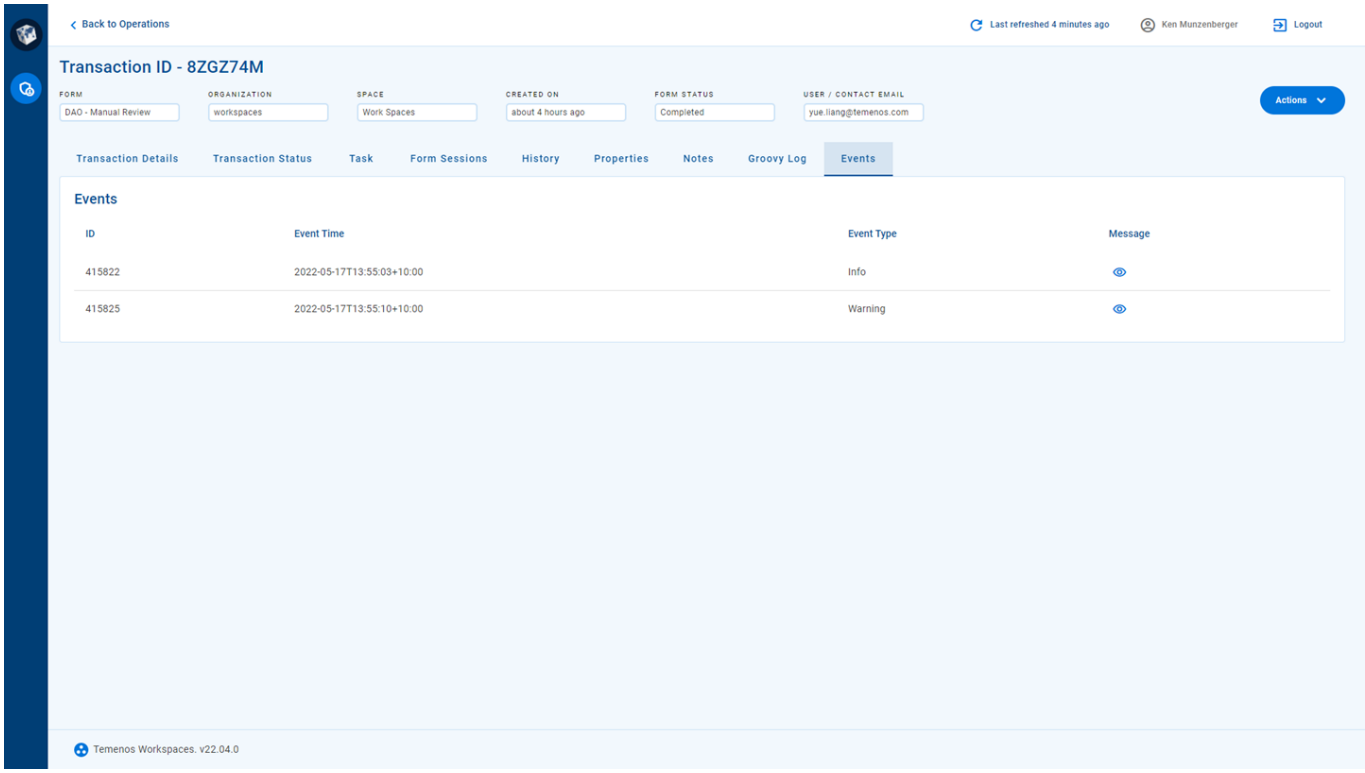
Transaction Events Tab

The Transaction Events tab displays a read-only view of all events logged against a selected transaction.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Events** tab is hidden if all of its cards are hidden.



The Events tab contains a table with the list of events, displaying the following information for each event.

- ID
- Event Time
- Event Type
- Message: Click the (Message) icon to display the event text in a modal dialog.

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Events** tab.

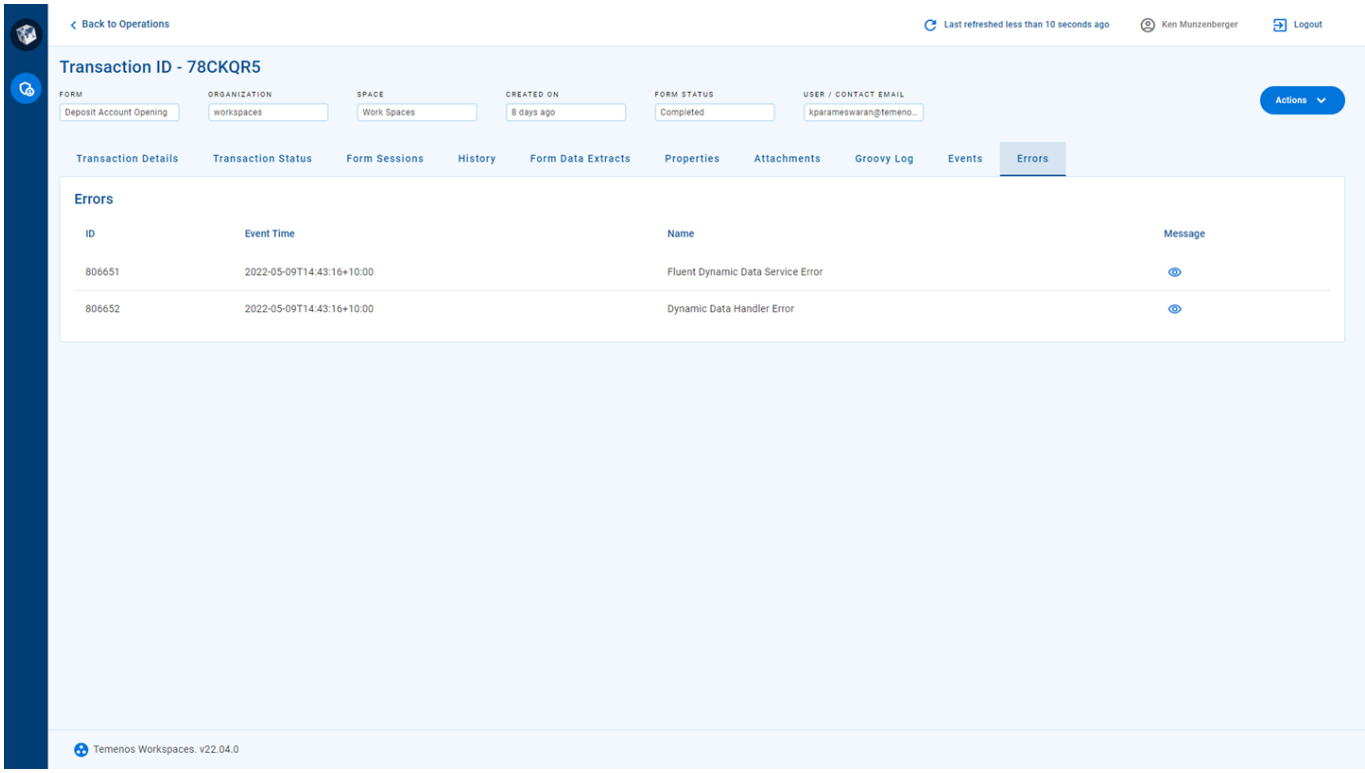
Transaction Errors Tab

The Transaction **Errors** tab displays a read-only view of any errors encountered during a transaction's life cycle.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Errors** tab is hidden if all of its cards are hidden.



The Errors tab contains a table with the list of errors, displaying the following information for each error.

- ID
- Event Time
- Name
- Message
- XML

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Errors** tab.

Transaction Properties Tab

Journey Manager allows you to view and create transaction properties, which represent arbitrary data as string objects. The main persistence model in Manager is an XML[1] file, which is stored on the server and sent to a browser when a form is rendered. It's then sent back to Manager after the form is submitted. It is also saved to the server in background save events.

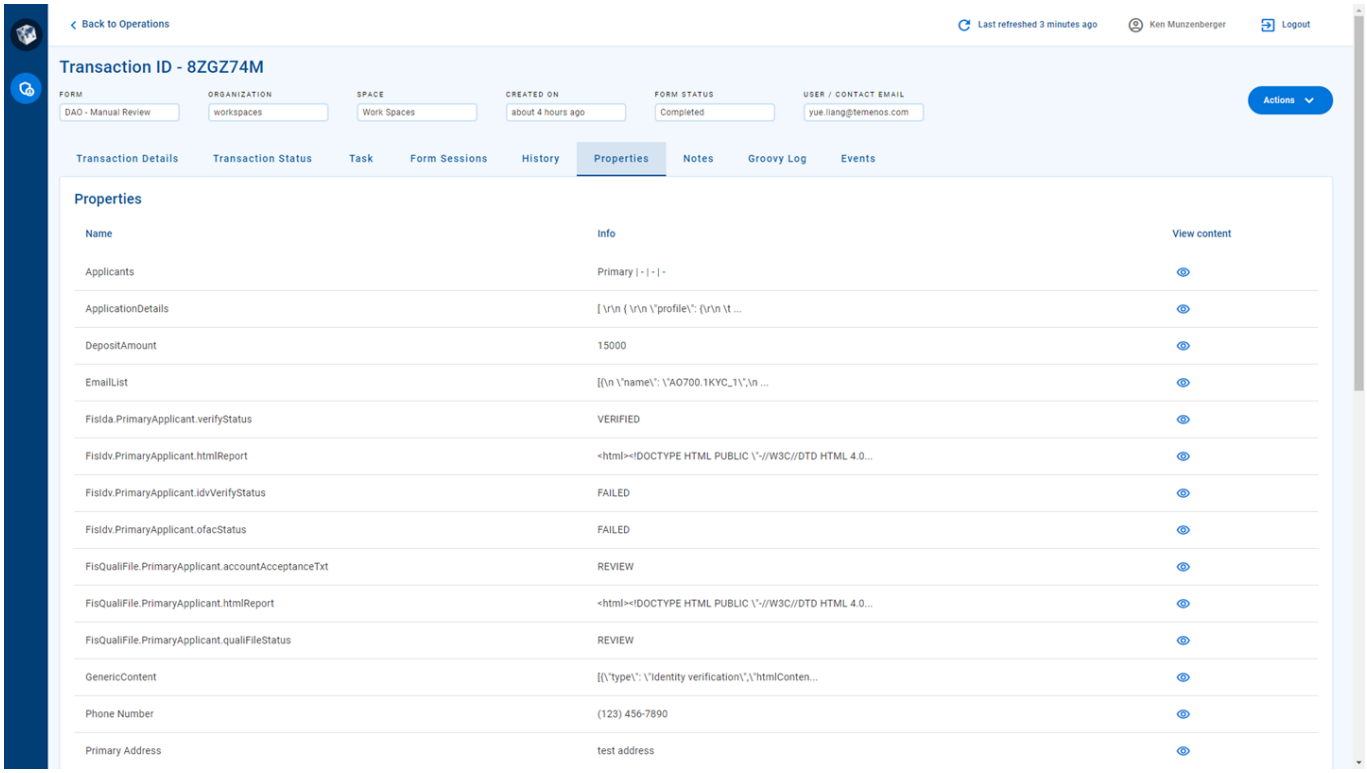
There are situations when it is useful to store some information, which is not part of the XML file, on the server only. For example, you may want to record a user's actual credit score, but not show it to the user. In this case, you can store results from the back-end server calls and send only a pass or fail result to the form, so form users won't see their actual score or the reasons why they failed. You can also store in-progress data that has been used in the back-end service calls temporarily to ensure that form users don't "game" the system by changing the data in the form at different points of their journey.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Properties** tab is hidden if all of its cards are hidden.

[1]XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.



The Properties tab presents a read-only view of transaction property items, displaying the following information for each item.

- Name
- Info: This value may be displayed truncated in the list. You can view the full value using the (View content) option.
- View content: Click the (View content) icon to display the full value of the **Info** field in a modal dialog.

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Properties** tab.

Transaction Notes Tab

As an application progresses through its life cycle, it may pass from one user to another to perform various assessments or other activities. Each user that touches the application can add information such as how the application is processed, details of user inquiries, or changes in prioritization. An application can pass through many different sets of hands, so it's important to capture this information in case it's needed by someone else later in the application life cycle, or to assist in responding to inquiries. Information captured this way is stored as notes against the relevant tasks. Notes can be added to any task at any time, even if the task is completed.

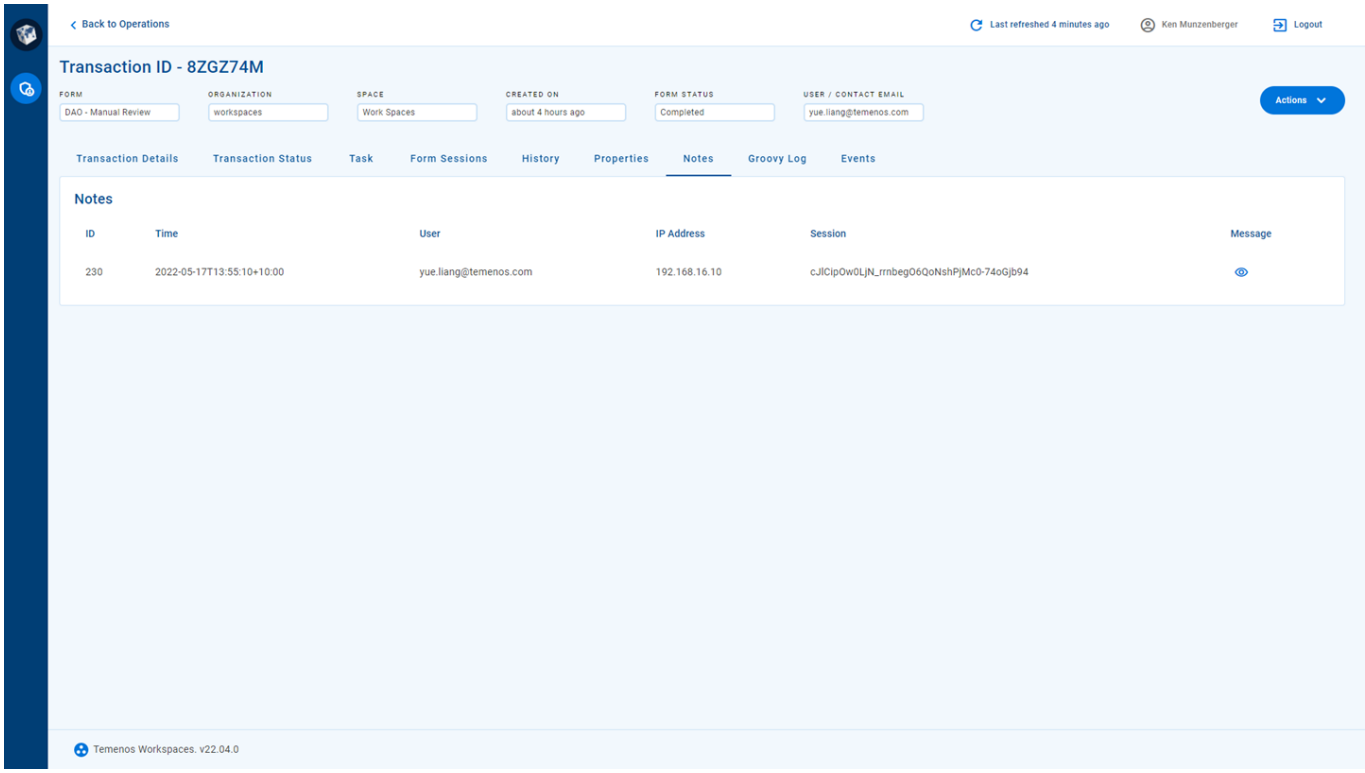
INFO

In Workspaces 22.04 and earlier releases, notes can be added to any task that is in progress, but not to completed tasks.

NOTE

The Operations portal implements a "hide empty" policy on UI elements when no information is available for them.

- Each card is hidden if all of the fields it contains are empty.
- The **Notes** tab is hidden if all of its cards are hidden.



The Notes tab contains a list of comment items associated with the selected transaction, displaying the following information for each item.

- ID
- Time
- User
- IP Address
- Session
- Message
- Message: Click the (Message) icon to view a note's full message text in a modal window.

Similar information is available in the Journey Manager console on the **Transaction Details** page, **Comments** tab.

System Logs

Manager captures a wide range of system and service run-time information and stores it in different log files. It provides a convenient interface for viewing this information, making it easier to troubleshoot diverse production problems.

Manager creates and maintains the following logs, which reside both on the file system and in the database:

- Event Log
- Error Log
- Error Log Trend
- Database Version Log
- T.Field Device Log | 21.11 This feature was removed in 21.11
- T.Field Sync Log | 21.11 This feature was removed in 21.11
- Import Log
- Audit Log
- Groovy Service Log
- Security Manager Log
- Apache Log

You can use System Event Publisher to publish some logging events into various queues that are used by 3rd party tools, for example, Splunk.

A good way to detect issues is to keep an eye on the logs provided by Manager. Inspect the error log and the event log regularly to be alerted to irregularities and problems of all sorts. Make sure you look into issues that you do not fully understand, using additional logs such as the submission and request log to track users' activities, the audit log to track changes made by administrators, the system health log to track vital system statistics or the log files maintained by the application server.

If you have multiple administrators set up in Manager, the audit log should also be monitored to make sure no unintended or harmful changes have been made.

When you log various events into a log file, there are a few recommended practices, as shown below:

Dos:

- Do choose the correct log file for your scenario, for example, error log to keep track of all relevant to your application.

- Do make your log entries meaningful.
- Do add a correlation key to track multiple log entries. In some cases, a submission key is often sufficient.
- Do use try-catch blocks for known issues.

Don'ts:

- Don't log any Personally Identifying Information (PII)[1] data.
- Don't log too much data as it's detrimental for overall performance of the system.
- Don't use logs for auditing and reporting purposes.

Next, learn more about the Event Log.

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

Error Log

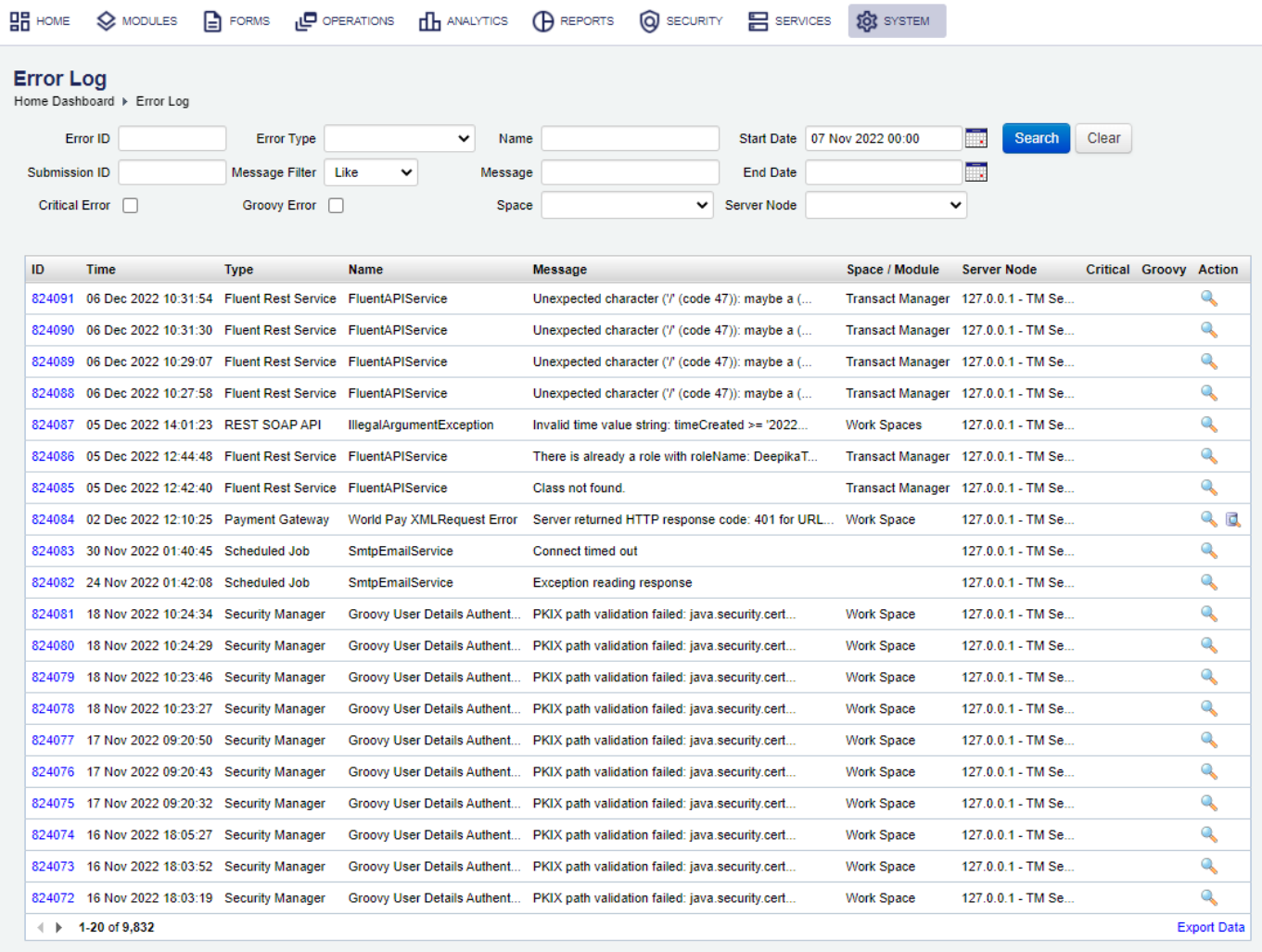
Manager records different types of errors that occur across all form spaces and environments. The examples of errors are:

- Lost Database connections
- Wrong form data configurations
- Delivery errors

Manager allows you to view, filter, and search for specific errors.

To view the error log, select System > Error Log.

Errors are shown page by page, with the maximum number of records per page configured via the user's preferences.



The list displays the following details:

- ID:
- Time:
- Type:
- Name:
- Message:
- Space/Module:
- Server Node:
- Critical:
- Groovy:

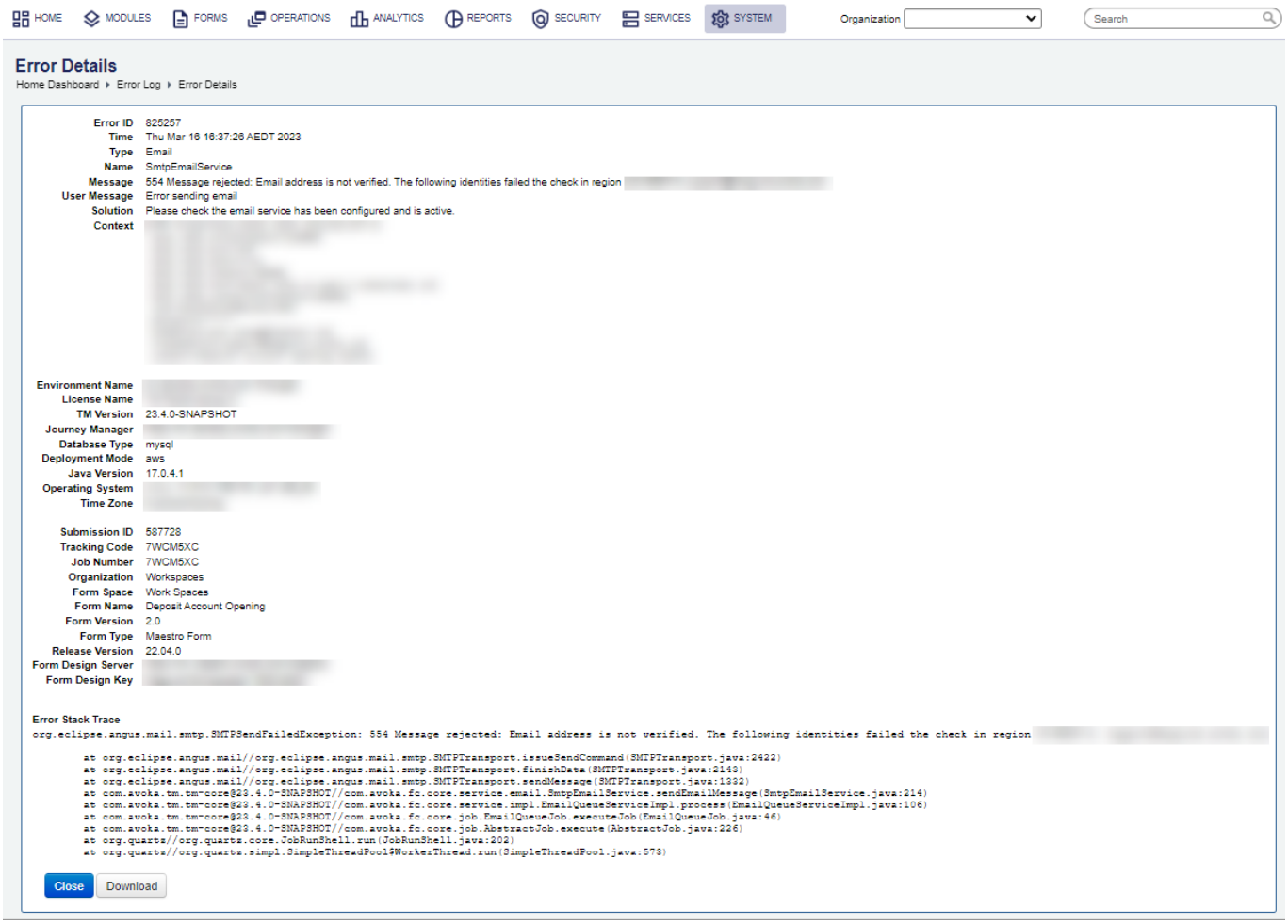
To filter or search the Error Log by one or more criteria, specify the following settings and click Search:

- Error ID: a unique error ID
- Submission ID: a submission ID
- Critical Error: select it to filter Critical Errors only
- Groovy Error: select it to filter Groovy Errors only
- Error Type: select an error type, which can be one of the following:
-
  - Async Message Expiry:
  - Collaboration Job: errors encountered during collaboration job execution.
  - Data Retention Management:
  - Email:
  - Form Abandonment:
  - Form Delivery: delivery of form data errors.
  - Form Dynamic Data:
  - Form Render:
  - Form Submission:
  - Form Transaction History:
  - Fluent Rest Service:
  - Insights
  - Management: console errors, for example, errors cased by importing, modifying or deleting entries such as forms, spaces, or organizations.
  - Payment Gateway:
  - Receipt Render:
  - REST SOAP API:
  - Scheduled Job:
  - Security Manager:
  - T.Field Sync:
  - Transaction Processor:

    - Unclassified: any other error.
- Message text, or part of a message text. The Message Filter provides control on how the message text is used in the search, so select one of these: Like, Not Like, Equal, Not Equal.
- Name
- Start Date and End Date of the event.
- Space
- Server Node

Click Clear to reset the search criteria.

To view additional information, click Details.



This is a great source of information for error troubleshooting and it typically includes:

- Error details, such as Error ID, Time, Type, Name, Message, User Message, Context. Context often has a transaction ID or tracking code, so you can find this transaction in [Transaction Support Search](), where you are able to see more details as well as [transaction]

properties. They may contain some crucial information to help you diagnose this error.

- Environment details, such as Environment Name, License Name, TM Version, Journey Manager, Database Type, Deployment Mode, Java Version, Operating System, Time Zone.
- Submission details, if present, such as Submission ID, Tracking Code, Organization, Form Space, Form Name, Form Version, Form Type, Release Version, Form Design Server, Form Design Key.
- Error stack trace.

Click Close to return to the previous page.
Click Download to save this error as an HTML file on your computer.

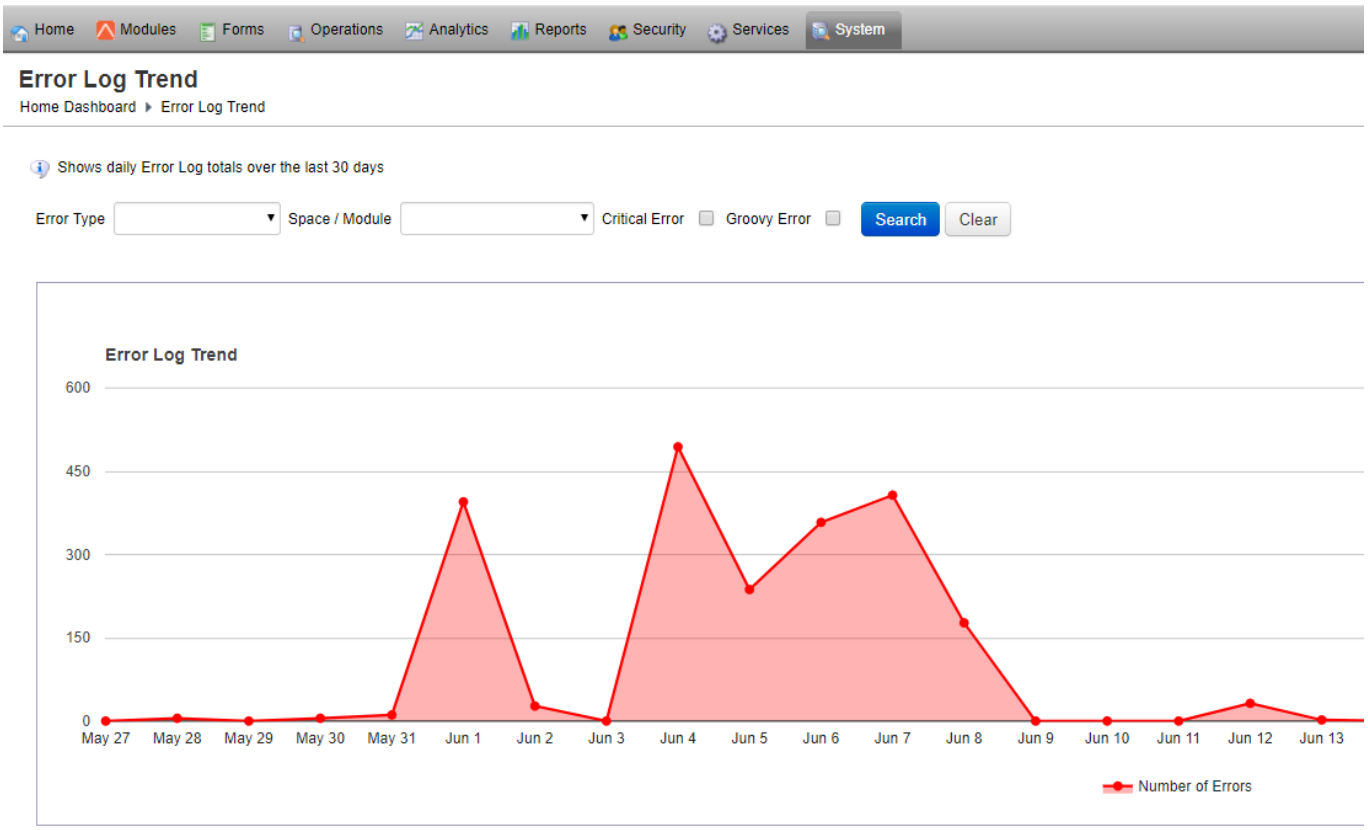
Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn about the import log.

Error Log Trend

The Error Log Trend chart shows the number of error entries logged by Manager for the last month. It provides a quick way of checking the health of the server.

To view Error Log Trend, select System > Error Log Trend.



To filter or search the errors by one or more criteria, specify the following settings and click Search:

- Error Type. For more information, see error log
- Space.
- Critical Error.
- Groovy Error.

Next, learn about other Manager logs.

Event Log

Manager records different events in the Event Log, such as:

- A user notification for a submission delivery
- A notification of a submission is delivered to the organization
- A Manager restart or shutdown

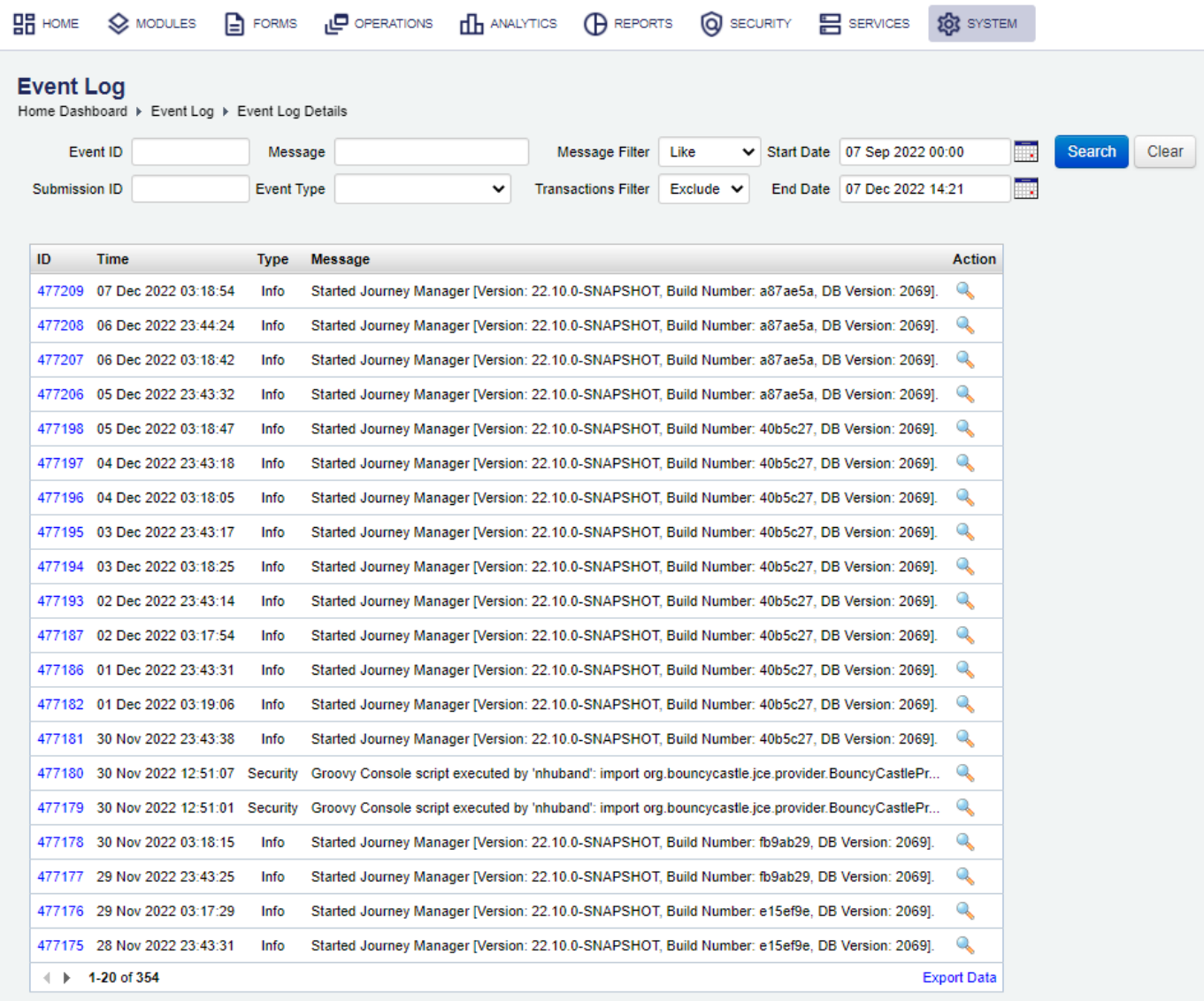
There are the following Event Types in the Event Log:

- Info - Management Dashboard information
- Warning - a warning message to a user
- Error - a Manager error
- Security - a security error, for example, user accounts or log in authentication errors
- Send Event/Delivered - an event was delivered to a Kafka's topic | 22.10 This feature was introduced in 22.10.
- Send Event/Failed - an event failed to be delivered to a Kafka's topic | 22.10 This feature was introduced in 22.10.
- Database Integrity - an event generated by the Database Integrity Checker job during its run. | 23.04 This feature was introduced in 23.04

Manager allows you to view, filter, and search for specific system events.

To access the event log, select System > Error Log.

Events are shown page by page, with the maximum number of records per page configured in the users preferences.



To filter or search the event log by one or more criteria, specify the following settings and click Search:

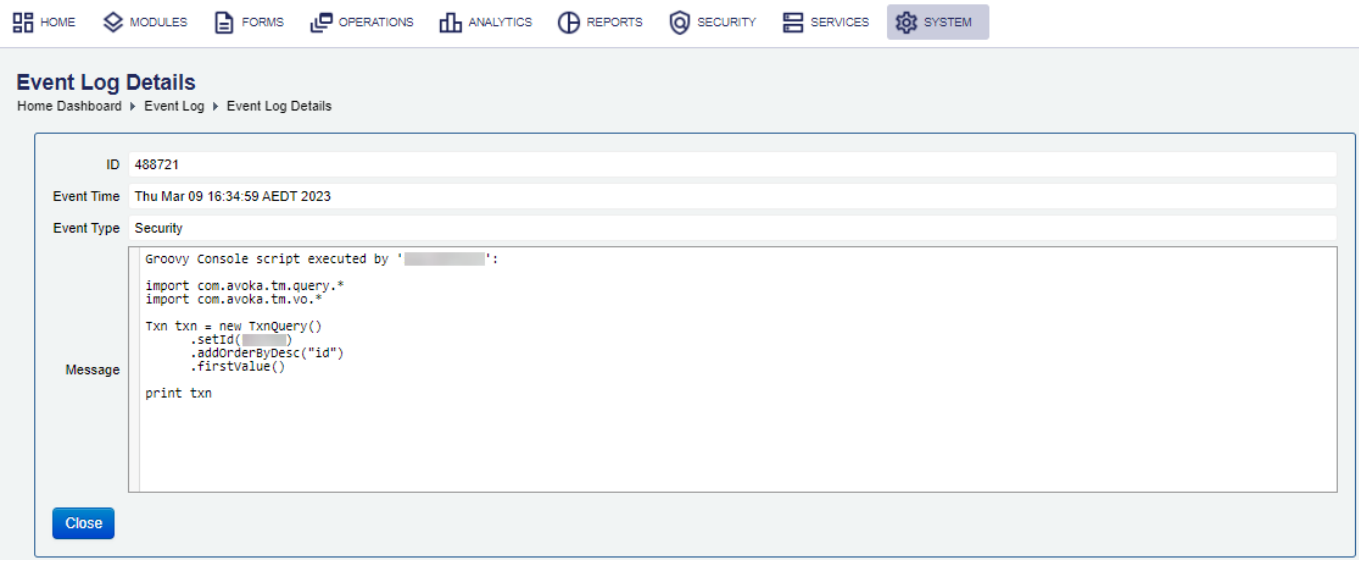
- Event ID
- Submission ID
- Event Type:
  - Info - Management Dashboard information
  - Warning - a warning message to a user
  - Error - a Manager error
  - Security - a security error, for example, user accounts or log in authentication errors
  - Send Event/Delivered - an event was delivered to a Kafka's topic | 22.10 This feature was introduced in 22.10.
  - Send Event/Failed - an event failed to be delivered to a Kafka's topic | 22.10 This feature was introduced in 22.10.

    - Database Integrity - an event generated by the Database Integrity Checker job during its run. | 23.04 This feature was introduced in 23.04
- Message text, or part of a message text. The Message Filter provides control on how the message text is used in the search, so select one of the following:
    - Like
    - Not Like
    - Equal
    - Not Equal
- Start Date and End Date of the event
- Transaction Filter with the Includeand Exclude options

Click Clear to reset the search criteria.

To view additional information, click Details.

The Groovy script execution information is also logged to the Event Log, as shown below.



Click Close to return to the previous page.

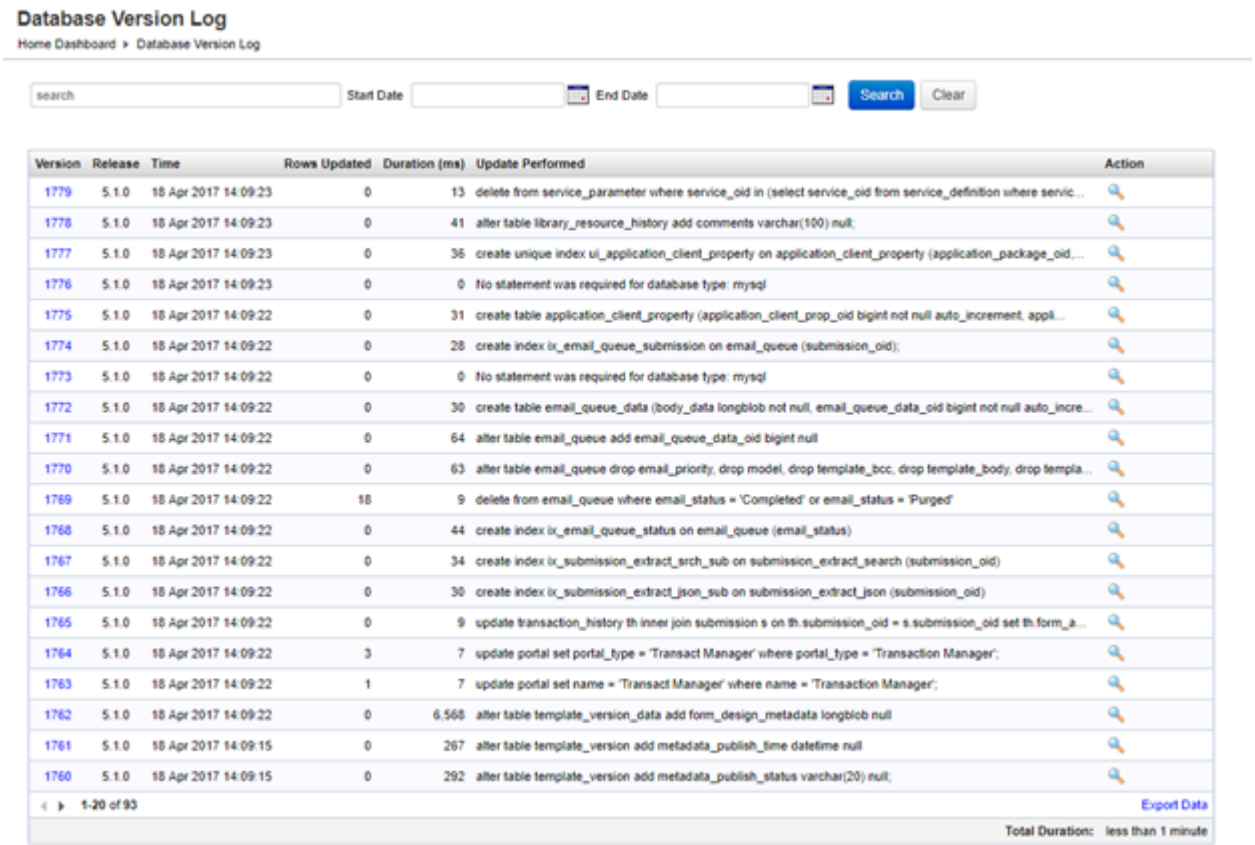
Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn about the error log.

Database Version Log

Every time it starts, Manager checks the database version and schema name, runs update statements, and stores this information in the database version log. The database version log lists all updates applied to the database since Manager was installed. Each entry contains a version number and the description of an update.

To view the database version log, select System > Database Version Log.



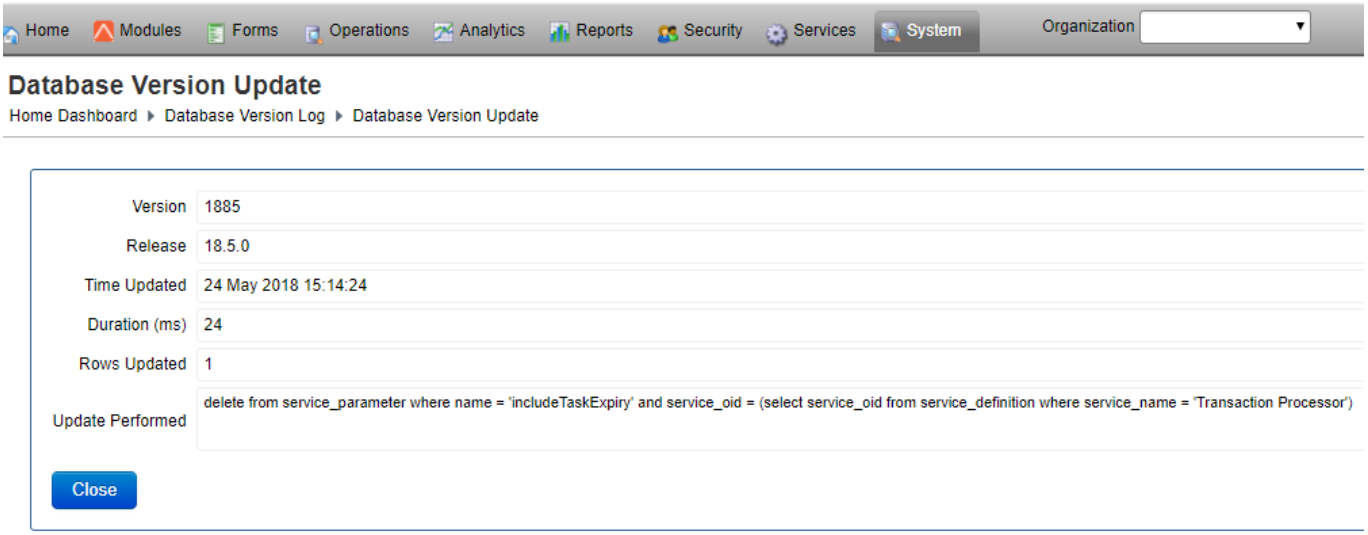
Log records are shown page by page, with the maximum number of records per page configured via the user's preferences.

To filter or search the database version log by one or more criteria, specify the following settings and click Search:

- Search text, which can be a release number, for example, 18.5.0, or a description of the update, as shown in the Update Performed column.
- Start Date and End Date of the event.

Click Clear to reset the search criteria.

To view additional information, click Details.



Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

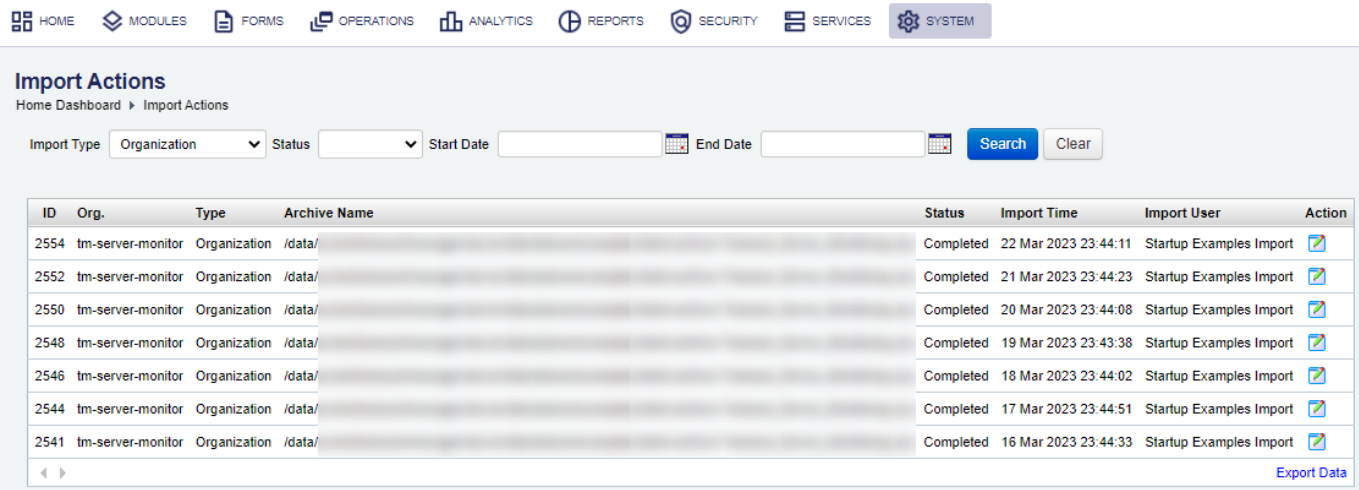
Next, learn about the error log.

Import Log

Manager keeps a log of all import operations and actions for tracking and auditing purposes. The log contains records of each import outcome and any warnings or issues that occurred during the import process.

To access the import log, select System > Import Log.

Import records are shown page by page, with the maximum number of records per page configured via the user's preferences.



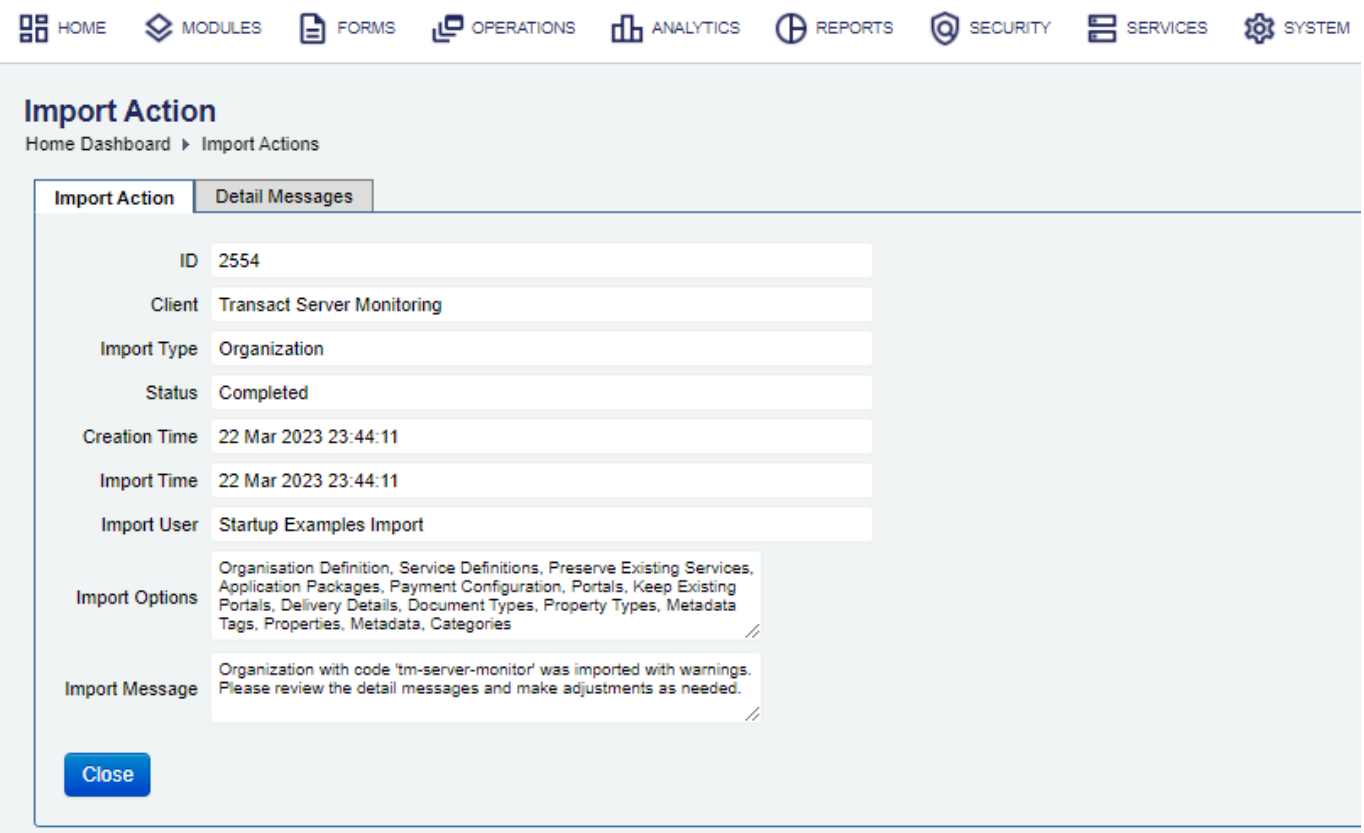
To filter or search the import log by one or more criteria, specify the following settings and click Search:

- Import Type - a type of an import operation, which can be one of the following:
  - Application Package
  - Form Version
  - Groups
  - Library
  - Metadata Tag
  - Organization
  - Portal
  - Roles
  - Service
  - Service Connection
  - Security Manager
  - TPac
  - Configuration Service - | 23.04 This feature was introduced in 23.04

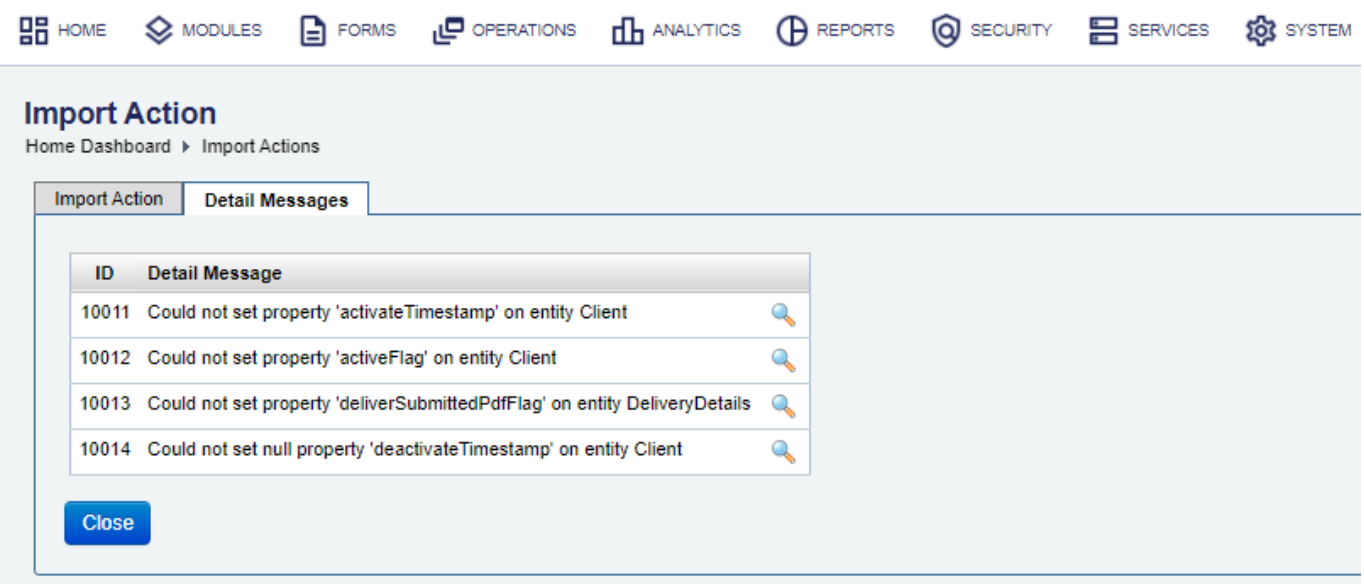
- Status - a status of the import operation, which can be one of the following:
  - Ready
  - Completed
  - Error
- Start Date and End Date - a date range when the import operation happened

Click Clear to reset the search criteria.

Click Edit to view the import information.



Click Detailed Message to view the additional import information.



Click Close to return the main page of the imported component type, that is the TPac in this example.

To remove an import record, click Remove and confirm it.

> WARNING
>
> We don't recommend removing import log records.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

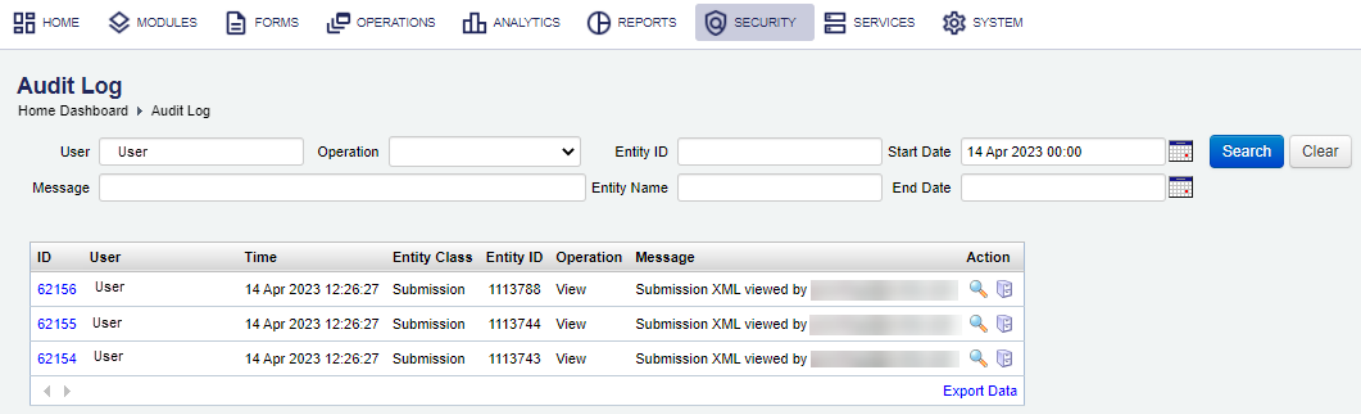
Next, learn about database version log.

Audit Log

Manager keeps track of all operations performed on entities[1] by users in Audit Log. For example, when an administrator views or changes form configuration, Manager logs that operation to Audit Log. Or, when a user changes a delivery status of a submission by selecting a new status from the Delivery Status dropdown list, an audit record is added to Audit Log with the entity name as Submission.

To view the audit log, select Security > Audit Log.

Audit events are shown page by page, with the maximum number of records per page configured via the user's preferences.



The Audit Log list displays the following details:

- ID - an audit record ID. Click ID to see additional information of a selected audit record.
- User - an actor who made the change. The actor can be an administrator (most common case), but also a form space user or even a scheduled job. In any case, the actor's login name (for administrators and form space users) or name (for jobs) is logged.

  > NOTE
  >
  > Unknown is used when a user can't be identified, for example, when an anonymous transaction is involved or when the changes were made by certain automated internal Manager systems.

- Time - a date and time when the change was made

---

[1]Entity is a generic name for various objects, which Journey Manager uses internally, for example, an organization is an entity that owns forms.

- Entity Class - a class name of the object that a user performed an operation on. Some examples are shown below:
  - ApplicationPackage
  - Form
  - Form Version
  - ImportAction
  - Organization
  - SchemaSeed
  - Submission
  - Template Version
  - UserAccount
  - UserGroup
  - UserProperty
  - VersionPropertyValue
- Entity ID - an ID of an entity that was affected by the change, which is unique within the entity type. For example, two different forms always have different IDs, but a form may have the same ID as an organization.
- Operation:
  - Create - audit records that are associated with the create event
  - Delete - audit records that are associated with the delete event
  - Export - audit records that are associated with the export event
  - Import - audit records that are associated with the import event
  - Update - audit records that are associated with the update event
  - View - audit records that are associated with the view event, for example, submissions viewed by a user
- Message - a detailed description of the operation performed on an entity, for example, `Form Version with name 'form-testpuppeteer-v1.0.2.zip' imported by user-1`. For Create and Updateoperations, a message contains attributes that were changed and their new values. For a Delete operation, the message only contains the name of the entity that was deleted and its ID. For more information, see Message Examples below.

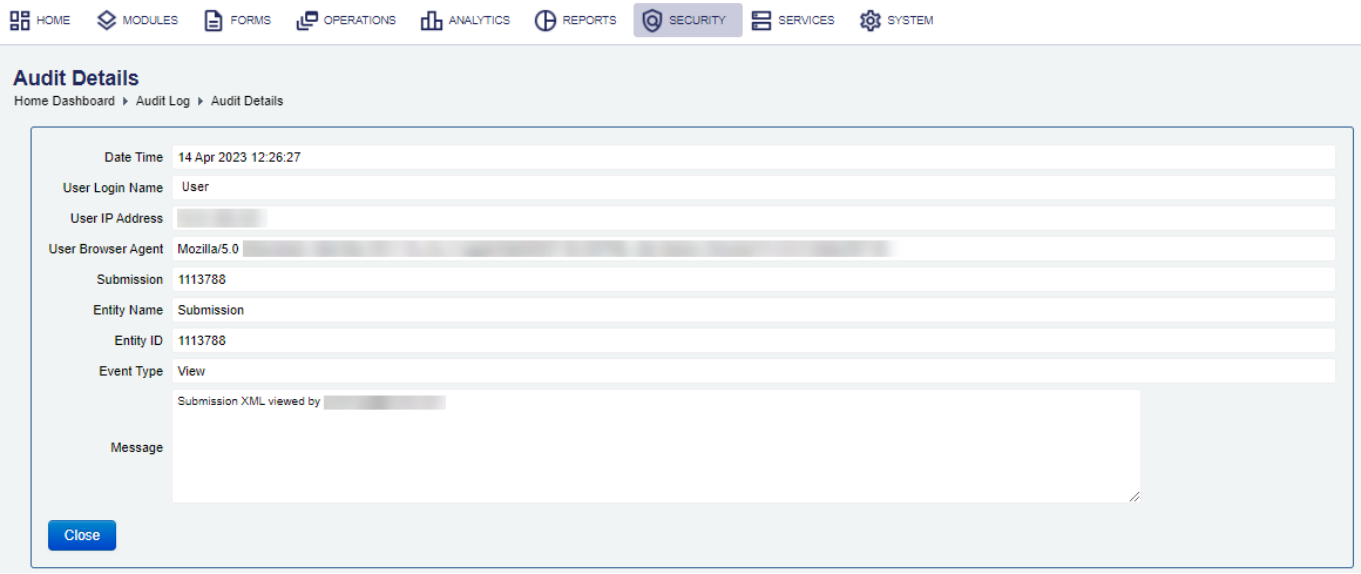
To filter or search the audit log by one or more criteria, specify the following settings and click Search:

- User name or email - provider an actor's username or email address
- Operation - select on of the following operations:
  - Create - audit records that are associated with the create event
  - Delete - audit records that are associated with the delete event

    - Export - audit records that are associated with the export event
    - Import - audit records that are associated with the import event
    - Update - audit records that are associated with the update event
    - View - audit records that are associated with the view event, for example, submissions viewed by a user
- Message text - enter a part of message content
- Entity ID - provide an entity ID to find what changes (operations) were applied to a corresponding entity
- Entity Name- type an Entity Class name, for example, Submission
- Start Date - select the start date of your search from the date picker
- End Date - select the end date of your search from the date picker

Click Clear to reset the search criteria.

Click ID or View to see additional information of a selected audit record, as shown below:



NOTE

Each audit log entry contains the same fields. However, depending on the nature of the change logged, the fields have to be interpreted in different contexts.

The Audit Log record displays the following details:

- Date Time - the date and time when the change was made
- User Login Name - the actor who made the change
- User IP Address - the IP address of the actor who made the change
- User Browser Agent - the browser of the actor who made the change

- Submission - a submission ID, if the Entity Name is Submission

> NOTE
>
> You can look up submission details by using this ID as Reference. For more information, see Transaction Support Search.

- Entity Name:
  - ApplicationPackage
  - Form
  - Form Version
  - ImportAction
  - Organization
  - SchemaSeed
  - Submission
  - Template Version
  - UserAccount
  - UserGroup
  - UserProperty
  - VersionPropertyValue
- Entity ID - the ID of an entity (object) in Manager that this audit record pertains to
- Event Type - the operation that was performed on the entity:
  - Create - audit records that are associated with the create event
  - Delete - audit records that are associated with the delete event
  - Export - audit records that are associated with the export event
  - Import - audit records that are associated with the import event
  - Update - audit records that are associated with the update event
  - View - audit records that are associated with the view event, for example, submissions viewed by a user
- Message - the detailed description of the operation performed on an entity

Click Close to return the audit log.

Click Submission View to see transaction details of a selected audit record, if it is related to an entity of the Submission type.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Message Examples

`Created UserProperty 15737`

This message means a UserProperty object (an entity with the UserProperty Entity Name) with the internal ID 15737 was created. Manager doesn't provide an UI interface to query a user property by ID, so, if required, you must run a query in your database directly.

`Submission Receipt Status changed from 'Ready' to 'Ready' by Unknown. Delivery Status reset to Not Ready`

This message means the receipt status was not changed - it is set to Ready, but the delivery status was set to Not Ready. The operation was performed by an Unknown user, meaning Manager didn't have user information to log into Audit Log at that time. It could be that a Manager internal job executed that delivery status change.

Next, learn about the event log.

Groovy Service Log

Manager allows you to view messages generated and logged by functions and services written in Groovy, so you can debug and profile them. When a form calls a Transact function, it executes a Groovy script, which can be programmed to write some information into a log file by using a `Logger` class, for example, `logger.debug("Variable: " + varaibleValue)`. In this case, this message is logged into Groovy Service Log and can be viewed in the Groovy Service Log tab.

WARNING

Do not use Groovy Service Log for any other purposes but troubleshooting of Groovy services.
Do not use Groovy Service Log for auditing and reporting.

WARNING

Messages logged to the Groovy Service Log are stored in the `GroovyServiceLog` table, which can grow big, so you need to set up data purging according to your requirements.

NOTE

All log messages associated with a transaction or PII[1] data are encrypted so you can't search for a part of encrypted text. If you want to search encrypted messages, you need to implement different logging solutions, for example, using the System Event Publisher.

The log can be empty if there is no debugging configured or if the debugging is disabled.

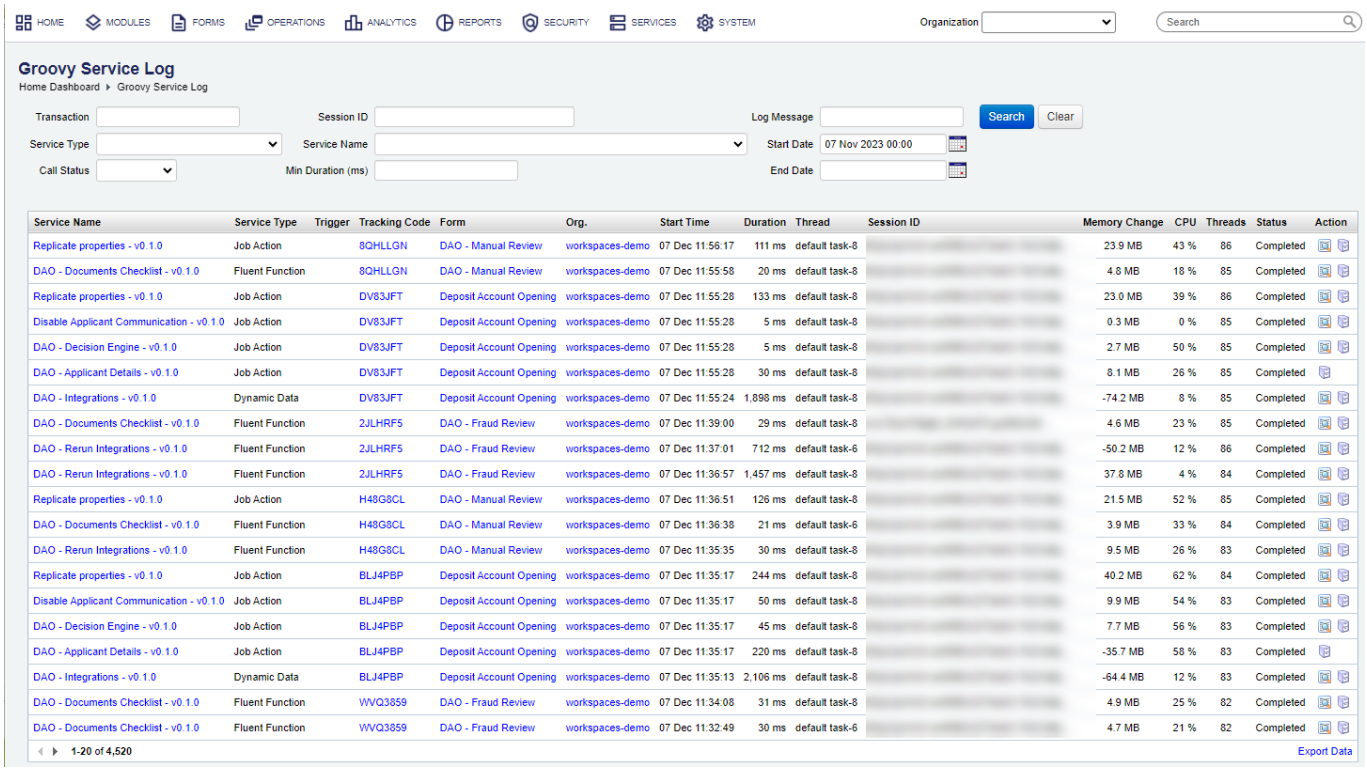
[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

NOTE

Do not log PII[1] data to the service log, for example, to debug a service, as there's a risk of exposing customer's details through the log file.

To view all Groovy service log messages, select Services > Groovy Service Log.

Messages are shown page by page, with the maximum number of records per page configured via the user's preferences.



The Groovy Service Log list displays the following details for each log entry:

- Service Name - a service that logged info into the Groovy Service Log. Click the service name to see more details
- Service Type - a type of a service
- Trigger - a trigger of a function

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

- Tracking Code - a tracking code of a transaction associated with the service invocation. Click a tracking code to see more details
- Form - a name of a form that called the service or the function
- Org - a client code of an organization where this form is hosted
- Start Time - date and time when the service or the function was invoked
- Duration - how long the service or the function ran, in milliseconds
- Thread - a name of a Java thread in which this service was executed
- Session ID - a form session ID
- Memory Change - how much memory was allocated for the service of the function invocation
- CPU - the CPU utilization at the service invocation time
- Threads - a number of threads used by a service or a function
- Status - a service invocation status:
  - Completed
  - Error

To filter or search the Groovy Service Log by one or more criteria, specify the following settings and click Search:

- Transaction
- Session ID
- Log Message, or part of a message text, This searches through unencrypted messages logged by services that are not transaction-related.

  NOTE

  You can't use this functionality to look for a part of encrypted text.

- Service Type. Select a type of a service from the dropdown list to see all messages logged by services of this type.
- Service Name. Select a service name from the dropdown list to see all messages logged by services of this name.

  NOTE

  Only services of a selected service type are shown in the dropdown list, if you used the Service Type filter above.

- Call Status. Select a status of the message from the dropdown list, that can be: Completed, Error, or both.

- Min Duration (ms). Select a minimum duration of a service call. This is handy to troubleshoot services and overall server performance by filtering down services that took longer to do work.
- Start Date and End Date of the message.

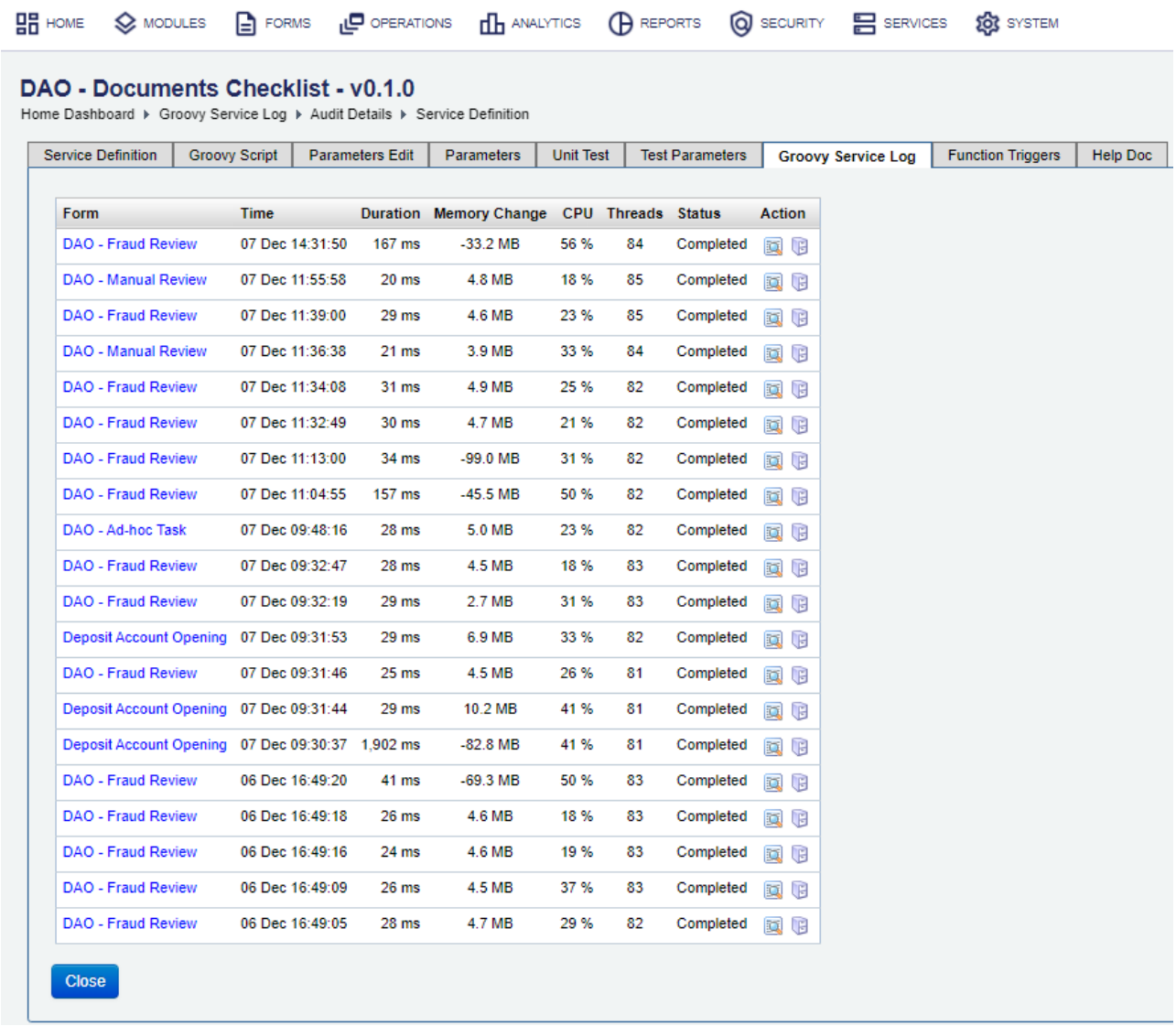
Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Alternatively, you can view Groovy Service Log directly from a Transact function's or a service's definition.

To view a Transact function's or a service's Groovy service log:

1. Select Services > Transact Functions or Services > All Services.
2. Locate a Transact function or a service and click Edit.
3. Click the Groovy Service Log tab to see the list of forms from where functions were invoked with the following details:



Form	Time	Duration	Memory Change	CPU	Threads	Status	Action
DAO - Fraud Review	07 Dec 14:31:50	167 ms	-33.2 MB	56 %	84	Completed	
DAO - Manual Review	07 Dec 11:55:58	20 ms	4.8 MB	18 %	85	Completed	
DAO - Fraud Review	07 Dec 11:39:00	29 ms	4.6 MB	23 %	85	Completed	
DAO - Manual Review	07 Dec 11:36:38	21 ms	3.9 MB	33 %	84	Completed	
DAO - Fraud Review	07 Dec 11:34:08	31 ms	4.9 MB	25 %	82	Completed	
DAO - Fraud Review	07 Dec 11:32:49	30 ms	4.7 MB	21 %	82	Completed	
DAO - Fraud Review	07 Dec 11:13:00	34 ms	-99.0 MB	31 %	82	Completed	
DAO - Fraud Review	07 Dec 11:04:55	157 ms	-45.5 MB	50 %	82	Completed	
DAO - Ad-hoc Task	07 Dec 09:48:16	28 ms	5.0 MB	23 %	82	Completed	
DAO - Fraud Review	07 Dec 09:32:47	28 ms	4.5 MB	18 %	83	Completed	
DAO - Fraud Review	07 Dec 09:32:19	29 ms	2.7 MB	31 %	83	Completed	
Deposit Account Opening	07 Dec 09:31:53	29 ms	6.9 MB	33 %	82	Completed	
DAO - Fraud Review	07 Dec 09:31:46	25 ms	4.5 MB	26 %	81	Completed	
Deposit Account Opening	07 Dec 09:31:44	29 ms	10.2 MB	41 %	81	Completed	
Deposit Account Opening	07 Dec 09:30:37	1,902 ms	-82.8 MB	41 %	81	Completed	
DAO - Fraud Review	06 Dec 16:49:20	41 ms	-69.3 MB	50 %	83	Completed	
DAO - Fraud Review	06 Dec 16:49:18	26 ms	4.6 MB	18 %	83	Completed	
DAO - Fraud Review	06 Dec 16:49:16	24 ms	4.6 MB	19 %	83	Completed	
DAO - Fraud Review	06 Dec 16:49:09	26 ms	4.5 MB	37 %	83	Completed	
DAO - Fraud Review	06 Dec 16:49:05	28 ms	4.7 MB	29 %	82	Completed	

- A form name - a form name that invoked the service or a function. Click the form name to see more details in a form dashboard
- A function invocation time
- Duration of the invocation
- Memory change - how much memory was allocated for a function or service invocation. It shows the delta (difference) in JVM free memory before and after the service invocation. The higher the absolute value is, the bigger memory consumption is.

> NOTE
>
> The memory change value can be negative, but it is the delta value what is important for you to analyze.

- CPU - the CPU utilization at the service invocation time

    - Number of Java threads used by the function or the service
    - Status of the invocation:
        - Completed
        - Error
4. Click Groovy Logger to view the output created during the function invocation.

    NOTE

    Select the Groovy Log tab to see the same logged messages.

5. Click Submission View to see details of a transaction associated with this function call.
6. Click Close.

NOTE

To enable Groovy logging, select the Groovy Scripting Enabled checkbox in the Parameters Edit tab.

Next, learn how configure Transact function service definition.

Groovy Log to Splunk Integration

Splunk is popular software for searching, monitoring, and analyzing machine-generated data, such as log files, which you can use it to have insights into errors and perform proactive monitoring and alerts for your applications.

Although Manager doesn't provide a direct integration with Splunk, you can configure a System Event Publisher with an AWS SQS adapter to push log records to an AWS queue. Then, you integrate Splunk with the AWS SQS, so it pulls Manager log records from there.

The Splunk integration is summarized below:

1. Enable and configure a system event publisher in Manager
2. Configure an AWS SQS adapter and configure a service connection to use the AWS SQS service
3. Configure Splunk AWS SQS adapter to consume (pull) the AWS SQS messages, transform and store them in the local Splunk data storage
4. Start using Splunk to analyze and monitor your applications

It is important to understand that you will deal with messages as events, not raw exports of log data, so you need to distinguish what an event and log record mean in this context.

Downside of such integration is that an event may contain redundant data, such as link to a transaction, within a set of messages (events) for the same transaction, which can dramatically increase amount of raw data to be stored on a client's side. The logic importing this data to Splunk should consider that.

It means that, for instance, a submission status change event will have event logs and error logs linked to the submission. If the submission status changes multiple times (and it often does), all event messages to the AWS SQS will contain all submission's event and error log records up to that milestone. In short, you will be getting duplicates of the early recorded log records.

> NOTE
>
> This solution creates significant amount of data in Splunk, as compared to Manager logs.

If you want to analyze the Groovy service log records in Splunk, you need to know that the Manager's `GroovyLogger` class internally has a SLF/Log4J logger which writes messages to a `ser-`

`vice.log` file. The default logging level is `ERROR`, which means `DEBUG`, `INFO` and `WARN` messages are not logged to disk. The reason for this default setting is to prevent PII[1] data being logged to disk.

If this is configured to `INFO` or `DEBUG` level, then `GroovyLogger` statements at those levels will be logged to the `server.log` file as well. From here they can be feed into Splunk or other log management systems, regardless of the Groovy Service Logging configurations for the database.

The System Event Publisher can feed Groovy Service Log Data message to the AWS SQS endpoint to support integration with other log management systems without having to write PII data to disk. The Groovy Service Log Data messages will include the same content as would be included in the Groovy Service Log Data records, so it cannot be used as a complete substitute to the Groovy Service Log records.

Next, learn about the system logs.

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

Apache Log

Being a web application that runs on an Apache server, Manager writes some important system information, such as event and error messages, into Apache log files. You generally don't need to check these log files unless you are troubleshooting some server errors, for example, FormDynamicDataServlet - Servlet not authorized.

To view the content of Apache log files, contact your Support team or Cloud Services team and ask them to send you a log file or a snippet of a file as per your requirements, for example, a date and time interval when an error took place.

NOTE

You can't access Apache log files from the Manager console.

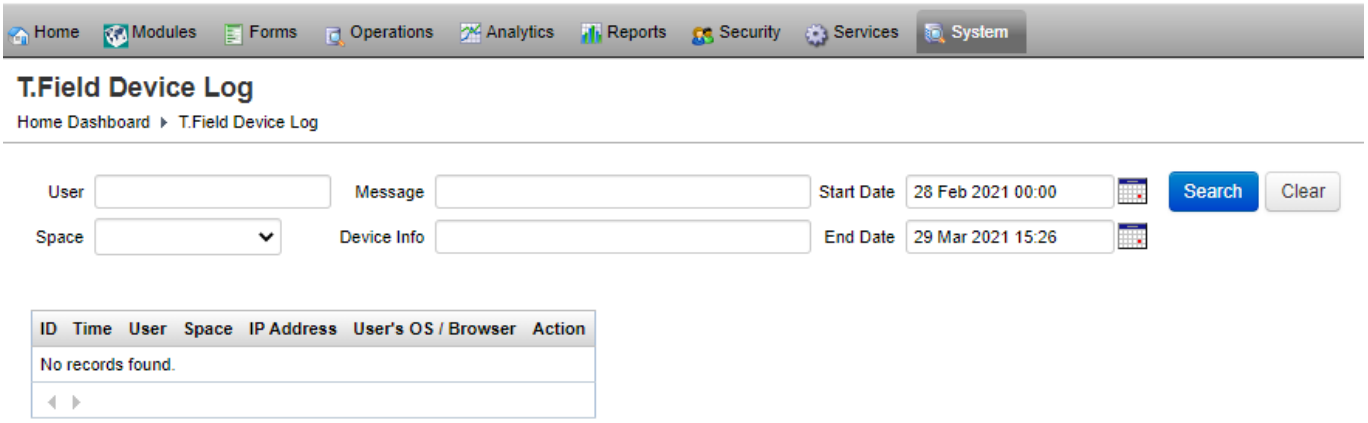
Next, learn about the error log.

T.Field Device Log

Manager comes with many types of log files and one of them is T.Field Device Log that allows you to view log files of a device, which runs the TransactField App. A TransactField App user must elect to upload their device log to a Manager server first. This can be very helpful in debugging issues occurring on the device running the TransactField App.

To view T.Field Device Log, select System > T.Field Device Log.

Log entries are shown page by page, with the maximum number of records per page configured in the users preferences.



To filter or search log entries by one or more criteria, specify the following settings and click Search:

- User
- Message
- Space - select a form space.
- Device Info
- Start Date and End Date of the event.

Click Clear to reset the search criteria.

To view additional information, click Details. You can also view the corresponding synchronization log entry.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

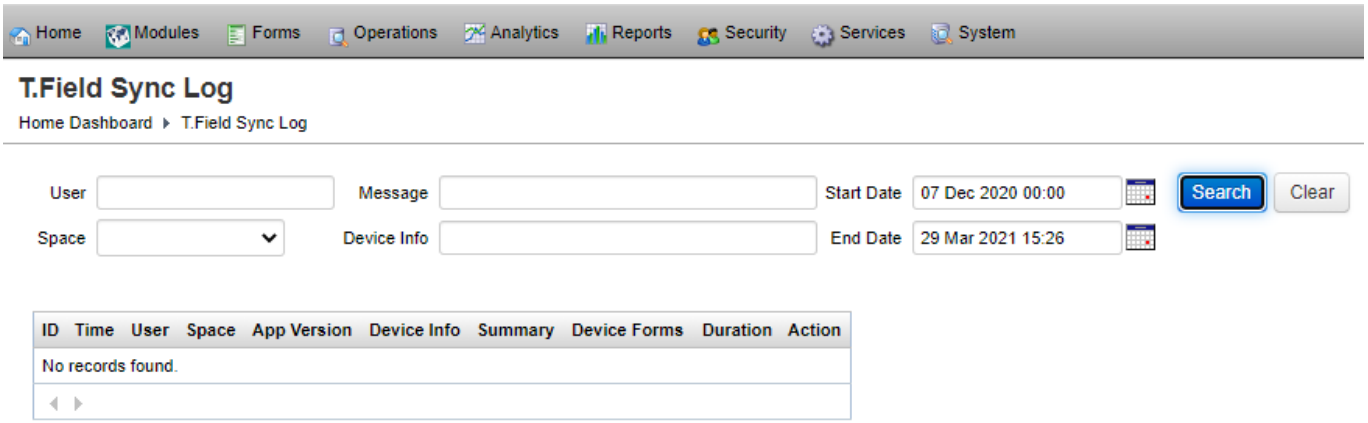
Next, learn about T.Field sync log.

T.Field Sync Log

Manager comes with many types of log files and one of them is T.Field Sync Log that allows you to view operations invoked by devices running the TransactField App. This is useful in debugging the TransactField App issues.

To view the T.Field Sync Log, select System > T.Field Sync Log.

Log entries are shown page by page, with the maximum number of records per page configured in the users preferences.



To filter or search log entries by one or more criteria, specify the following settings and click Search:

- User
- Message
- Space - select a form space.
- Device Info
- Start Date and End Date of the event.

Click Clear to reset the search criteria.

To view additional information, click Details.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

> NOTE
>
> TransactField App users can elect to upload their device log to the Manager server, which you can viewe using T.Field Device Log

Next, learn about T.Field device log.

Journey Manager Analytics

Journey Manager collects a broad range of data related to user's interactions with forms, such as form submissions, and visualizes it with via the following charts.

- Transactions trends
- Forms activity
- Referer submissions
- Form submissions
- Device types submissions
- Reader Versions
- Submission Trend - | 18.05 This feature was removed in 18.05

Most charts are based on an additional set of database tables that hold a subset of submission and request data. These tables are regularly updated are more lightweight than the submission and request tables themselves. The data contained in the tables can also be purged automatically and independently from the submission and request tables.

Manager also provides A/B testing functionality, so you can perform A/B testing and analyze the results to decide which form version to use.

> NOTE
>
> For more extensive transaction analysis, you should use Journey Analytics.

Analyze Transaction Trend

Manager provides an easy to use tool to visualize and analyze transaction trends using the following charts:

- Transaction volume
- Completion rate
- Conversion rate
- Form abandonment rate
- Bounce rate
- Median form completion time taken by users to complete a submission.

Transactions are often long running and can take many days or even weeks to complete, so Manager makes the following assumptions while calculating the trends:

- Analytics for completion, conversion, form abandonment, and bounce rates exclude in-progress transactions, such as opened, saved, and assigned.
- Completion, conversion, form abandonment, and bounce rates are trailing indicators which become more accurate over time.
- Daily bounce rates and form started abandonment rates become available 48 hours after forms were first opened (request time).
- The start date and end date filters are based on the transaction request time (start time).
- Analytics exclude collaboration job, task, and Manager server monitoring transactions.

All transaction trend charts display a daily value on the x-axis. If the selected date range is too wide, Manager plots the trend graph for every two or more days to fit this date range in the given screen size, so we advise you to restrict date filtering to shorter ranges. However, you can see an exact daily figure for volume, percentage, rate, and time, by hovering over specific points plotted on any of the graphs.

NOTE

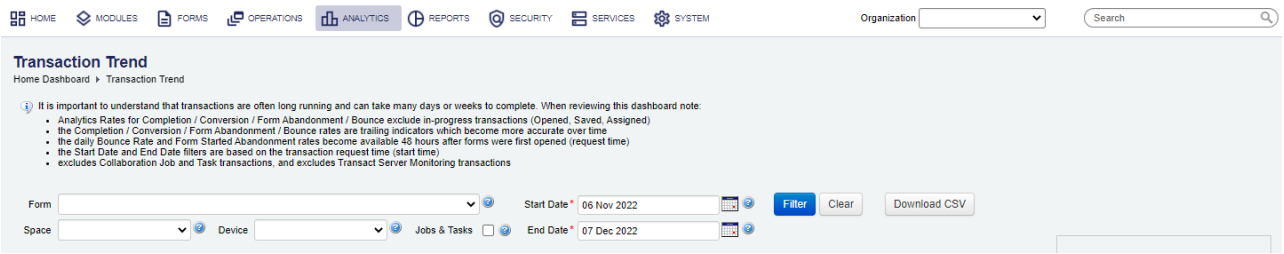
You can also use Journey Analytics to analyze transaction trends as it provides more options to drill down into user transactions and better visualization. You will, however, need the Journey Analytics module installed in your Manager environment.

NOTE

Manager and Journey Analytics calculate transaction trends differently depending on multiple factors, such as analytics implementation, and are beyond the scope of this documentation.

To view the transaction trend:

1. Select Analytics > Transaction Trend.



2. Select an organization from the Organization dropdown list to retrieve statistics relevant to this organization only. If not selected, statistics for all organizations is shown.
3. Select a form from the Form dropdown list to include only the transactions related to that form. If left blank, the transactions related to all forms are included in the chart.
4. Select a form space from the Space dropdown list to include only the transactions of that form space.
5. Select a device from the Device dropdown list to include transactions accessed by users from the following devices:
   - Desktop
   - Phone
   - Tablet
6. Select the Jobs & Tasks checkbox to include collaboration job and task transactions. Otherwise, they are excluded. | 18.05 This feature was introduced in 18.05..
7. Specify the date range of transactions to be included in the chart from the Start Date and the End Date. By default, this is set to a date range starting one year before, up to and including today.

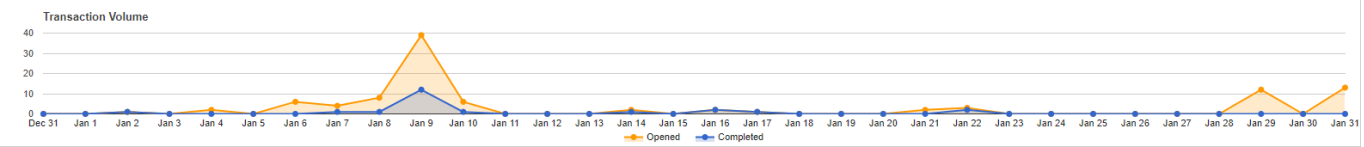
   NOTE

   The maximum date range is 6 months only.

8. Click Filter to generate the transaction trends based on the selection criteria and review them as shown below.

Transaction Volume

This chart displays the number of Opened (both started and bounced) and Completed transactions per day. Transactions completed on a later date are counted against the day they were started. Transactions not completed on the same day as they were started, appear against their start date and not their completion date. Collaboration job and tasks are not included.



The opened transactions are plotted in yellow and the completed transactions are plotted in blue.

Completion Rate (%)

This chart provides the percentage completed transactions divided by the number of started form transactions excluding bounced transactions as they are typically form visits where the user did not start filling in the form.

Conversion Rate (%)

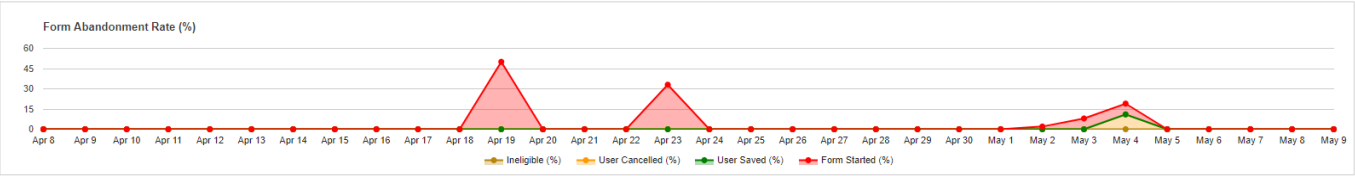
This chart provides the percentage completed transactions divided by the number of opened transactions or visits (includes bounces). Form bounces or visits are included in the conversion rate. | 17.10 This feature was introduced in 17.10.

Form Abandonment Rate (%)

This chart displays the percentage breakdown of the different types of user abandonment (never completed or submitted). These types of form abandonment include:

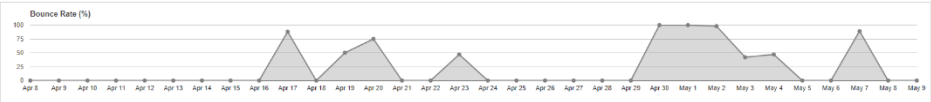
- Ineligible - the form determined that the user is not eligible to complete the transaction, and the transaction was terminated.
- User Cancelled - the user explicitly clicked on the Cancel form button. Note not all form designs include this button. This information becomes available immediately.

- User Saved - the user has explicitly saved the form and then closed it but has never returned to resume the form. This type of abandonment is determined over time, and can take up to 30 days depending upon the form's configuration.
- Form Started - the user has started interacting with the form (entering data, clicking on buttons or navigating around) but they never perform an explicit save or submit. Form Start abandonment is determined 24 hours after the transaction has been opened.



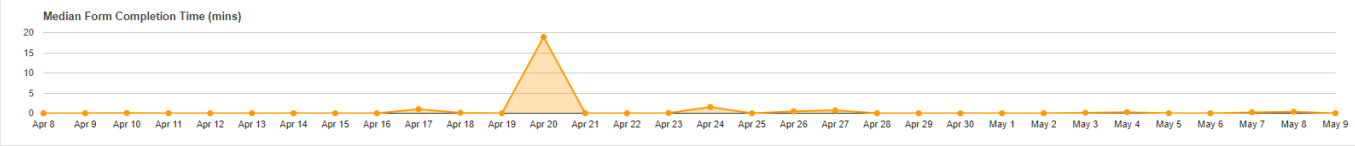
Bounce Rate (%)

This chart displays the percentage of bounced[1] transactions divided by opened (rendered) transactions per day.



Median Form Completion Time (mins)

This chart displays the median form completion time in minutes or the cumulative amount of time it took users to complete the transaction, potentially across multiple form sessions.



NOTE

The averages of the above stats are displayed on the right hand side of the screen.

Click Download CSV to export the displayed data as a CSV file.

[1]Bounce is when a user views a form but never starts it and simply closes the window immediately or navigates away.

NOTE

Manager prior to version 17.10, uses Export Excel Data link instead of the Download CSV button.

Next, learn about all Manager analytics.

Form A/B Testing Overview

A/B testing, also known as split testing, is a process of comparing two versions of a form to find out which one performs better. Two variants of the same form, A and B, are shown to similar visitors and the one that gives a better conversion rate will only be used. An A/B test is considered industry best practice for optimizing a goal based digital experience. The practice arose in digital publishing where, for example, two different versions of an article headline are published to see which one of them gets the most click-throughs. Instead of two headlines, Manager allows a financial institution to run two different versions of the same onboarding journey at the same time to see which performs better. A/B testing requires multiple capabilities to be successful, all of which have first-class support by the Temenos Journey Manager platform:

- The capability to collect detailed analytics in order to understand user behavior and abandonment (pain points) in an onboarding journey
- The capability to make rapid changes to the onboarding data collection flow (the applicant facing forms), whilst maintaining a common data model
- The capability to rapidly (and automatically) test and deploy new versions of the application journey
- The capability for two or more versions of the same application journey to be deployed in a production system at the same time (the A and B versions)
- The capability for a single URL to be randomly directed to either the A or B version, thus partitioning a live audience

Improving conversion rates is often imperative to any form modification. To easily measure the effectiveness of any proposed changes made to a form prior to adopting them, the form A/B testing feature allows for a trial period. After the trial period, it is hoped that a declared clear winner can be implemented as the newly adopted version of the form.

Manager provides the form A/B testing functionality, which you can use to compare versions of a form, usually the new proposed one versus the previous, to test the effectiveness of certain modifications. During this trial period, the modified form is tested in the following way:

- Whenever a user requests this form, Manager randomly chooses a form version from the candidates to display
- All user activities, especially submissions by form version, are captured
- Form version metrics, for example completed transactions, are monitored and analyzed

You can view form A/B testing results as a set of graphs to determine the effectiveness of form changes between the versions.

NOTE

For more extensive form A/B testing, you should use Journey Analytics.

Next, learn how to configure form A/B testing.

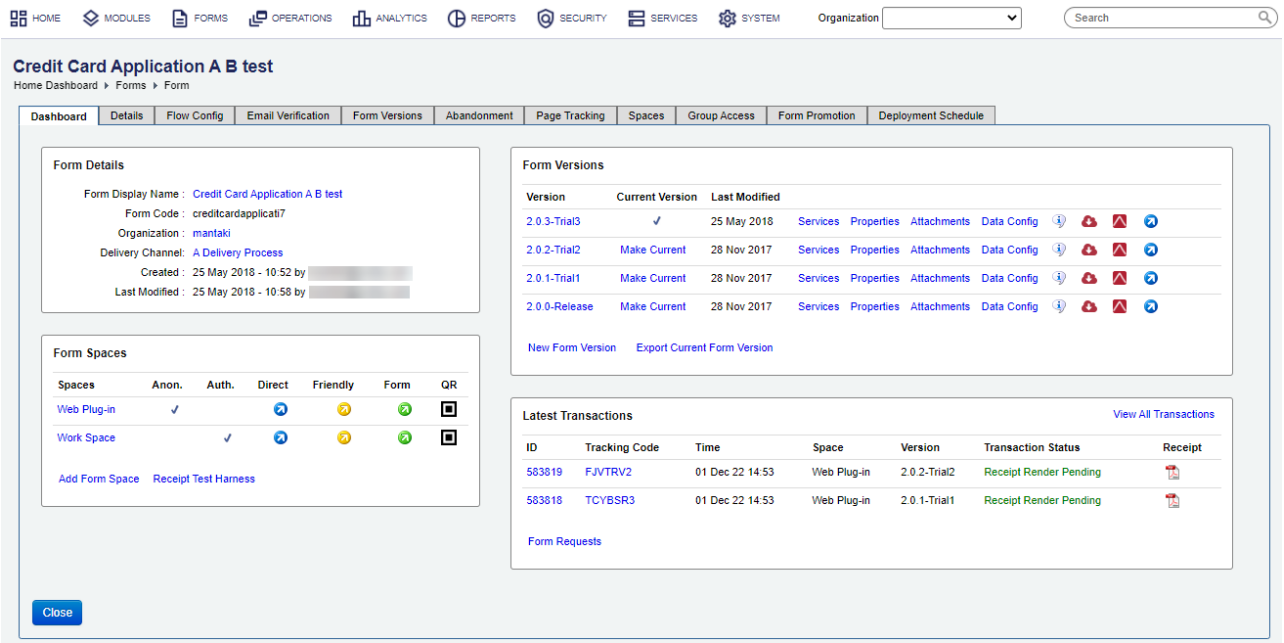
Configure Form A/B Testing

Manager allows you to perform A/B testing of a form, if it uses the A/B Testing Form Version Selector service which captures and visualizes all user's requests to this form, so you can analyze this information and decide which form is most suitable and should be used.

After you have finished with the A/B testing, you need to change the form configuration back to the original one.

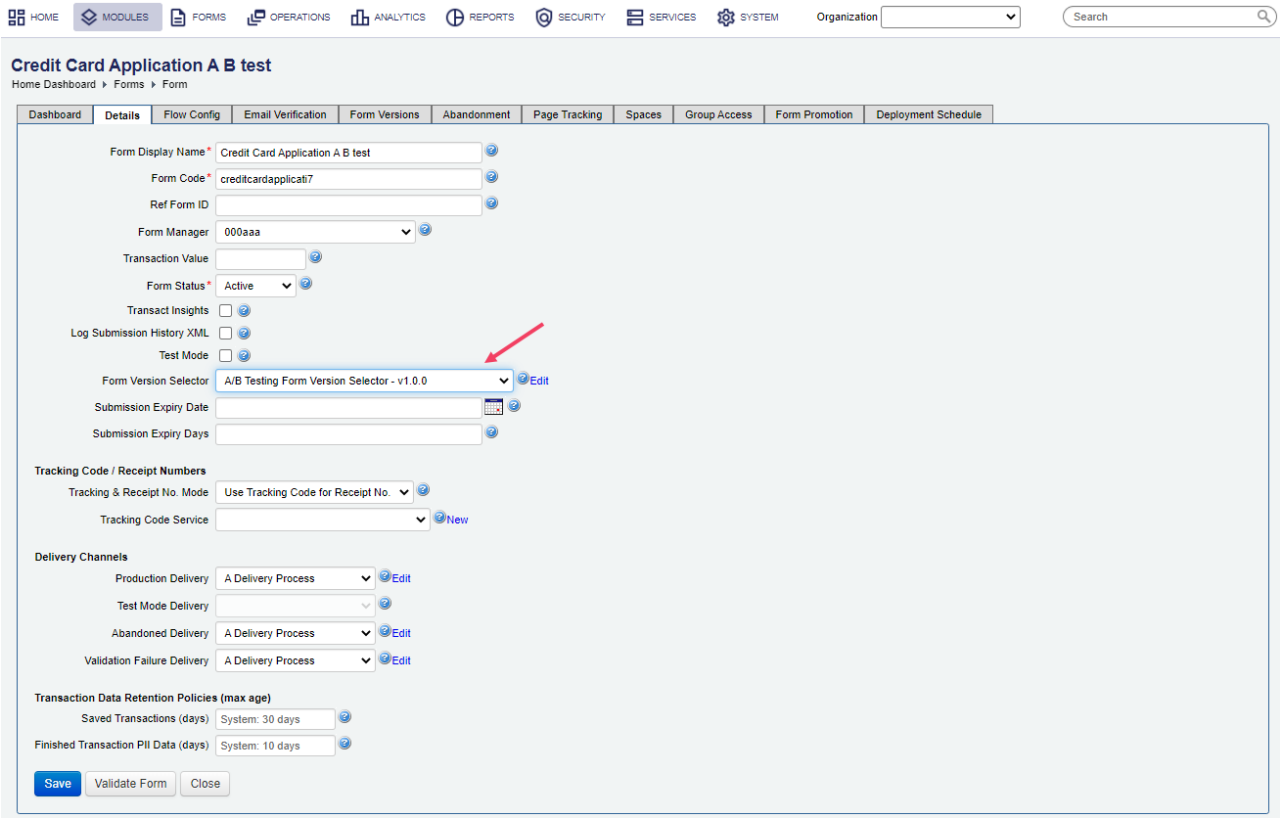
To start the form A/B testing:

1. Select Forms > Forms.
2. Locate a form and click Edit.



   In this example, the form has not yet been configured for A/B testing, so only the current version of the form is offered to users.

3. Click the Details tab.
4. Select the A/B Testing Version Selector - v1 service from the Form Version Selector drop-down list.



5. Click Save to update the changes.

Based on this change, the A/B Testing Form Version Selector will randomly select and display one of the form's versions to form users. To have more control over which versions are tested, learn how to elect versions a form for A/B testing.

To stop the form A/B testing:

1. Select Forms > Forms.
2. Locate your form and click Edit.
3. Select the Details tab.
4. Select the blank option from the Form Version Selector dropdown list.
5. Click Save to update the changes. Users will now be presented with the current form version.

Next, learn how to view form A/B testing results.

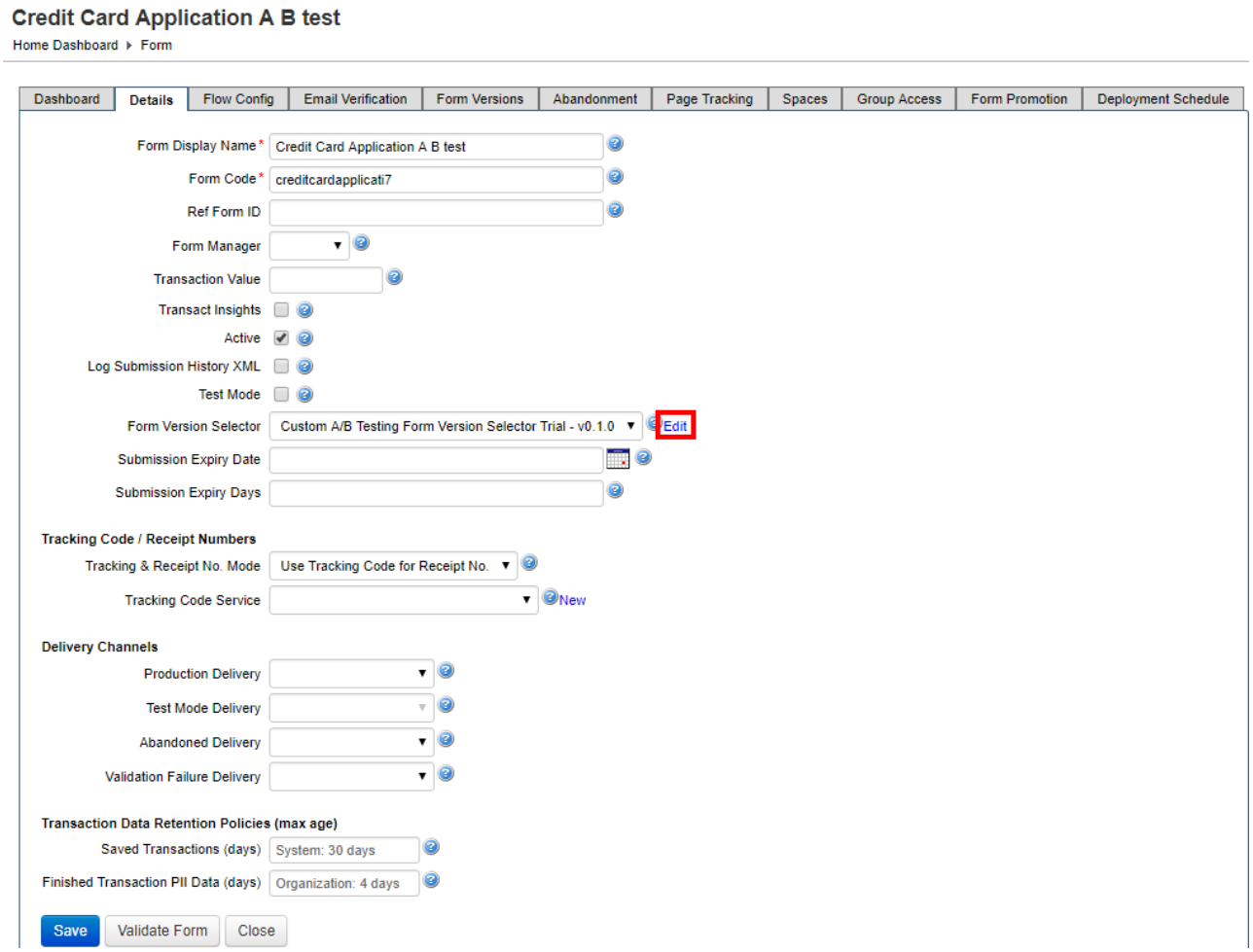
Elect Form A/B Testing

Manager allows you to perform A/B testing of a form. In most cases, the form A/B testing is for a new version of a form versus its previous version. To accomplish this, you must *elect* the latest two versions of the form to include in the form A/B testing.

> NOTE
>
> Form A/B testing is handled by the A/B Testing Form Version Selector service, which is a core global service that must never be modified. You should copy this service before continuing.

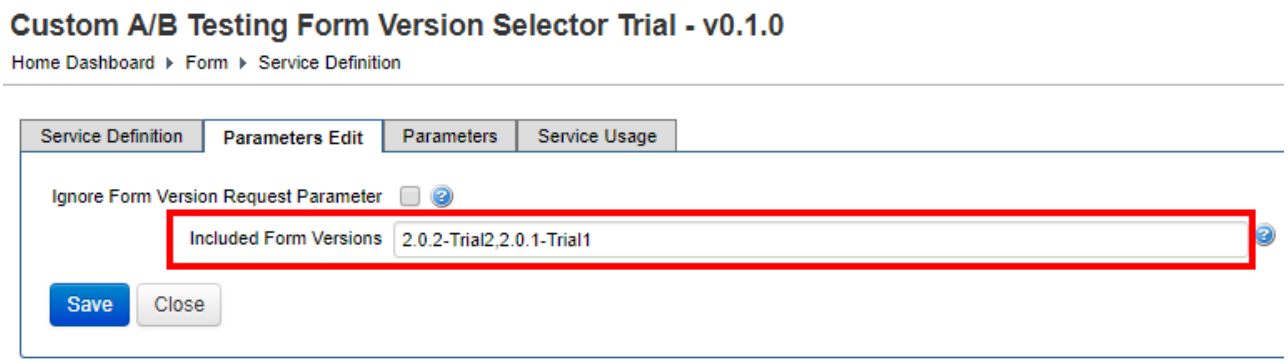
To nominate specific form versions to be included in the form A/B testing:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select the Details tab.
4. Select the copied service from Form Version Selector dropdown list. In this example, the copied service is named Custom A/B Testing Form Version Selector Trial.



5. Click Save to update the changes.
6. Click Edit to elect only specific versions of your form.
7. Enter the nominated form versions, separated by commas (and without spaces), in the Included Form Versions field.

> NOTE
>
> If Included Form Versions is left blank, all form versions are included in the form A/B testing. This is the default option.



8. Click Save to update the changes. From now on, whenever a user selects this form, the Custom A/B Testing Form Version Selector service will randomly choose either 2.0.2-Trial2 or 2.0.1-Trial1 form version to be rendered to the users.

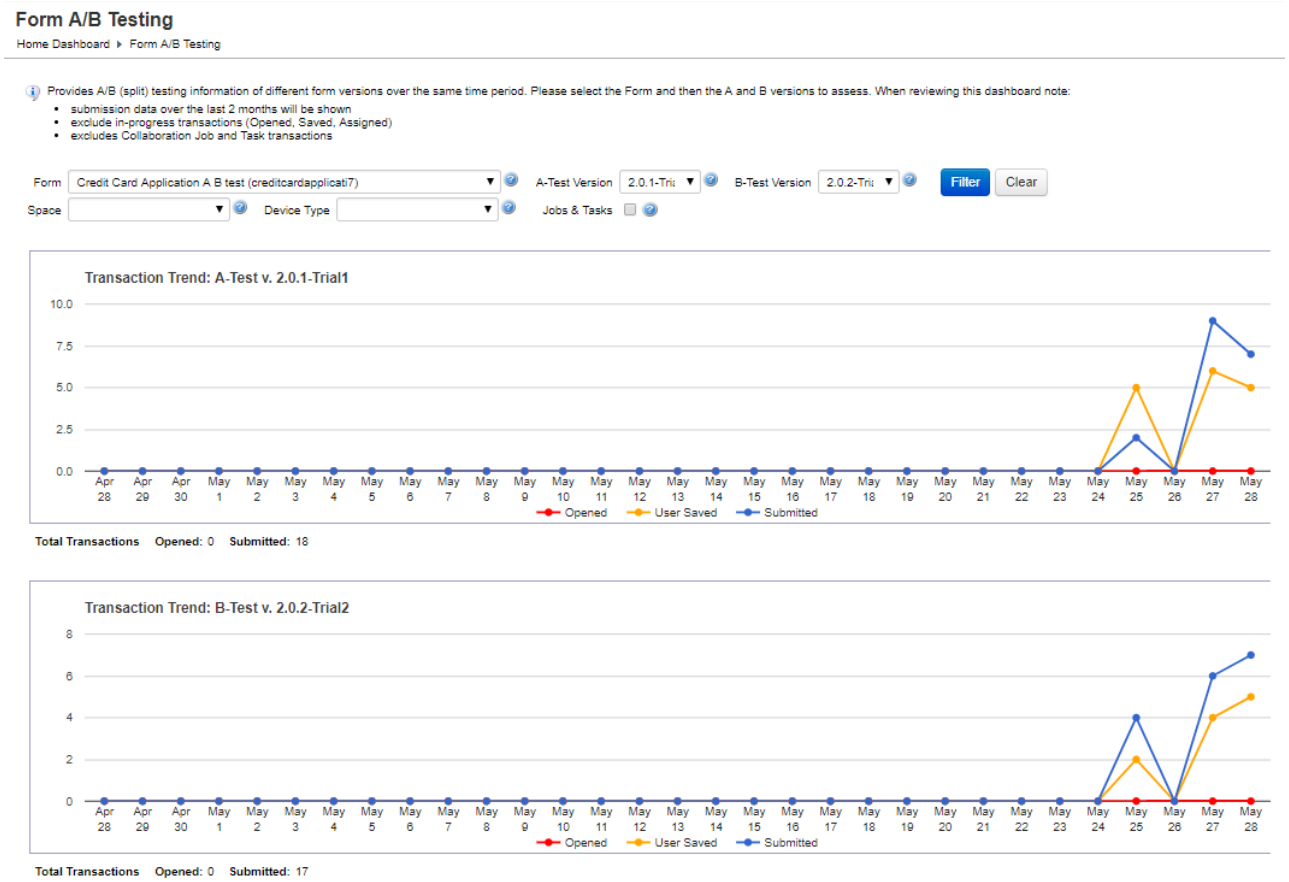
Next, learn how to view form A/B testing results.

Analyze Form A/B Testing

Manager provides out of the box analytics for A/B testing, which allow you to compare any two form versions, from the elected group of form versions for a duration of a form A/B testing trial. You can configure the form A/B testing at any time during the trial period.

To view the form A/B testing chart:

1. Select Analytics > Form A/B Testing.



2. Select an organization from the Organization dropdown list to retrieve statistics relevant to this organization only. If not selected, statistics for all organizations is shown.
3. Select a form from the Form dropdown.
4. Select a form version to represent the A-test results from the A-Test Version dropdown list.
5. Select a form version to represent B-test results from the B-Test Version dropdown list.
6. Select a form space from Space dropdown to include form transactions of this form space only.
7. Select a device from the Device Type dropdown to include only the transactions by users who accessed their form using this device.

A device type can be one of the following:

- Desktop
- Phone
- Tablet

8. Select the Jobs & Tasks checkbox to include collaboration job and task transactions, otherwise they are excluded.

6. Click Filter to generate the report. The comparison for each form A/B test is plotted as Opened, User Saved, and Submitted.

Next, learn about form A/B Testing analytics and reporting in Journey Analytics.

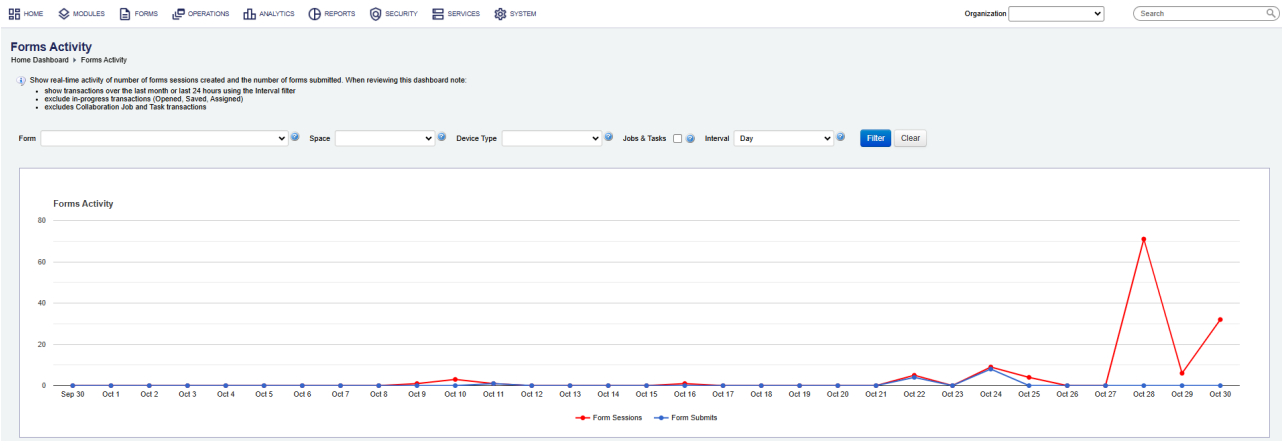
Analyze Forms Activity

Manager generates this chart to visualize the real-time activity of the number of form sessions created and the number of forms submitted. The following assumptions are applied:

- The chart shows transactions over the last month or last 24 hours depending on the filter selection.
- The chart excludes in-progress transactions, such as opened, saved, assigned, collaboration job, and task transactions.

To view the forms activity chart:

1. Select Analytics > Forms Activity.



2. Select an organization from the Organization dropdown list to retrieve statistics relevant to this organization only. If not selected, statistics for all organizations is shown.
3. Select a form from the Form dropdown list to include the transactions related to that form only. If left blank, the transactions related to all forms are included in the chart.
4. Select a form space from the Space dropdown list to include the transactions of that form space only.
5. Select a device from the Device dropdown list to include transactions accessed by users from the following devices:
   - Desktop
   - Phone
   - Tablet
6. Select the Jobs & Tasks checkbox to include collaboration job and task transactions. Otherwise, they are excluded. | 18.05 This feature was introduced in 18.05..
7. Select an activity interval from the Interval dropdown list, which can be one of the following:

- Day - to see the daily form activity for the last month. This is the default.
- Hour - to see the hourly form activity for the last 24 hours.

8. Click Filter to generate the chart based on the selection criteria.

Next, learn about referrer submissions.

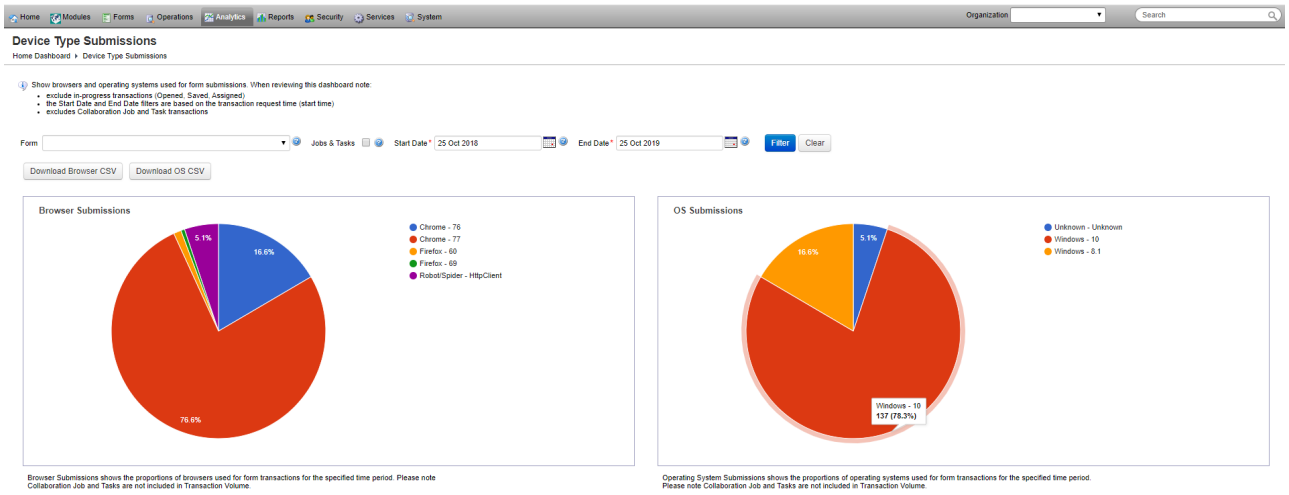
Analyze Device Type Submissions

Manager generates charts to visualize the proportion of browsers and operating systems used for form transactions for a specified time period. The following assumptions are applied:

- Charts exclude in-progress transactions, such as opened, saved, assigned, collaboration job and task transactions.
- The Start Date and End Date filters are based on the transaction request time (start time).

To view the device types submissions charts:

1. Select Analytics > Device Types Submissions.



2. Select an organization from the Organization dropdown list to retrieve statistics relevant to this organization only. If not selected, statistics for all organizations is shown.
3. Select a form from the Form dropdown list to include the submissions of that form only. If left blank, submissions of all forms are included in the charts.
4. Select the Jobs & Tasks checkbox to include collaboration job and task transactions only. Otherwise, they are excluded. | 18.05 This feature was introduced in 18.05..
5. Specify the date range of submissions to be included in the chart from the Start Date and the End Date. By default, this is set to a date range starting one year before up to and including today.
6. Click Filter to generate the charts based on the selection criteria.
7. Click Download Browser CSV to export the browser submissions data as a CSV file.
8. Click Download OS CSV to export the operating system submissions data as a CSV file.

NOTE

Manager prior to v18.05 has two separate charts: browser submissions and OS submissions.

Next, learn about reader versions.

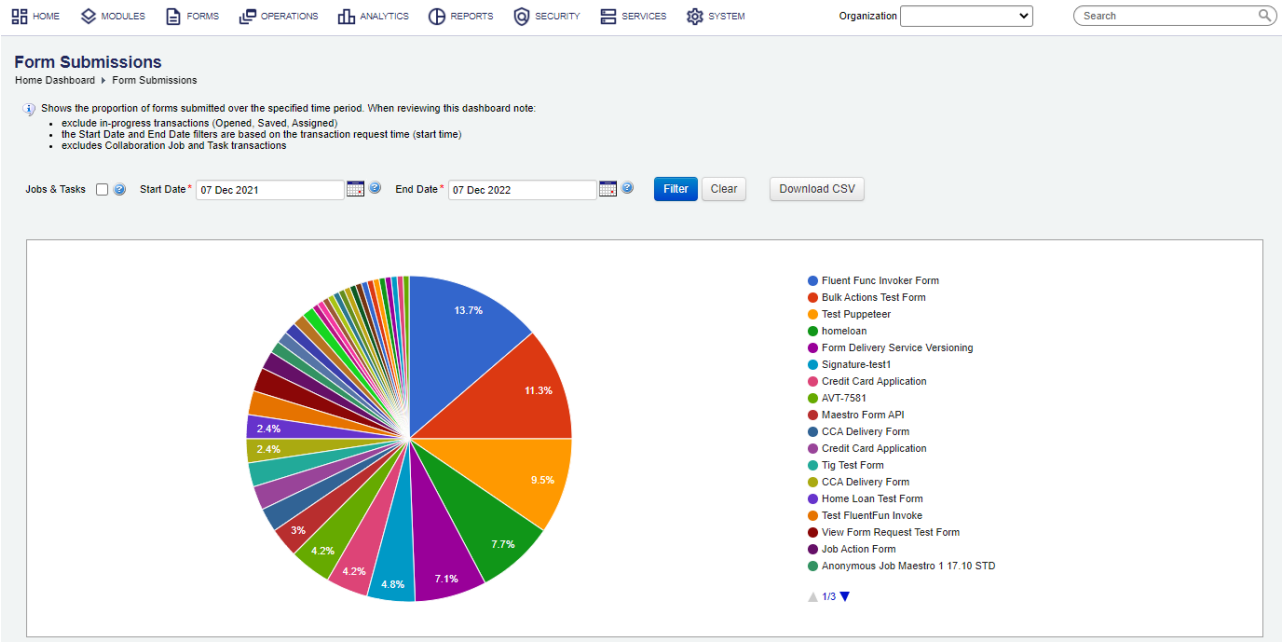
Analyze Form Submissions

Manager generates this chart to visualize the most active forms and the proportion of forms submitted over the specified time period. The following assumptions are applied:

- Charts exclude in-progress transactions, such as opened, saved, assigned, collaboration job and task transactions
- The Start Date and End Date filters are based on the transaction request time (start time)

To view the form submissions chart:

1. Select Analytics > Form Submissions.



2. Select an organization from the Organization dropdown list to retrieve statistics relevant to this organization only. If not selected, statistics for all organizations is shown.
3. Select the Jobs & Tasks checkbox to include collaboration job and task transactions only. Otherwise, they are excluded. | 18.05 This feature was introduced in 18.05.
4. Specify the date range of submissions to be included in the chart from the Start Date and the End Date. By default, this is set to a date range starting one year before up to and including today.
5. Click Filter to generate the chart based on the selection criteria.

NOTE

You can see the form name and the total number of submissions made for each form by hovering the mouse over the chart slice.

NOTE

The chart legend is shown on the right-hand side of the chart with forms ordered according to highest percentage of submissions.

6. Click Download CSV to export the displayed data as a CSV file.

NOTE

Manager prior to version 17.10, uses Export Excel Data link instead of the Download CSV button.

7. Click up and down arrows to scroll the list of the forms visualized on this pie chart

Next, learn about device type submissions.

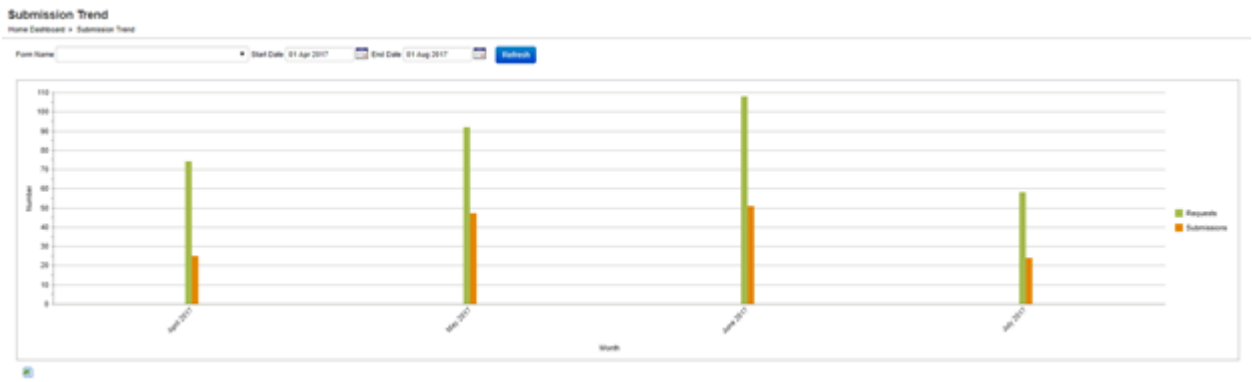
Analyze Submission Trend

This functionality has been removed in Manager v18.05. Use transaction trend instead.

Manager generates charts to visualize the number of form requests compared to the number of completed submissions for each month during the selected date range.

To view the submission trend chart:

1. Select Analytics > Submission Trend.



2. Select a form from the Form Name dropdown list to see submission trends for a specific form only.
3. Specify the date range of submissions to be included in the chart from the Start Date and the End Date. By default, this is set to a date range starting one year before up to and including today.
4. Click Refresh to generate the charts based on the selection criteria.
5. Click to export the displayed data as an XLS file.

Next, learn about transaction trends.

Analyze Referer Submissions

Manager allows you to see URLs of the web pages where form users clicked a link that sent them to your application form so they could submit the form. For each form, Manager generates a pie chart diagram that visualizes Referer URLs[1] of web pages and a number of times each Refer URL appeared in returned set of submission data for that page.

When a user submits a form, the request that is made to Manager will have a HTTP Referer parameter. This is the page the user came from when they clicked the link to the form. The referrer url is set when a submission is rendered, this is stored on the submission object. The referer submission report does a count of each unique referer URL for a particular form. This helps clients who might have multiple places users can access the link to their form from for analytical purposes.

This information is especially important when users can access a link to an application form multiple places, such as pages or even websites.

> NOTE
>
> For more user behavior analysis, we recommend using Journey Analytics.

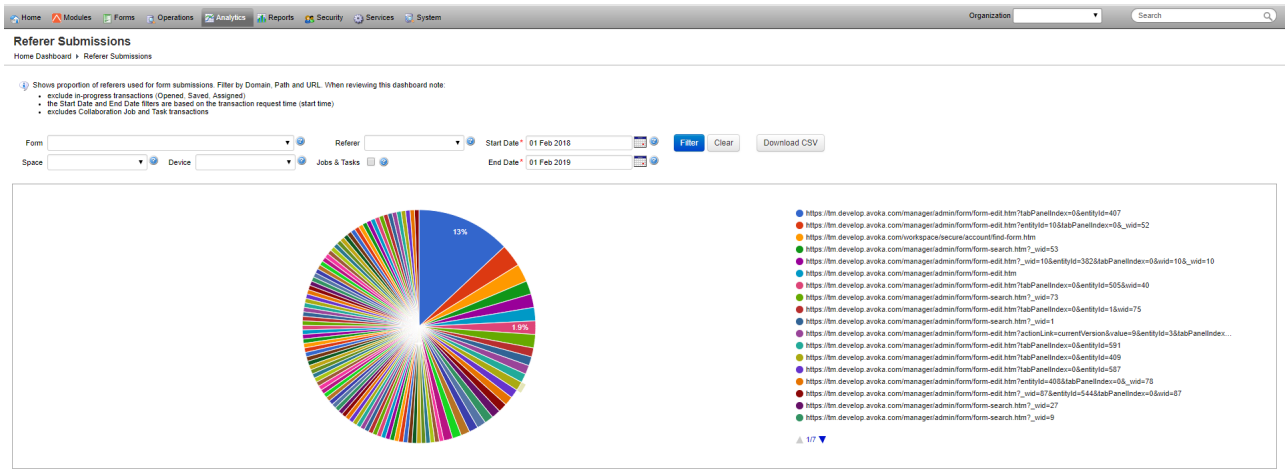
Manager applies the following assumptions when generating a report:

- Charts exclude in-progress transactions, such as opened, saved, assigned, collaboration job and task transactions
- The Start Date and End Date filters are based on the transaction request time (start time)

To view the referer submissions chart:

[1]The Referer HTTP request header contains an absolute or partial address of the page that makes the request. The Referer header allows a server to identify a page where people are visiting it from. This data can be used for analytics, logging, optimized caching, and more.

1. Select Analytics > Referer Submissions.



2. Select an organization from the Organization dropdown list to retrieve statistics relevant to this organization only. If not selected, statistics for all organizations is shown.
3. Select a form from the Form dropdown list to see referers[1] used for this form only. If left blank, referers and their counts for all forms are included in the charts.
4. Select a form space from the Space dropdown list to include transactions of that form space only.
5. Select a device from the Device dropdown list to include transactions accessed by users from the following devices:

   A device type can be one of the following:

   - Desktop
   - Phone
   - Tablet

6. Select the Jobs & Tasks checkbox to include collaboration job and task transactions only. Otherwise, they are excluded. | 18.05 This feature was introduced in 18.05..
7. Specify the date range of submissions to be included in the chart from the Start Date and the End Date. By default, this is set to a date range starting one year before up to and including today.

---

[1]The Referer HTTP request header contains an absolute or partial address of the page that makes the request. The Referer header allows a server to identify a page where people are visiting it from. This data can be used for analytics, logging, optimized caching, and more.

8. Click Filter to generate the chart based on the selection criteria.
9. Click Download CSV to export the displayed data as a CSV file. The CSV file has the `submission_referer,referer_count` format, with records ordered by the referer_count in descending order. The sample report is shown below:

```
submission_referer,referer_count
https://my-website-1.com,21
https://my-website-2.com,17
https://my-website-3.com,1
```

Next, learn about form submissions.

Analyze Browser Submissions

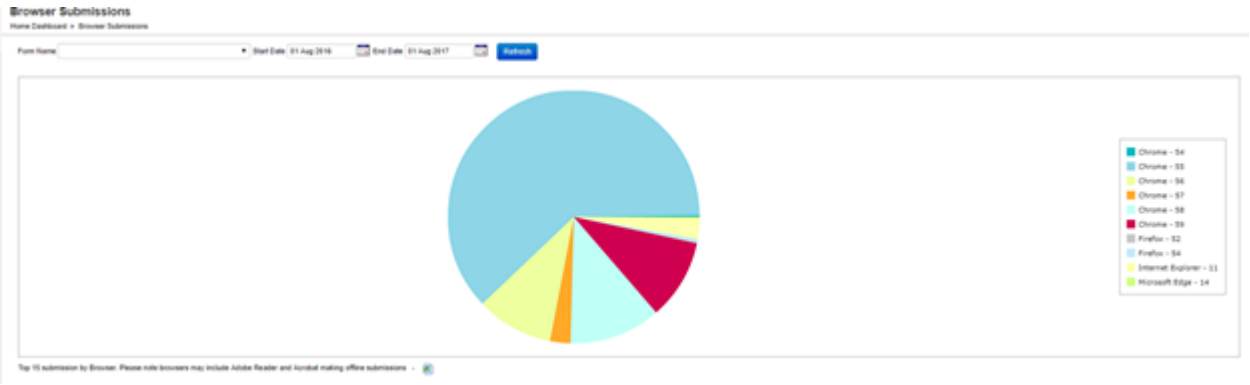
NOTE

This functionality was removed in Manager v18.05. Use transaction trend instead.

Manager generates browser submissions charts to visualize the number of the internet browsers most used by form users.

To view Browser Submissions chart:

1. Select Analytics > Browser Submissions.



2. Select a form from the Form Name dropdown list to see submission trends for a specific form only.
3. Specify the date range of submissions to be included in the chart from the Start Date and the End Date. By default, this is set to a date range starting one year before up to and including today.
4. Click Refresh to generate the charts based on the selection criteria.
5. Click to export the displayed data as an XLS file.

Next, learn about transaction trends.

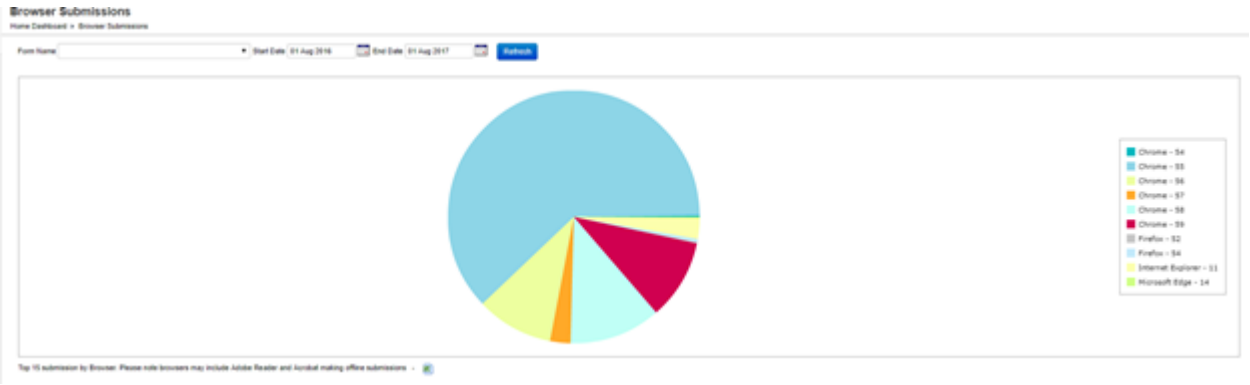
Analyze OS Submissions

This functionality has been removed in Manager v18.05. Use device type submissions instead.

Manager generates charts to visualize the operating systems most used by form users.

To view the operating system submissions chart:

1. Select Analytics > Operating Systems Submissions.



2. Select a form from the Form Name dropdown list to see submission trends for a specific form only.
3. Specify the date range of submissions to be included in the chart from the Start Date and the End Date. By default, this is set to a date range starting one year before up to and including today.
4. Click Refresh to generate the charts based on the selection criteria.
5. Click to export the displayed data as an XLS file.

Next, learn about device type submissions.

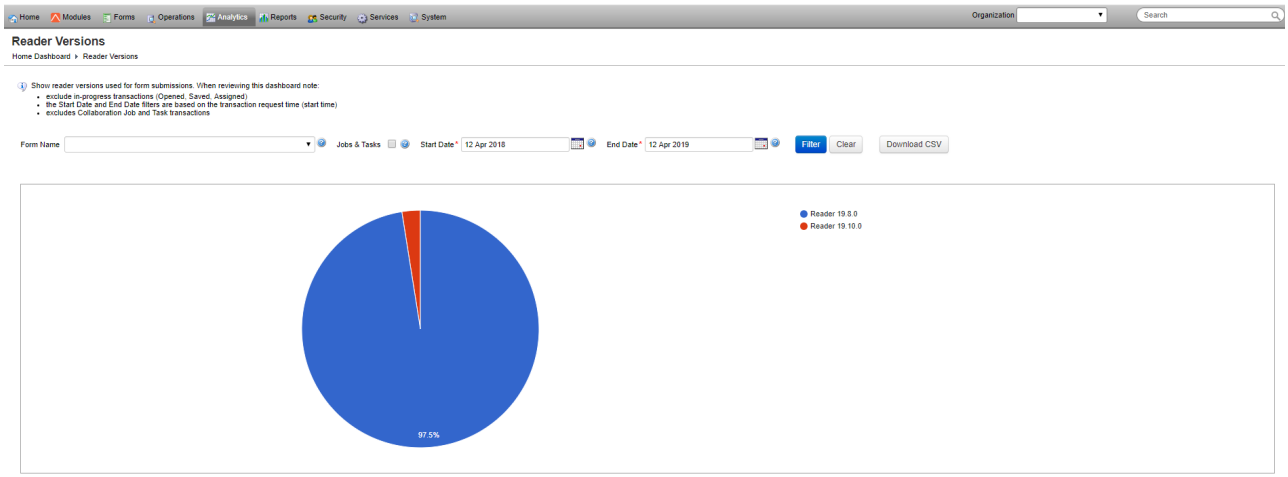
Analyze Reader Versions

Manager generates this chart to visualize a breakdown of the most common Adobe Reader versions used to submit forms. The following assumptions are applied:

- Charts exclude in-progress transactions, such as opened, saved, assigned, collaboration job and task transactions.
- The Start Date and End Date filters are based on the transaction request time (start time).

To view the form submissions chart:

1. Select Analytics > Reader Versions.

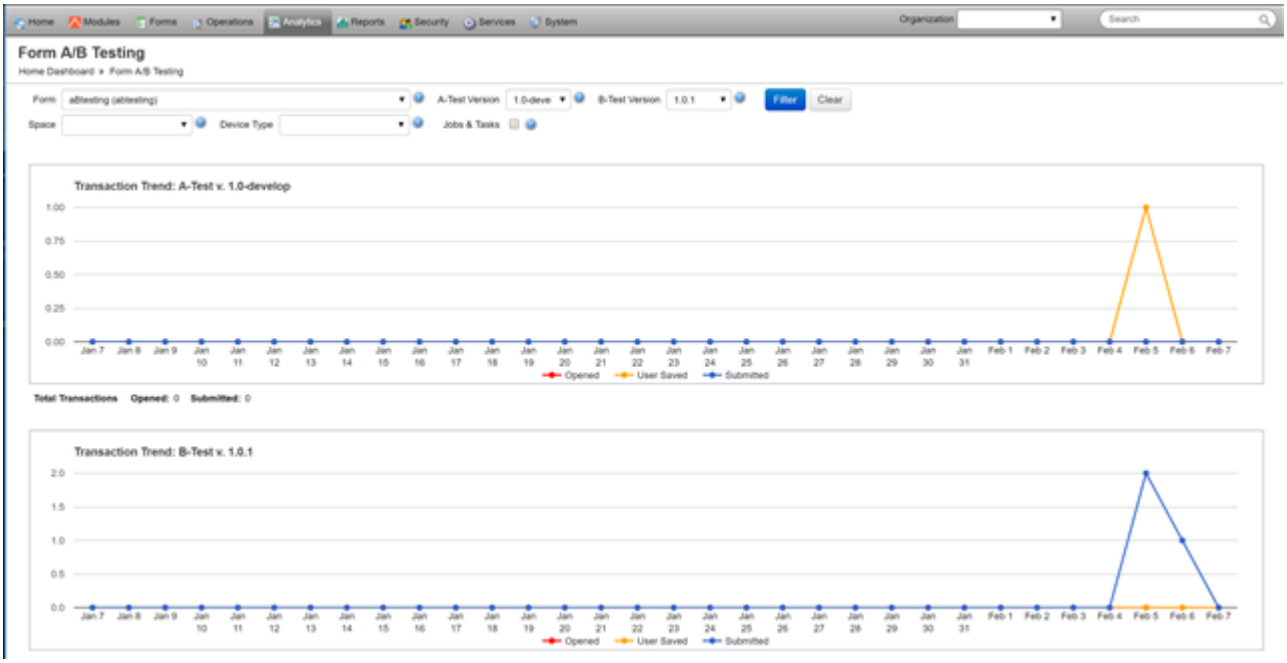


2. Select an organization from the Organization dropdown list to retrieve statistics relevant to this organization only. If not selected, statistics for all organizations is shown.
3. Select a form from the Form dropdown list to include the submissions of that form only. If left blank, submissions of all forms are included in the charts.
4. Select the Jobs & Tasks checkbox to include collaboration job and task transactions only. Otherwise, they are excluded. | 18.05 This feature was introduced in 18.05..
5. Specify the date range of submissions to be included in the chart from the Start Date and the End Date. By default, this is set to a date range starting one year before up to and including today.
6. Click Filter to generate the charts based on the selection criteria.
7. Click Download CSV to export the displayed data as a CSV file.

Next, learn about transaction trends.

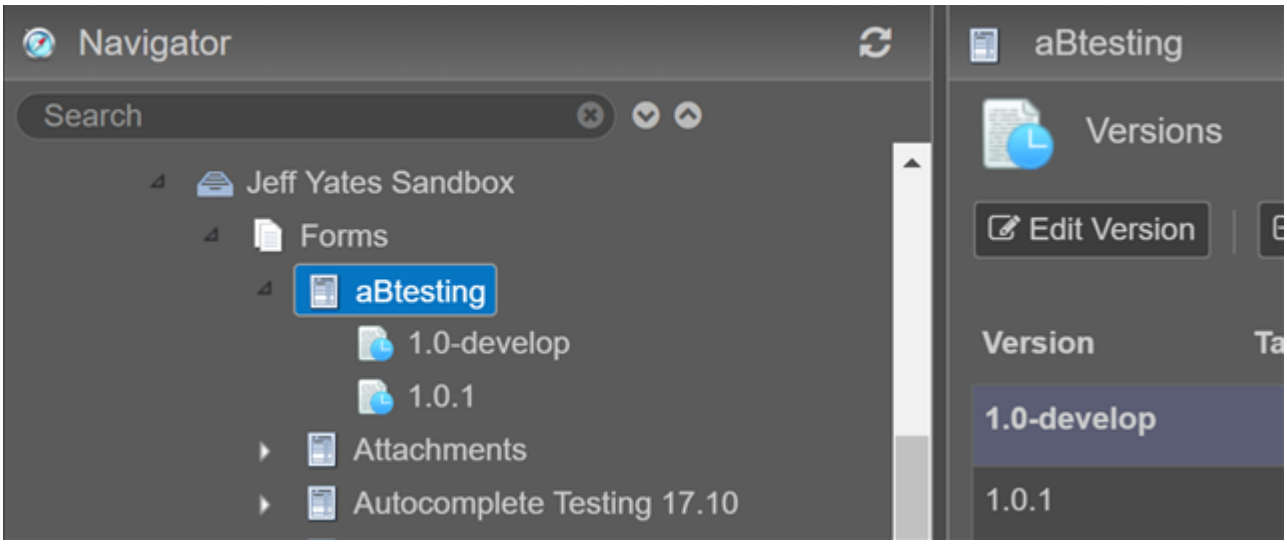
Maestro and Composer Forms A/B Testing

First thing to do is to determine a need to do A/B Testing at all. Avoka Insights does an excellent job of highlighting problems in the User Experience that a change in Form design may address. Or, more proactively, investigating a new Form feature can involve A/B Testing, to determine that the new additions won't stop end users from submitting.



Maestro

To test multiple versions of a Form, the versions must of course first be created. As an example, a Form named aBtesting was made. It is a default Maguire Template form, with two Text Fields added.



In this example, the Text Fields are arranged horizontally in the first version, and vertically in the second. This is similar to A/B Testings intended use case how does a possibly small change to a form impact the user experience, as seen in completion rates.

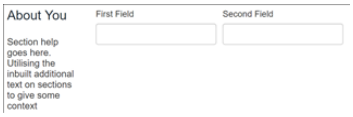


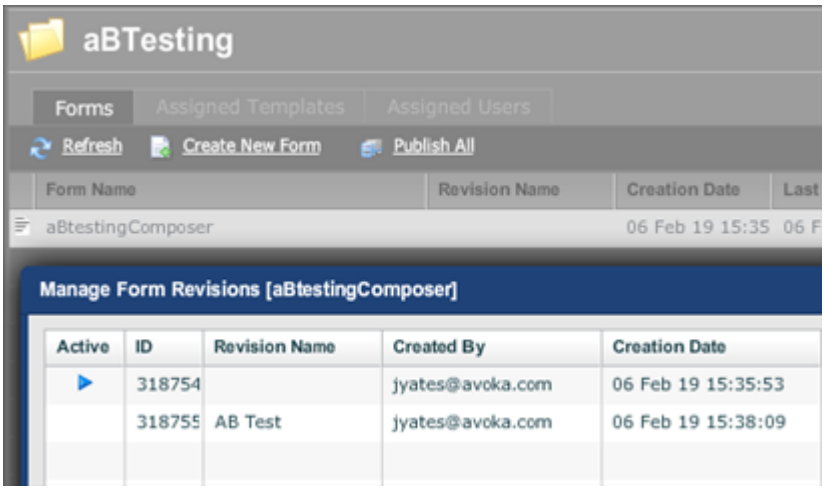
Figure 1 - the "A" design

Figure 2- the "B" design

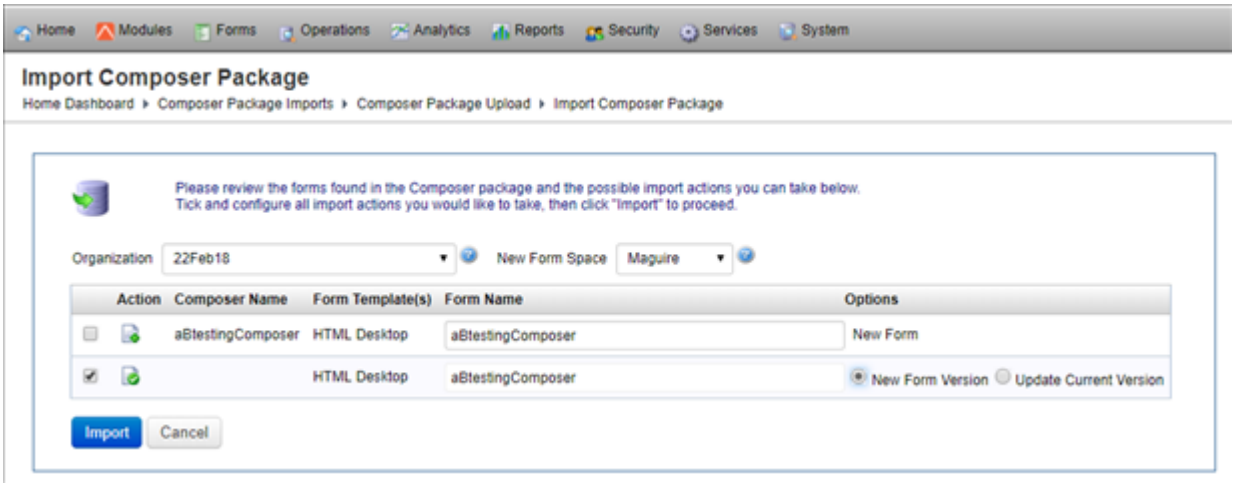
Build and deploy both versions to Manager as you normally would.

Composer

The same concepts apply when using Composer as the form designer. Build the first design, the A version, and publish it to a ZIP file. Then use Composers Form Revisions feature to create the B version. Publish the B version to a ZIP file as well.

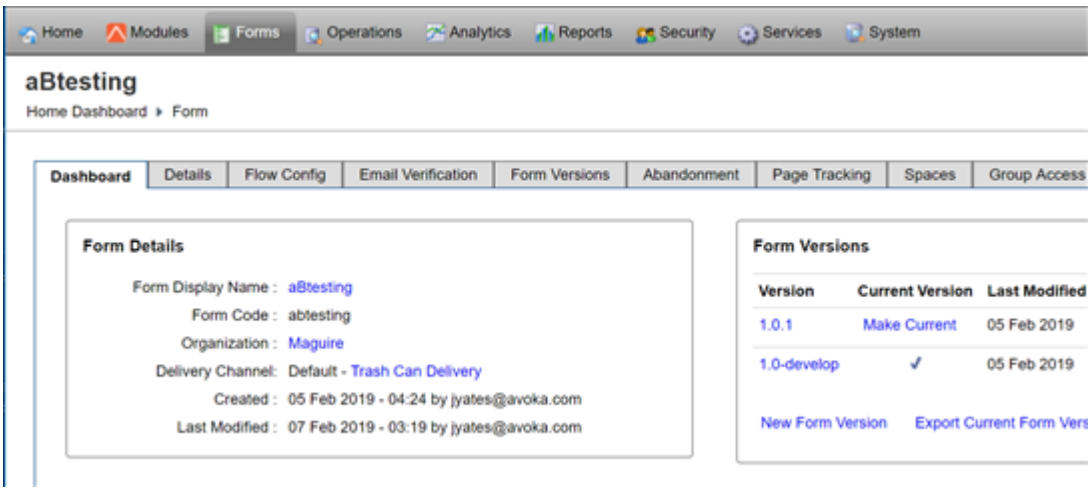


Import both ZIP files to Manager using the Import Composer Package feature. You will create a new form for the first ZIP file imported, and a new version of that form when you import the second ZIP file.



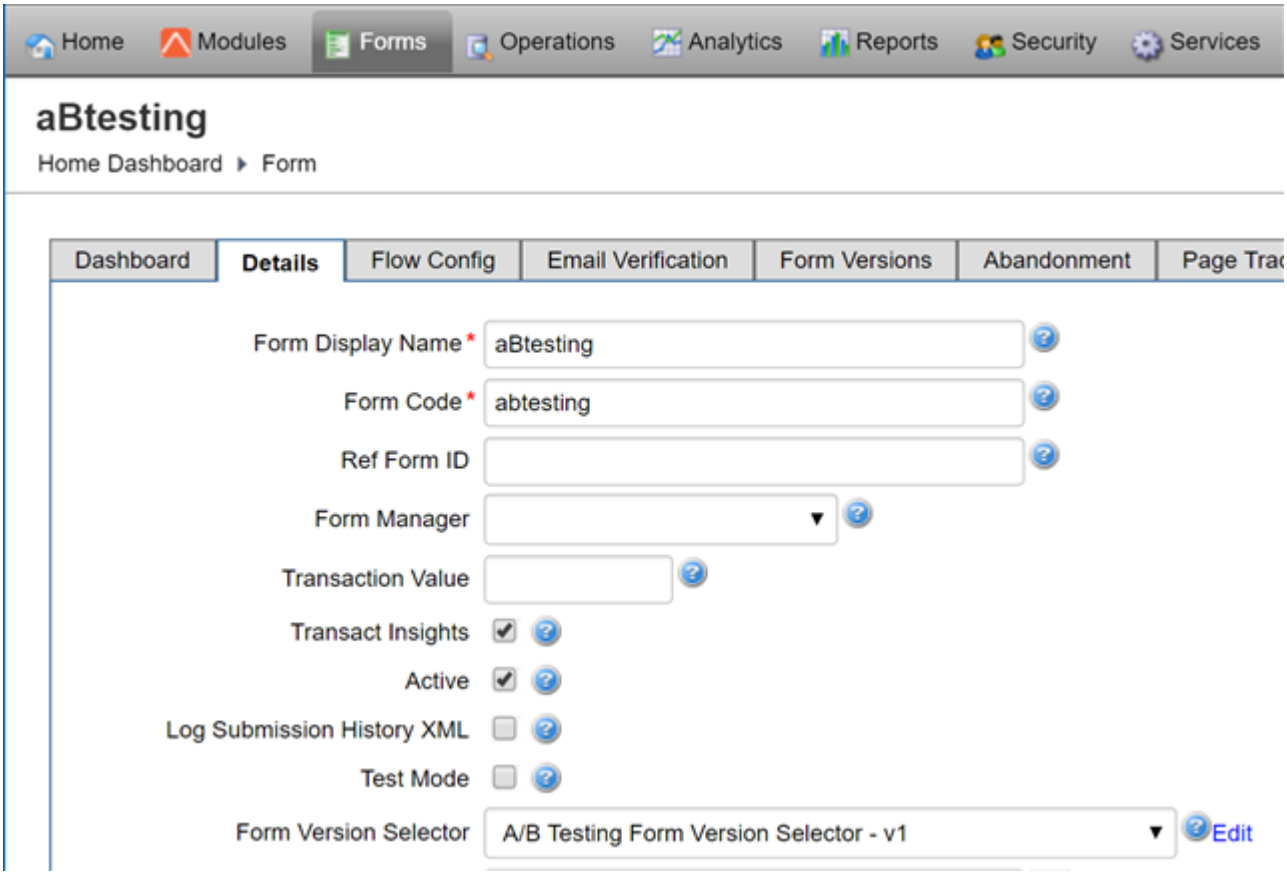
Manager

Once deployed to Transact, the Form dashboard would look like shown below .

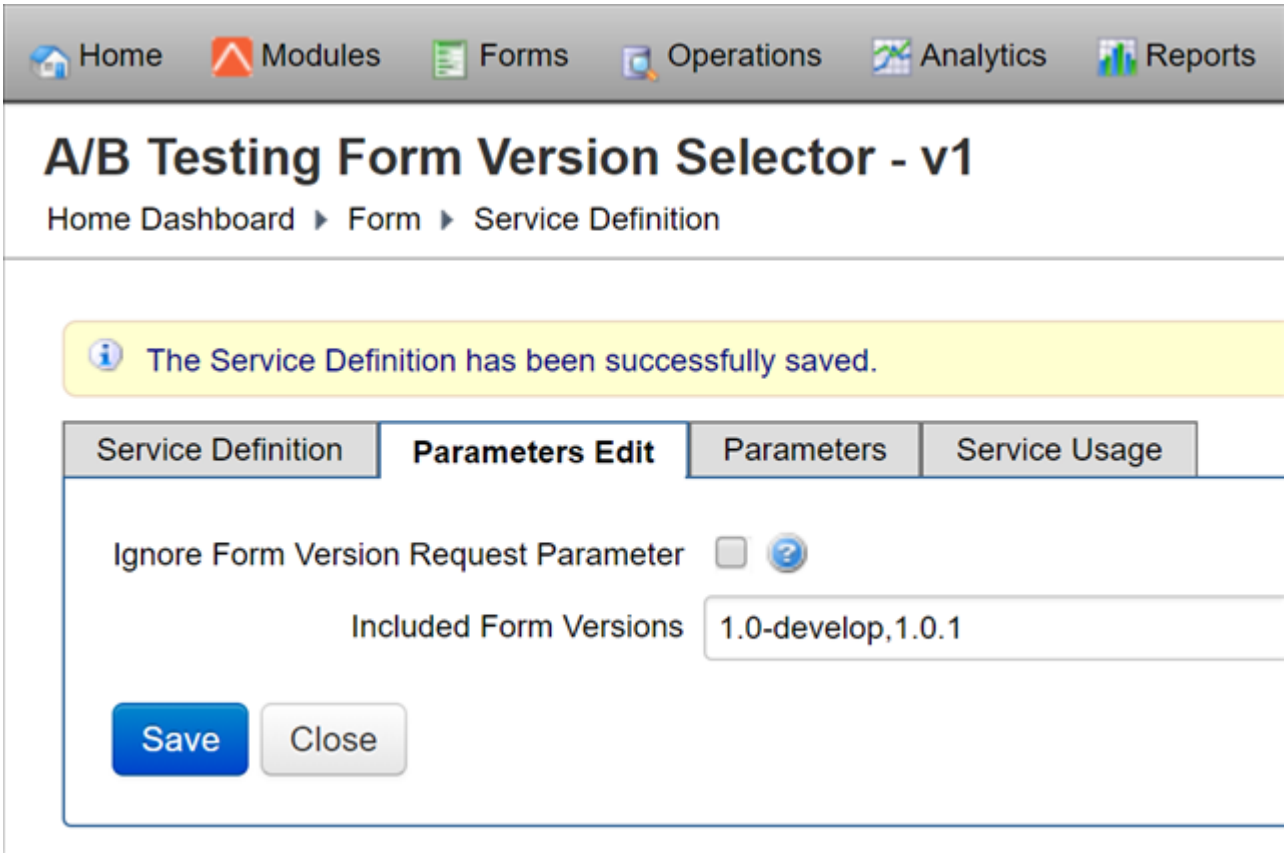


This shows that both Form versions are deployed. For A/B testing, it does not matter which version is marked as Current Version. In fact, neither needs to be current. The Form Version Selector Service will chose which version to render.

The next step is to then add a Form Version Selector Service to the Form. Switch to the Details tab and add one.



The Form Version Selector Service shown above is a default service. In practice, a copy of this Service should be made and that copy used for this Form's testing. This needs to be done to allow the Service to be configured with the Form Version numbers specific to this Form. Click the Edit link to setup the Form Version Selector Service.

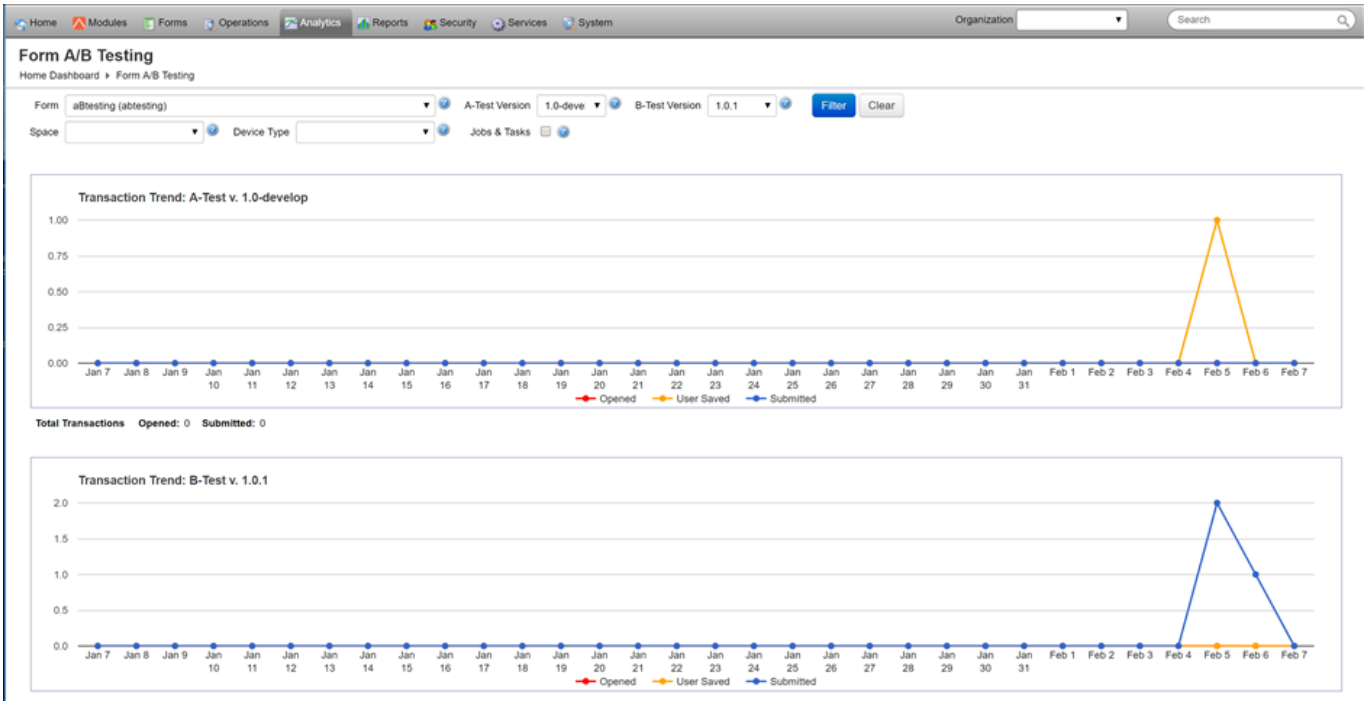


On this screen you add the Form version numbers, separated by commas, that need to be tested. Note that A/B Testing can actually be used against three or more different Form Versions!
The Form Version Selector Service will randomly choose from the supplied list of versions, evenly distributing the selections. So in our example, version 1.0-develop will be selected 50% of the time, and version 1.0 will be selected 50% of the time.
If one of the versions needs to be presented more often than the other, include it in the list more than once. Using 1.0-develop,1.0.1,1.0.1 will result in version 1.0.1 being selected 2/3 of the times.
That's all that's needed. The A/B Testing is now live and functioning. Accessing the Form's base url will trigger the Form Version Selector Service, and it will return a randomly selected version. Normally the base url (like https://training.avoka.com/web-plugin/servlet/SmartForm.html?formCode=abtesting) would return the Form Version marked Current.

Performing the Tests, Reviewing Results

Now that the A/B Test is live and running, let it collect data. There is nothing additional to do to perform the test other than let the system run. After a suitable amount of time (to be determined

by each project team for each form Forms under heavy use may only need a day's data, others may need a week), review the results and turn off the test.



The Analytics menu item on Transact's main menu contains the Form A/B Testing link. This page will show a graph of the relative performance of the entire Form, one graph for each version tested (as shown above). Of course, Insights can also be used to review the results of A/B Testing. And with Insights, you can see differences between the two Form Version's performance on a much more granular level.
After reviewing the results, the A/B Test should be shut off. Go back to the Form Dashboard tab and mark the proper Form Version as Current. Then on the Details tab, clear the Form Version Selector setting.

What is implied above is that this is a production server test. Live Forms being used by end users. So all versions tested this way must go through all of the normal development processes full testing on lower environments, etc. After all, the desired goal is to find a new production Form design, that removes friction from the user experience.

Next, learn about form A/B Testing analytics and reporting in Journey Analytics.

Reports Overview

Manager automatically records licensing data, which includes organizations, forms, transactions, and services, and generates a license usage report for each organization.

Next, learn how to view job licensing report.

Licensing Reports

Manager keeps a record of transactions for licensing purposes. You can view this transaction license usage per each organization as two reports: transaction licensing report and job licensing report.

> NOTE
>
> For licensing reports to work, you need to configure Manager with correct system licensing details, which you obtain from Temenos.

The licensing reports provide a comparison between the number of completed transactions recorded in Manager and Amazon Redshift[1] using Amazon S3. This is useful when you need to confirm that the number of transactions, that you have been billed for, matches the number of transactions recorded in Manager.

There may be a slight variance between the number of transactions recorded in Manager and the number recorded by Amazon Redshift when Manager has not yet published new transactions to Amazon Redshift or when Manager has purged transactions but they remain in Amazon Redshift records.

> NOTE
>
> For more information about your licensing model, refer to your Master Service Agreement (MSA).

Manager comes with a dedicated Transaction Licensing scheduled job to process transaction data daily and generate licensing reports which are sent to our billing servers. You can't see the Transaction Licensing scheduled job in the list of standard scheduled jobs, but you can configure it and run manually from System > System Licensing.

---

[1]Amazon Redshift is an Internet hosting service and data warehouse product which forms part of the larger cloud-computing platform Amazon Web Services. It is built on top of technology from the massive parallel processing (MPP) data-warehouse company ParAccel to handle large scale data sets and database migrations.

Billing Records

Manager allows you to customize tracking of the billable aspects of clients onboarding journeys.
| 24.10 This feature was introduced in 24.10

You can create a bill record with the `BillBuilder` Fluent API to track both transaction and service usage. These records indicate a billable transaction and can be created on demand within any Fluent service. Bill records can be linked together to form a journey (`setJourneyId()`) and are then periodically published out to the Temenos transaction record keeping service.

A simple example below shows how to create a transaction (TXN) record:

```
Bill bill = new BillBuilder()
   .setBillType("TXN")
   .setClientCode("client")
   .setSubmissionId(123)
   .setFormCode("BillFORM")
   .setFormVersion("1.0")
   .setLicenseKey("LicenseKey")
   .setBillingScore("1.5")
   .setJourneyId(2)
   .build();
```

To create a service (SVC) record, use the following code:

```
Bill bill = new BillBuilder()
   .setBillType("SVC")
   .setClientCode("TC")
   .setSvcName("serviceName1")
   .setSvcVersion("80")
   .setLicenseKey("testLicenseKey")
   .setBillingScore("1.2")
   .build();
```

For more information, see Fluent API.

NOTE

To generate and read billing records, you must have the Bill Edit and Bill View permissions correspondingly.

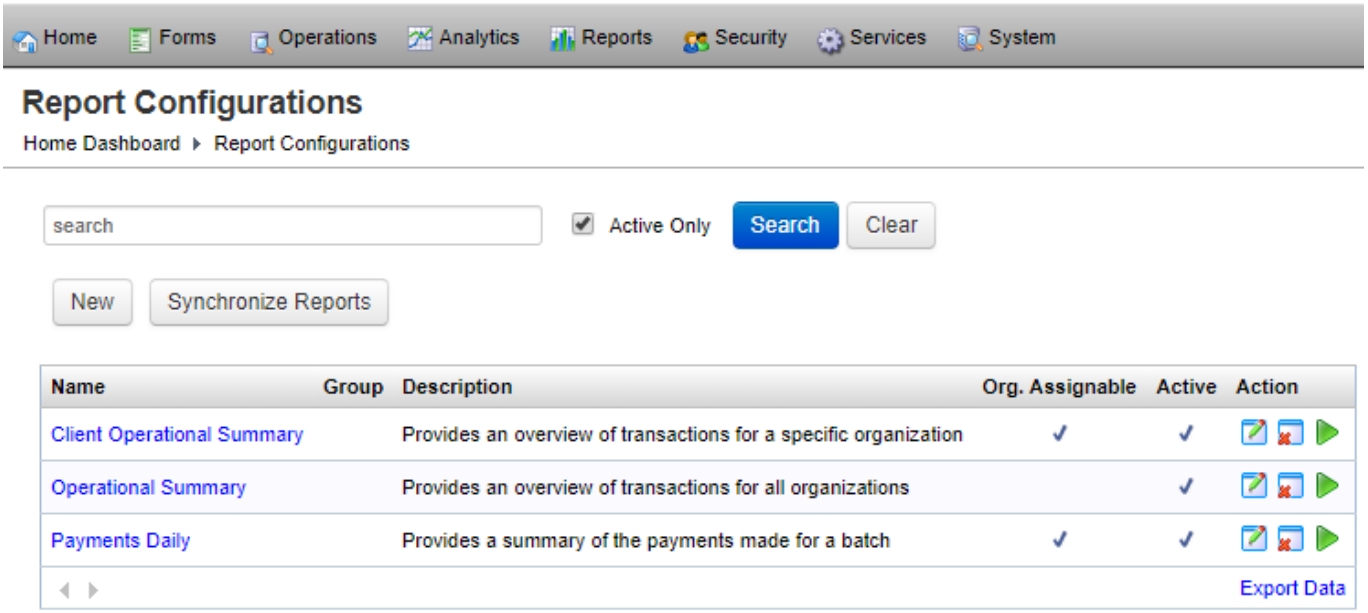
Next, learn how to view a transaction licensing report.

Report Configurations

Manager, prior to version 19.05, allows you to view, create and edit reports.

> NOTE
>
> You can try using the standard Job Licensing and Transaction Licensing reports before creating your own.

To view report configurations, select Reports > Report Configurations.



To filter reports by one or more criteria, specify the following settings and click Filter:

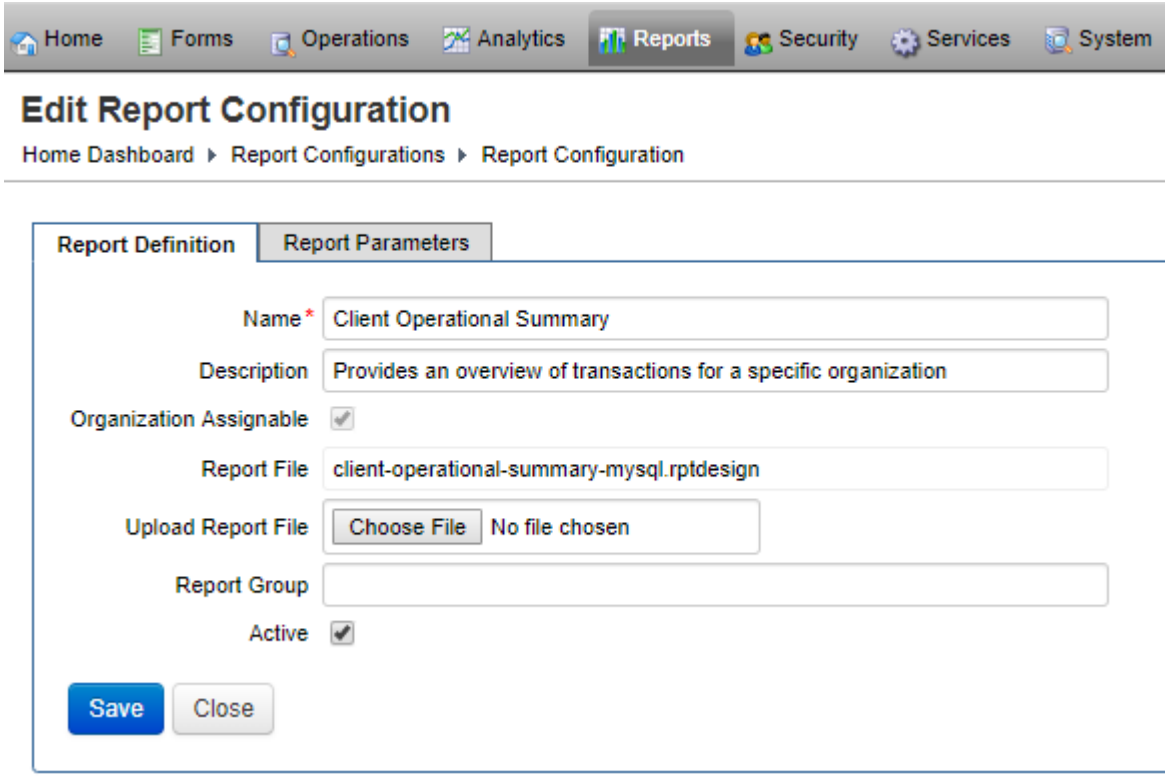
- A report name or part of the name
- Active Only

Click Clear to reset the search criteria.

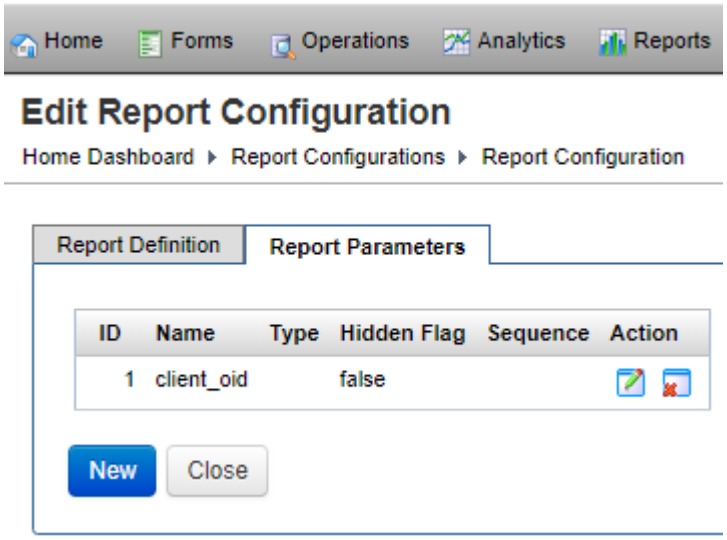
Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

To configure a report:

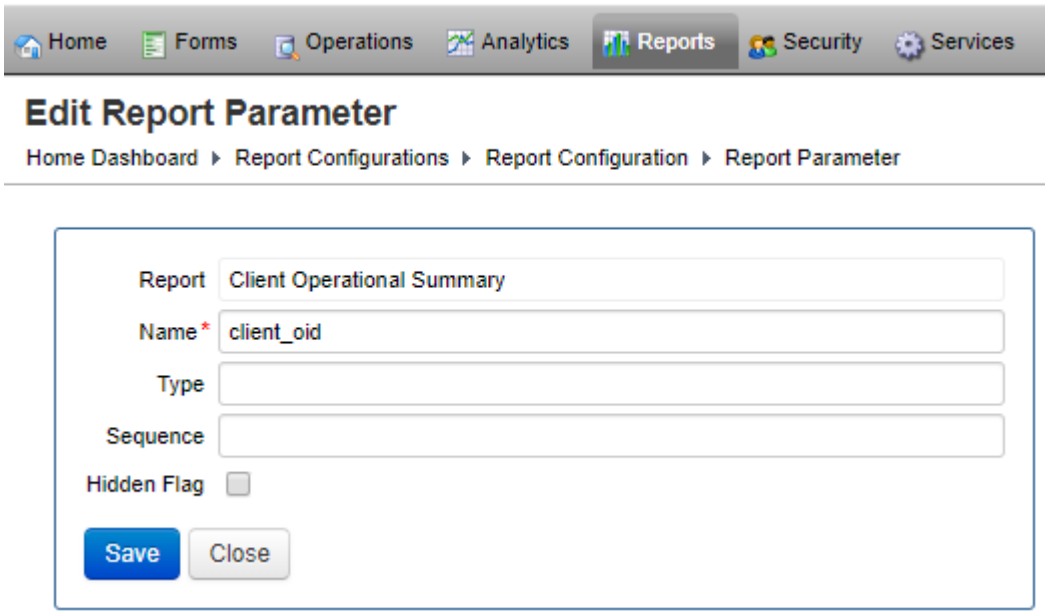
1. Select Reports > Report Configurations.
2. Locate a report and click Edit or click New to create a new report.



3. Edit the report name in the Name field.
4. Provide a report description in the Description field.
5. Select the Organization Assignable checkbox to make the report available to individual organizations.
6. Click Choose File to upload a report file.
7. Specify the Report Group.
8. Select the Active checkbox to make it the active report.
9. Click Save to update the changes.
10. Select the Report Parameters tab to create or configure report parameters.



11. Locate a parameter and click Edit or click New to create a new parameter.



12. Edit the parameter name in the Name field.
13. Provide the parameter type in the Type field.
14. Update the sequence number in the Sequence field.
15. Select the Hidden Flag checkbox to make it hidden.
16. Click Save to update the changes.

To remove a report:

1. Select Reports > Report Configurations.
2. Locate a report and click Remove Report to delete a report.

Click Synchronize Report to synchronize reports.

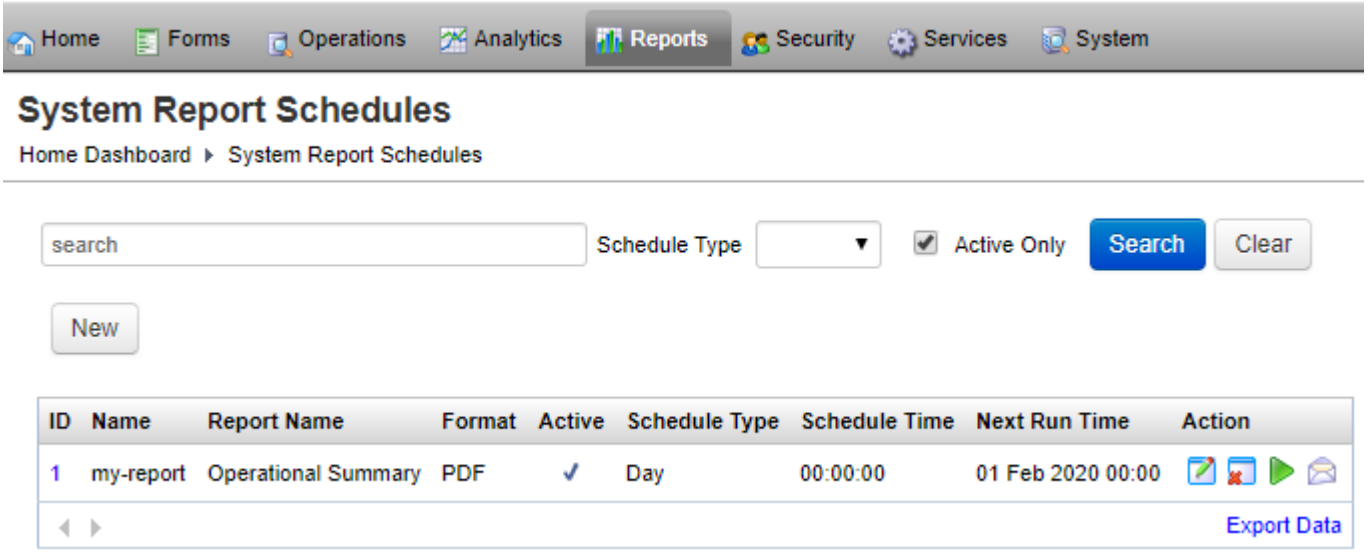
Click View the Report Using the Report Viewer to see the rendered report.

Next, learn about system report schedules.

System Report Schedules

Manager allows you to view, create new and edit existing system report schedules.

To view system report schedules, select Reports > System Report Schedules.



To filter reports by one or more criteria, specify the following settings and click Filter:

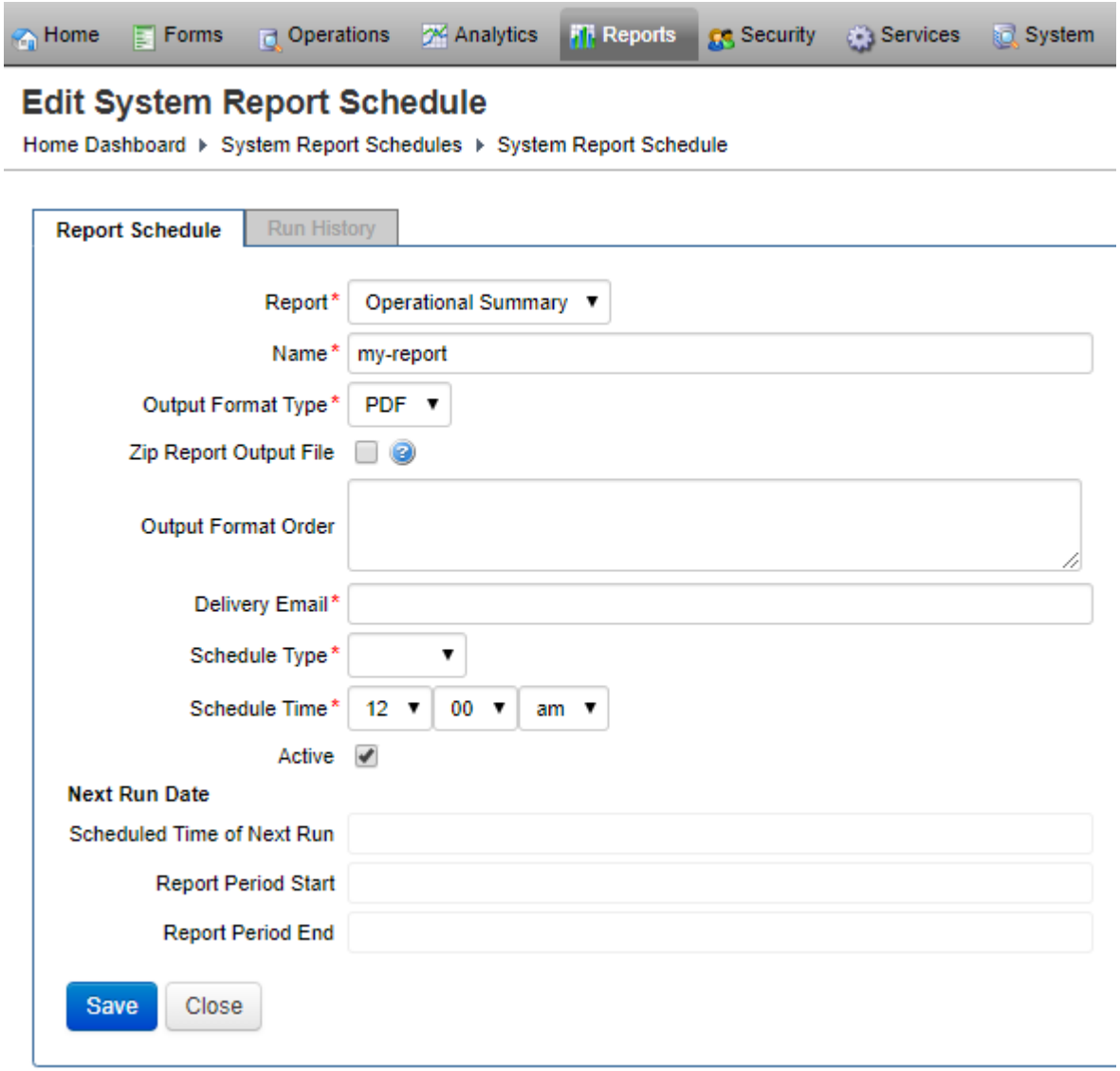
- A scheduler type
- Active Only

Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

To configure a system report scheduler:

1. Select Reports > System Report Schedules.
2. Locate a report and click Edit or click New to create a new report.



3. Select a report from the Report dropdown list.
4. Edit the report name in the Name field.
5. Select an output format type from the Output Format Type dropdown list.
6. Select the Zip Report Output File checkbox to ZIP the generated report.
7. Specify the output format order in the Output Format Order field.
8. Provide a valid email address in the Delivery Email field.
9. Select a scheduler type from the Schedule Type dropdown list.
10. Specify scheduler time from the Schedule Time dropdown lists.
11. Select the Active checkbox to make this scheduler active.
12. Click Save to update the changes.

To remove a system report scheduler:

1. Select Reports > System Report Schedules.
2. Locate a report and click Remove ReportScheduler to delete a scheduler.

Click View the Report On-Screen to see the rendered report.

Click Trigger the report scheduler to run the scheduler

Next, learn how to view job licensing report.

View Transaction Licensing Report

Manager automatically records licensing data, which includes organizations, forms, transactions, and services, and generates a license usage report for each organization. When the report is generated, the following assumptions are made:

- Report excludes In Progress transactions and Abandoned transactions.
- Report excludes Server Monitoring transactions.
- Reporting dates are based on transaction finish time when a form is Completed or Abandoned.
- Analytics reporting dates are based on transaction request time and should not be compared with this report.

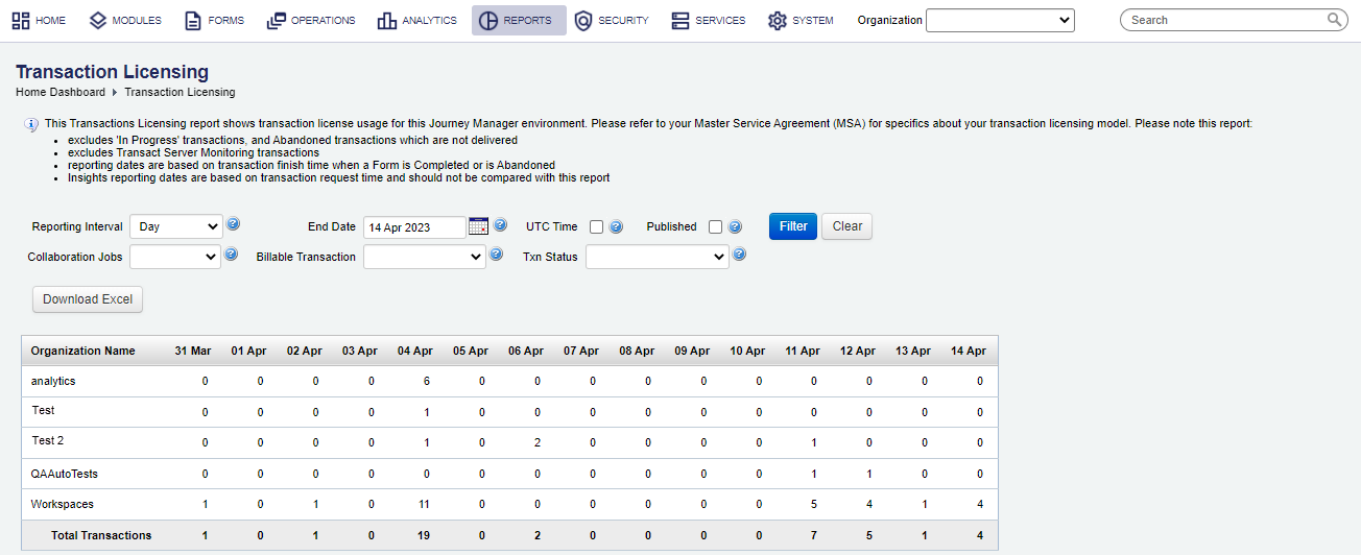
NOTE

By default, the transaction licensing report is run nightly.

NOTE

You can generate a transaction report using the `com.avoka.tm.query.TxnHistoryQuery` and `com.avoka.tm.svc.BillBuilder` Fluent APIs. For more information, see Fluent API.

To view the transaction licensing report, select Reports > Transaction Licensing.



The list displays the following details:

- Organization Name: a name of an Organization, sorted in the alphabetical order.
- Number of transactions per day for each organization.
- Total Transactions: a number of transactions per day for each organization.

To filter the transaction licensing report by one or more criteria, specify the following settings and click Filter:

- Select the reporting intervals as Day, Week, or Month from the Reporting Interval drop-down list.
- Select a collaboration job type from the Collaboration Jobs dropdown list:
  - Jobs Txns: transactions associated with a collaboration job.
  - Standalone: standalone transactions which are not part of a collaboration job.
  - Blank: both types of transactions. This is the default.
- Edit the End Date of the reporting period.
- Select a transaction type from the Billable Transaction dropdown list: | 22.10 This feature was introduced in 22.10.
  - Billable: only billable transactions
  - Non-Billable: only non-billable transactions
  - Blank: both types of transactions. This is the default.

> NOTE
>
> Prior to version 22.10, you could configure these options in the Scope dropdown list and they were called:
>
> - Internal Txns: transactions made by internal users.
> - External Txns: transactions made by external users, who are customers and applicants.

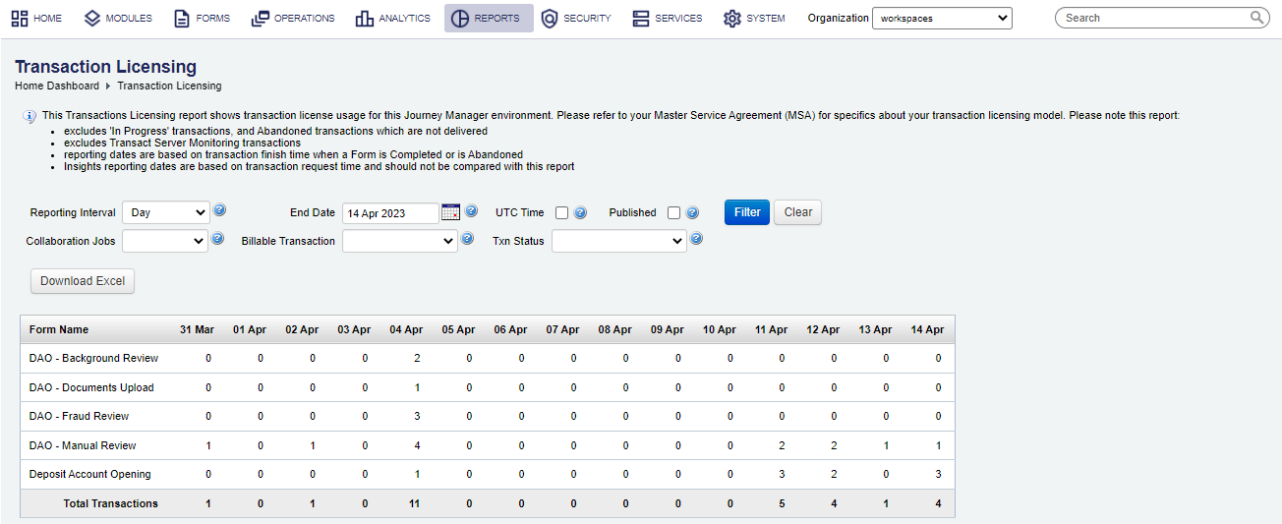
> NOTE
>
> You can change what type of licensing transactions a form space supports.

- Select the UTC Time checkbox to use the Coordinated Universal Time[1] (UTC) in the report.
- Select the Published checkbox to display transactions that have been published to Temenos. Otherwise, all transactions are included in the report.

---

[1]Coordinated Universal Time (UTC) is the primary time standard by which the world regulates clocks and time.

- Select whether the report displays completed transactions or delivered abandoned transactions in the Txn Status dropdown list. Otherwise, all transactions are included in the report.
- Select an organization from the Organization dropdown list to show the transaction breakdown for each form belonging to the selected organization. For example, if you select Workspaces from the Organization dropdown list, the following report is shown:



Form Name	31 Mar	01 Apr	02 Apr	03 Apr	04 Apr	05 Apr	06 Apr	07 Apr	08 Apr	09 Apr	10 Apr	11 Apr	12 Apr	13 Apr	14 Apr
DAO - Background Review	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
DAO - Documents Upload	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
DAO - Fraud Review	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
DAO - Manual Review	1	0	1	0	4	0	0	0	0	0	0	2	2	1	1
Deposit Account Opening	0	0	0	0	1	0	0	0	0	0	0	3	2	0	3
Total Transactions	**1**	**0**	**1**	**0**	**11**	**0**	**0**	**0**	**0**	**0**	**0**	**5**	**4**	**1**	**4**

Click Clear to reset the search criteria.

Click Download Excel to download a copy of this report as a Microsoft spreadsheet XLS file.

Next, learn how to view a job licensing report.

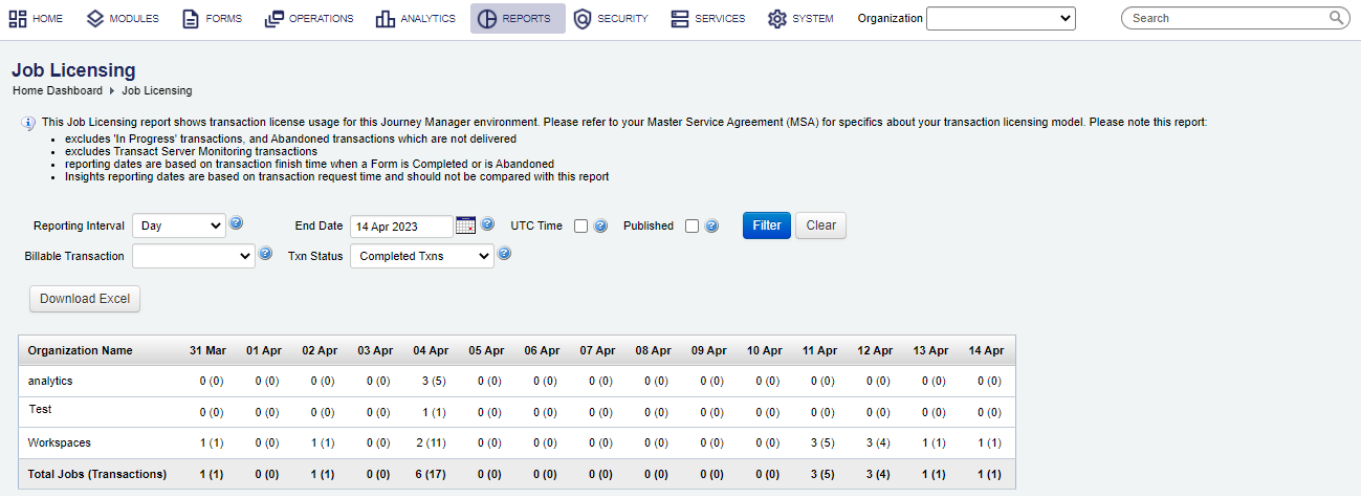
View Job Licensing Report

Journey Manager allows you to view job licensing data for transactions completed by collaboration jobs. When the report is generated, the following assumptions are made:

- Report excludes In Progress transactions and Abandoned transactions.
- Report excludes Server Monitoring transactions.
- Reporting dates are based on transaction finish time when a form is Completed or Abandoned.
- Analytics reporting dates are based on transaction request time and should not be compared with this report.

> NOTE
>
> You can generate a transaction report using the `com.avoka.tm.query.TxnHistoryQuery` and `com.avoka.tm.svc.BillBuilder` Fluent APIs. For more information, see Fluent API.

To view the job licensing report, select Reports > Job Licensing.



Organization Name	31 Mar	01 Apr	02 Apr	03 Apr	04 Apr	05 Apr	06 Apr	07 Apr	08 Apr	09 Apr	10 Apr	11 Apr	12 Apr	13 Apr	14 Apr
analytics	0 (0)	0 (0)	0 (0)	0 (0)	3 (5)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Test	0 (0)	0 (0)	0 (0)	0 (0)	1 (1)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Workspaces	1 (1)	0 (0)	1 (1)	0 (0)	2 (11)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	3 (5)	3 (4)	1 (1)	1 (1)
Total Jobs (Transactions)	**1 (1)**	**0 (0)**	**1 (1)**	**0 (0)**	**6 (17)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**0 (0)**	**3 (5)**	**3 (4)**	**1 (1)**	**1 (1)**

The list displays the following details:

- Organization names in the alphabetical order.
- Number of collaboration jobs and transactions (shown in brackets) per day for each organization.
- Total number of collaboration jobs and transactions (shown in brackets) per day for each organization.

To filter the job licensing report by one or more criteria, specify the following settings and click Filter:

- Select the reporting intervals as Day, Week, or Month from the Reporting Interval dropdown list.
- Edit the End Date of the reporting period.
- Select a transaction type from the Billable Transaction dropdown list: | 22.10 This feature was introduced in 22.10.
  - Billable - to show only billable transactions.
  - Non-Billable -to show only non-billable transactions.
  - Blank - to show both types of transactions. This is the default.

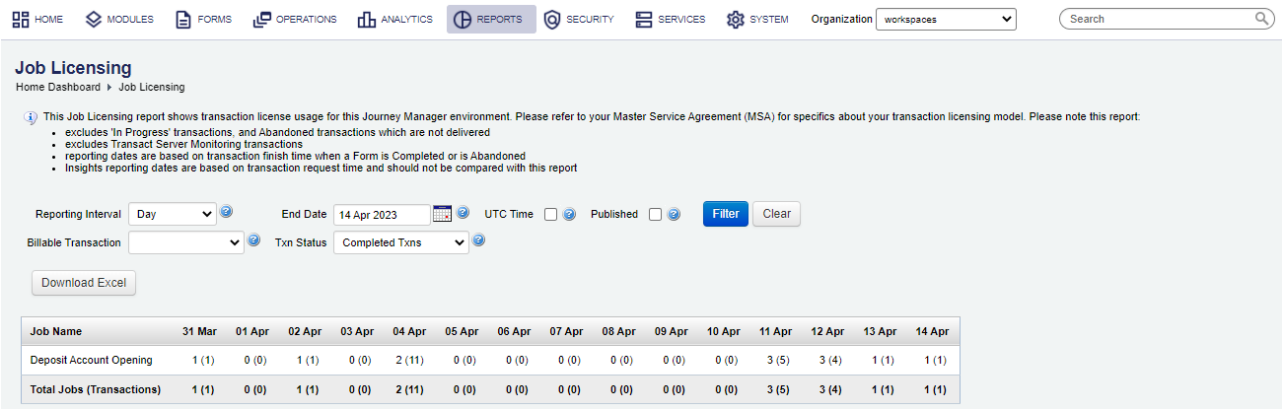
> NOTE
>
> Prior to version 22.10, you could configure these options in the Scope dropdown list and they were called:
>
> - Internal Txns - to show transactions made by internal users.
> - External Txns - to show transactions made by external users, who are customers and applicants.

> NOTE
>
> You can change what type of licensing transactions a form space supports.

- Select the UTC Time checkbox to use the Coordinated Universal Time[1] (UTC) in the report.
- Select the Published checkbox to display transactions that have been published to Temenos. Otherwise, all transactions are included in the report.
- Select whether the report displays completed transactions or delivered abandoned transactions in the Txn Status dropdown list. Otherwise, all transactions are included in the report.
- Select an organization from the Organization dropdown list to show the transaction breakdown for each form belonging to the selected organization. For example, if you select Work-

[1]Coordinated Universal Time (UTC) is the primary time standard by which the world regulates clocks and time.

spaces from the Organization dropdown list, the following report is shown:



Click Clear to reset the search criteria.

Click Download Excel to download a copy of this report as a Microsoft spreadsheet XLS file.

Next, learn how to view transaction licensing report.

Create a Custom Report

Manager allows you to create and present a custom report.

A quite common requirement at the end stages of a project is to create "on-demand" reports on the data within TM. Reports help management and operational users clearly see data that could even be PII data. One benefit of creating reports that are presented within a formspace is that we are not responsible for the delivery of PII data. The results are displayed on screen or within a CSV file and it's up to the end user to safely and securely download and store the file. Reports can be heavily customized to show data from submission objects, data extracts, or form xml. Remember that the form xml will be purged based on data retention settings. Reports can also be quite processor heavy so special care should come with load testing.

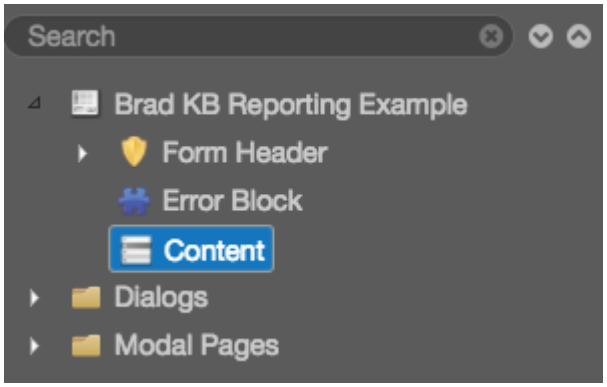
Let's walk though a very simple report that uses the Core API to query submissions. Typically I use the Core API because the Fluent API doesn't have a start and end date as part of the querying criteria. Start and end dates are a standard for most reporting criteria. Most reports are based on submissions and all DAOs query off of submission date except for the TransactionHistoryDAO which queries off of form start/opened date.

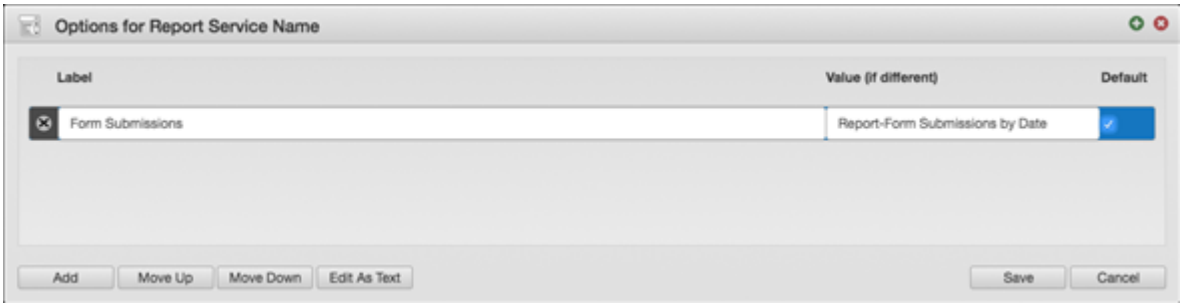
The following example only shows how to display 1 report but the architecture is setup in a way to easily add more reports.

Form

After creating a new blank form with your customer's branding:

1. Delete all non-essential components, especially the submit and save buttons. We don't want any transactions getting submitted to Manager or the customer will incur a fee.
    - On the Maguire form I deleted: Wizard Nav Bar, Form Footer, Function Bar, Navigation Second Level, Navigation Chevrons, Job Details Block, and everything within the "Content" Page Controller.



2. Add a Page named"Reporting" to the Content Page Controller.
3. Add a Section named"Report Criteria" to your "Reporting" Page.
4. Add a dropdownnamed "Report Service Name" to your "Report Criteria" Section. We will only have one report but this is where you can add multiple reports if required.
    - Add one option to the dropdown. Make the label "Form Submissions" and the value has to be the name of the Groovy Service that creates the report data. We'll call our service "Report-Form Submissions by Date".
    - 

5. Add a block called "Criteria1" to the "Reporting Criteria" Section. The reason for the block is to add visibility rules if you have different reports with different criteria.
6. Add a 2 date fields called "Start Date" and "End Date".
    - Prefill the "Start Date" and "End Date" with today's date.
    - Add Validation that they must be Past Dates and Include Today.
    - Make both fields Mandatory.
    - Add the following validation rule to "End Date" (optional):

```
if (Calc.daysBetween(data.startDate, data.endDate) > 7)
return "Date range cannot exceed 7 days.";
```

7. Add a Text Display field beside the dates that states the valid date range. i.e. "Date range cannot exceed 7 days."
8. Add a dropdown called "Output Format" with 2 options: CSV and HTML to the "Reporting" Section.
9. Add a "Run Report" button.
10. Add a Block called "OutputBlk" to house all the hidden data and output fields.
11. Add a Text Display field with "Output records have been returned." to the "OutputBlk".
12. Add a "recordsReturned", "rawData", and "outputReturned" Data Fieldsto the "OutputBlk".
13. Add a "Download CSV"Button to the "OutputBlk" with the following Visibility rule: `data.outputreturned === "CSV"`
14. Add a "Show HTML" Button to the "OutputBlk" with the following Visibility rule: `data.outputreturned === "HTML"`

15. At this point, all the necessary fields have been added. Now it's time to insert the logic to the fields.

16. Add the following "Click" script to the "Run Report" button:

```
var runReport = function()
{
data.rawdata = null;
data.recordsreturned = 0;
data.outputreturned = data.outputFormat;
Form.showProgress("Your report is being generated...");
var inputData = {
startDate: data.startDate,
endDate: data.endDate,
reportServiceName: data.reportServiceName,
outputType: data.outputFormat
}
DynamicData.call("GenerateDynamicDataReport", inputData).then(
function(response) {
Form.showProgress("");
console.log(response);
data.rawdata = response.data;
data.recordsreturned = response.recordsReturned;
}).catch(
function(err) {
Form.showProgress("");
console.log(err);
}
);
}
Form.validate("data.reportCriteria").then(
function(response) {
if (response.valid == true) {
runReport();
} else {
return;
}
}
).catch(
function(err) {
console.log(err);
}
);
```

17. Add the following Click script to the "Download CSV" Button:

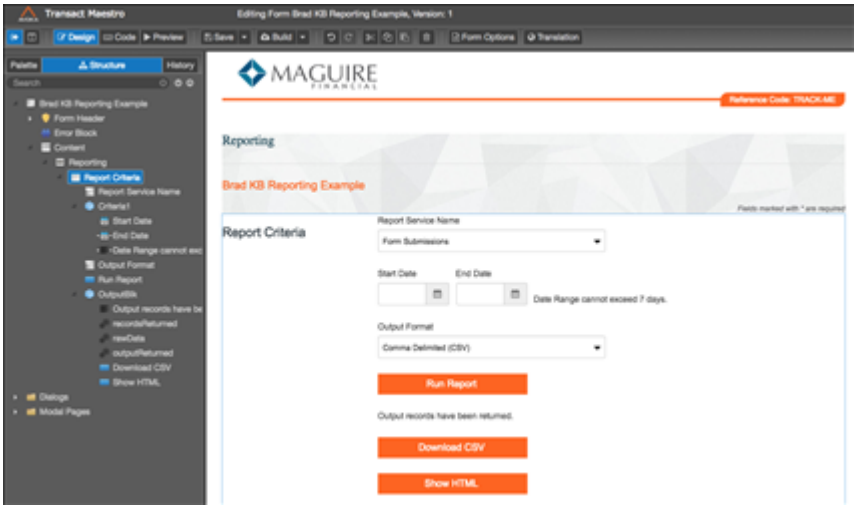
```
function requiredFeaturesSupported()
{
return ( BlobConstructor() && msSaveOrOpenBlobSupported() );
}
function BlobConstructor()
{
return (window.Blob)
}
function msSaveOrOpenBlobSupported()
{
return (window.navigator.msSaveOrOpenBlob)
}
function b64toBlob(b64Data, contentType, sliceSize)
{
contentType = contentType || '';
sliceSize = sliceSize || 512;
var byteCharacters = atob(b64Data);
var byteArrays = [];
for (var offset = 0; offset < byteCharacters.length; offset += sliceSize) {
var slice = byteCharacters.slice(offset, offset + sliceSize);
var byteNumbers = new Array(slice.length);
for (var i = 0; i < slice.length; i++) {
byteNumbers[i] = slice.charCodeAt(i);
}
var byteArray = new Uint8Array(byteNumbers);
byteArrays.push(byteArray);
}
var blob = new Blob(byteArrays, {type: contentType});
return blob;
}
if (requiredFeaturesSupported()) {
blobObject = b64toBlob((data.rawdata), "text/csv");
window.navigator.msSaveOrOpenBlob(blobObject, 'report.csv');
} else {
var a = document.body.appendChild(document.createElement("a"));
a.download = "report.csv";
a.href = "data:text/csv;charset=utf-8; base64," + data.rawdata;
a.innerHTML = "download report csv";
a.hidden = true;
a.click();
```

```
}
```

18. Add the following code to the "Show HTML" Button:

```
if (data.rawdata) {
var win = window.open("", "Report", "toolbar=no, location=no, directories=no,
status=no, menubar=no, scrollbars=yes, resizable=yes, width=780, height=200, top="+
(screen.height-400)+", left="+(screen.width-840));
win.document.body.innerHTML = data.rawdata;
}
```

All done... now it's time to write the services! Here's what it should look like:



Groovy Script

The form is designed to call one generic DDS with the name of the report service as a parameter. There are 3 services which sort of represent a typical 3 layer java framework of "Service Layer", "Manager Layer", and "DAO Layer". Only the "Manager Layer" has to change in most situations since the Submission object that gets returned gives you access to everything you'll need.

1. GenericDynamicDataReport - This is the main DDS service that the form calls. It's just passes on the parameters to the right Groovy Service. In a perfect situation you would also handle any bubbled exceptions here too.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.fc.core.service.ServiceLocator
import com.avoka.fc.core.dao.*
import com.avoka.fc.core.entity.*
import groovy.json.*
String reportServiceName = request.getParameter('reportServiceName')
String startDate = request.getParameter('startDate')
String endDate = request.getParameter('endDate')
String outputType = request.getParameter('outputType')
def reportService = ServiceLocator.getServiceForName(reportServiceName)
Map<String,Object> result = reportService.invokeService([
startDate:startDate,
endDate:endDate,
outputType:outputType
])
return JsonOutput.toJson(result)
```

2. Report-Form Submissions by Date - This is the name of the service that you configured in your Maestro form. It has to match perfectly with the value of the dropdown in "Report Service Name" and the parameters have to match as well. This service assembles the data returned from the next DAO call. This is where you can get really creative and return back submission properties, data extracts, form XML, etc... But this is where you also have to be careful with performance. The more introspecting and filtering you do the slower your report will be. The "filterData" function.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.fc.core.dao.*;
import com.avoka.fc.core.entity.*;
import com.avoka.fc.core.service.ServiceLocator
import groovy.json.*
DATEFORMAT = 'yyyy-MM-dd'
String formCode = serviceParameters.formCode // You'll have to add a service para-
meter or hardcode your formcode here.
String startDate = parameters.startDate
String endDate = parameters.endDate
String outputType = parameters.outputType
String encodeCsv = parameters.encodeCsv ?: 'false'
//logger.info "--------------------------------------------------"
//logger.info serviceDefinition.getServiceName()
//parameters.each { logger.info "$it.key: $it.value" }
//logger.info "--------------------------------------------------"
def reportHeaders = createReportHeaders()
def reportData = retrieveData(formCode, startDate, endDate)
reportData = filterData(reportData)
def output = null
if (outputType.toUpperCase() == "CSV") {
String csvString = createCsvReport(reportHeaders, reportData)
if (encodeCsv == 'true')
output = Base64.getEncoder().encodeToString(csvString.getBytes())
else
output = csvString
} else if (outputType.toUpperCase() == "HTML") {
StringBuilder reportHtmlStr = new StringBuilder();
reportHtmlStr.append startHtmlBody()
reportHtmlStr.append createHtmlReport(formCode, reportHeaders, reportData,
startDate, endDate)
reportHtmlStr.append endHtmlBody()
output = reportHtmlStr.toString()
}
return ["data":output, "recordsReturned": reportData.size]
/*******************
* HELPER FUNCTIONS
*******************/
def createReportHeaders()
{
```

```
def reportHeaders = ["ApplicationId" // 1
,"Submit Date" // 2
,"Submit Time" // 3
,"Current Status" // 4
,"Last Updated" // 5
]
return reportHeaders
}
def retrieveData(formCode, startDate, endDate)
{
def getSubmissionDeliveryListService = ServiceLocator.getServiceForName("GetSub-
missionDeliveryList")
def results = getSubmissionDeliveryListService.invokeService(["formCode":formCode,
"startDate":startDate, "endDate":endDate])
return results
}
def filterData(results)
{
StringBuilder dataStr = new StringBuilder();
def dataTable = [][]
def filteredResults = results.sort { it.timeSubmission }.reverse()
filteredResults.eachWithIndex { it, i ->
dataTable[i] = [
it.trackingNumber, // 1
it.timeSubmission.format(DATEFORMAT), // 2
it.timeSubmission.format(TIMEFORMAT), // 3
it.job?.currentStep?.name, // 4
it.job?.currentStep?.timeCreated.format(DATETIMEFORMAT) // 5
]
}
return dataTable
}
def createHtmlReport(formCode, reportHeaders, dataTable, startDate, endDate)
{
StringBuilder dataStr = new StringBuilder();
dataStr.append "<p>Applications submitted for ${formCode} from ${startDate} to
${endDate}</p>"
dataStr.append "<table>"
dataStr.append "<tr>"
reportHeaders.each {
```

```
dataStr.append "<th>${it}</th>"
}
dataStr.append "</tr>"
dataTable.each{ it1 ->
dataStr.append "<tr>"
it1.each {
dataStr.append "<td>${it ?: ""}</td>"
}
dataStr.append "</tr>"
}
dataStr.append "</table>"
return dataStr.toString()
}
def startHtmlBody()
{
StringBuilder dataStr = new StringBuilder();
dataStr.append "<!DOCTYPE html><html>"
dataStr.append '''<head>
<style type="text/css">
body {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 10px;
}
table {
border-collapse: collapse;
}
table, th, td {
font-size: 10px;
padding: 3;
border: 1px solid black;
}
td {
color: #898989;
font-size: 10px;
padding: 3;
text-align: right;
}
</style>
</head>'''
dataStr.append "<body>"
```

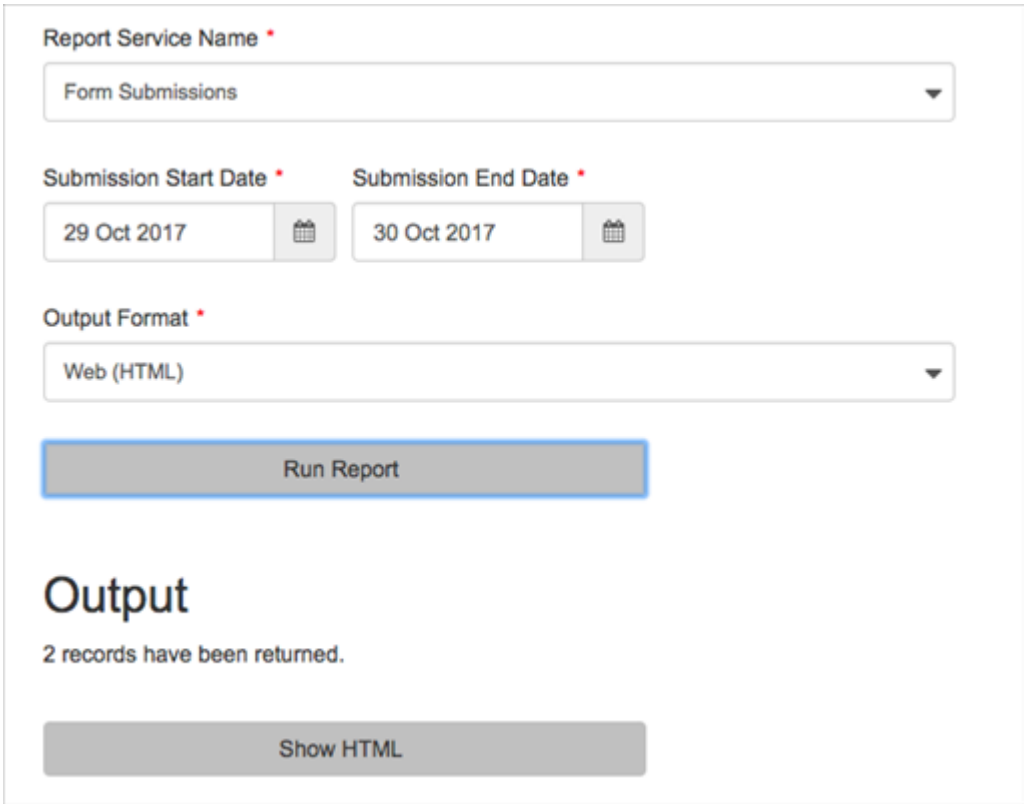
```
return dataStr.toString()
}
def endHtmlBody()
{
return "</table> </body> </html>"
}
def createCsvReport(reportHeaders, dataTable)
{
StringBuilder dataStr = new StringBuilder();
reportHeaders.each {
dataStr.append "\"${it}\","
}
dataStr.append "\n"
dataTable.each{ it1 ->
it1.each {
dataStr.append "\"${it ?: ""}\","
}
dataStr.append "\n"
}
return dataStr.toString()
}
```

3. GetSubmissionDeliveryList - This "DAO Layer" extracts submission objects for a given date range for a specific formcode. I've played with many DAOs and maybe one suits your purpose more than this but what I've found is that many of the DAOs cut off after a certain limit regardless of what you specify. The requirements I've had in many reports goes beyond these limitations so I've found this script/DAO to be the best and surprisingly performs very well. You can also pass a parameter to include Saved transactions.

```
import com.avoka.core.groovy.GroovyLogger as logger
import com.avoka.fc.core.service.ServiceLocator
import com.avoka.fc.core.dao.*
import com.avoka.fc.core.entity.*
import groovy.json.*
String formCode = parameters.formCode
Date startDate = Date.parse('yyyy-MM-dd', parameters.startDate)
startDate.set(hourOfDay : 0)
startDate.set(minute : 0)
startDate.set(second : 0)
Date endDate = Date.parse('yyyy-MM-dd', parameters.endDate)
endDate.set(hourOfDay : 23)
endDate.set(minute : 59)
endDate.set(second : 59)
String includeSaved = parameters.includeSaved ?: 'false'
//logger.info "--------------------------------------------------"
//logger.info serviceDefinition.getServiceName()
//parameters.each { logger.info "$it.key: $it.value" }
//logger.info "--------------------------------------------------"
List deliveryStates = ["Not Ready", "Ready", "Sent Email", "In Progress", "Pend-
ing", "Completed", "Error", "Undeliverable", "Not Required"]
String clientId = DaoFactory.getClientDao().getClientByCode("digital").getId();
String deliveryDetailsId = null
String deliveryMessageLike = null
int fetchLimit = 10000
SubmissionDao submissionDao = DaoFactory.getSubmissionDao();
List allResults = []
deliveryStates.each {
allResults.addAll(submissionDao.getSubmissionDeliveryList(it,
clientId,
deliveryDetailsId,
formCode,
deliveryMessageLike,
startDate,
endDate,
fetchLimit))
}
if (includeSaved == 'true') {
def savedList = submissionDao.getSavedSubmissionList(null, clientId, null, null,
startDate, endDate)
def filteredResults = savedList.findAll{ it.formCode == formCode }
```

```
allResults.addAll(filteredResults)
}
//logger.info "Total records retrieved - getSub-
missionDeliveryList.size:${allResults.size}"
return allResults
```

The interface for the custom report is shown below:



NOTE

You can also use the `TxnHistoryQuery` class to create a custom report using a Groovy script.

Next, learn how to [view job licensing report](#).

Outgoing Email Service Overview

Journey Manager uses a SMTP[1] server to send emails to various types of users, such as application form users, system administrators and DevOps, to notify them about the following events:

- A link to a user to download a PDF receipt of their completed transaction
- A PDF receipt as an attachment to a user directly
- A form is saved, shared, submitted, and so on
- Delivery process
- Form promotions
- Scheduled reports
- Alerts and system notifications

Emails are either sent directly as part of application form processes or can be scheduled using the email queue. The later provides a mechanism of sending emails, keep track of sent emails including their statuses, and re-send emails.

Manager doesn’t process incoming emails and it doesn't monitor the SMTP server for bounced emails. Bounced emails can take place when form users enter incorrect email addresses while filling in forms. In this case, Manager still sends emails to these addresses, which are bounced on the SMTP level. If you want to monitor bounced emails, you need to ask your cloud hosting team or your SMTP server administrator to configure the bounces to go to a mailbox where you can monitor them and action accordingly.

Manager doesn't directly support a full secure email mechanism through SMTP, so you can not send encrypted emails from a Manager server to a customer email server. However, Manager allows you to enable TLS connections for the Manager's mail server. If Manager is configured to

---

[1]SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with one of two other protocols, POP3 or IMAP, that let the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving e-mail. On Unix-based systems, sendmail is the most widely-used SMTP server for e-mail. A commercial package, Sendmail, includes a POP3 server. Microsoft Exchange includes an SMTP server and can also be set up to include POP3 support.

use Amazon Simple Email Service (SES)[1] to send emails, it uses TLS up to the point when SES send out the emails to the recipient SMTP.

NOTE

Only one SMTP server can be configured per Journey Manager instance.

NOTE

You can change an email notification format to adjust it to your requirements.

Next, learn how to configure the SMTP server.

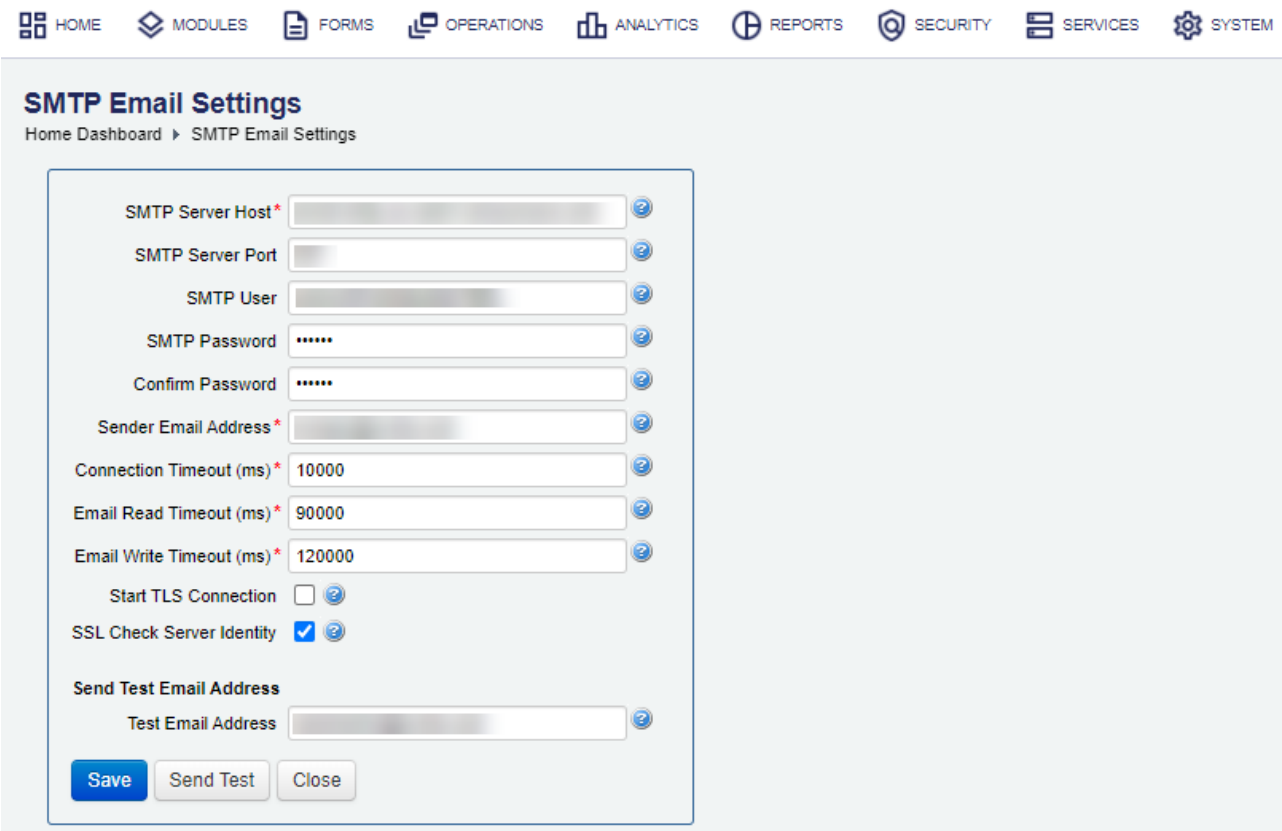
[1]Amazon Simple Email Service (SES) is a pay-per-use service that allows you to build in email functionality into an application that you are running on AWS.

Configure a SMTP Server

Manager requires you to configure an SMTP server before you can start sending queued emails notification emails. The SMTP server can be a local one included with the installer or an external one, such as Amazon Simple Email Service (SES)[1].

To configure the SMTP server:

1. Select System > SMTP Email Settings.



2. Provide an SMTP server host name in the SMTP Server Host field, for example, `email-smtp.my-domain.amazonaws.com`.

[1]Amazon Simple Email Service (SES) is a pay-per-use service that allows you to build in email functionality into an application that you are running on AWS.

3. Specify a port number in the SMTP Server Port field, for example, `1234`.

   NOTE

   The port 465 is used for encrypted SMTP.
   The port 25 is blocked in AWS by default. If you must use it, please contact out Support Team.

4. Enter an SMTP email server user name in the SMTP User field.
5. Choose a strong SMTP email server user password in the SMTP Password field.
6. Provide the default email sender address in the Sender Email Address field, for example, `noreply@my-domain.com`.
7. Specify the email server connection timeout in the Connection Timeout (ms) field.
8. Specify the email server read timeout in the Email Read Timeout (ms) field.
9. Specify the email server write timeout in the Email Write Timeout (ms) field.
10. Select the Start TLS Connection checkbox to enable TLS[1] connection to the SMTP server. This is set as the `mail.smtp.starttls.enable=true` property. TLS version depends on a 3-rd party library used by Manager as well as recipient's mail server's capability.

    NOTE

    Check that the SMTP server supports TLS before enabling this setting, otherwise emails won't be delivered and you'll receive an error.

11. Select the SSL Check Server Identity checkbox to check the server from the socket connection against a server name(s) as specified in the server certificate (RFC 2595 check). This is set as the `mail.smtp.ssl.checkserveridentity=true` property.
12. Provide the test email address in the Test Email Address field to send a test to, for example, `test-user@my-domain.com`.
13. Click Send Test to verify SMTP configuration by sending an email to the Test Email Address.
14. Click Save to update the changes.

Next, learn how to view the email queue.

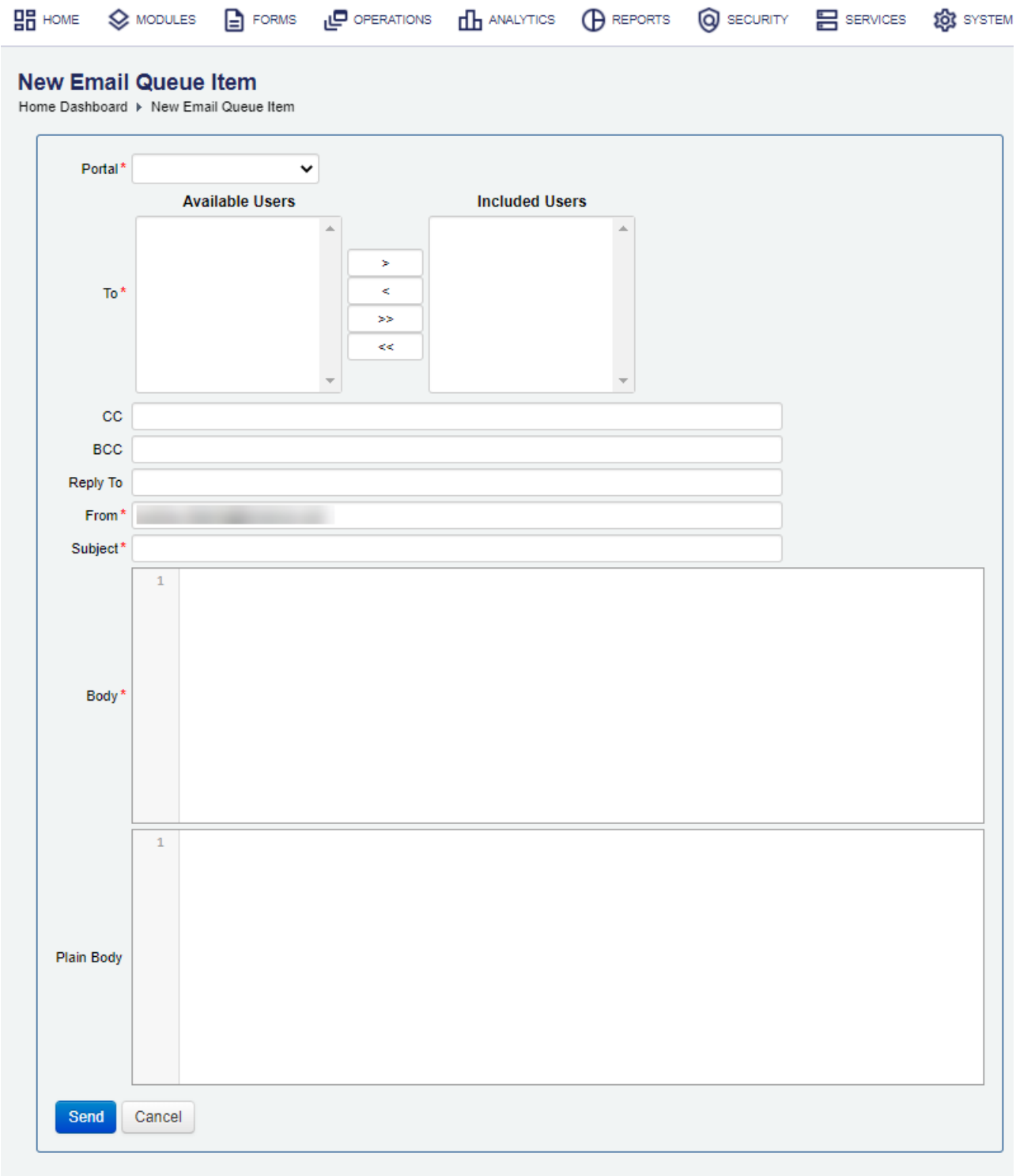
[1]Transport Layer Security (TLS) is a cryptographic protocol designed to provide communications security over a computer network.

Create an Email Queue Item

Manager allows a Manager administrator to create emails in the email queue, which are automatically sent and tracked.

To send an email, you need to add it to the email queue first:

1. Select System > Email Queue and click New.



2. Select a space from the Portal dropdown list. It populates the Available Users list with all users associated with this form space.
3. Select one or more email addresses from the Available Users and click > to add them as recipients. To add all users, click >>. Click < or << to remove selected users.

4. Add additional email addresses into the CC and BCC fields.
5. Edit the Reply To address if it is different to the From address. This address is used when a user replies to an email sent out via the email queue.
6. Edit the From field with a custom email, if required, otherwise the default value of the Sender Email Address, defined in the SMTP Email Settings, is used.
7. Edit an email subject in the Subject field.
8. Enter the email message into the Body email content or the Plain Body for plain text content.
9. Click Send to add this email to the email queue, which will send it.

> NOTE
>
> There is a maximum of two automatic retries in case of any errors.

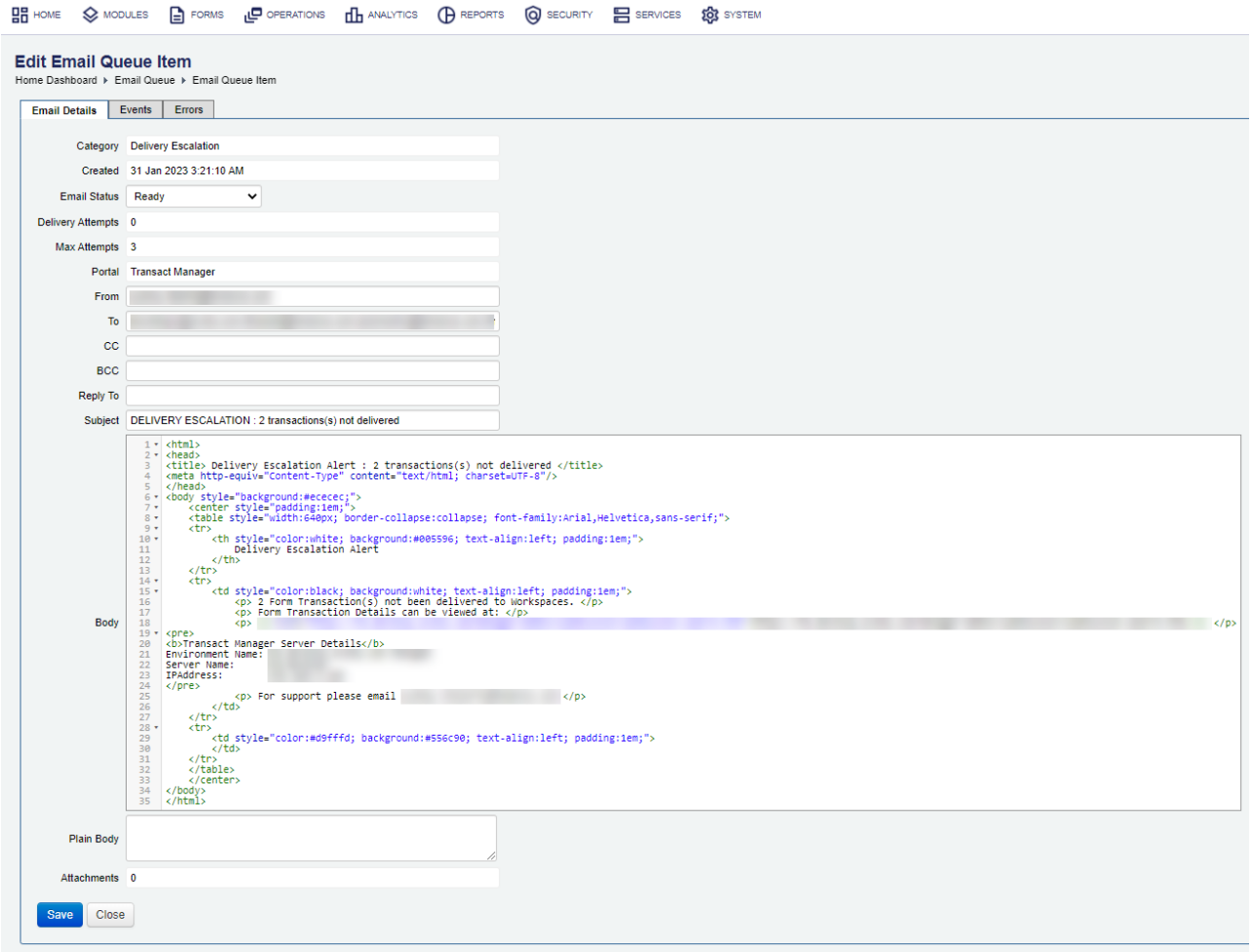
Next, learn how to view the email queue.

Edit an Email Queue Item

Manager allows you to edit an email that was added to the email queue.

To edit an email:

1. Select System > Email Queue.
2. Locate an email and click Edit.
3. Click the Email Details tab to edit some of the fields such as the recipients and email's body.



4. Select a new email status from the Email Status dropdown list to make Manager perform a desired action on the email. For example, select:
   - Ready to send the email again. This is handy to resolve email delivery issues when an email has been unsuccessfully resent several times and has exceeded the maximum number of retries.
   - Completed to mark the email as sent, so it can be removed from the email queue during the next data purge

5. Click Save to update the changes.

Click the Events tab to see any events associated with this email delivery.

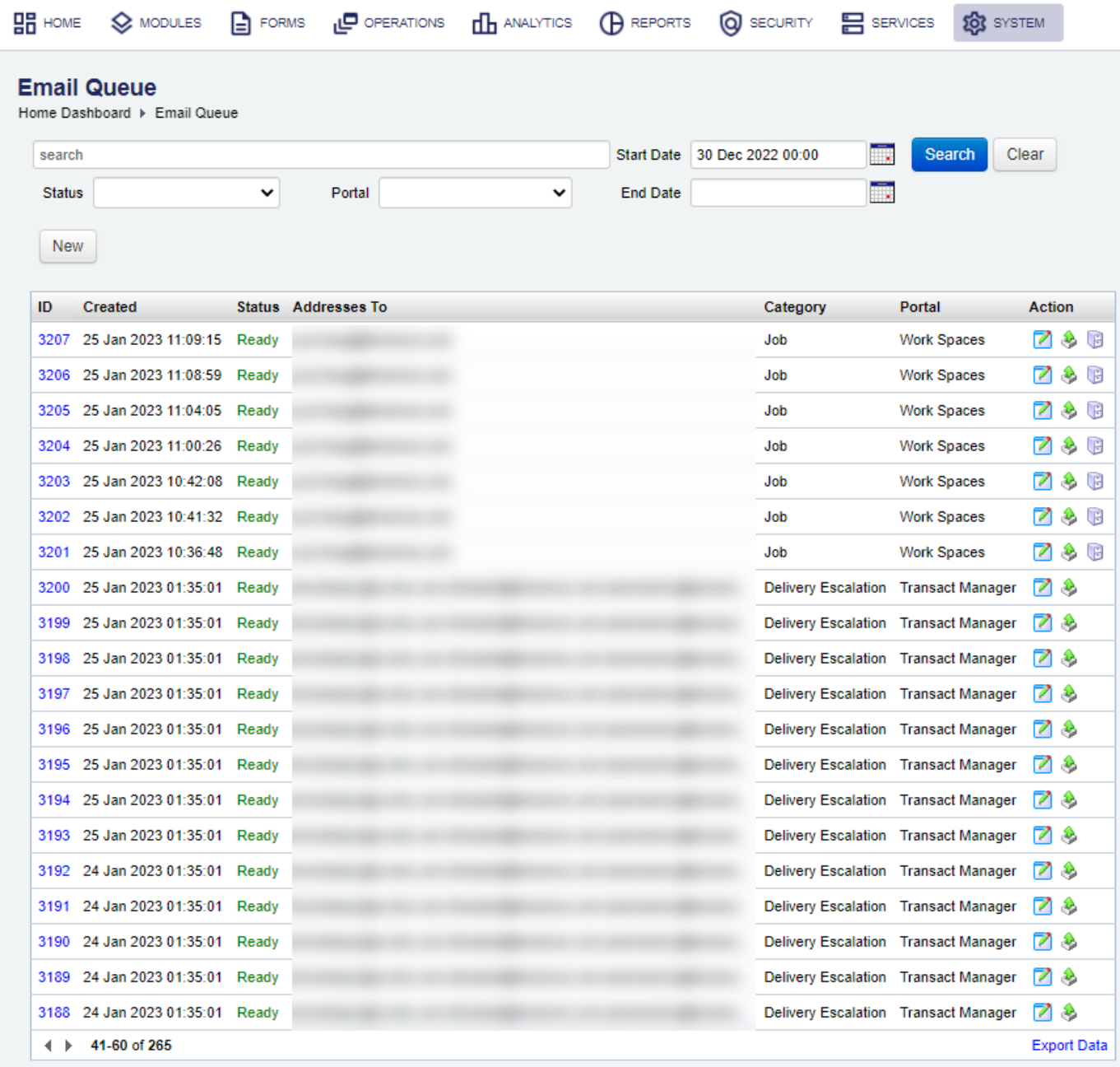
Click the Errors tabs to see any errors that occurred when attempting to send the email. This can be useful for debugging email failures.

Next, learn how to view email queue items.

View the Email Queue

Manager allows you to view and manage emails in the outgoing email queue.

To view the list of all emails, select System > Email Queue.



The list displays the following details:

- ID: an email ID.
- Created: a day and time when the email was created.

- Status: a status of the email, which can be one of the following:
  - Not Ready
  - Ready
  - Completed
  - Error
  - Max Retries Exceeded
- Addresses To: a list of email addresses the email was sent to.
- Category: a category of the email.
- Portal: a form space the email was sent from.

To filter or search the email queue by one or more criteria, specify the following settings and click Search:

- A part of the email, for example an email address.
- Status
- Portal
- Start Date - select a start date of the date range.
- End Date - select an end date of the date range.

Click Clear to reset the search criteria.

You can also perform the following actions:

- Create a new email
- Edit an email
- Click Send Email Now to resent the email manually. An administrator can do this task after identifying and fixing the email delivery issues.
- Click View Submission Details to check the transaction detail

NOTE

You can't delete an email from the queue, but you can change its status to Completed so it will be purged by the Data Retention Schedule during its next run.
To accelerate the purging process, you can set the Email Queue to 1 day and click Apply Policy. Don't forget to set it back to its previous value after emails are purged.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to create email queue item.

Data Retention Management Overview

Manager has a high volume of user transactions associated with forms and their resources, such as data records and log files. This amounts to significant data even during a rather short period of operation. This can have a detrimental effect on the overall system performance, which, in turn, can negatively impact user experience and various third party integrations. To maintain the system optimal performance over time, Manager provides data storage, retention and purging capabilities that you can adjust according to your company's data policies. This is instrumental in achieving the following:

Control the growth of database tables

In Manager, the Transaction table is a heavy weight table used to store a lot of information about in progress transactions and is not designed for long term storage. When a transaction is completed an entry is created in the Transaction History table which is designed to provide a long term audit trail. Purging data and records quickly from the Transaction table will help maintain high system performance.

Reduced Exposure of PII

Purge transaction form data which contains sensitive information

Form transactions can include sensitive business or personal data, such as a form user's Personally Identifiable Information[1] (PII), depending on how transactions have been designed. Purging this data after delivery provides a stronger security story.

Data retention management is an automated process - what you need to do is to adjust the default data retention configuration and periodically review it to make sure the data is purged as per your SLAs[2].

> TIP
>
> If you back up a database that is external to Manager, you must ensure that the back-ups are encrypted and securely stored.

Manager purges the following data:

---

[1]Personally identifiable information (PII) is any data that could potentially identify a specific individual. Any information that can be used to distinguish one person from another and can be used for de-anonymizing anonymous data can be considered PII.

[2]A service-level agreement (SLA) is a commitment between a service provider and a client. Particular aspects of the service – quality, availability, responsibilities – are agreed between the service provider and the service user.

- Transaction data: the forms XML, Data Extracts, Attachments and Receipts. This will include the majority of the users personal information.
- Transaction records: the actual record from the Transaction table. The main reason to keep this record in place for short period of time is for troubleshooting.
- Transaction history records: the audit log of transaction activities. For systems with high throughput there is a argument for archiving these records for long term storage and purging these records from Manager.
- System logs: numerous areas of system logs whose purge time can the controlled separately for troubleshooting purposes.

Manager collects data retention information for each transaction including whether any PII data has been purged or is due to be purged and details about the data retention stage reached in the life cycle of this transaction.

Manager allows you to configure data retention on the global (system), organization and form levels that gives you flexibility in overriding important settings for individual organizations. Let's look at this in more detail.

Global Level

You can define data retention on the global (system) level, which is applicable to an entire server or environment, including all server nodes. This configuration is initially set to default values that depend on your environment's data retention policy mode: strict or relaxed.

In addition to data retention, you can also configure submission data storage. This means that you can choose how and where submission-related data, such as submission data, attachments and submission history data for saved forms, is stored and retrieved by Manager.

By default, submission-related data is stored in a Manager database. However, it can be configured and changed at runtime as well. The functionality is encapsulated in a set of service definitions of type Submission Data Storage. The storage service that is marked as the default for this service type is used when storing submission data.

Organization Level

By default, Manager applies global data retention configuration defined for the entire server to its organizations, unless an organization has its own data retention configured.

However, you can customize retention policies for all transactions associated with forms, for example, saved and finished transaction's PII[1] data.
The data retention policies you can configure are the maximum age of form submissions in general as well as delivered (including undeliverable) and saved (including abandoned) submission data.
You can also set the data storage encoding for the organization if it is different to the system-wide setting. We recommend that you use the Compressed / Encrypted option for maximum security and minimal storage space requirements. For more information, see File System Submission Data Storage Service.
Finally, for additional security you can set a rollover interval for the security key for the organization. The security key is used by Manager to encrypt submission-related data. All encrypted data is associated with its security key, so data encrypted before a security key change will still be accessible. If you choose a value other than Never, a new security key will be automatically generated by a background job when required.

Form Level

Manager applies global data retention policies, defined for the entire server or individual organizations, to its forms. However, you can customize transaction retention policies on the form level, for example, saved and finished transaction's PII[2] data. These settings then apply to all submissions made for this form.

Manager has two modes of data retention policy, strict and relaxed, that determines the length of time transactional data is retained. The selected policy can be customized at the environment organization, or form levels.

> NOTE
>
> It's recommended to use the strict policy mode, which enforces less transactional, historical and log data retention as opposed to the relaxed policy mode.

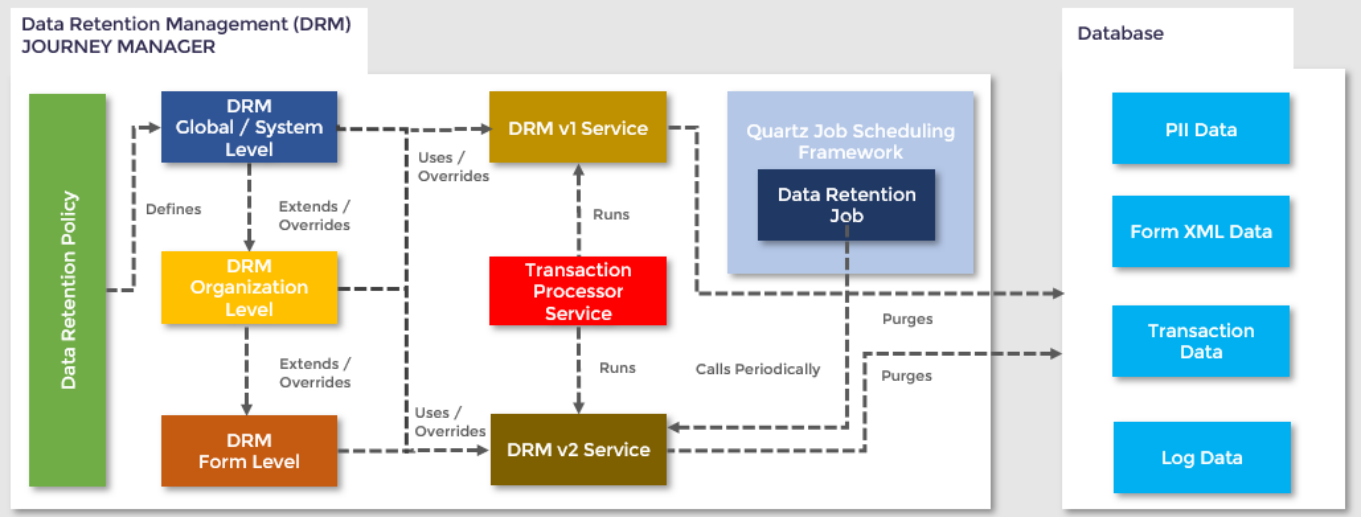
---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual.

[2]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual.

Manager uses various scheduled jobs that automatically purge transactions that have reached their maximum age.

The diagram below summarizes all this functionality.



You can check the data retention summary, which is useful for monitoring the overall performance of the data retention management and assists in diagnosing and tuning of any data retention management issues.

Next, learn about data retention policies.

Global Data Retention Configuration

Manager comes with highly customizable Data Retention Management that allows you to implement various customer requirements relating to their data retention policies. You can define data retention on the global (system) level, which is applicable to an entire server or environment, including all server nodes. This configuration is initially set to default values that depend on your environment's data retention policy mode: strict or relaxed. Then, you can refine data retention configuration per an organization and, further, per form.

NOTE

The Temenos Journey Manager platform is a System of Engagement[1] that enables you to create onboarding journeys for many types of products. It's strictly designed to store onboarding data for a short period of time, just long enough for a customer to complete their applications, and uses functionality, such as a configurable Data Retention policy to keep the amount of stored data low at any given time.
Temenos Journey Manager is not intended to be used as a System of Record[2] that stores large amount of data as this can affect system performance.

Global data retention configuration properties are grouped into the following sections:

Transaction Records

Transaction records are created the moment a user opens a form. The status of the form changes as the user saves, cancels, and submits the form. Each transaction corresponds to the user entered data, and is associated with other details, such as submission history, submission properties, data extracts, and attachments. In relation to data retention, transactions incur two major concerns:

---

[1]Systems of Engagement (SoE) is the technology used by an organization to help facilitate and orchestrate the customer journey via more personalized, seamless interactions across the various touchpoints. These include social media channels, email marketing platforms, mobile apps, and content management systems.
[2]System of Record (SOR) is an ISRS (information storage and retrieval system) that is the authoritative source for a particular data element in a system containing multiple sources of the same element. To ensure data integrity, there must be one -- and only one -- system of record for a given piece of information.

- Transactions are continually generated and contain a large amount of data. This can lead to large database sizes and degraded performance if data is not purged regularly.
- Transactions contain user-entered Personally Identifiable Information. This can pose a security risk. Manager comes with several data security measures and we recommend purging PII data as soon as transactions have finished.

Generally, you want to keep your purge settings strict, which ensures that transaction data is purged regularly, maximizing your system's performance and security. Manager supports this approach by a frequent job that checks for and purges any PII data and completed submissions that had been completed more than the maximum age time frame. The transactions eligible for deletion are transactions in a finished state, such as completed transactions (successful submissions) and abandoned transactions. For more information, see the Transaction Records sections.

System Logs

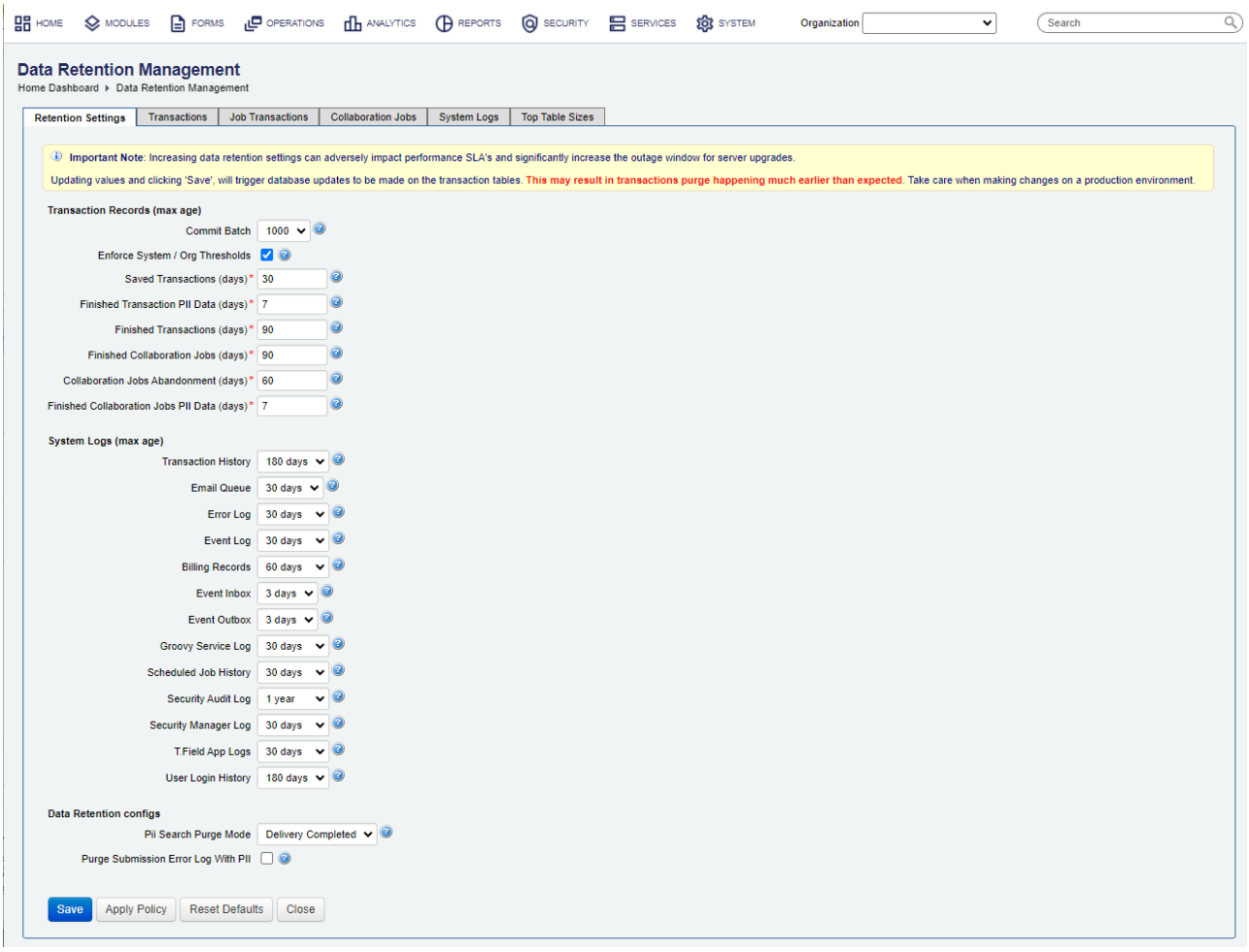
This section allows you to define how long to keep various logs that Manager collects as per transaction life cycle.

Data Retention Configs

This section allows you to configure additional data retention settings.

To configure global data retention:

1. Select System > Data Retention Management and click the Retention Settings tab.



NOTE

You must have the Service Edit permission to view this configuration.

Transaction Records (max age)

2. Select a size of a commit batch, which is used for transaction purging, from the Commit Batch dropdown list. The value must be within the fetch limit specified below. The default is 1000.
3. Select the Enforce System/Org Thresholds checkbox to ensure that data retention policies on lover levels (organizations and forms) can't exceed policies on higher levels (system or organizations). This prevents you from configuring a more lenient policies at the organization and form levels.

Furthermore, where there is a stricter corresponding setting at the organization level, you will be prevented from setting a more lenient corresponding override at the form level.

For example, if the global Finished Transaction PII Data policy is set to 14 days, you can't enter values greater than 14 days for the Finished Transaction PII Data policy for the organization or form level. In addition, if an organization's value of the Finished Transaction PII Data policy is 10 days, you can't configure any of this organization's forms to have the same policy be greater than 10 days.

> NOTE
>
> This setting is only relevant to the Saved Transactions and Finished Transaction PII Data policies, as only they can be overridden on all 3 levels.

> NOTE
>
> Even if Enforce System/Org Thresholds is selected, changes to global or organization policies don't affect the values already set on lower levels. For example, if an organization has the Finished Transaction PII Data policy set as 14 days and you change the global Finished Transaction PII Data policy to 7 days, the organization's value will remain unchanged until you start editing it. This logic avoids problems around accidental changes of global policies that would start trickling through the entire system.

4. Edit the maximum age of user saved transactions, after which they will be automatically abandoned, in the Saved Transactions field. The enforced range of this value is:
   - Min: 1 day
   - Recommended: 30 days
   - Max (Strict mode): 180 days
   - Max (Relaxed mode): 730 days

> NOTE
>
> Manager enforces the maximum limits of data retentions to provide required system stability.

The abandonment date is reset each time the form is saved by the user. This means that the allowable length of time between when the form is first opened and its eventual

submission may be extended by the user resuming the form regularly to complete and save the entered data.

Once a saved transaction is abandoned, it is considered a finished transaction like when a user completes and successfully submits a form. Note that the transaction will not be deleted at this point, but as a finished transaction, the countdown for PII data deletion and submission record deletion starts ticking.

> NOTE
>
> You can override this setting on both the organization and form levels.

> WARNING
>
> Be aware that 24-hour automatic abandonment of transactions can happen even they are rendered but never actually captured any data, or transactions that are never user-saved, even though they are background-saved, for example, on page navigation.

5. Edit the amount of time to keep PII transaction data for after the transaction is finished (Completed or Abandoned) in the Finished Transaction PII Data field. The enforced range of this value is:
   - Min: 1 day
   - Recommended: 30 days
   - Max (Strict mode): 30 days
   - Max (Relaxed mode): 180 days

   Transaction PII data consists of user entered information, such as XML data, attachment data, PDF receipts, submission data extracts and submission properties, which takes up a large amount of space and poses some security risk due to containing user entered data. The transaction record itself will stay after data deletion for a period of time defined with Finished Transactions.

   > NOTE
   >
   > You can override this setting on both the organization and form levels.

6. Edit a number of days transaction records and any associated details are kept once a transaction has finished in the Finished Transactions field. If PII data has been deleted for a

submission, the header records remain in the system until this policy kicks in, allowing administrators to find the submission in the logs.

7. Edit the number of days in the Finished Collaboration Jobs field to specify how long collaboration jobs and their submissions are kept. The default is 90 days.
8. Edit the number of days in the Collaboration Jobs Abandonment field to specify how long to keep collaboration jobs before automatic abandonment. The enforced range of this value is:
   - Min: 1 day
   - Recommended: 30 days
   - Max (Strict mode): 180 days
   - Max (Relaxed mode): 180 days

   The Transaction Processor service abandons a collaboration job when either its Last Processed Time or Creation Time parameter expires, that is, when either time is greater than the above specified period of time. The default is 60 days.

   > NOTE
   >
   > We recommend understanding the abandonment process for collaboration jobs as well as transactions and tasks before configuring this property.

   | 22.04 This feature was introduced in 22.04.

   > NOTE
   >
   > You can disable collaboration job abandonment by setting the Max Collaboration Job Abandon Age Days to `0` in the Data Retention Management Service.

   > NOTE
   >
   > You can override this value on the organization level.

9. Edit the number of days to keep PII collaboration job data for after the collaboration job has been finished (Completed, Canceled, or Expired) in the Finished Collaboration Jobs PII Data field. The collaboration job PII data consists of job properties. The enforced range of this value is:
   - Min: 1 day
   - Recommended: 30 days

- Max (Strict mode): 30 days
- Max (Relaxed mode): 730 days

System Logs (max age)

10. Select a number of days to keep transaction history records, counting from the time the submission is started, from the Transaction History dropdown list. The default is 180 days.
11. Select how long an email queue item is kept from the Email Queue dropdown list. Time is counted from the moment the email is created. | 18.05 This feature was introduced in 18.05.
12. Select how long error log entries and data that are not associated with a submission are kept, counting from the time that the error occurred, from the Error Log dropdown list.
13. Select how long event log entries that are not associated with a submission are kept, counting from the time that the event was logged, from the Event Log dropdown list.
14. Select the amount of time in days when billing records become eligible for purging from the Billing Records dropdown list. Default is 60 days. | 24.10 This feature was introduced in 24.10
15. Select the amount of time in days when Event Inbox records (both Delivered and Failed) become eligible for purging from the Event Inbox dropdown list. Default is 3 days. | 24.10 This feature was introduced in 24.10
16. Select the amount of time in days when Event Outbox records (both Delivered and Failed) become eligible for purging from the Event Outbox dropdown list. Default is 3 days. | 24.10 This feature was introduced in 24.10
17. Select how long log entries for Groovy service invocations are kept, counting from invocation time, from the Groovy Service Log dropdown list.
18. Select how long scheduled job history entries, which are created every time a scheduled job runs, are kept from the Schedule Job History dropdown list.
19. Select how long audit log entries are kept, counting from the time the log was created, from the Security Audit Log dropdown list. The audit log can be crucial to tracking down changes made by administrators, so on non-test servers, we recommend allowing adequate time before purging this log. The default is 1 year.
20. Select the maximum age to keep Security Manager log records for from the Security Manager Log dropdown list. The default is 30 days.
21. Select the amount of time to keep T.Field App device log, sync log and sync call records for from the T.Field App Logs dropdown list. The default is 30 days.
22. Select how long events relating to user sessions, such as login, logout, session expiry, and account lock, are kept, from the User Login History dropdown list. The default is 180 days.

Data Retention Configs

23. Select the required purge mode from the PII Search Purge Mode dropdown list to specify when PII searching metadata, such as data extracts and transactions properties, will be

deleted:

- Form Completed: the smallest set of data to be deleted.
- Delivery Completed: a bigger set of data to be deleted.
- PII Purge Time: the biggest set of data to be deleted and it takes the longest time to do it.

The default is Delivery Completed.

24. Select the Purge Submission Error Log With PII checkbox to delete all PII[1] data of a submission from the error log.
25. Click Save to update the changes.
26. Click Apply Policy to apply the retention policies right away and permanently delete records exceeding the maximum age.
27. Click Reset Defaults to reset this configuration with the recommended default values.

| 24.10 This feature was introduced in 24.10

NOTE

Open the event log and search for events using data retention keywords to see the actual number of purged transactions of each category.

Next, learn how to configure organization data retention.

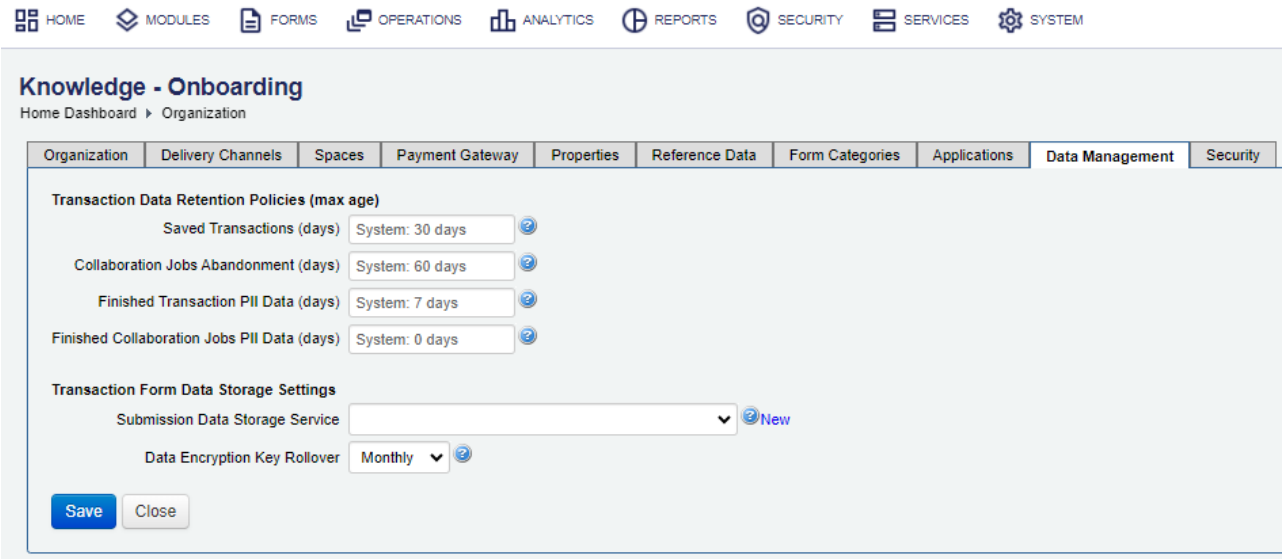
[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

Configure Organization Data Retention

By default, Manager applies global data retention configuration defined for the entire server to its organizations, unless an organization has its own data retention configured. However, you can customize retention policies for all transactions associated with forms, for example, saved and finished transaction's PII[1] data.

To configure organization's data retention:

1. Select Forms > Organizations.
2. Locate an organization and click Edit.
3. Select the Data Management tab to check and edit data retention settings.



> NOTE
>
> You can only update these settings if the Enforce System/Org Thresholds checkbox in the Retention Settings tab is selected and no values have been entered for each of these settings.

4. Edit the maximum age of user saved transactions, after which they will be automatically abandoned, in the Saved Transactions field. This value must not exceed the Saved

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual.

Transactions organization threshold, if one exists at the organization level (from the Data Management tab for the related organization), otherwise at the global level (from the Retention Settings tab). In this example, the value must not exceed the system active threshold of 30 days.

> NOTE
>
> You can also override this property using form abandonment configuration.

5. Edit the number of days in the Collaboration Jobs Abandonment field to specify how long to keep collaboration jobs before automatic abandonment. The default is 60 days. | 22.04
   This feature was introduced in 22.04.

   > NOTE
   >
   > You can use this property to override the value defined in global data retention configuration, but you can't set the value greater than the one defined in the global data retention configuration.

6. Edit Finished Transaction PII Data with a value which doesn't exceed the Finished Transaction PII Data organization threshold, if one exists at the organization level (from the Data Management tab for the related organization), otherwise at the global level (from the Retention Settings tab). In this example, the value must not exceed the organization active threshold of 7 days.

NOTE

However, when the Enforce System/Org Thresholds checkbox in the Retention Settings tab is not selected, no such input prompts are provided, and different constraints apply. From the Details tab, if you wish to override the global and/or organization Data Retention Management settings, enter custom values for:

- Edit Saved Transactions with a value which doesn't exceed the maximum value for the Saved Transaction setting as determined by the data retention management policy. That is, the value cannot exceed 180 days for a strict mode and cannot exceed 730 days (2 years) for a relaxed mode.
- Edit Finished Transaction PII Data with a value which doesn't exceed the maximum value for the Finished Transaction PII Data setting as determined by the data retention management policy. That is, the value cannot exceed 30 days for a strict mode and cannot exceed 730 days (2 years) for a relaxed mode.

7. Edit the in the Finished Collaboration Jobs PII Data (days) field.
8. Select a submission data storage service from the Submission Data Storage Service dropdown list.
9. Select a submission data encryption key rollover policy from the Data Encryption Key Rollover dropdown list, which can be one of the following:
   - Never
   - Annually
   - Monthly
   - Weekly. This is the default option
   - Daily
10. Click Save to update the changes.

Next, learn how to configure form data retention.

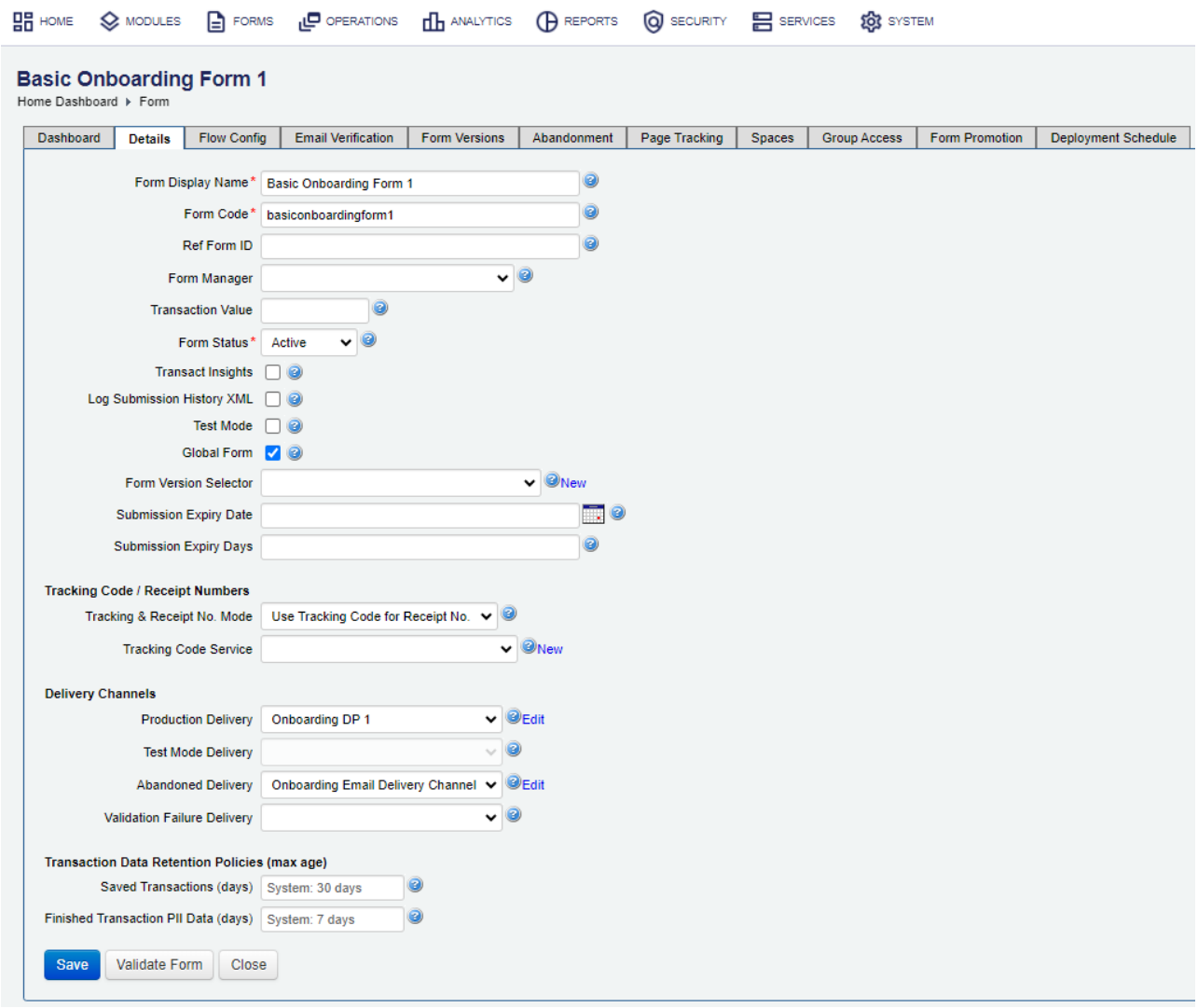
Configure Form Data Retention

Manager applies global data retention policies, defined for the entire server or individual organizations, to its forms. However, you can customize transaction retention policies on the form level, for example, saved and finished transaction's PII[1] data. These settings then apply to all submissions made for this form.

To configure form data retention:

1. Select Forms > Forms.
2. Locate a form and click Edit.
3. Select the Details tab to check and edit data retention policies in the Transaction Data Retention Policies section.

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual.



NOTE

You can only update this configuration if the Enforce System/Org Thresholds checkbox in the Retention Settings tab is selected and no values have been entered for each of these settings.

4. Edit Saved Transactions with a value which must not exceed the configured Saved Transactions organization's threshold, or the global threshold. In this example, the value must not exceed the system active threshold of 30 days.

NOTE

You can also configured the saved transaction threshold from the Abandonment tab of a form.

5. Edit Finished Transaction PII Data with a value which must not exceed the Finished Transaction PII Data organization's threshold, or the global threshold. In this example, the value must not exceed the organization active threshold of 4 days.

NOTE

If theEnforce System/Org Thresholds checkbox in the Retention Settings tab is not selected, no default values are provided, and different constraints apply. To override the global or organization data retention configuration:

- Edit Saved Transactions with a value which must not exceed the maximum value for the Saved Transaction setting of the data retention management policy, which is 180 days for a strict mode and 730 days (2 years) for a relaxed mode.
- Edit Finished Transaction PII Data with a value which must not exceed the maximum value for the Finished Transaction PII Data setting of the data retention management policy, which is 30 days for a strict mode and 730 days (2 years) for a relaxed mode.

6. Delete Form Data Extracts for relaxed mode only. | 5.1 This feature was removed in 5.1
7. Click Save to update the changes.

Next, learn how to configure organization data retention.

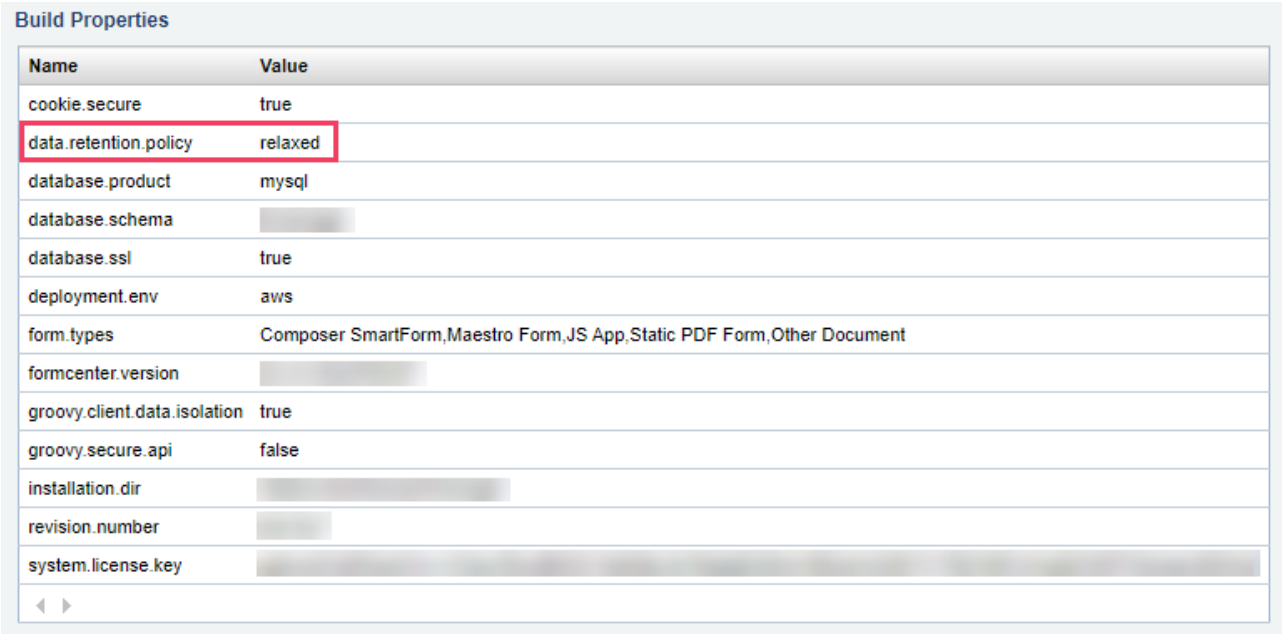
Data Retention Policy Mode

Manager can be installed using either strict or relaxed data retention policy mode. This determines the default values of the global data retention management configuration, which are used throughout an entire Manager server or an environment consisting of several server nodes. We recommend using the strict mode in production environments, while the relaxed mode in development and testing environments.

> NOTE
>
> As the data retention policy mode is selected during Manager installation or upgrade, you can't change it after wards, unless you wish to re-install software.

To check the data retention policy mode of your environment:

1. Select System > System Info.
2. Scroll down to the Build Properties section and look for `data.retention.policy`, as shown below:



Build Properties

Name	Value
cookie.secure	true
data.retention.policy	relaxed
database.product	mysql
database.schema	
database.ssl	true
deployment.env	aws
form.types	Composer SmartForm,Maestro Form,JS App,Static PDF Form,Other Document
formcenter.version	
groovy.client.data.isolation	true
groovy.secure.api	false
installation.dir	
revision.number	
system.license.key	

We recommend using the strict mode to constrain the potential amount of data retained within Manager by restricting the maximum age of transaction's data to be stored. It prevents problems related to long data retention, such as:

- Database growth
- Performance degradation

- Prolonged storage of sensitive PII[1] data

Both the strict and relaxed modes apply the global default settings to data retention configuration, however, you can override some of its values at the organization or form levels. The order in which data retention settings are applied is shown below:

- Global data retention configuration
- Organization's data retention configuration
- Form's data retention configuration

The default settings can be individually configured with a value that can't exceed the maximum value defined for each property.

> NOTE
>
> You can't change Submission Data Extracts and Submission Properties default values.

When you configure the data retention policy in Manager, you need to consider the following:

- Optimal conversion rate – a user must be given an adequate time to return to and complete a transaction.
- System efficiency – a lean database makes Journey Manager more responsive.
- Privacy – PII[2] data must be retained for the shortest possible time to minimize any potential security risks.

---

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.
[2]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.

NOTE

The Temenos Journey Manager platform is a System of Engagement[1] that enables you to create onboarding journeys for many types of products. It's strictly designed to store onboarding data for a short period of time, just long enough for a customer to complete their applications, and uses functionality, such as a configurable Data Retention policy to keep the amount of stored data low at any given time.
Temenos Journey Manager is not intended to be used as a System of Record[2] that stores large amount of data as this can affect system performance.

Manager is designed to keep data only for as long as it is required within the transaction life cycle. It is recommended to use the strictest possible data retention management policy to avoid excessive data base growth and potential problems during any recurring Manager upgrades.

However, often it is a balancing act between maintaining a lean, well performing and responsive system, while providing an adequate time frame for users to return to and complete their submitted forms. After sufficient time, given to users to complete their transactions, has elapsed, all incomplete transactions must be automatically abandoned.

You should ensure that Manager is correctly configured in a way that PII data is automatically purged from the transactions soon after the user has submitted their form to minimize any potential security risk. Therefore, we recommend using the strict data retention policy for any production Manager instances.

In some cases, you may want to change the data retention policy mode of your environment to test your data retention procedures. Switching relaxed to strict has no impact, but it all depends on how much you reduced retention days by. For example, if you want to have 180 days to start with and plan go back to 30 days after that, we recommend reducing days to 150, 120, 90, 60

---

[1]Systems of Engagement (SoE) is the technology used by an organization to help facilitate and orchestrate the customer journey via more personalized, seamless interactions across the various touchpoints. These include social media channels, email marketing platforms, mobile apps, and content management systems.
[2]System of Record (SOR) is an ISRS (information storage and retrieval system) that is the authoritative source for a particular data element in a system containing multiple sources of the same element. To ensure data integrity, there must be one -- and only one -- system of record for a given piece of information.

and then 30. Keep at least 1 day between these changes to allow the Transaction Processing scheduled job to purge records in small batches.

> NOTE
>
> If you have any questions regarding setting up the data retention policy mode, we recommend contacting our Customer Service team to discuss it before going to production.

Strict vs Relaxed Mode

There are differences between the strict and relaxed modes for the various data types. Let's look at each of them in more detail.

Saved Transactions

This setting determines how long to keep user saved transactions before they are abandoned by Manager and become finished transactions. The Manager environment setting is initially set to the default, but you can change it to any value that doesn't exceed the maximum. Organization and form level settings can be used to override the environment settings. However, they cannot exceed the maximum value.

- Strict mode: default is 30 days, maximum is 180 days
- Relaxed mode: default is 30 days, maximum is 2 years

Finished Transactions with PII Data

This setting determines how long to keep data entered by the user before it is deleted. The Manager environment setting is initially set to the default, but you can change it to any value that doesn’t exceed the maximum. Organization and form level settings can be used to override the environment settings. However, they cannot exceed the maximum value.

- Strict mode: default is 7 days, maximum is 30 days
- Relaxed mode: default is 7 days, maximum is 2 years

Finished Transactions

This setting determines how long to keep finished transactions before they are deleted. Finished transactions include transactions that have been marked as abandoned or completed. The Manager environment setting is initially set to the default, but you can change it to any value that doesn’t exceed the maximum. Organization and form level settings can be used to override the environment settings. However, they cannot exceed the maximum value.

- Strict mode: default is 90 days, maximum is 2 years
- Relaxed mode: default is 90 days, maximum is 5 years

Finished Collaboration Jobs

This setting determines how long to keep finished collaboration jobs before they are deleted. The Manager environment setting is initially set to the default, but you can change it to any value that doesn’t exceed the maximum . Organization and form level settings can be used to override the environment settings. However, they cannot exceed the maximum value.

- Strict mode: default is 90 days, maximum is 2 years
- Relaxed mode: default is 90 days, maximum is 2 years

Submission Data Extracts

This setting determines how long to keep submission data extracts before they are deleted. You can't override this setting, unless it's Manager version prior to 17.10 and it's in the relaxed mode.

- Strict mode: default is 7 days, maximum is 30 days
- Relaxed mode: default is 7 days, maximum is 2 years

Submission Properties

The setting determines how long to keep submission properties before they are deleted. You can't override this setting, unless it's Manager version prior to 17.10 and it's in the relaxed mode.

- Strict mode: default is 7 days, maximum is 30 days
- Relaxed mode: default is 7 days, maximum is 2 years

Next, learn about data retention schedule.

Data Retention Schedule

Manager comes with a mechanism for processing transactions and managing data retention. It executes the Transaction Processing scheduled job in the background every 5 minutes, which purges up to 500 records each time. This keeps data from accumulating and spreads out the processing load over time.

Transaction Processing scheduled job performs the following important role in the transaction life cycle:

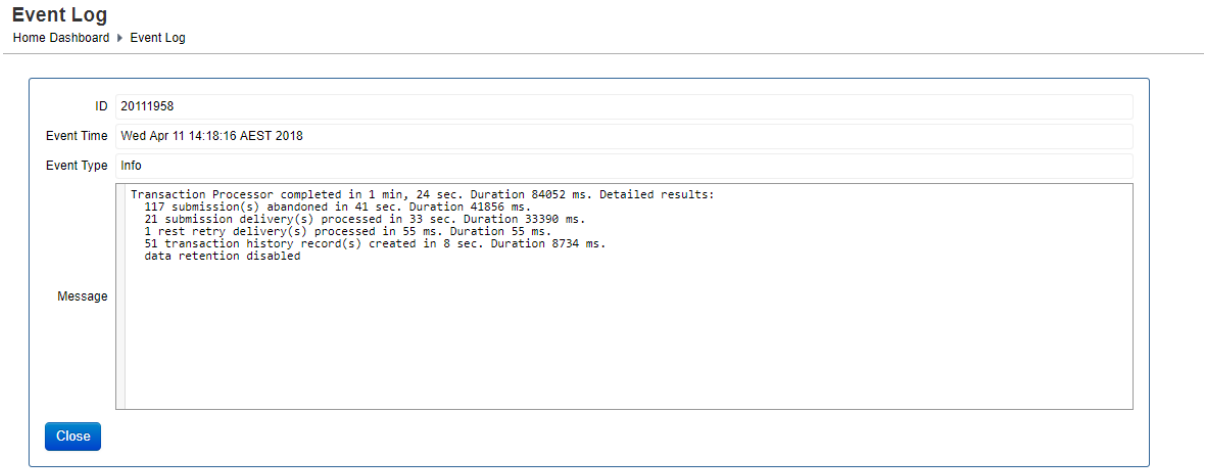
- Delivery
- Task expiry
- Abandonment
- Data purging by calling the Transaction Processor and the Data Retention Management services that purge data.

> NOTE
>
> To change the maximum number of purged records, see the Transaction Processor Service and the Data Retention Management Service documentation.

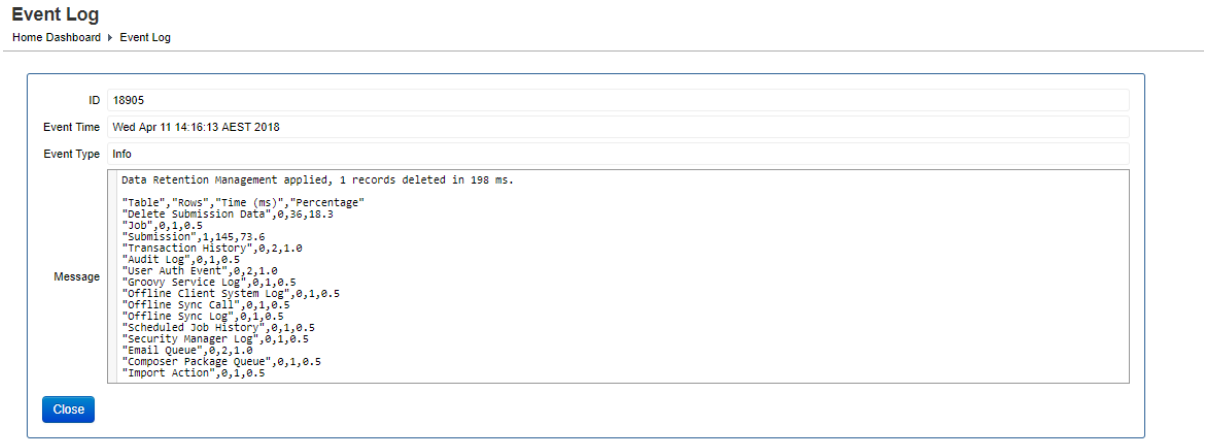
To view the Transaction Processing scheduled job as part of all scheduled jobs:

1. View scheduled jobs.
2. Search for Transaction Processing to check the following details:
    - Status
    - Type
    - Schedule
    - TZ
    - Next Run
    - Last Run
    - First Run
3. Check Data Retention Management audit information in the Event Log, which captures the following details:

- Transaction Processor service info details.



- Data Retention Management service info details.



Next, learn how to view transactions data retentions summary.

Manual Data Purge

Manager has an automated process in place that continuously applies data retention policies to all transaction data, including PII[1] data, and purges it accordingly. However, you can purge data of a selected transaction manually using UI. This will mark the transactions for immediate deletion so they will be removed from the system during the next run of the Transaction Processor service, which is usually run every 5 minutes by the Transaction Processing scheduled job.

> WARNING
>
> The data is not removed immediately, so check the Repeat Interval of the Transaction Processing scheduled job configuration in your environment.
> The data is not removed if the transaction is a part of a collaboration job, so you must manually purge it.

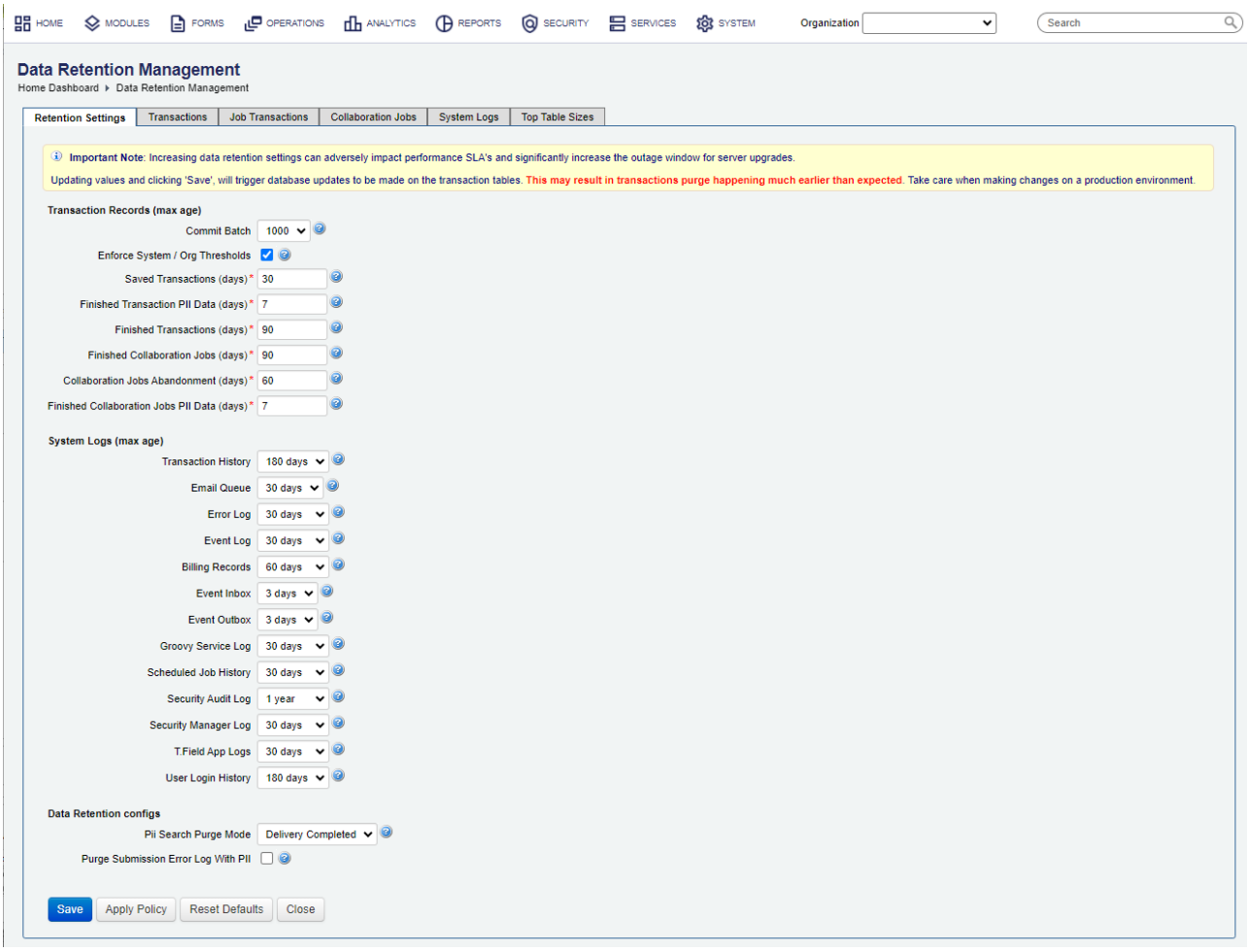
You can use this functionality to:

- Delete transaction's data as per customer's request.
- Free up server resources, such as OS file system or a database space, by delete old transactions.

To trigger a data purge manually:

1. Select System > Data Retention Management.
2. Click the Retention Settings tab to verify the current settings and adjust them so the manual purge removes data that needs to be removed.

[1]Personally Identifiable Information (PII) is information about an individual that can be used to distinguish or trace an individual‘s identity, such as name, social security number, date and place of birth, mother‘s maiden name, or biometric records; and any other information that is linked to an individual. In Europe, PII is known as personal data.



3. Click Apply Policy to run the Transaction Processor service, which purges data and displays a report with actual number of records been purged.

Next, learn about data retention policies.

Data Retention Summary

Manager keeps track of how much data is currently stored for each server instance and breaks it down by categories. This allows you to choose the most appropriate data retention policy and verify whether data has been purged as expected or not.

Manager provides the following data retention summaries for each category:

- Transactions
- Job transactions
- Collaboration jobs
- System logs
- Top table sizes

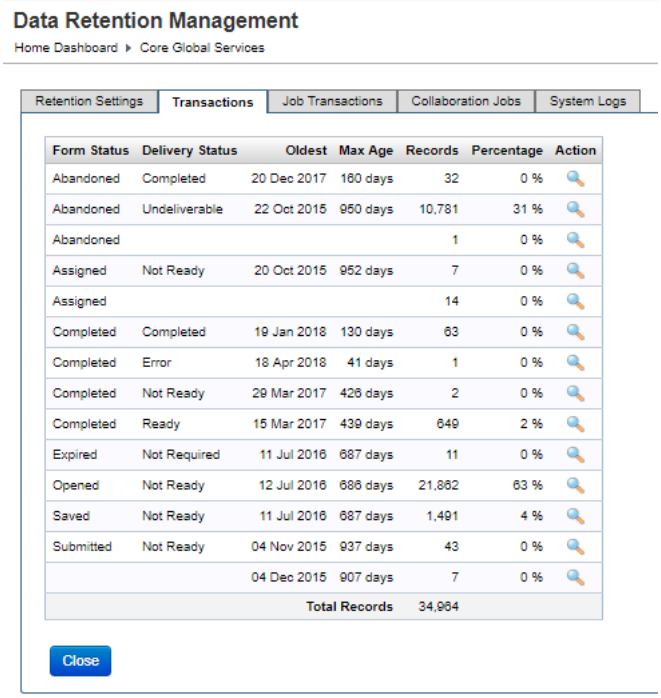
Next, learn about Data Retention Schedule.

Data Retention Summary: Transactions

Manager allows you to view data retention summary of transactions.

> NOTE
>
> Transactions don't include submissions associated with collaboration jobs. | 18.05 This feature was introduced in 18.05.

To view transactions data retention summary, select System > Data Retention Management and click the Transactions tab. The data usage summary is grouped by Form Status and Delivery Status.



Form Status	Delivery Status	Oldest	Max Age	Records	Percentage	Action
Abandoned	Completed	20 Dec 2017	160 days	32	0 %	
Abandoned	Undeliverable	22 Oct 2015	950 days	10,781	31 %	
Abandoned				1	0 %	
Assigned	Not Ready	20 Oct 2015	952 days	7	0 %	
Assigned				14	0 %	
Completed	Completed	19 Jan 2018	130 days	63	0 %	
Completed	Error	18 Apr 2018	41 days	1	0 %	
Completed	Not Ready	29 Mar 2017	426 days	2	0 %	
Completed	Ready	15 Mar 2017	439 days	649	2 %	
Expired	Not Required	11 Jul 2016	687 days	11	0 %	
Opened	Not Ready	12 Jul 2016	686 days	21,862	63 %	
Saved	Not Ready	11 Jul 2016	687 days	1,491	4 %	
Submitted	Not Ready	04 Nov 2015	937 days	43	0 %	
		04 Dec 2015	907 days	7	0 %	
			Total Records	34,964		

- Oldest - the date of the oldest transaction for this group.
- Max Age - the maximum age of the oldest transaction for this group.
- Records - the number of transaction records for this group.
- Percentage - the percentage transaction records for this group of the total number of transaction records.

- Abandoned: a transaction that has been abandoned by the user, or based on the data retention policy
- Assigned: a task which is waiting to be completed by a user
- Completed: a transaction that has been successfully submitted and contain all required data

- Opened: a transaction that has been started by a user. When a transaction is created by a user (by starting a new form), the initial form status is set to Opened. This status will be changed once the user saves, cancels, or submits the transaction.
- Saved: a transaction that has been saved by the user or automatically by the system
- Submitted: a transaction that has been submitted but may require additional information, such as attachments, payment, and so on. Once the required data is collected, the transaction status will change to Completed. This status is not common for Maestro forms as all information, such as attachments and payments, are collected prior to the user submitting the form. Most Maestro forms will skip the Submitted status and will be marked as Completed after the form is submitted by the user. This is a legacy feature.
- Expired: a form associated with this transaction has expired and is no longer available

A delivery status can be one of the following:

- Completed: a transaction has been successfully delivered, based on the configured delivery channel.
- Undeliverable: a delivery is not possible. Check the error logs for further details of the error.
- Not Ready: a transaction is not ready for delivery yet. This is the start state before a transaction has been completed by the user. Generally, the receipt render service will kick the transaction from this state to the Ready state when the receipt is rendered.
- Not Required: a delivery is not required. It may be that this form is part of a collaboration job and the delivery is to be handled separately say as a consolidation of the transaction details from all the associated forms.
- Ready: a transaction is ready to be delivered. A transaction will be marked as Ready until the delivery channel is processed, or it may remain marked as Ready if a delivery channel is not configured.
- In Progress: a delivery has been started but is waiting to be completed.
- Paused: a transaction delivery is paused. This transaction's status can be changed by some other means, for example by a Groovy Scheduled Task.
- Error: a transaction delivery has failed and is waiting to be retried.
- Sent Email: a secure email has been sent and now a transaction is waiting for the user to log in and process it.
- Pending: a transaction delivery is pending.

Only transactions with the following combinations of statuses are purged:

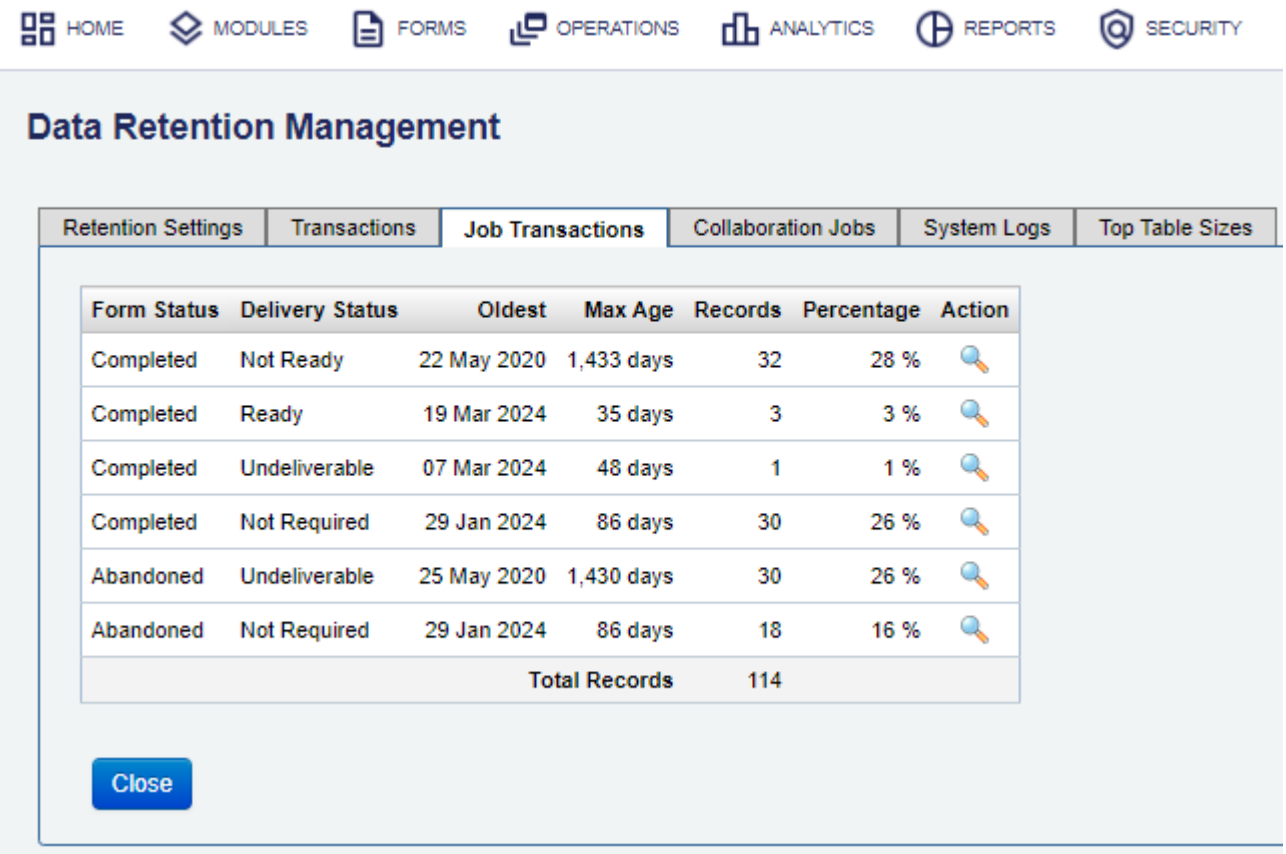
- Form Status: Completed or Abandoned,
- Delivery Status: Completed, Undeliverable or Not Required.

Next, learn how to view job transactions data retention summary.

Data Retention Summary: Job Transactions

Manager allows you to view data retention summary of job transactions.

To view data retention summary, select System > Data Retention Management and click the Job Transactions tab. The data usage summary is grouped by Form Status and Delivery Status.



Form Status	Delivery Status	Oldest	Max Age	Records	Percentage	Action
Completed	Not Ready	22 May 2020	1,433 days	32	28 %	
Completed	Ready	19 Mar 2024	35 days	3	3 %	
Completed	Undeliverable	07 Mar 2024	48 days	1	1 %	
Completed	Not Required	29 Jan 2024	86 days	30	26 %	
Abandoned	Undeliverable	25 May 2020	1,430 days	30	26 %	
Abandoned	Not Required	29 Jan 2024	86 days	18	16 %	
			Total Records	114		

- Oldest - the date of the oldest transaction for this group.
- Max Age - the maximum age of the oldest transaction for this group.
- Records - the number of transaction records for this group.
- Percentage - the percentage transaction records for this group of the total number of transaction records.

- Abandoned: a transaction that has been abandoned by the user, or based on the data retention policy
- Assigned: a task which is waiting to be completed by a user
- Completed: a transaction that has been successfully submitted and contain all required data

- Opened: a transaction that has been started by a user. When a transaction is created by a user (by starting a new form), the initial form status is set to Opened. This status will be changed once the user saves, cancels, or submits the transaction.
- Saved: a transaction that has been saved by the user or automatically by the system
- Submitted: a transaction that has been submitted but may require additional information, such as attachments, payment, and so on. Once the required data is collected, the transaction status will change to Completed. This status is not common for Maestro forms as all information, such as attachments and payments, are collected prior to the user submitting the form. Most Maestro forms will skip the Submitted status and will be marked as Completed after the form is submitted by the user. This is a legacy feature.
- Expired: a form associated with this transaction has expired and is no longer available

A delivery status can be one of the following:

- Completed: a transaction has been successfully delivered, based on the configured delivery channel.
- Undeliverable: a delivery is not possible. Check the error logs for further details of the error.
- Not Ready: a transaction is not ready for delivery yet. This is the start state before a transaction has been completed by the user. Generally, the receipt render service will kick the transaction from this state to the Ready state when the receipt is rendered.
- Not Required: a delivery is not required. It may be that this form is part of a collaboration job and the delivery is to be handled separately say as a consolidation of the transaction details from all the associated forms.
- Ready: a transaction is ready to be delivered. A transaction will be marked as Ready until the delivery channel is processed, or it may remain marked as Ready if a delivery channel is not configured.
- In Progress: a delivery has been started but is waiting to be completed.
- Paused: a transaction delivery is paused. This transaction's status can be changed by some other means, for example by a Groovy Scheduled Task.
- Error: a transaction delivery has failed and is waiting to be retried.
- Sent Email: a secure email has been sent and now a transaction is waiting for the user to log in and process it.
- Pending: a transaction delivery is pending.

Only transactions with the following combinations of statuses are purged:

- Form Status: Completed or Abandoned,
- Delivery Status: Completed, Undeliverable or Not Required.

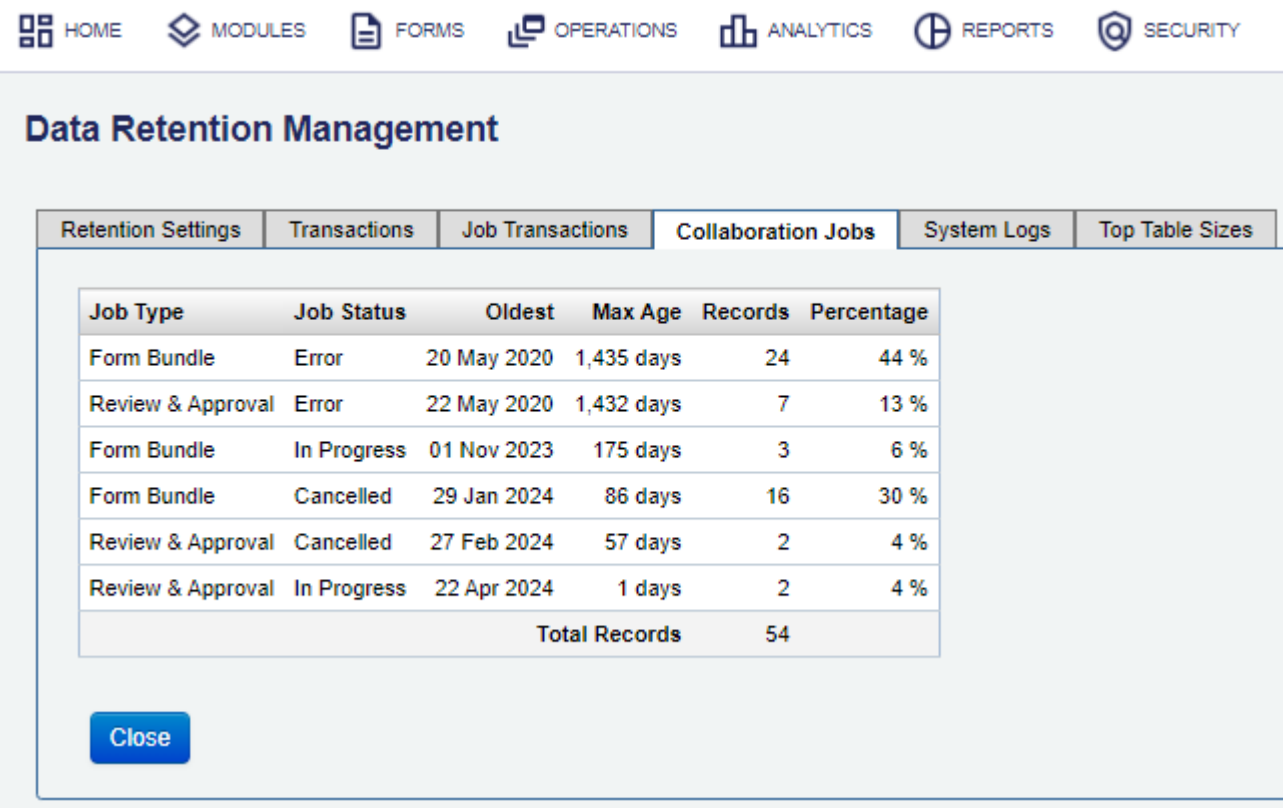
Next, learn how to view collaboration job data retention summary.

Data Retention Summary : Collaboration Jobs

Manager allows you to view data retention summary of collaboration jobs.

> NOTE
>
> Only collaboration jobs with the Completed, Cancelled, and Expired combinations of statuses are purged:

To view collaboration jobs data retention summary, select System > Data Retention Management and click the Collaboration Jobs tab.



The collaboration job list displays the following details grouped by a Job Type and Job Status:

- Job Type - a job type, for example, Form Bundle, Review or Approval.
- Job Status - a job status, which can be one of the following:
  - In Progress - the collaboration job is active and in progress.
  - Completed - the collaboration job has been completed. This happens when a endpoint step has completed.
  - Expired - this happens when an endpoint expiry collaboration job has been automatically expired by Journey Manager.

    - Cancelled - the collaboration job has been manually canceled by an administrator.
    - Error - the collaboration job has a critical or unrecoverable error that requires an administrator intervention before it can be run again.
    - Paused - the collaboration job has been paused temporarily.
- Oldest - the date of the oldest transaction for this group.
- Max Age - the maximum age of the oldest transaction for this group.
- Records - the number of transaction records for this group.
- Percentage - the percentage transaction records for this group of the total number of transaction records.

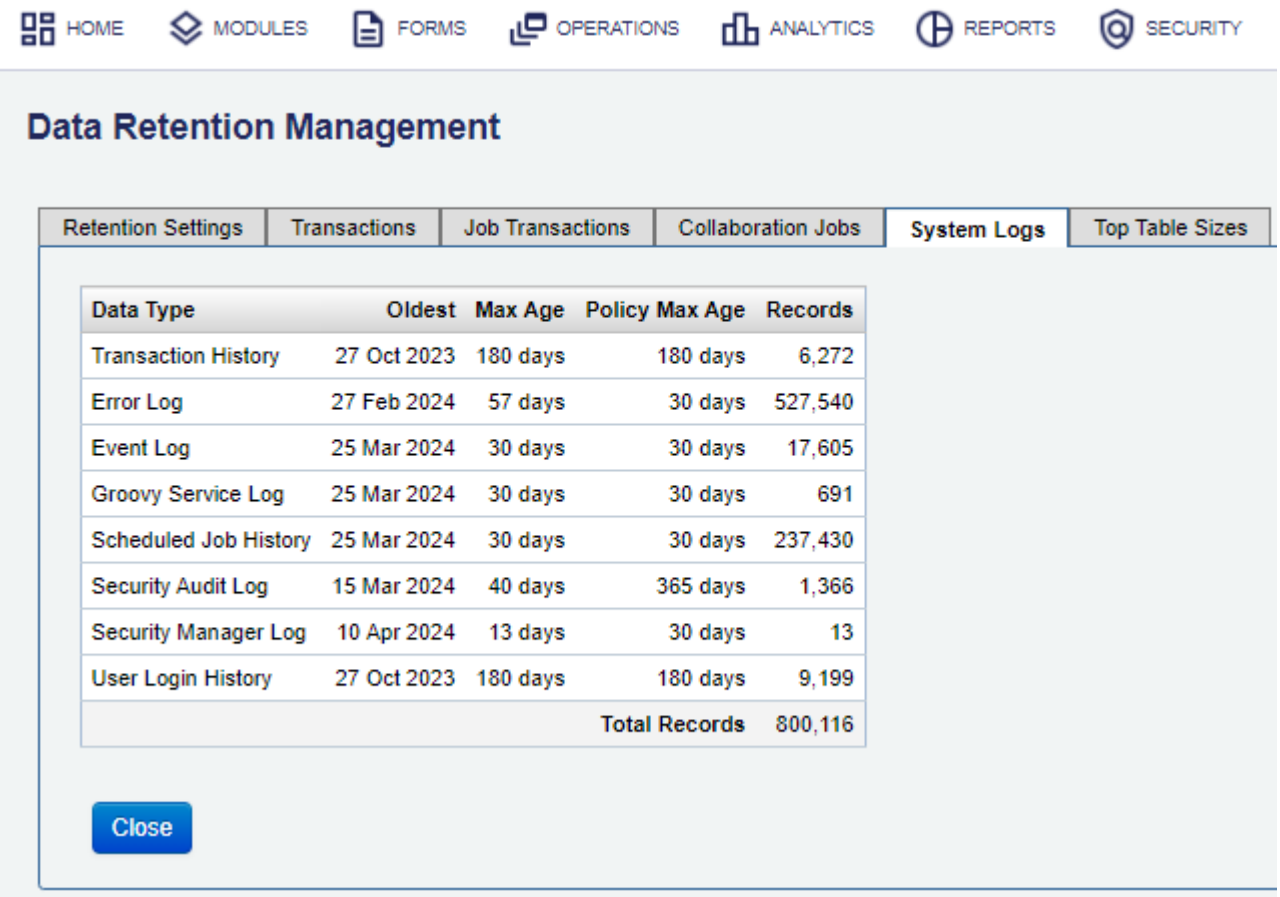
Next, learn how to view system log data retention summary.

Data Retention Summary: System Logs

Manager allows you to view data retention summary of system logs.

System log records associated with a transaction are normally purged based on the standard data retention policy, as long as the transaction had already been purged. If not, it must wait for the transaction to be purged. If there are existing old transactions, then it is possible that the related system log entries will show the Max Age being greater than the Policy Max Age.

To view system logs data retention summary, select System > Data Retention Management and click the System Logs tab. The data usage summary is .



Data Type	Oldest	Max Age	Policy Max Age	Records
Transaction History	27 Oct 2023	180 days	180 days	6,272
Error Log	27 Feb 2024	57 days	30 days	527,540
Event Log	25 Mar 2024	30 days	30 days	17,605
Groovy Service Log	25 Mar 2024	30 days	30 days	691
Scheduled Job History	25 Mar 2024	30 days	30 days	237,430
Security Audit Log	15 Mar 2024	40 days	365 days	1,366
Security Manager Log	10 Apr 2024	13 days	30 days	13
User Login History	27 Oct 2023	180 days	180 days	9,199
			Total Records	800,116

The log list displays the following details grouped by a log file type:

- Oldest - the date of the oldest transaction for this group.
- Max Age - the maximum age of the oldest transaction for this group.
- Policy Max Age - the policy maximum age as specified in the data retention settings for your server instance.
- Records - the number of transaction records for this group.

- Percentage - the percentage transaction records for this group of the total number of transaction records.

Next, learn how to view top table size data retention summary.

Data Retention Summary: Top Table Sizes

Manager allows you to view current top table sizes data summary.

> NOTE
>
> This information is only available if a Manager instance uses a MySQL[1] DB.

To view top table sizes data retention summary, select System > Data Retention Management and click the Top Table Sizes tab.

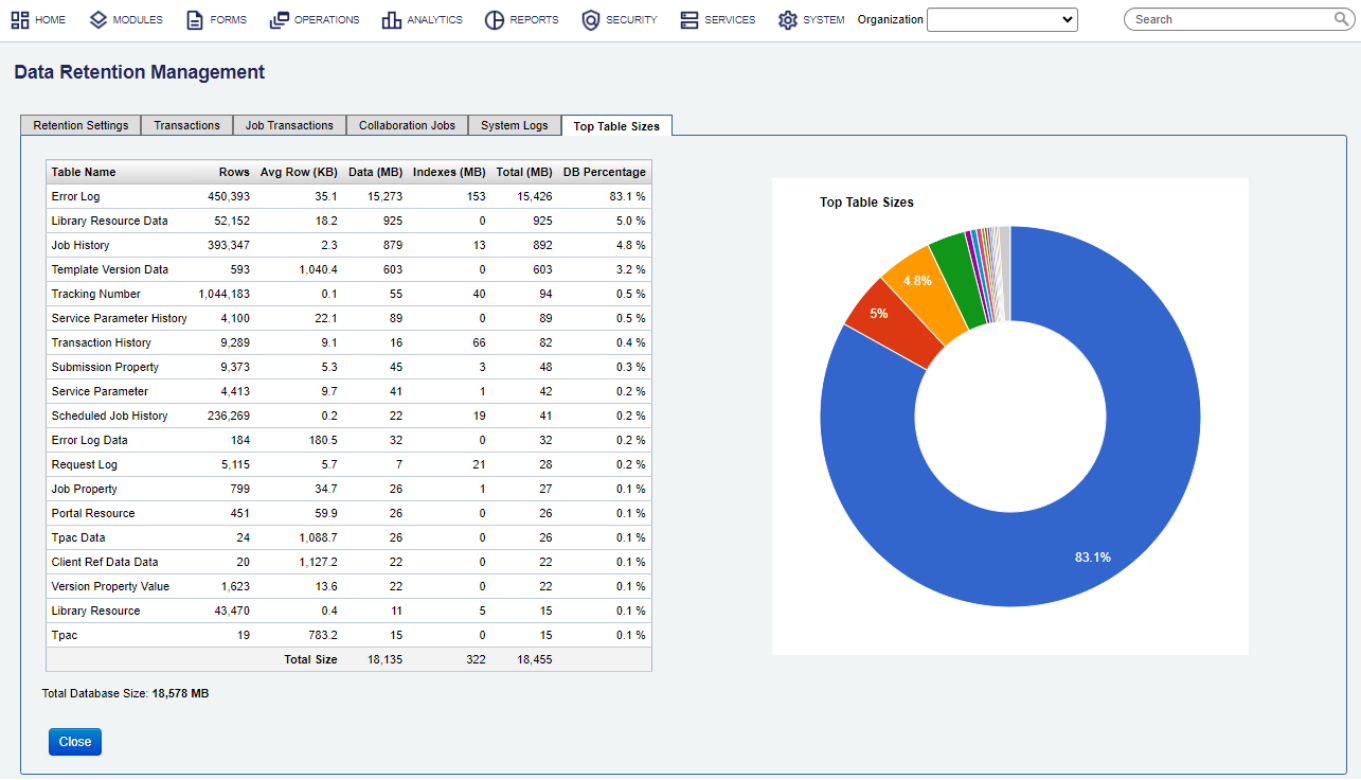


Table Name	Rows	Avg Row (KB)	Data (MB)	Indexes (MB)	Total (MB)	DB Percentage
Error Log	450,393	35.1	15,273	153	15,426	83.1 %
Library Resource Data	52,152	18.2	925	0	925	5.0 %
Job History	393,347	2.3	879	13	892	4.8 %
Template Version Data	593	1,040.4	603	0	603	3.2 %
Tracking Number	1,044,183	0.1	55	40	94	0.5 %
Service Parameter History	4,100	22.1	89	0	89	0.5 %
Transaction History	9,289	9.1	16	66	82	0.4 %
Submission Property	9,373	5.3	45	3	48	0.3 %
Service Parameter	4,413	9.7	41	1	42	0.2 %
Scheduled Job History	236,269	0.2	22	19	41	0.2 %
Error Log Data	184	180.5	32	0	32	0.2 %
Request Log	5,115	5.7	7	21	28	0.2 %
Job Property	799	34.7	26	1	27	0.1 %
Portal Resource	451	59.9	26	0	26	0.1 %
Tpac Data	24	1,088.7	26	0	26	0.1 %
Client Ref Data Data	20	1,127.2	22	0	22	0.1 %
Version Property Value	1,623	13.6	22	0	22	0.1 %
Library Resource	43,470	0.4	11	5	15	0.1 %
Tpac	19	783.2	15	0	15	0.1 %
		Total Size	18,135	322	18,455	

The table list displays the following details of the 19 largest tables listed by their size in descending order:

- Rows - the number of transaction records for this table.
- Avg Row (KB) - the average row size in kilobytes for this table.

---

[1]MySQL is an open-source relational database management system (RDBMS). It's an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web applications and online publishing.

- Data (MB) - the data size in megabytes for this table.
- Indexes (MB) - the indexes size in megabytes for this table.
- Total (MB) - the total data size in megabytes for this table.
- DB Percentage - the percentage of total data size in megabytes for this table of the total megabytes for the top table sizes.

Next, learn how to view transactions data retention summary.

Server Nodes Overview

Journey Manager uses a scalable architecture based on server nodes. Each server node corresponds to a Java Virtual Machine[1] (JVM) instance where a Manager server is installed. Each server nodes can be configured to host a combination of the Manager console, host form spaces, and operate as background PDF receipt generators.

Manager allows you to configure a single server node to host multiple servers. For example, a single server node can be configured to host both the form space server and the Manager server. Receipt generation is handled by an individual server node, but you can change this configuration if required.

Manager creates default server nodes when it is initially installed. For example, a server node dedicated to PDF receipt generation is created when Manager is installed. Though these server nodes are installed automatically, it is still important that you check and confirm the configuration, including the IP address and credentials, of every server node to ensure that each of them operates as expected in the Manager environment.

NOTE

All server nodes share the same database.

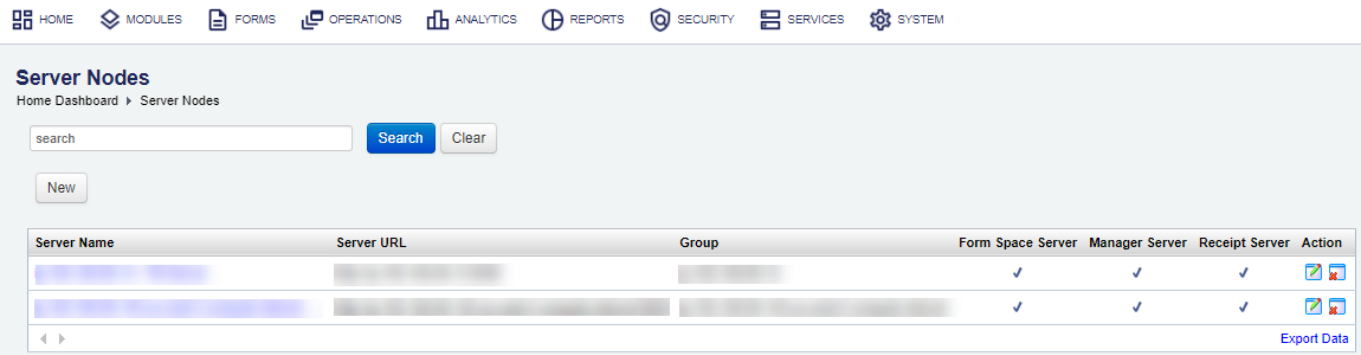
Next, learn how to view server nodes.

[1] Java Virtual Machine (JVM) is a software-based machine that runs Java programs. It can be installed on several different operating systems, including Windows, OS X, and Linux. JVMs allow Java apps to run on almost any computer.

View Server Nodes

Manager allows you to view, create, configure, remove, and test server nodes.

To view the list of server nodes, select System > Server Nodes.



The list displays the following details:

- Server Name: a name that uniquely identifies the server node.
- Server URL: a URL of to the server node.
- Group: the local data server group to where the server node belongs.
- Form Space Server: a form space server that handles and hosts web traffic and form spaces.
- Manager Server: a Manager admin dashboard server that supports development and operations staff.
- Receipt Server: a background PDF receipt generation server.

To search for a server node, type a text relevant to its configuration, for example `9080`, and click Search.

Click Clear to reset the search criteria.

You can also do the following from this page:

- Create a server node.
- Configure a server node.
- Remove a server node.
- Check Health Status

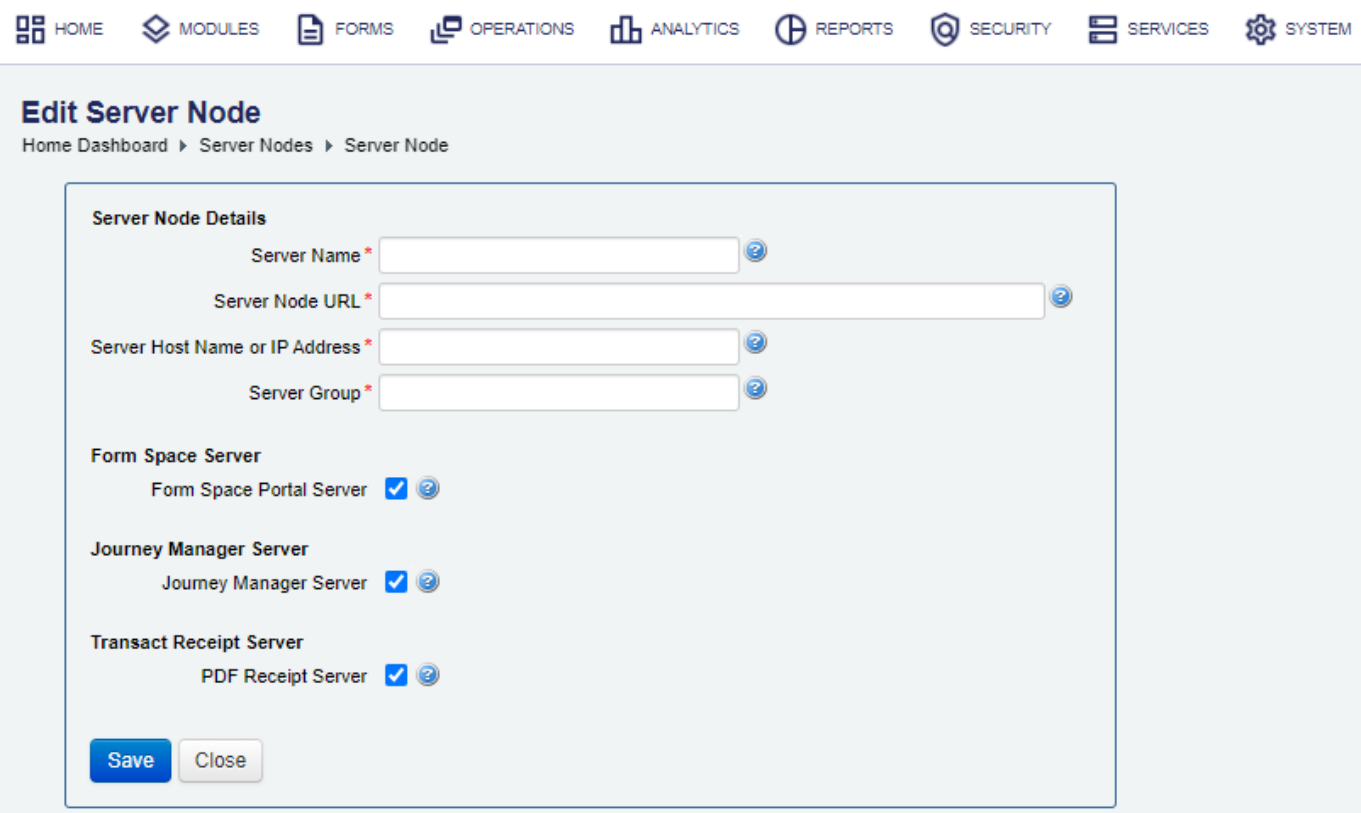
Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

Next, learn how to create a server node.

Create a Server Node

Manager creates server nodes during its installation, but you can add new server nodes at any time when required. Typically, you create server nodes for specific purposes such as to host form spaces or generate PDF receipts.

To create a new server node, select System > Server Nodes and click New.



Next, learn how to configure a server node.

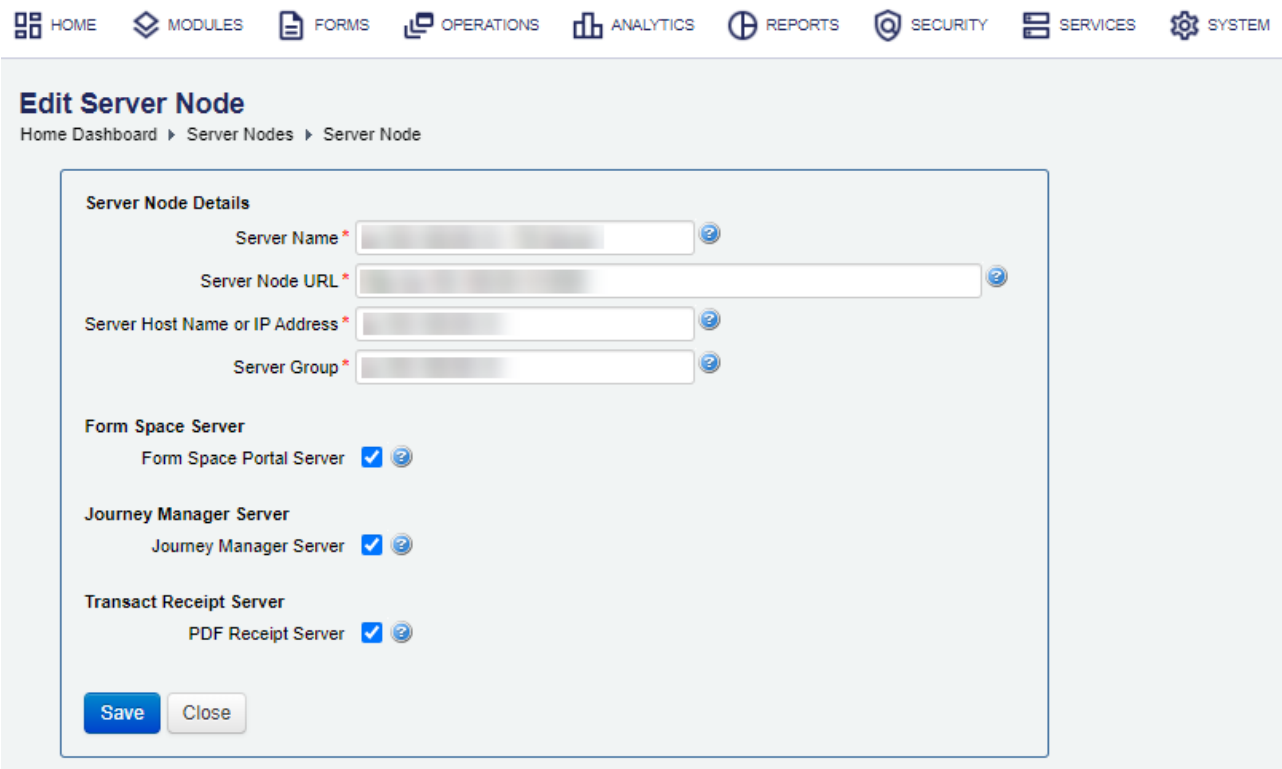
Configure a Server Node

Manager allows you to configure server nodes, which you need to do after a server node is created.

> NOTE
>
> You can configure a server node to host one or several Manager servers.

To configure a server node:

1. Select System > Server Nodes.
2. Locate a service node and click Edit.



3. Provide a unique server name in the Server Name field. For example, `my-server-node-1`.
4. Provide an URL of the server node in the Server Node URL field. For example, `http://my-server-node-domain-name:9080`. You can use HTTP protocol (not HTTPS) because it's for internal communication between server nodes.
5. Specify a server host name or IP address in the Server Host Name or IP Address field.
6. Specify a server group where the server node belongs in the Server Group field.
7. Select the Form Space Portal Server checkbox if a server node handles public facing Internet traffic. It is not selected by default.

8. Select the Journey Manager Server checkbox to mark it as a Manager admin dashboard server that supports development and operations staff.
9. Select the PDF Receipt Server checkbox to mark it as a background PDF receipt generator.
10. Click Save to update the changes.

Next, learn how to remove a server node.

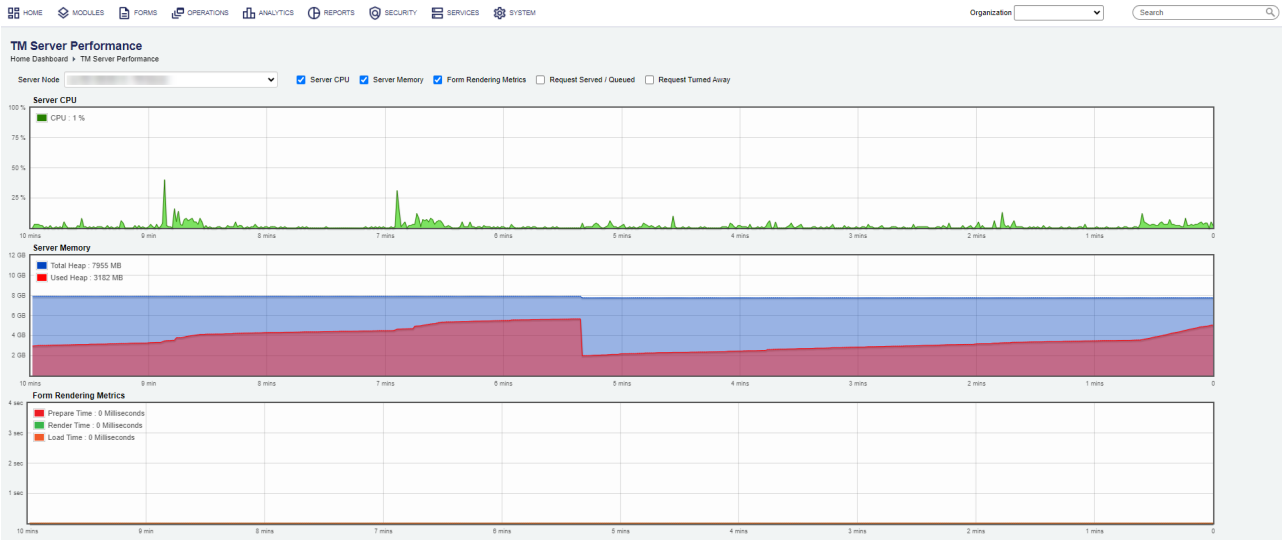
Monitor Server Node Performance

Manager provides graphical visualization of a server node real-time performance, which you can use to troubleshoot server performance issues.

> NOTE
>
> The performance graphs have 10 minutes time span, which can't be changed.

To view server node performance information:

1. Select Operations > JM Server Performance.
2. Choose a server node from the dropdown list.



3. Select all performance metrics you wish to visualize:
   - Server CPU
   - Server Memory
   - Form Rendering Metrics
   - Request Served / Queued
   - Request Turned Away

Next, learn how to view server nodes.

Remove a Server Node

Manager allows you to remove a server node.

> NOTE
>
> We recommend changing a server node mode to inactive rather than removing it.

To remove a server node:

1. Select System > Server Nodes.
2. Locate a server node.
3. Click Remove.
4. Click OK to confirm your decision.
5. Verify the server node has been removed from the list of server nodes.

Next, learn how to create a server node.

System Info

Manager allows you to see some important server's information, which you can use to troubleshoot server performance issues.

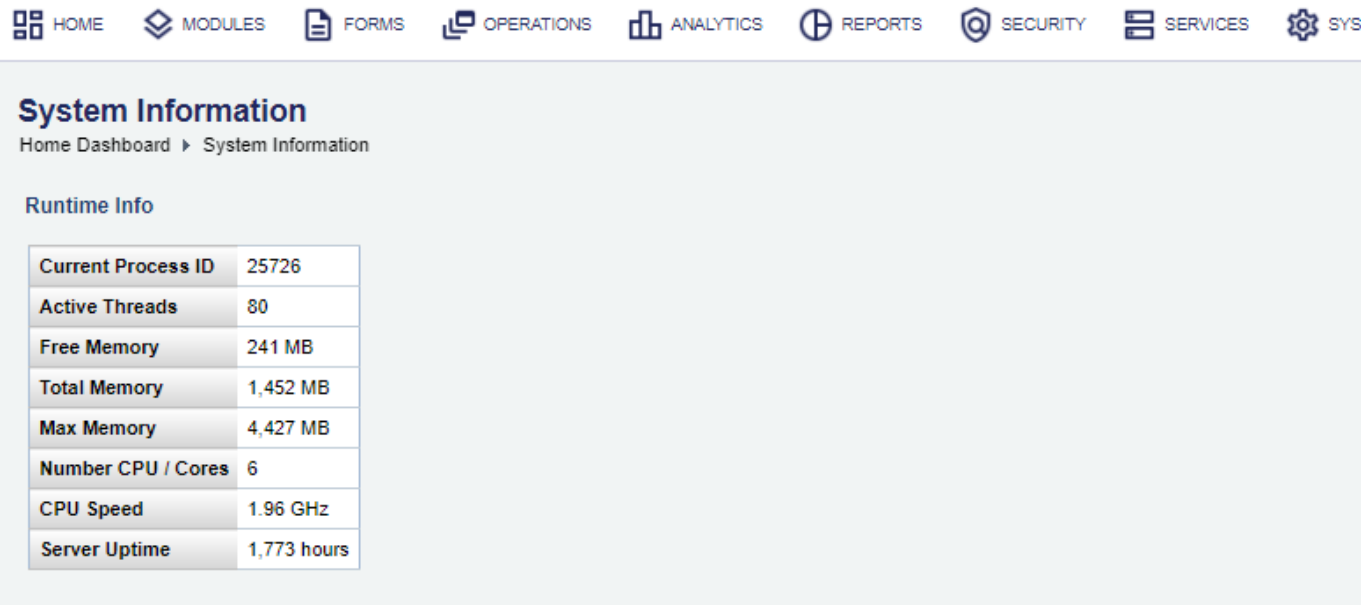
To view the server's system information, select System > System Info.

The system information is broken down into the following sections:

- Runtime Info
- Build Properties
- System Properties

Runtime Info

This section shows the current runtime load of the server, which is useful to check the server performance.



Build Properties

This section shows the server's build and installation configuration.

Build Properties

Name	Value
cookie.secure	true
data.retention.policy	relaxed
database.product	mysql
database.schema	
database.ssl	true
deployment.env	aws
form.types	Composer SmartForm,Maestro Form,JS App,Static PDF Form,Other Document
formcenter.version	
groovy.client.data.isolation	true
groovy.secure.api	false
installation.dir	
revision.number	
system.license.key	

Some important properties are:

`groovy.client.data.isolation`
: If set to `true`, Manager uses a client's organization to execute the Groovy Script for users who don't have global access.

`deployment.env`
: Shows the deployment mode of Manager, which was chosen during the installation. For example, `aws` means the AWS Cloud deployment.

System Properties

This section shows the server's Java[1] and Groovy[2] configuration.

---

[1]Java is a programming language that produces software for multiple platforms. When a programmer writes a Java application, the compiled code (known as bytecode) runs on most operating systems (OS), including Windows, Linux and Mac OS. Java derives much of its syntax from the C and C++ programming languages.

[2]Groovy is a powerful scripting language which runs on the Java Virtual Machine. Across theTemenos Journey Manager platform, Groovy is used to create services in Manager. These services are primarily used to create plug-in style systems that interact with Manager.

System Properties

Name	Value
[Standalone]	
cayenne.server.contexts_sync_strategy	false
com.amazonaws.services.s3.enableV4	true
com.avoka.component.httpclient.connectTimeout	10000
com.avoka.component.httpclient.socketTimeout	60000
com.avoka.core.cayenne.query.cache.enabled	true
com.avoka.core.cayenne.query.cache.size	2048
com.avoka.core.cayenne.query.cache.timeout	30
com.avoka.core.cayenne.query.profiling.detail	false
com.avoka.core.cayenne.query.profiling.enabled	false
com.avoka.core.cayenne.query.profiling.threshold	30000
com.avoka.core.cayenne.query.timeout	600
com.avoka.core.groovy.script.cache.timeout	3600
com.avoka.core.util.XmlUtils.enableDoctypeProcessing	false
com.avoka.core.util.XmlUtils.stripSurrogateCharacters	true
com.avoka.fc.admin.disableGroovyConsole	false
com.avoka.fc.core.event.processor.threads	2
com.avoka.fc.core.event.queue.size	100000
com.avoka.fc.core.service.phantomjs.delFiles	true
com.avoka.fc.core.service.reporting.logErrorEvent	true

1-20 of 102

Some important properties are:

`user.timezone`
: Checks the server's timezone, which is important for correct database configuration, see configure timezone for MySQL database.

In some cases, you need to change some of the System Properties to implement your specific requirements or adjust Manager performance. You can do this via the Fluent API. For example, the Fluent API's `HttpRequest` base class has `setSocketTimeout()` and `setConnectTimeout()` methods, so you can set timeout values after creating a `GetRequest` or `PostRequest`. However, in most cases, you need to contact your System Administrator to request the change.

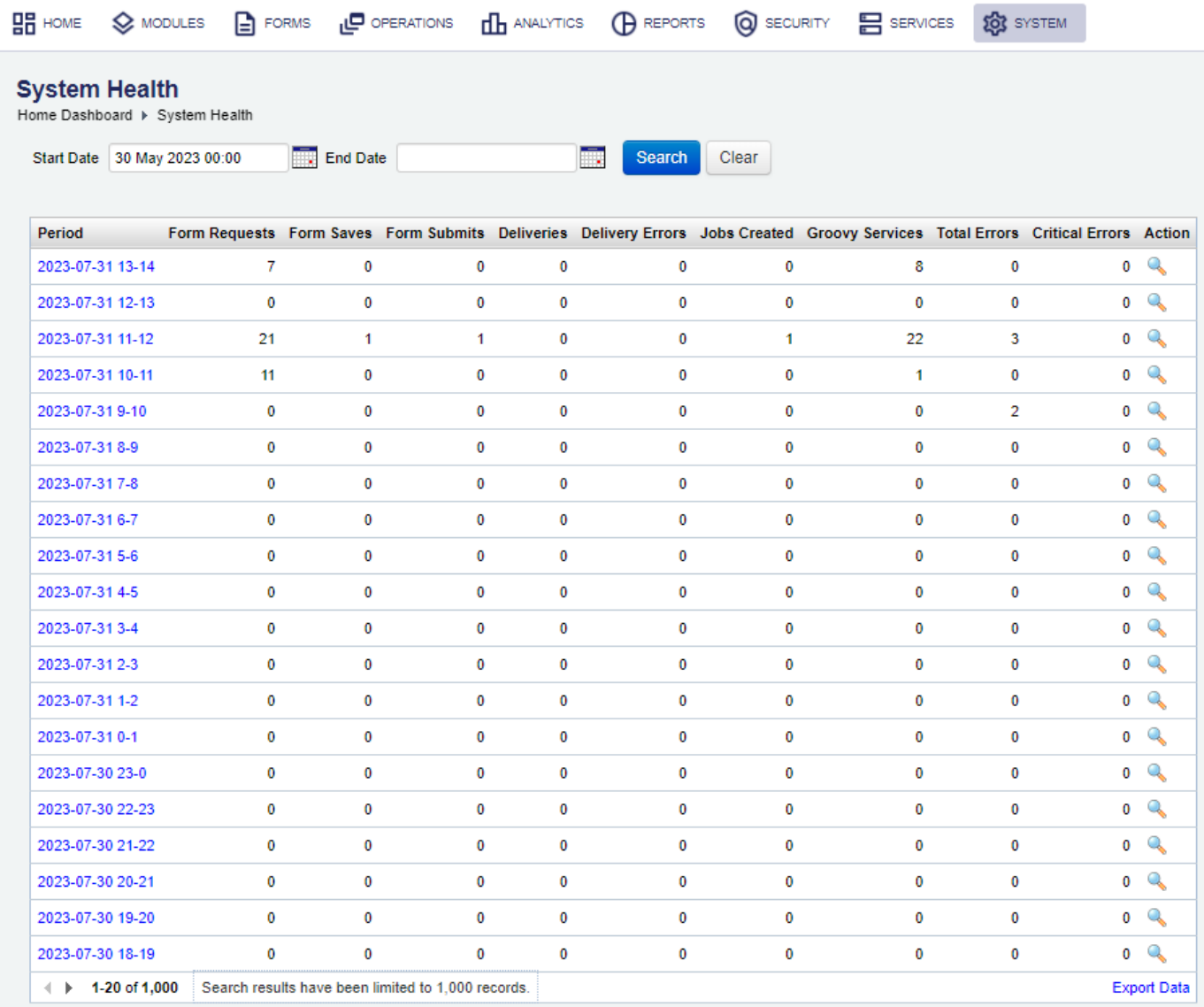
Next, learn how to check system health.

System Health

Manager allows you to view server's system health information that includes transaction statistics, system info, and error log. Some of this information can be sent to external systems using System Event Publisher System Health Events.

To view system health, select System > System Health.

System health records are shown page by page, with the maximum number of records per page configured in the users preferences.



Period	Form Requests	Form Saves	Form Submits	Deliveries	Delivery Errors	Jobs Created	Groovy Services	Total Errors	Critical Errors	Action
2023-07-31 13-14	7	0	0	0	0	0	8	0	0	
2023-07-31 12-13	0	0	0	0	0	0	0	0	0	
2023-07-31 11-12	21	1	1	0	0	1	22	3	0	
2023-07-31 10-11	11	0	0	0	0	0	1	0	0	
2023-07-31 9-10	0	0	0	0	0	0	0	2	0	
2023-07-31 8-9	0	0	0	0	0	0	0	0	0	
2023-07-31 7-8	0	0	0	0	0	0	0	0	0	
2023-07-31 6-7	0	0	0	0	0	0	0	0	0	
2023-07-31 5-6	0	0	0	0	0	0	0	0	0	
2023-07-31 4-5	0	0	0	0	0	0	0	0	0	
2023-07-31 3-4	0	0	0	0	0	0	0	0	0	
2023-07-31 2-3	0	0	0	0	0	0	0	0	0	
2023-07-31 1-2	0	0	0	0	0	0	0	0	0	
2023-07-31 0-1	0	0	0	0	0	0	0	0	0	
2023-07-30 23-0	0	0	0	0	0	0	0	0	0	
2023-07-30 22-23	0	0	0	0	0	0	0	0	0	
2023-07-30 21-22	0	0	0	0	0	0	0	0	0	
2023-07-30 20-21	0	0	0	0	0	0	0	0	0	
2023-07-30 19-20	0	0	0	0	0	0	0	0	0	
2023-07-30 18-19	0	0	0	0	0	0	0	0	0	

The System Health list displays the following details:

- Period - a time span system health has been collected. Click the link to view more details.
- Form Requests - a number of form requests occurred during the period of time

- Form Saves - a number of form saves occurred during the period of time
- Form Submits - a number of form submits occurred during the period of time
- Deliveries - a number of form deliveries occurred during the period of time
- Delivery Errors - a number of delivery errors occurred during the period of time
- Jobs Created - a number of collaboration job created during the period of time
- Groovy Services - a number of Groovy services run during the period of time
- Total Errors - a number of all errors occurred during the period of time
- Critical Errors - a number of critical errors occurred during the period of time

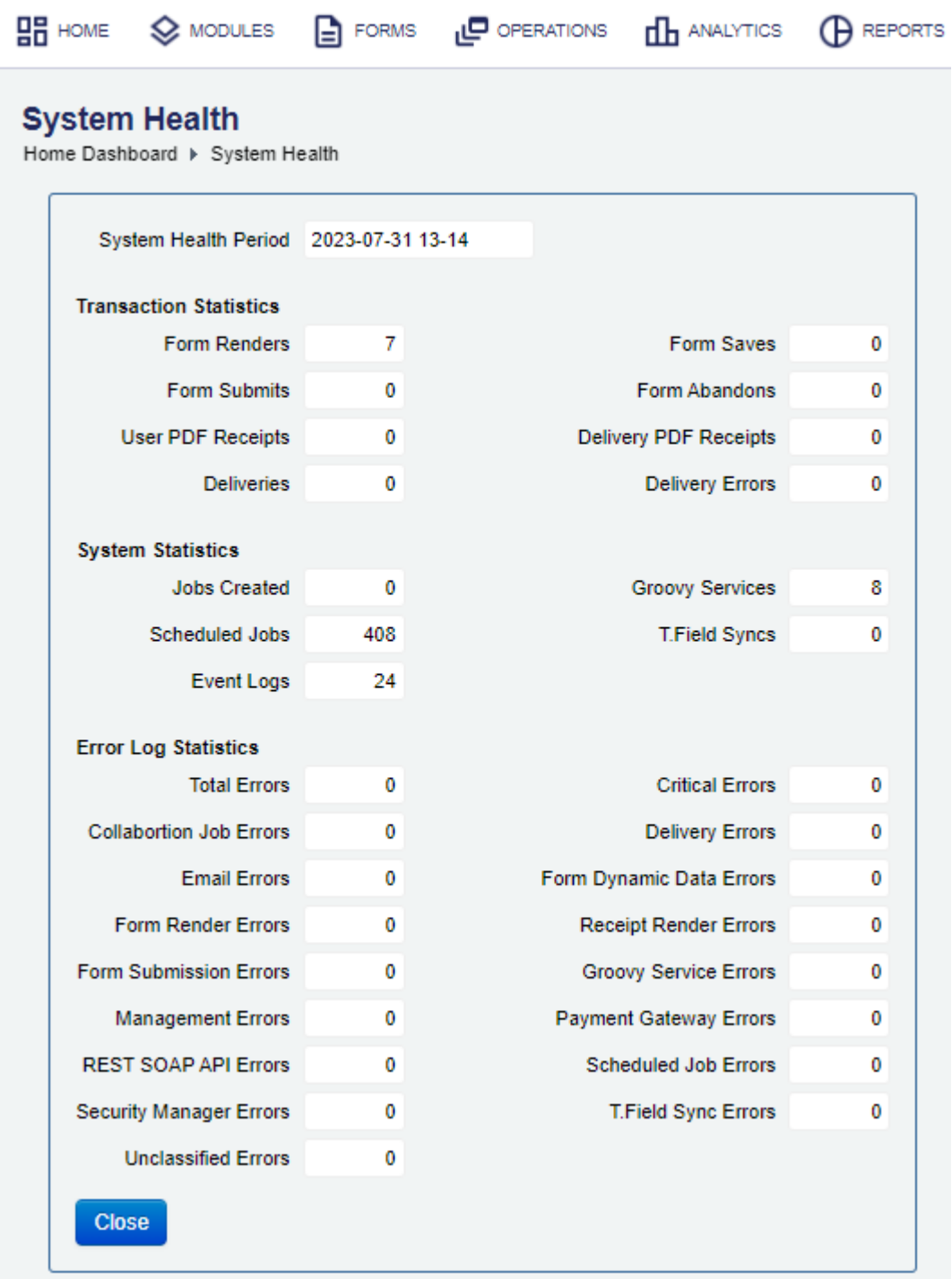
To filter or search system health records by one or more criteria, specify the following settings and click Search:

- Start Date.
- End Date.

Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

To view additional information, click View. The example below shows system health for a specified period.



The System Health record displays the following details:

- Form Renders -
- Form Submits -
- User PDF Receipts -
- Deliveries -
- Form Saves -

- Form Abandons -
- Delivery PDF Receipts -
- Jobs Created -
- Scheduled Jobs -
- Event Logs -
- Groovy Services -
- T.Field Sync -
- Total Errors -
- Collaboration Job Errors -
- Email Errors -
- Form Render Errors -
- REST SOAP API Errors -
- Security Manager Errors -
- Unclassified Errors -
- Critical Errors -
- Delivery Errors -
- Form Dynamic Data Errors -
- Receipt Render Errors -
- Groovy Service Errors -
- Payment Gateway Errors -
- Scheduled Job Errors -
- T.Field Sync Errors -

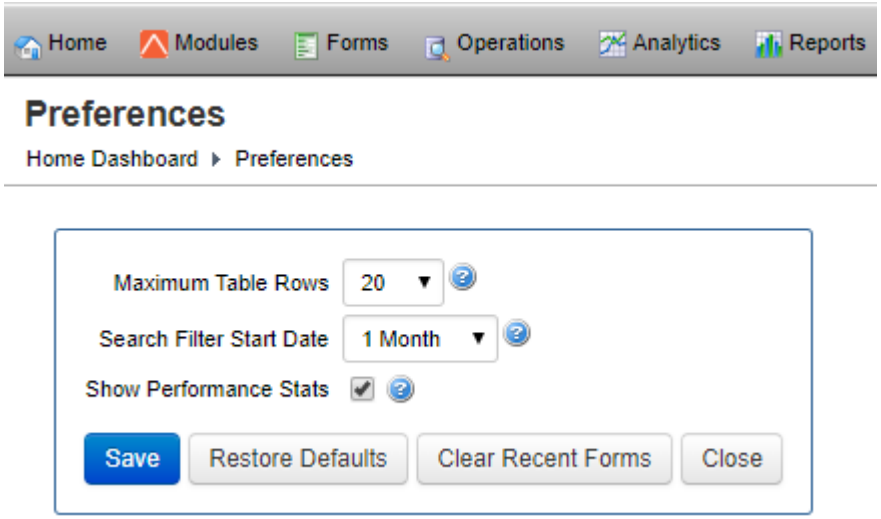
Next, learn how to check database version log.

System User Preferences

Manager allows you to customize the user's preferences.

To configure the user's preferences:

1. Select System > Preferences.



2. Select the maximum number of table rows to display in one page from the Maximum Table Rows dropdown list. Default is 20.
3. Select the default search start date filter criteria from the Search Filter Start Date dropdown list. Default is 1 Month.
4. Select the Show Performance Stats checkbox to show performance statistics in the form requests, Groovy service log, and scheduled job history. It is selected by default.
5. Click Restore Defaults to restore your preferences to the default Manager preferences.
6. Click Clear Recent Forms to clear your form history.
7. Click Save to update the changes.

Next, learn how to check Manager system info.

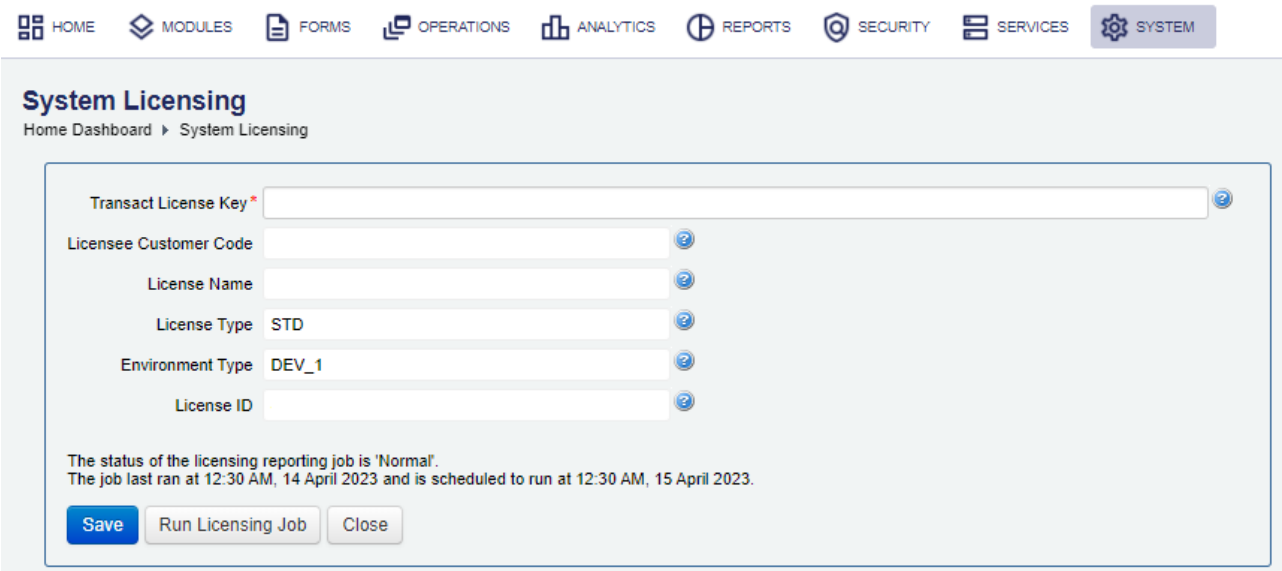
System Licensing

Journey Manager is a licensed product, so you must have a valid license to run it. Our licensing model includes an automated transaction licensing report, which Manager generates periodically and sends it to our servers. The report contains daily summaries on the number of submissions, broken down by organizations and other metrics, such as number of completed transactions. For this to work, you must configure Manager with licensing details, which we provide to you.

> NOTE
>
> You must have an Administrator role to view and configure the license details.

To view and configure the system licensing details of your server:

1. Select System > System Licensing.



2. Enter the Transact License Key provided to you by Temenos. Manager will populated all other read-only fields below from this license key, so you don't need to enter any additional information.

   > NOTE
   >
   > A separate license key must be entered for every Journey Manager instance as well as every Journey Manager version upgrade of the instance.

3. The Licensee Customer Code is provided by Temenos, for example, My Company.

4. The License Name is the name of your Journey Manager system, for example, My Company UAT.
5. The License Type is a type of the license of your Manager system.
6. The Environment Type you are licensed for, for example, `DEV`, `TEST`, `UAT`, `STAG`, and `PROD`.
7. The License ID is the unique code identifying your Manager system for licensing purposes.
8. Click Save to update the changes.
9. Click Run Licensing Job to run the transaction licensing report manually, which will also upload the data to Temenos. When the job is run, you check the details of the last transaction licensing job, its status, for example Normal, time it was run, and when the next scheduled job will be run.

   NOTE

   By default, the transaction licensing report is run nightly.

Next, learn how to check system health.

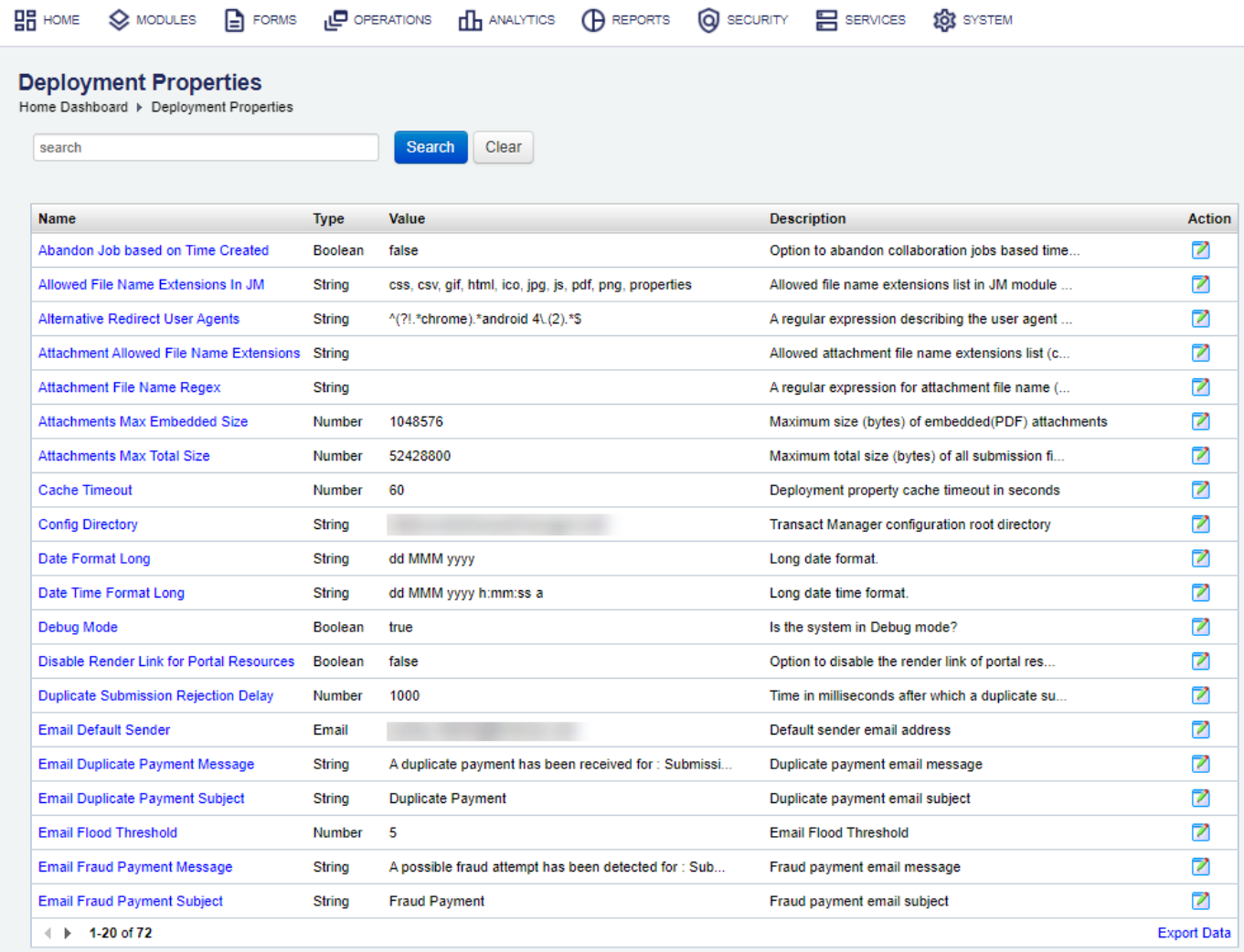
Deployment Properties

Manager comes with a set of deployment properties that allow you to control and customize applications' functionality without the need to rebuild them. You can view and configure deployment properties. Each property is initialized to its default value during Manager installation, but you can change it. However, you must understand the impact of the change as it may affect various parts of Manager.

Some deployment properties are pertinent to other products, such as Journey Analytics and Journey Maestro, so you may need to update these properties to control product's features.

To view the deployment properties, select System > Deployment Properties.

Deployment properties are shown page by page, with the maximum number of records per page configured in the users preferences.



Name	Type	Value	Description	Action
Abandon Job based on Time Created	Boolean	false	Option to abandon collaboration jobs based time...	
Allowed File Name Extensions In JM	String	css, csv, gif, html, ico, jpg, js, pdf, png, properties	Allowed file name extensions list in JM module ...	
Alternative Redirect User Agents	String	^(?!.*chrome).*android 4\.(2).*$	A regular expression describing the user agent ...	
Attachment Allowed File Name Extensions	String		Allowed attachment file name extensions list (c...	
Attachment File Name Regex	String		A regular expression for attachment file name (...	
Attachments Max Embedded Size	Number	1048576	Maximum size (bytes) of embedded(PDF) attachments	
Attachments Max Total Size	Number	52428800	Maximum total size (bytes) of all submission fi...	
Cache Timeout	Number	60	Deployment property cache timeout in seconds	
Config Directory	String		Transact Manager configuration root directory	
Date Format Long	String	dd MMM yyyy	Long date format.	
Date Time Format Long	String	dd MMM yyyy h:mm:ss a	Long date time format.	
Debug Mode	Boolean	true	Is the system in Debug mode?	
Disable Render Link for Portal Resources	Boolean	false	Option to disable the render link of portal res...	
Duplicate Submission Rejection Delay	Number	1000	Time in milliseconds after which a duplicate su...	
Email Default Sender	Email		Default sender email address	
Email Duplicate Payment Message	String	A duplicate payment has been received for : Submissi...	Duplicate payment email message	
Email Duplicate Payment Subject	String	Duplicate Payment	Duplicate payment email subject	
Email Flood Threshold	Number	5	Email Flood Threshold	
Email Fraud Payment Message	String	A possible fraud attempt has been detected for : Sub...	Fraud payment email message	
Email Fraud Payment Subject	String	Fraud Payment	Fraud payment email subject	

The list displays the following details:

- Name: a name of a deployment property.
- Type: a type of a deployment property, which can be one of the following:
  - Boolean
  - CSV
  - HTML
  - Image
  - JSON
  - List
  - Long Text
  - Number
  - String
- Value: a value of a deployment property.
- Description: a meaningful description of a deployment property.

To filter or search deployment properties, enter the property name or part of the name and click Search. For example, to find all deployment properties that contain Maestro keyword, type maestro and click Search.

Click Clear to reset the search criteria.

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

To configure a deployment property:

1. Locate a deployment property and click Edit.

2. Update the deployment property's value in the Value field.
3. Click Save to update the changes.

Default Deployment Properties

Manager has the following default deployment properties:

Abandon Job based on Time Created

Allows you to specify whether to abandon collaboration jobs based on their Creation Time or Last Processed Time parameter. You can do it by setting this property to:

- `true`: to use Creation Time
- `false`: to use Last Processed Time

This property affects how a time interval defined with the Collaboration Jobs Abandonment parameter in the Data Retention configuration is used by the Transaction Processor service. | 22.04 This feature was introduced in 22.04.

Allowed File Name Extensions In JM

Lists the file name extensions, as a comma separated list, that are allowed for uploading as form spaces' resources for the JM module. This is to prevent users from uploading malicious files in Journey Manager using the portal resource page and subsequently rendering them, causing the malicious code to execute.
The default list is `css, csv, gif, html, ico, jpg, js, pdf, png, properties`. A warning message The file <YOUR_FILE> has an extension that is not allowed in JM is shown when you try to upload a file with a black-listed file name extension. | 24.04 This feature was introduced in 24.04

> NOTE
>
> Use Disable Render Link for Portal Resources to completely stop rendering of all form space resources.

Alternative Redirect User Agents

Describes a set of user agents requiring alternative HTTP redirect mechanisms.

Attachment Allowed File Name Extensions

Lists the allowed file name extensions of file attachments, as a comma separated list, that Maestro forms will accept as valid attachment files, for example, `jpeg, jpg, png`. This rule is enforced when a the Maestro form sends the attachment to Manager for processing. | 19.11 This feature was introduced in 19.11.

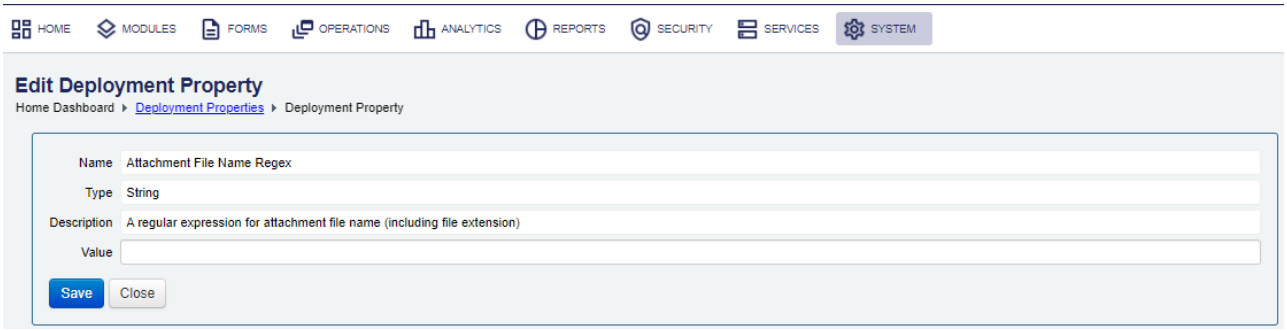
Attachment File Name Regex

A regular expression[1] (regex) for validating a file name, including a file name extension, when a customer uploads an attachment. This is the system level deployment property. If it

[1]A regular expression (regex) is a sequence of characters that define a search pattern. Usually such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation.

doesn't exist, then the attachment functionality operates as normal. If the regex fails, the attachment upload operation fails.
You can use a standard regular expression, for example, `^[\w,\s-]+\.[A-Za-z]{3}$`.

| 20.11 This feature was introduced in 20.11.



Attachments Max Embedded Size
: The maximum size of a single embedded (PDF) attachment.

Attachments Max Total Size
: The global maximum combined size of all attachments for a single submission.

> NOTE
>
> Contact your System Administrator or cloud service team to request this value changed.

Cache Timeout
: Deployment property cache timeout in seconds

Config Directory
: Manager configuration root directory.

Date Format Long
: Long date format.

Date Time Format Long
: Long date time format.

Debug Mode
: Set it to `true` to enable system-wide debugging of all Groovy services and Transact functions, so you don't need to enable the Groovy Debug Logging property for each service.

> WARNING

Use the Groovy Debug Logging option only for troubleshooting of a Groovy script. We don't recommend enabling it in a production environment nor running a service with this option enabled for a prolonged period of time because it generates significant amount of verbose information in the `groovy_service_log` and `groovy_service_log_data` tables. This can deplete a database of free space and degrade the overall server performance.
Also, the size of these tables impact the performance of the Manager UI, such as Transaction Groovy Log and Groovy Service Log, where the Groovy service log records are displayed.

There are several settings that can prevent enabling system debugging. For more information, see Production Mode and Groovy Debug Logging configuration properties.
| 23.04This feature was updated in 23.04

Disable Render Link for Portal Resources

Set to `true` to disable the View Content render link of form space's resources. This is to prevent users from rendering malicious files, causing the malicious code to execute within Manager. The default is `false`. | 24.04 This feature was introduced in 24.04

NOTE

Use Allowed File Name Extensions In JM to whitelist file extensions to be allowed as form space resources and therefor to be rendered, if you don't wish to disable them all.

Duplicate Submission Rejection Delay

Time in milliseconds after which a duplicate submission is rejected.

Email Default Sender

Default sender email address for emails sent by Manager.

Email Duplicate Payment Message

Body text template for emails notifying administrators of a duplicate payment.

Email Duplicate Payment Subject

Subject template for emails notifying administrators of a duplicate payment.

Email Flood Threshold

Email Flood Threshold

Email Fraud Payment Message

Body text template for emails notifying administrators of a possibly fraudulent payment.

Email Fraud Payment Subject

Subject template for emails notifying administrators of a possibly fraudulent payment.

Email Lost Password Message

Lost password email message.

Email Lost Password Subject

Lost password email subject.

Email Promotion Message

Form promotion notification body.

Email Promotion Sender

The email address for the promotion notification messages.

Email Promotion Subject

Form promotion notification subject.

Email Report Message

Report delivery email message.

Email Report Sender

Report delivery sender email address.

Email Report Subject

Report delivery email subject.

Email Submission Status Message

Submission processing status update email message. You can overwrite it on the organization and form level by using organization and form version properties.

Email Submission Status Subject

Submission processing status update email subject. You can overwrite it on the organization and form level by using organization and form version properties.

Email Task Notification Message

Task notification body.

Email Task Notification Subject

Task notification subject.

Email Task Sender

The email address shown as the sender for task notification messages.

Email Welcome User Message

User welcome email body.

Email Welcome User Subject

User welcome email subject.

Enable REST/Web Service Delivery

Enable REST and web service delivery for submissions.

Enable Script Submissions

Enable test submission scripts by switching off submission request log checks.

Enable transforming javax to jakarta

Enable the transformation of packages from `Javax` to `Jakarta`, which are used in Groovy scripts of services, inside archive files during the import process. This is mainly used for Exchange components. This option is disabled by default. | 23.10 This feature was introduced in 23.10

Enable Virus Scan for JM Uploads

Enable virus scan check for all uploads via the Manager UI. The check is triggered with HTML FileField control. It's enabled by default. | 23.04 This feature was introduced in 23.04

Enable XSS Protection On Controls

Enable XSS[1] protection for HTML controls, such as a TextField, TextArea, and so on, by validating user's input text for specific "at risk" HTML tags, deemed to be a part of the XSS attack, and removing them. This option is disabled by default. | 22.04 This feature was introduced in 22.04.

For more information, see https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html

Eventing Error Log Flood Threshold Count

Set a number of errors that can be logged by "Event Listener Configuration Service" on page 1279 and the "Event Inbox Processor Service" on page 1276 within a time interval specified with the Eventing Error Log Flood Threshold Time deployment property.

This is to prevent the error log from being filled up. Once the number of errors has reached the threshold, one more generic error is logged saying that the threshold has been exceeded and further errors will only be logged to the new event error log file.

The default is 10. | 24.04 This feature was introduced in 24.04

Eventing Error Log Flood Threshold Time

Set a time interval (in second) within the Eventing Error Log Flood Threshold Count deployment property is applied. This allows you to configure various scenarios, such as only log 10 errors in 5 mins or log 100 errors per an hour.

The default is 300 seconds. | 24.04 This feature was introduced in 24.04

Eventing Feature Enabled

Enable the Eventing Feature to publish events to Kafka topics. This option is disabled by default. | 22.10 This feature was introduced in 22.10.

Exception Debug Logging Enabled

---

[1]Cross-site scripting (XSS) is a type of security vulnerability that can be found in some web applications. XSS attacks enable attackers to inject client-side scripts into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy.

Exception Debug Logging is enabled by default

Export Service Connection Passwords On

Option to enable or disable exporting of passwords in service connection.

Fluent API User Permission Check

Option to enable or disable Fluent API user permission check.

Groovy Logging Enabled

Enable logging of Groovy service calls.

Groovy Script Timeout

Groovy script execution timeout in milliseconds.

Insights Application ID

Insights Application ID. This should be `PROD` for production instances. Other options are: `DEV`, `DEV1`, `DEV2`, `DEV3`, `LOCAL`, `TEST`, `TEST50` or `STAGING`.

Insights Enabled

Option to enable or disable Insights Analytics.

Insights Version

Insights version identifier. Leave it blank to use the default version.

Maestro Preview PDF Renderer

Receipt renderer for Maestro preview, use 'PhantomJS 2.0' or 'PhantomJS 2.1'

Online Save Enabled

Option to globally disable Online Saving of forms. If set to true, the individual form's setting will be used.

Portal Resources File Name Regex

A regular expression for validating portal resources file name on upload, including a file extension.

Regex Pattern For Url Fields

A regular expression used to validate the URL fields of form space properties, module properties, security managers and form flow configuration. This validation is not applicable to context paths of form spaces and modules.

The pattern is `^[a-zA-Z0-9._@$:=\\-\\{}*?/]+$`. | 24.04 This feature was introduced in 24.04

Report Internal URL

Location where reports are run internally by the application, that is `http://localhost:9080/webreport`.

Report Reschedule Delay.

Amount of time in minutes to wait before rescheduling a running report.

Run Data Retention Independently

Option to allow or disallow running Data Retention independently of Transaction Processor. This requires a form space restart for the change to take effect.

Secure Session Enabled

Option to tighten session security on ajax calls to FormDynamicDataServlet. (Requires a portal restart to take effect).

Server Real Time Health Monitoring

Enable real-time server health monitoring. Set it to false to disable the Manager server metrics on your local server instance temporarily.

SMTP Connection Timeout

SMTP email connection timeout (`mail.smtp.connectiontimeout`).

SMTP Host

SMTP email server hostname (`mail.smtp.host`).

SMTP Password

SMTP email server authentication Password (`mail.smtp.password`).

SMTP Port

SMTP email server Port (`mail.smtp.port`).

SMTP SSL Check Server Identity

SMTP SSL Check Server Identity (`mail.smtp.ssl.checkserveridentity`).

SMTP Start TLS

SMTP Start TLS email connection (`mail.smtp.starttls.enable`).

SMTP Timeout

SMTP email read timeout (`mail.smtp.timeout`).

SMTP User

SMTP email server authentication User (`mail.smtp.user`).

SMTP Write Timeout

SMTP email write timeout (`mail.smtp.writetimeout`).

Status Filter Enabled

Option to enable/disable Status Filter.

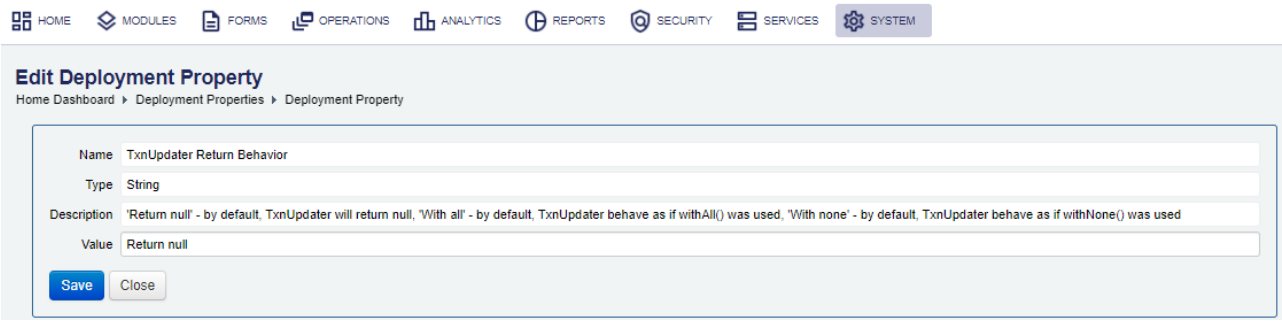
Supported Delivery Methods

Supported delivery methods.

TxnUpdater Return Behavior

This allows you can specify the default behavior of the `TxnUpdater` to control the unforeseen performance side affects of the previous change to the `TxnUpdater` when asked to return a `Txn` object. Now, you can specify the following options for TxnUpdater Return Behavior:

- Return null: `TxnUpdater` will return null. This is the default option.
- With all: `TxnUpdater` behaves as if `withAll()` was used.
- With none: `TxnUpdater` behaves as if `withNone()` was used.



| 19.05 This feature was introduced in 19.05.

Important Deployment Properties

You should modify the following deployment properties after installing Manager:

Directories

Manager uses directories to store various objects such as attachments and report files. By default, these files are stored in a sub-folder of the folder where the Manager server is installed. You can adjust this by setting the Config Directory property to any directory you want. However, make sure that the service has read and write access to the directory.
Edit the property LiveCycle Repository Domain and enter the path, relative to the LiveCycle repository, where Manager should publish its forms. If multiple Manager instances are sharing the same LiveCycle repository, make sure to enter a unique path, for example, `/tm-test-thirdparty/JourneyManagerRepository/`.

SMTP Server

If you want Manager to send emails, for example, for delivery purposes, you have to configure the SMTP settings.

Duplicate Form Submissions

Manager offers some options to control duplicate submissions, which are multiple submission attempts of a requested form. This can happen either because the user attempts to submit the same form multiple times (by double-clicking the submit button or saving the form offline and then opening multiple copies of it) or commonly in system testing, where scripts are used to automate a number of submissions.
By default, Manager blocks duplicate submissions when they are more than a certain time interval apart, which is configured using the Duplicate Submission Rejection Delay property. If duplicate submissions occur within a certain short period, Manager assumes that they originate due to accidental double-clicking of the submit button and ignores the duplicates.
To enable duplicate submissions , for example, for load testing, change the Enable Script Submissions deployment property to true.

To customize the message displayed to form space users when a duplicate submission is rejected, use the in-built form property Submission Duplicate Message.

Journey Analytics Deployment Properties

Manager has the following deployment properties pertinent to Journey Analytics:

Insights Application ID
: Journey Analytics Application ID. This should be PROD for production instances. Must be one of PROD, DEV, DEV1, DEV2, DEV3, LOCAL, TEST, TEST50 or STAGING.

Insights Enabled
: Enable Journey Analytics.

Insights Version
: Journey Analytics version identifier. Leave blank for the default version.

Journey Maestro Deployment Properties

> NOTE
>
> You must have the certain permissions set by an Administrator to see this section. For more information, see Administrator's documentation.

Next, learn how to configure modules.

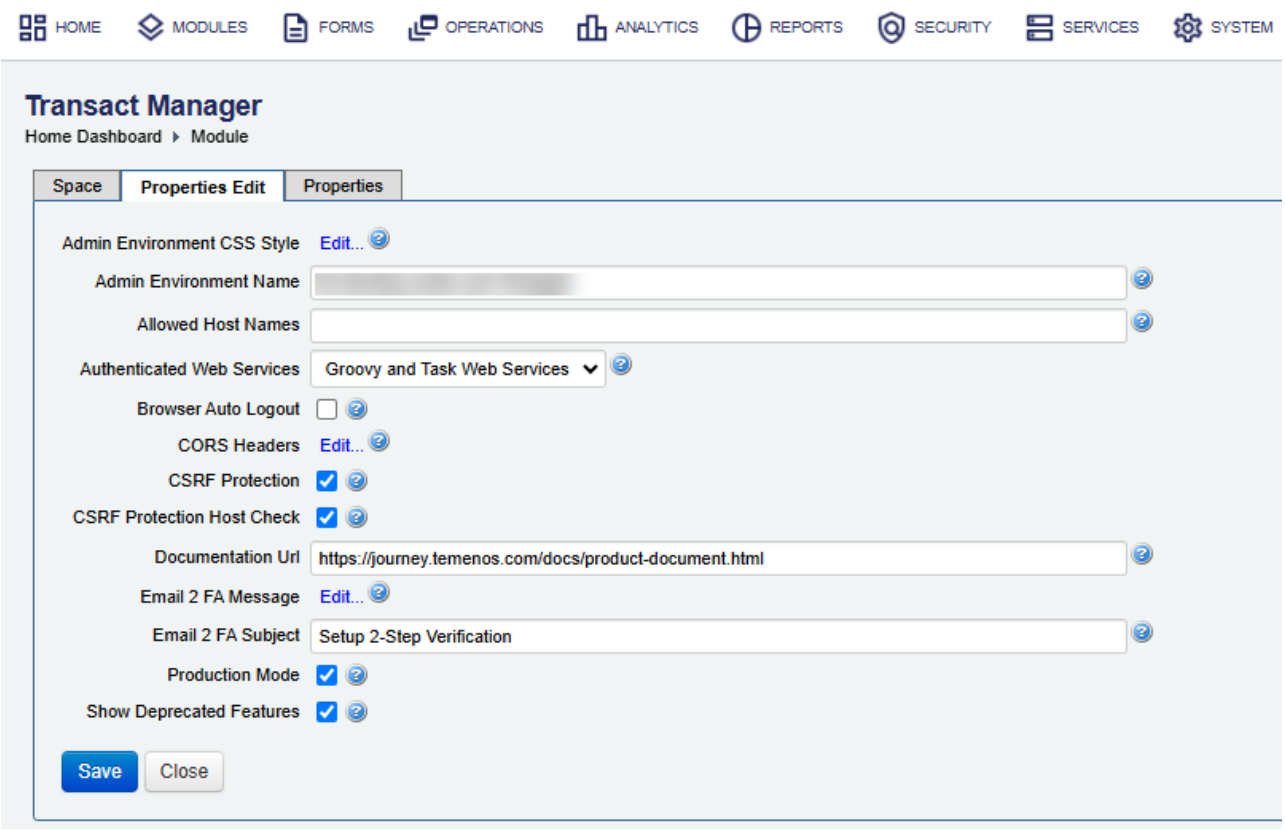
Enable Deprecated Features

Manager allows you to enable deprecated features, if you depend on them in your current installation and configuration.

> NOTE
>
> Manager 18.05+ removes deprecated features of Manager module by default. You can, however, enable these features if they're required.

To enable deprecated features:

1. Select System > Modules Config.
2. Locate the Transact Manager module and click Edit.
3. Select the Properties Edit tab.



4. Select the Show Deprecated Features checkbox.
5. Click Save to update the changes.

The list below identifies the current deprecated features that support the Show Deprecated Features property.

Version	Deprecated Features	Details
18.05	Forms menu options	• New Form Wizard removed. • The New button from the Forms UI removed. • Document Types removed. • Metadata Tag Definitions removed.
18.05	Specific form selection	• Form Version Promotion tab from the form dashboard removed. • The Form Tags tab or for a selected version of the form removed.
18.05	The Organizations option from the Forms menu bar when a specific organization is selected	• The Form Tags tab removed. • Switch to the Delivery Channel tab removed. • By adding or editing a delivery channel where the Delivery Method is REST Service, the PDF Receipt Embed Files checkbox removed.

The following deprecated features can't be enabled using the Show Deprecated Features property.

Version	Deprecated Features	Details
21.11	TransactField App functionality removed	Removed support for the TransactField App product as it has reached End-Of-Life, which also includes T.Field Device Log and T.Field Sync Log.
21.11	FOP removed	FOP receipt service has been removed.
21.05	Remove the PDF export option from licensing reports	Remove the PDF export option (the Download PDF button) from the Transaction and Job Licensing reports because it uses iText[1], which requires a commercial license. The Excel spreadsheet export is still supported.
21.05	Wet Signature Template	LiveCycle Wet Signature receipt generation removed.

Next, learn about all Journey Manager features.

[1]iText is a library for creating and manipulating PDF files in Java and .NET.

Configuration Service

Manager provides a functionality enabling you to quickly and easily reconfigure a server instance by uploading a configuration file containing all relevant information. The configuration file is a JSON[1] file with a flat structure defining server's properties as name / value pairs. The Configuration Service consumes this file and runs all configuration tasks that it defines.

The Configuration Service comes with several template files, which you can use to create your own configuration deployments. As you upload the JSON configuration file, the Configuration Service parses it and runs import operations for each section of configuration, such as form spaces, organizations, forms, service connections and so on. This means that you need to have correct import permissions assigned to your account.

Currently, the Configuration Service comes with the following template files:

- Template version (2.1) - | 24.04 This feature was introduced in 24.04
- Template version (2.0) - | 23.10 This feature was introduced in 23.10
- Template version (1.1) - | 23.04 This feature was introduced in 23.04
- Template version (1.0)

Create Entities

Originally, the Configuration Service was designed to update only existing configurations; there was no functionality to create new ones. However, starting from 24.04, the Configuration Service allows you to create new entities by using the template version 2.1. At the moment, it is limited to service parameters only, albeit, future releases might add ability to create more entities via the Configuration Service.This means that you need to install a Manager instance and configure it prior to being able to use the Configuration Service.

To create a new entity for `serviceParameters`, you must use a new flag `createOrUpdate` that can be defined in 2 scopes: global and entity scopes.

- Global scope: the `configurations` object to define it as a global parameter across the whole file:

[1]JSON (JavaScript Object Notation) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.

```
"version": "2.1",
  "configurations": {
    "createOrUpdate" : "false",
    "manager": {
```

- Entity scope: the `serviceParameters` object to overrides the global parameter set above:

```
"services": [
    {
        "serviceName": "My Service 2",
        "versionNumber": "1.0.0",
        "serviceParameters": [
            {
                "name": "Service parameter name (String)",
                "description ": "Service parameter description",
                "bindParameter": "false",
                "readonly ": "true",
                "required  ": "true",
                "clearValueOnExport  ": "true",
                "unitTest": "false",
                "value": "Service parameter value",
                "createOrUpdate": "false"
            }
        ]
    }
]
...
```

The following logic applies when using the `createOrUpdate` parameter:

1. The local `createOrUpdate` parameter defined on the entity level takes precedence over the global parameter.
2. The global parameter is used if there is no local `createOrUpdate` defined on the entity level.

Then for the `serviceParameters` only, if it has the `createOrUpdate` parameter and set to true, then:

- If the service parameter exists then it will be an update operation.
- If the service parameter doesn't exist, then it will be a create operation that will also require the presence of type parameter under the service parameter values.
- If the service parameter of type `List`, an addition parameter called `listValues` is required to set the initial value of the list option. For example, `label1:value1|label2:value2`, the value of

a parameter must be set to a value from the `listValues`, such as `value1` or `value2`. The snippet of the configuration is shown below:

```
{
    "name": "Service parameter type (List)",
    "'escription ": "Service parameter description",
    "bindParameter": "false",
    "readonly ": "false",
    "required ": "true",
    "clearValueOnExport ": "true",
    "unitTest": "false",
    "createOrUpdate": "true"
    "type": "List",
    "ListValues": "1: test1|2: test2",
    "value": "1"
}
```

You can upload and deploy the Configuration Service file using one of following options:

- UI
- SDK
- API

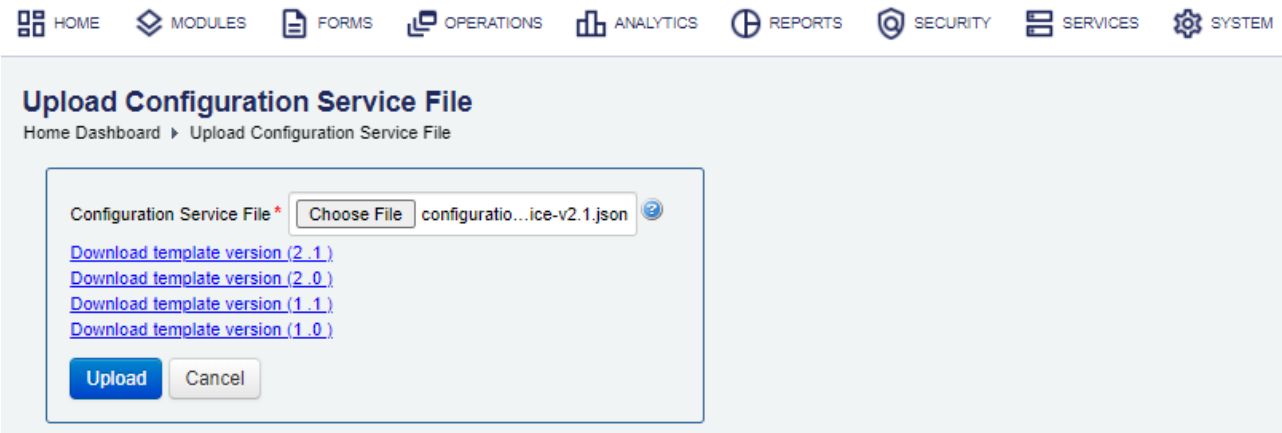
> NOTE
>
> To deploy the Configuration Service file , you must have the REST Configuration Service API permissions correspondingly.

Let's look at each of them in more detail.

Deploy Using UI

To upload and deploy a Configuration Service file via Manager:

1. Select System > Configuration Service.

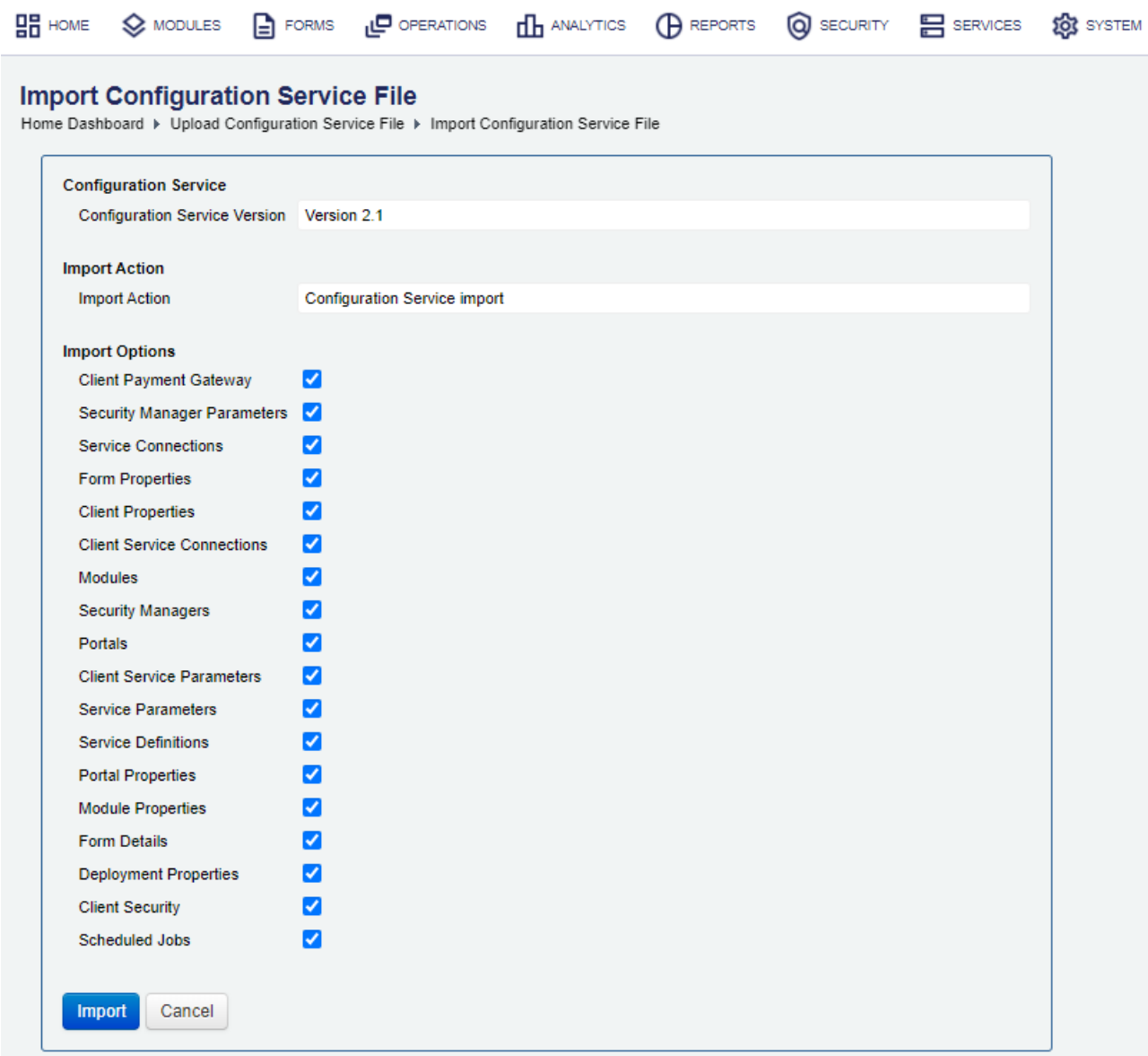


2. Click a template link, for example, Download template version (2 .1), to view and download a template configuration service JSON file.
3. Create your own specific configuration by populating the template file. If you wish to create new configurations for service parameters, see Create Entities.

   NOTE

   Make sure all JSON tags are properly formatted and closed, so there are no validation errors during the uploading process.

4. Click Choose File to select a configuration file you. The supported file formats are ZIP and JSON.
5. Click Upload to start the uploading process. Manager validates the JSON file and shows a list of import options so you can choose what part of the configuration file to upload.



If the configuration file is not valid, the following error message is shown: Error loading import file: Import file is not a valid json file.

6. Clear Portal Properties to exclude form spaces' properties from the import. For more information, see import form spaces.
7. Clear Client Service Connections to exclude organizations' service connections from the import.
8. Clear Portals to exclude organizations' form spaces from the import.
9. Clear Client Properties to exclude organizations' properties from the import. For more information, see import organizations.
10. Clear Form Details to exclude form details from the import.
11. Clear Service Definitions to exclude service definitions from the import.

12. Clear Deployment Properties to exclude deployment properties from the import.
13. Clear Scheduled Jobs to exclude organizations' scheduled jobs from the import. | 24.04
    This feature was introduced in 24.04
14. Clear Service Connections to exclude service connectionss from the import. For more information, see import service connections.
15. Clear Service Parameters to exclude services' parameters from the import. For more information, see import services.
16. Clear Client Service Parameters to exclude organizations' service parameters from the import.
17. Clear Client Payment Gateway to exclude client payment gateways from the import.
18. Clear Module Properties to exclude module properties from the import.
19. Clear Modules to exclude modules from the import. The configuration section of the template JSON file defines the following options:

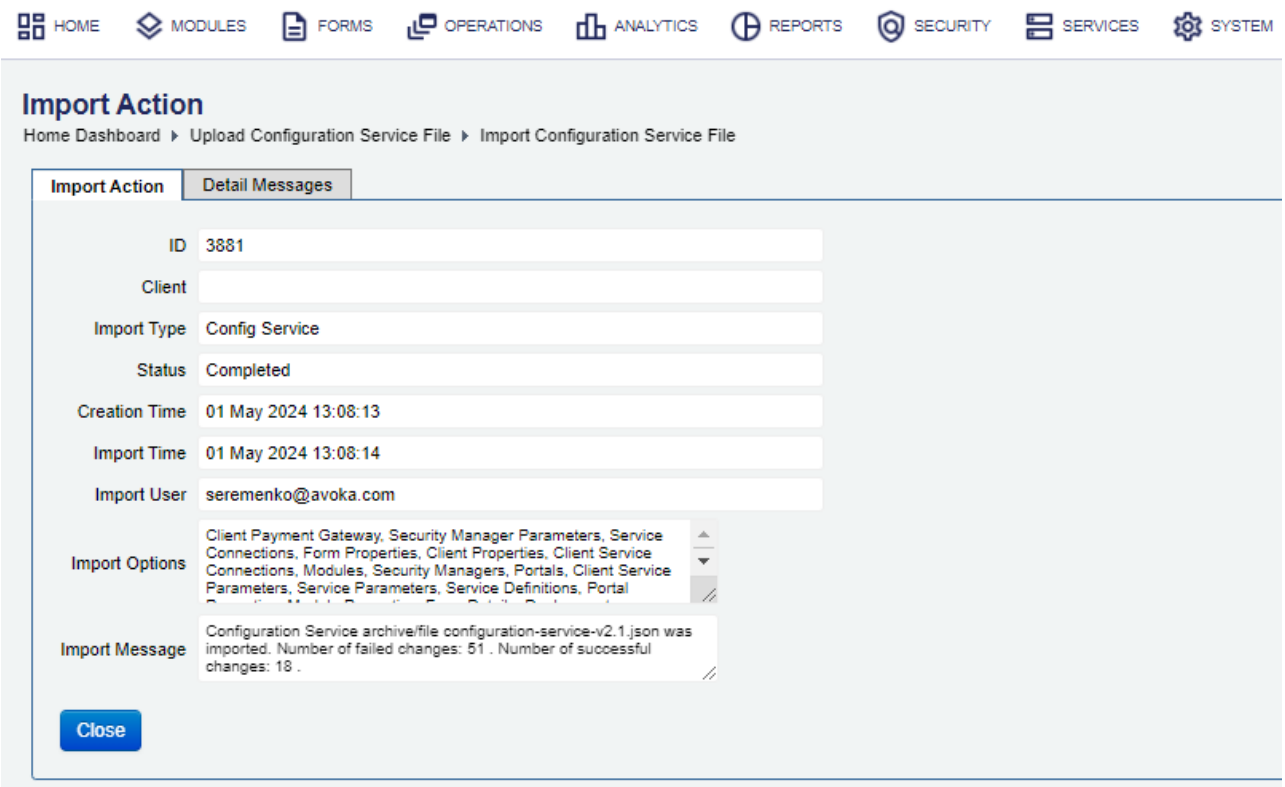
```
"modules": [
        {
          "moduleName": "Module Name",
          "moduleContextPath": "http://tm-server/module/",
          "description ": "",
          "Security Manager": "",
          "moduleProperties": [
            {
              "name": "module property name",
              "value": "test",
              "description ": ""
            }
          ]
        }
      ]
```

20. Clear Security Managers to exclude organizations' security managers from the import.
21. Clear Security Manager Parameters to exclude organizations' security manager's parameters from the import.
22. Clear Form Properties to exclude form version properties from the import.
23. Clear Client Security to exclude organizations' security configuration from the import. The configuration section of the template JSON file defines the following options:

```
"clientSecurity": {
          "overrideSystemDefaultBrowserSupportPolicy": "false",
          "enableInternetExplorer10": "false",
          "enableInternetExplorer9": "false",
          "enableInternetExplorer8": "false",
          "overrideSystemDefaultCSP": "false",
          "organizationCSPHeader": ""
}
```

| 24.04 This feature was introduced in 24.04

24. Clear Scheduled Jobs to exclude scheduled jobs from the import.
25. Click Import to process the file in accordance to the selections above.
26. Review the import action status and the import message for any errors. Click Detail Messages to see more information.



In case of an error, you need to identify a root cause, update the configuration file accordingly and try to run the import again.

27. Click Close. It opens the Import Actions UI, so you can review the latest import log records.

Deploy Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:configuration-service-archive-deploy goal to upload and deploy a Configuration Service file.

Deploy Using API

The Journey Manager REST API provides the endpoint URL to upload and deploy a Configuration Service file.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s)://<JM_SERVER>/manager/secure/rest/configuration-service/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as base64.

`--form ‘archiveFile=@“/Users/my-configuration-service-v2.0.json”’` - a path to a Configuration Service file to be uploaded and deployed.

`--form ‘importOptions=“CLIENT_PROPERTIES,PORTAL_PROPERTIES,FORM_PROPERTIES,SERVICE_PARAMETERS,SCHEDULED_JOBS”’` - a comma-separated list of import options to perform. The option names are the same as described in the Deploy Using UI section but they use an underscore, for example, `DEPLOYMENT_PROPERTIES` for Deployment Properties.

The complete list of import options is shown below:

```
PORTALS
PORTAL_PROPERTIES
CLIENT_PROPERTIES
FORM_PROPERTIES
SERVICE_PARAMETERS
CLIENT_SERVICE_PARAMETERS
SERVICE_CONNECTIONS
SERVICE_DEFINITIONS
CLIENT_SERVICE_CONNECTIONS
DEPLOYMENT_PROPERTIES
CLIENT_PAYMENT_GATEWAY
CLIENT_SECURITY
MODULES
MODULE_PROPERTIES
SECURITY_MANAGERS
SECURITY_MANAGER_PARAMETERS
SCHEDULED_JOBS
FORM_DETAILS
```

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

```
{
  “archiveName”: “config-service.json”,
  “importMessage”: “Configuration Service archive/file config-service.json was
imported. Number of successful changes: 11 .“,
  “importStatus”: “Completed”,
  “importTime”: “2024-02-07T10:28+1100"
}
example with warnings
{
  “archiveName”: “my-configuration-service-v2.0.json”,
  “detailMessages”: [
    “Portal not found : [Work Space]“,
    “Service definition for service name [Amazon Submission Data Storage] not
found”,
    “Service definition for service name [Amazon Submission Data Storage] not
found”,
    “Service definition for service name [Amazon Submission Data Storage] not
```

```
found”,
    “Service connection for service connection name [Amazon KMS] not found”
  ],
  “importMessage”: “Configuration Service archive/file my-configuration-service-
v2.0.json was imported. Number of failed changes: 5 . Number of successful changes:
11 .“,
  “importStatus”: “Completed”,
  “importTime”: “2024-02-06T14:38+1100"
}
```

Example

```
curl --location --request PUT ‘https://<JM_SERVER>/-
manager/secure/rest/configuration-service/v1/’ \
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
--form ‘archiveFile=@“/Users/my-configuration-service-v2.0.json”’ \
--form ‘importOptions=“CLIENT_PROPERTIES,PORTAL_PROPERTIES,FORM_PROPERTIES,SERVICE_
PARAMETERS, \
CLIENT_SERVICE_PARAMETERS,SERVICE_CONNECTIONS,CLIENT_SERVICE_
CONNECTIONS,DEPLOYMENT_PROPERTIES”’
```

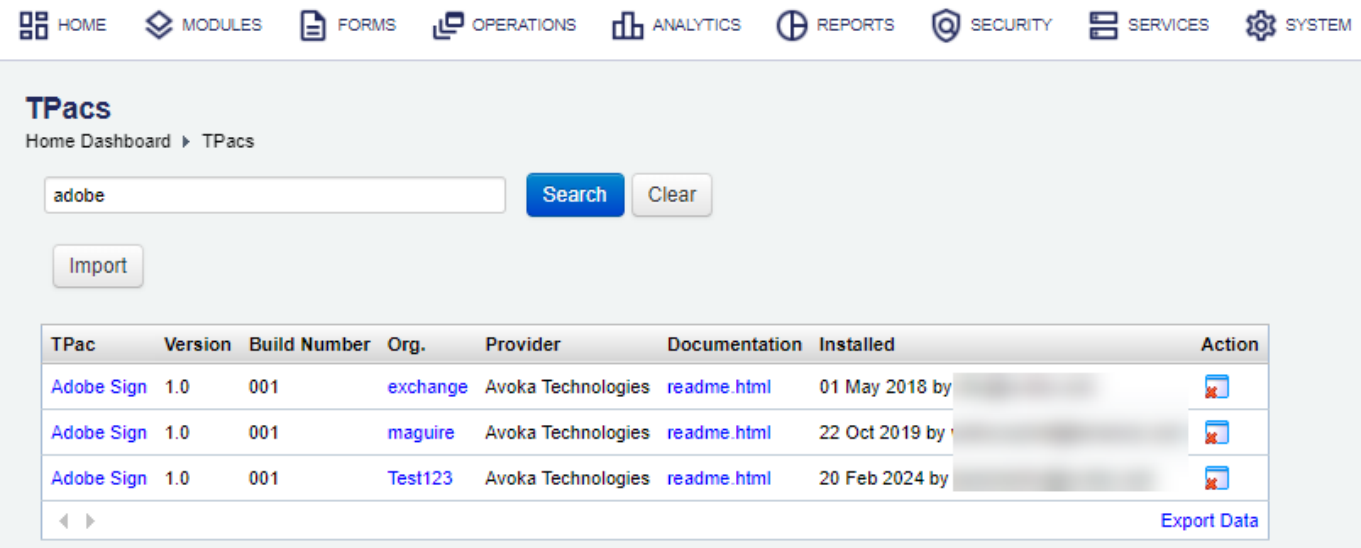
Next, learn how to create a server node.

View TPacs

Manager allows you to view all TPacs installed in your environment. A TPac is an Exchange package that provides some specific functionality that you require for various onboarding form applications. For more information, see Exchange Package Installation.

To view installed packages, select System > TPacs.

The packages are shown page by page, with the maximum number of records per page configured in the users preferences.



The list displays the following details:

- TPac: a name of a package. Click the name to view package details.
- Version: a version of a package.
- Build Number: a build number of a package.
- Org: an organization where a package is installed to.
- Provider: a name of a company that built and packaged a TPac.
- Documentation: a name of a link to a package documentation. Click the link to open it in a browser.
- Installed: a date when and a user who installed the TPac.

To filter or search packages, enter the package name and click Search:

Click Clear to reset the search criteria.

You can also perform the following actions:

- Remove a TPac
- Import TPacs

Click Export Data to export the displayed data as a Microsoft Excel (XLS) file.

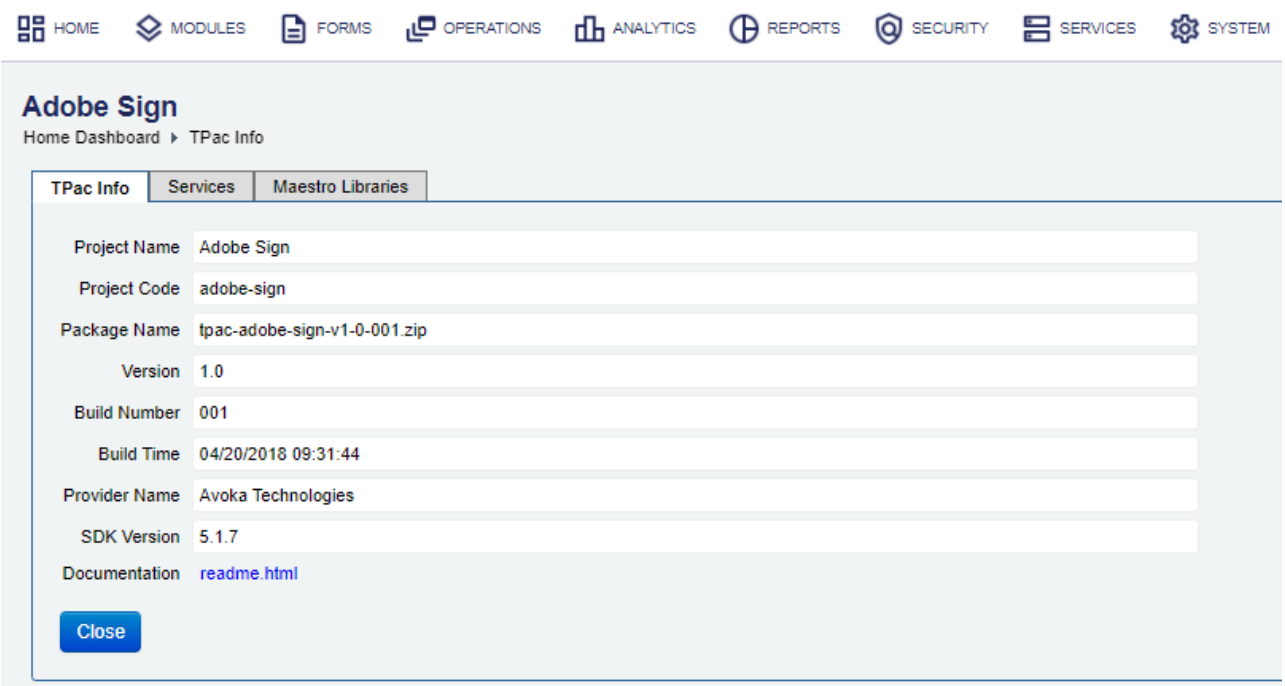
TPac Info

A TPac includes the following information:

- Manager's services and service connections to the 3rd party API.
- Maestro's libraries with components (widgets).
- Manifest configuration file.
- Documentation explaining how to set the package up and use it.

To check package's detailed information:

1. Click a package name and select the TPac Info tab.

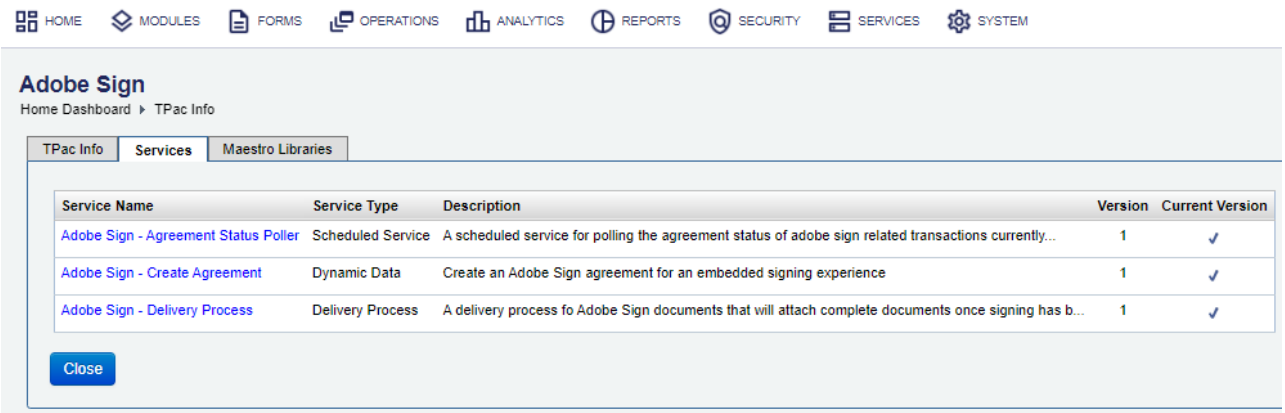


The window displays the following details:

- Project Name: a name of a project this package is relevant to.
- Project Code: a code of the project.
- Package Name: a name of the package.
- Version: a version of a package.
- Build Number: an incremental number identifying a build of the package file.
- Build Time: a date and time when the package file was built.

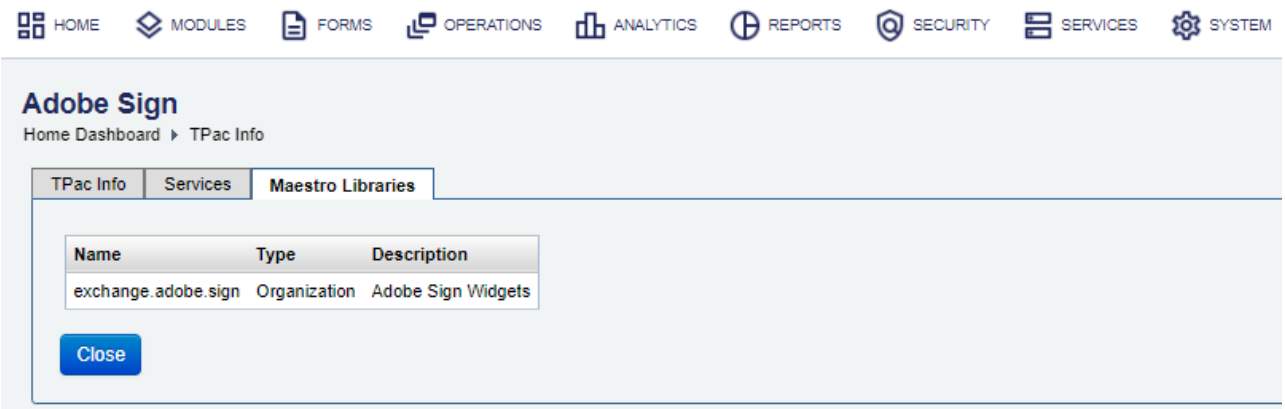
- Provider Name: who built the package file.
- SDK Version: a version of Journey SDK used to build the package file.
- Documentation: a link to a help file. Click the link to open it in a new browser window.

2. Select the Service tab to list services used in this package.



Click each service name to open service's definition.

3. Select the Maestro Libraries to list libraries used in this package.



4. Click Close.

Remove a TPac

You can remove a package when it's not used or required any more. Click Remove TPac to delete a selected package.

> NOTE
>
> Services and service connections, used by a TPac, are not automatically removed from Manager, so you have to do it manually. For more information, see Remove a Service and Remove a Service Connection.

Next, learn how to configure SMTP server.

Import TPacs

Manager allows you to import a TPac as a ZIP file, which is an Exchange package that provides some specific functionality that you require for various onboarding form applications.

You can also use this functionality to:

- Rollback one or several TPacs.
- Re-create one or several TPacs in another Manager.
- Promote one or several TPacs to other environments.
- Migrate forms from one environment to another.

> NOTE
>
> Check the list of Exchange packages to find which one you need for your requirements and then request it as a ZIP file from the Client Services team.

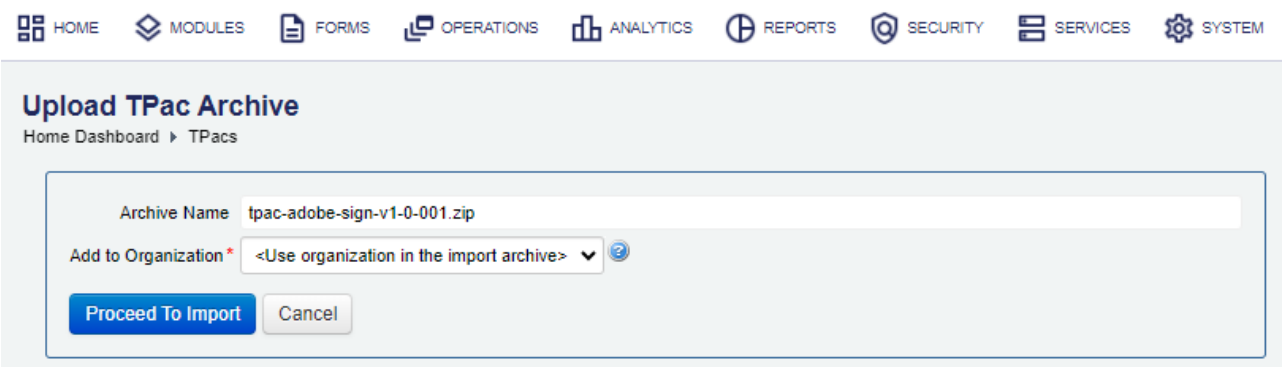
You can import a service using one of following options:

- UI
- SDK
- API

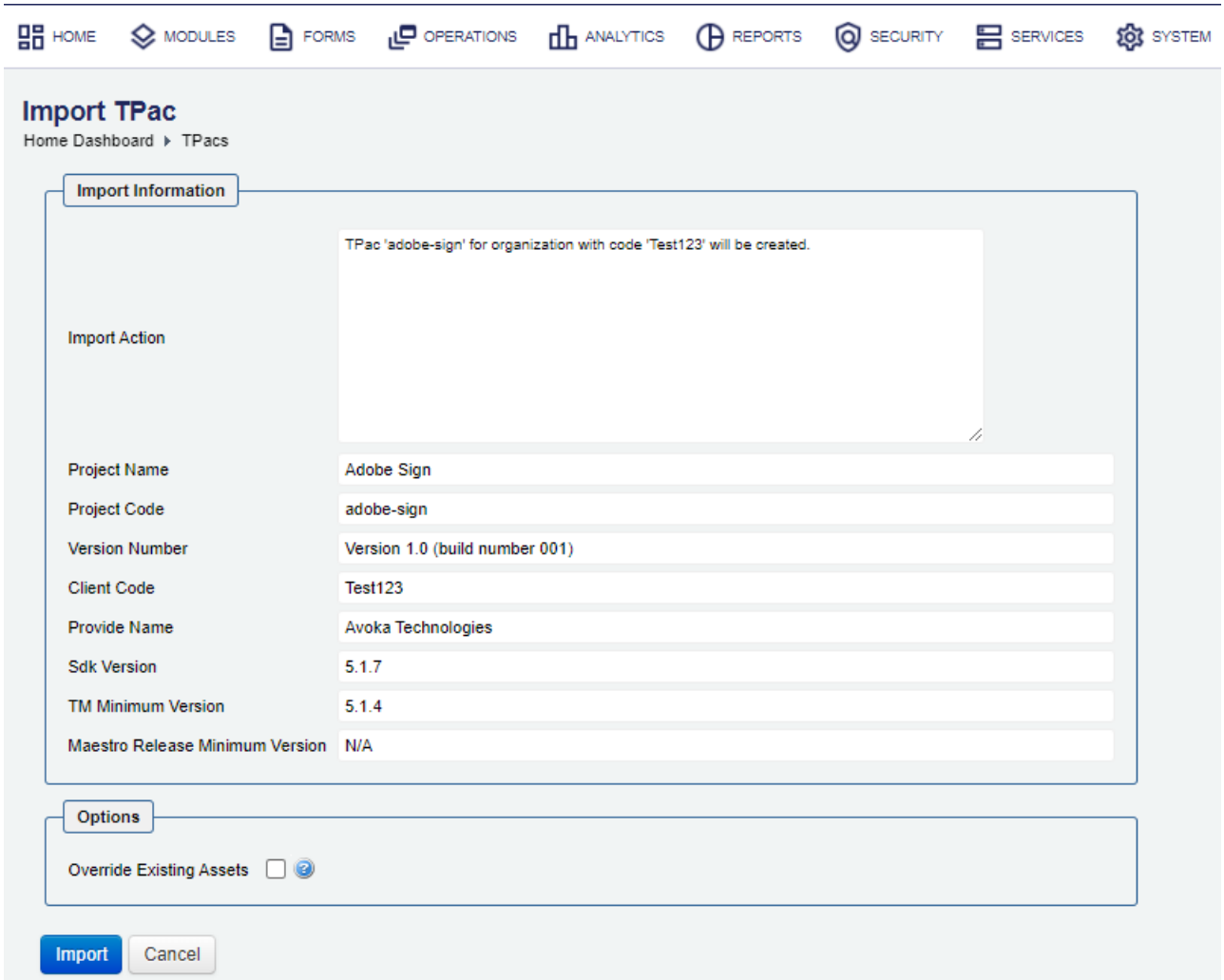
Import Using UI

To import a TPac:

1. Select System > TPacs.
2. Click Import.
3. Click Choose File and choose a TPac archive ZIP file.
4. Click Upload.



5. Select an organization from the Add to Organization dropdown list where you want to add the services to or use the organization already configured in the import archive file.
6. Click Proceed To import.
7. Review the Import Options as shown below:



8. Select the Override Existing Assets to override any existing services and Maestro libraries.
9. Click Import to complete the process.
10. Select the Detail Messages tab to see more information about the import. This tab pops up open if the import operation has any issues you may need to check and resolve.
11. Click Close and check the list of Exchange packages to verify the imported TPac has been added.

NOTE

You can check the Audit Log and Error Log for successful and failed import operations respectively, whilst the Import Log shows all import operations.

Import Using SDK

The Journey Manager SDK Maven plugin provides the tm-sdk:tpac-deploy goal to import an Exchange package.

Import Using API

The Journey Manager REST API provides the endpoint URL to import an Exchange package.

To call this REST API, you need:

HTTP Request

Both HTTP and HTTPS protocols are supported:

```
http(s)://{JM_SERVER}/manager/secure/rest/archive/v1/
```

Parameters

`--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’` - an HTTP header containing `'username:password'` encoded as Base64[1].

`--form ‘archiveFile=@“/Users/my-tpac.zip”’` - a path to an archive file to be imported.

`--form ‘importOptions=“OVERRIDE_EXISTING_ASSETS”’` - a comma-separated list of import options to perform. The option names are the same as described in the Import Using UI section but they use an underscore, for example, `OVERRIDE_EXISTING_ASSETS` for Override Existing Assets.

Response

If successful, this operation returns the following response with a HTTP `200 OK` status code.

Example

```
curl --location --request PUT ‘https://{JM_SERVER}/manager/secure/rest/archive/v1/’
\
--header ‘Authorization: Basic <USE-NAME-AND-PASSWORD-BASE64-ENCODED>’ \
--form ‘archiveFile=@“/Users/my-tpac.zip”’ \
--form ‘importOptions=“OVERRIDE_EXISTING_ASSETS,PRESERVE_SERVICE_CONNECTIONS”’
```

Next, learn how to view all installed TPacs.

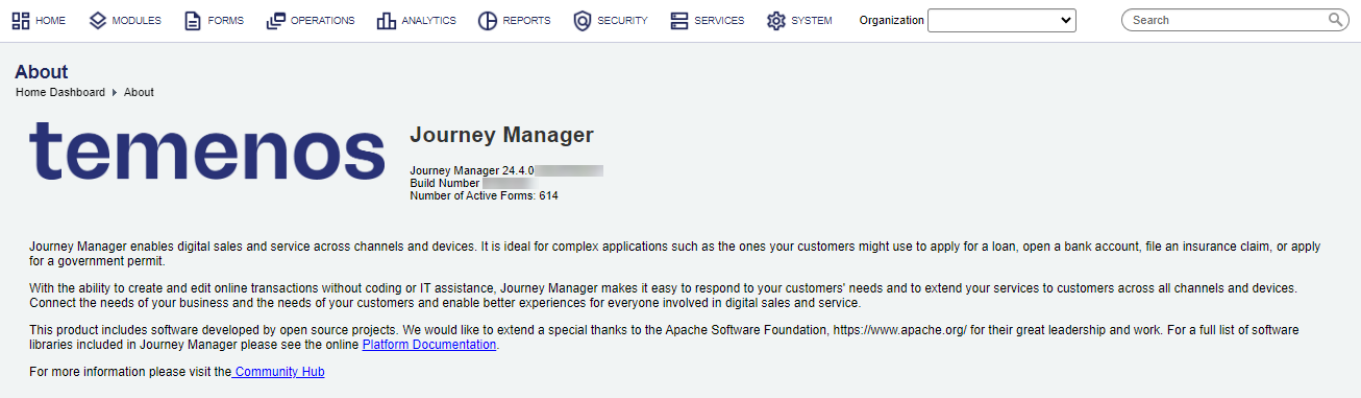
---

[1]Base64 is a binary to a text encoding scheme that represents binary data in an American Standard Code for Information Interchange (ASCII) string format.

About

Journey Manager allows you to view the product's basic information, such as its current version[1], its build number and number of active forms. You may want to use this information when updating Manager software or contacting our customer service to troubleshoot any issues.

To view this information, select System > About.



Click Platform Documentation to open out documentation website in a new browser window.

Click Community Hub to open our community website in a new browser window.

Next, learn how to check system health.

---

[1]Semantic Versioning is a formal convention for specifying compatibility using a three-part version number: major version; minor version; and patch. The patch number is incremented for minor changes and bug fixes which do not change the software's application programming interface (API). The minor version is incremented for releases which add new, but backward-compatible, API features, and the major version is incremented for API changes which are not backward-compatible. For example, software which relies on version 2.1.5 of an API is compatible with version 2.2.3, but not necessarily with 3.2.4.

Multi-Environment Support

Typically, you need several Manager instances to develop, test, and deploy a Maestro form into production. These are often referred to as environments.

An environment can be a dedicated hardware, a virtual machine[1], or a cloud[2] infrastructure. Several Manager instances can be installed and run on the same or different environments. Manager can also be run in multi-tenanted environments.

As you develop and enhance Journey Maestro forms and Manager configurations, you promote or migrate them from one environment to another until they are in the production environment. Normally, you implement a form in the development environment, test it in one or several test environments, and finally deploy it to the production environment.

Any Manager instance allows you to migrate forms and organizations from one environment to another. Additionally, you can simulate form migration by using the Manager Form Test Mode feature.

Next, learn how to promote your Manager to another environment.

---

[1]A virtual machine (VM) is a software program or operating system that not only exhibits the behavior of a separate computer, but is also capable of performing tasks such as running applications and programs like a separate computer. A virtual machine, usually known as a guest is created within another computing environment referred as a "host." Multiple virtual machines can exist within a single host at one time.

[2]Cloud computing is shared pools of configurable computer system resources and higher-level services that can be rapidly provisioned with minimal management effort, often over the Internet. Cloud computing relies on sharing of resources to achieve coherence and economies of scale, similar to a public utility.

Multi-Environment Migration

Journey Manager allows you to migrate or promote its assets from one environment to another. You must migrate them in the following order:

1. Services
2. Form spaces
3. Security managers[1]
4. Users
5. Roles
6. Organizations
7. Forms
8. Groups

For example, to migrate a space and an organization with all its forms from one environment to another, you must import them in this order:

1. A form space
2. An organization
3. Forms

You should review the server and environment specific configurations of the component before the migration, because it might need to be updated in the target environment. This is especially true for the following components:

- Services that contain endpoint URLs.
- Spaces that contain the context path.

---

[1]Security managers are the Journey Manager's components that are responsible for User Authentication, User Authorization, User Self-Registration, and Session Management. Security mangers are designed to handle various sets of users requiring access to different form spaces and can be associated with one or more form spaces. Users can be external or internal, and they can require access to form spaces or a combination of form spaces and modules in different environments. You can find security managers under Security > Security Managers.

Manager Deployment and Rollback Procedure

This guide covers a customer solution deployment and rollback specifics for an initial deployment and environment establishment. It provides a list of appendices, which is designed to provide any customer specific information, that need to be completed by the customer prior to any solution deployment or rollback.

Deployment

It is recommended to create a new version of the configuration and keep a previous one, if possible. Then you can make use of the new configuration by assigning it to your space in the Spaces tab. It will allow you to roll back to the previous configuration quickly and reliably.

Security Managers

Promote the SSL Certificate Store security manager definition as per the Journey Manager security manager environment migration process.

1. Generate the SSL trust Java keystore as outlined in Appendix - SSL Keystore Generation
2. Generate the SSL private Java keystore as outlined in Appendix - SSL Keystore Generation
3. Navigate to the Security Manager Certificates tab.
4. Upload the trust keystore as the Validation Certificate File.
5. Upload the private keystore as the Java Keystore File.
6. Enter the private key alias and password details into the Private Key fields.
7. Navigate to the Security Manager Parameters tab.
8. Enter the trust keystore password details into the truststorePass parameter.

Organisations

Promote the organisation definitions as per the Journey Manager organisation environment migration process. The list of organization definitions can be found in Appendix - Organizations Environment.

Form Spaces

Promote the form spaces definitions as per the Journey Manager form space environment migration (high availability) process. The list of form spaces definitions can be found in Appendix - Form Spaces Environment.

1. Navigate to the Form Space Space tab.
2. Update the Context Path URL field as per Appendix - Form Spaces Environment.

Organisation Applications

Promote the organisation applications as per the Journey Manager organisation application environment migration process. The list of organisation applications can be found in Appendix - Organization Applications Environment.

Service Connections

1. Navigate to a service connections definitions in Journey Manager. The list of form service connections definitions can be found in Appendix - Service Connections Environment.
2. Update the Endpoint as per Appendix - Service Connections Environment.

Verification

Execute post-deployment verification checks as per Appendix - Verification.

Rollback

You can make use of the previous configuration by assigning it to your space in the Spaces tab. It will allow you to roll back to the previous configuration quickly and reliably.

Security Managers

Revert to the previous security manager version or a backup copy as per the Journey Manager security manager environment migration process.

Organisations

Revert to the previous organisation version or a backup copy as per the Journey Manager organisation environment migration process.

Form Spaces

Revert to the previous form space version or a backup copy as per the Journey Manager form space environment migration (high availability) process.

Organisation Applications

Revert to the previous organisation application version or a backup copy as per the Journey Manager organisation application environment migration process. Please refer to the application

names listed in the Deployment chapter.

Service Connections

1. Navigate to a service connections definitions in Journey Manager. The list of form service connections definitions can be found in Appendix - Service Connections Environment.
2. Update the Endpoint as per Appendix - Service Connections Environment.

Verification

Execute post-deployment verification checks as per Appendix - Verification.

Appendix - SSL Keystore Generation

Document the process to generate SSL trust and private keystore files here.

Appendix - Organizations Environment

Document the process to update Organization configuration.

If you do not have a change control in place, document the current (working) and the new configuration here, so this information can be used during the rollback.

Appendix - Form Spaces Environment

Document the process to update Form Spaces configuration, e.g. Context Path URL.

Appendix - Organization Applications Environment

Document the process to update Organization Applications configuration.

Appendix - Service Connections Environment

Document the process to update Service Connections configuration, e.g. Endpoint URL.

Appendix - Verification

Document the process to verify the deployment and rollback success.

Rollback Overview

Manager is a complex and highly configurable software, which allows you to create, import and update majority of its components, such as an organization, form space, form version, service, user, role, group, and security managers. Most of the changes include customized Groovy scripts and configuration files, for example, collaboration jobs, that can be long and complex and are deployed manually. Even though, developers and testers do their best to ensure there are no errors or flaws in these changes, errors do happen and we have to deal with them. Manager is designed to deal with these possible error scenarios, but each component does it differently. The general approach is to rollback the faulty changes to the previous working version.

You can rollback the following components:

- Forms and applications
- Configurations
- Security Managers
- Organizations
- Forms spaces
- Service Connections
- Services

Next, learn about the form dashboard.

Rollback Configuration

If the results of the triage indicate that some configuration is incorrect, for example, someone has put wrong details or updated it with a typo, you can try to rollback to a previous working configuration. There is no easy way to revert to previous configurations in Journey Manager, so each configuration change may have its own rollback procedure. You must relay on your change logs to find the previous working configuration and then do it manually via Manager.

It is recommended to version any new configuration changes, for example, a new security manager v2, and keep the previous ones. Then you can make use of the new configuration by assigning it to your form space. It will allow you to roll back to the previous configuration quickly and reliably.

To give a practical example, let's assume you need to change your current Local Security Manager. Instead of changing its configuration directly, you create a new Local Security Manager version 2 and populate it with the configuration of the current one. Then you update some specific configuration according to your requirements and assign the new security manager with your space. If you discover later than the new configuration stopped working by some unknown reasons, you can easily roll back to the previous Local Security Manager version 1 by re-assigning it to your form space.

Even if you have managed to fix a problem by rolling back a configuration change, you should always ensure that you determine why it changed, and how to prevent it being changed again. In particular, application-level configuration changes may be overwritten when a new version of the application is deployed. This type of change needs to be updated in the application at the source-code level.

> NOTE
>
> In some cases, you may need to contact our Customer Care team to get help in restoring your Manager configuration.

You can also use the configuration rollback as a way to promote or migrate it from one to another Manager server, for example, in different environments.

Next, learn how to promote the Journey Manager to another environment.

Manager FAQ

General

- ### How to verify 2FA PIN from Amazon SNS was sent using SMS?

  You can check the transaction properties to see if the `MFAToken` exists there. This property should give a created date as well (`createdDate`).

- ### How to redirect a form user to an external URL instead of a Journey Manager form?

  You can set up a service on the specific form which throws a `RedirectException` with a redirect URL under the right conditions. A code snippet is shown below.

```
import com.avoka.tm.util.RedirectExceptior

//....
if (!params.request.getParameter("email")){
    throw new RedirectException("http://www.google.com")
}

//...
```

  If you need to submit a form at that point in time, you can use the Confirmation Page URL property.

- ### How to add a Java library to Journey Manager?

  If implementing your new business functionality requires a new Java library, which is not part of the product release, you have the following options:

  - Try to use an existing library that is part of the product release
  - Request us to add the library, so we will evaluate if this library provides an essential missing functionality and then add it to the new release
  - Request us to whitelist the library, so you can use it in the new release. For more information, see Secure Java Package Whitelist in your Journey SDK version.

-

How to turn off the auto-confirmation email?

If you don't want auto-confirmation emails to be sent to users, you need to set the Send Confirmation Email to User option to None for your form in Manager. You also need to clean the Auto-Send Confirmation Emails checkbox in the Form Options of your form in Maestro.

-

Does Manager support blue-green deployment?

No. At the moment, the Cloud Services team doesn't deploy Journey Manager, its modules or applications using a blue - green deployment strategy.