Avoka Transact

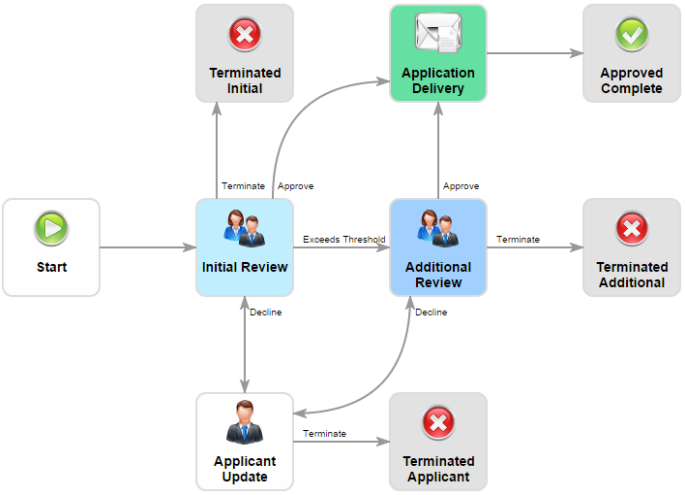
Collaboration Jobs (CollabJobs v4.3)

What Is the function of Collaboration Jobs?

Collaboration Jobs provide a simple ways in Transact to do the following:

Bundle Scenarios: such as customer on-boarding forms. For example a new banking customer fills out an initial form which has some common fields and a list of products which they select those are interested in. On submission the user is then presented with a list of forms, one for each product.

Simple Workflows* : such as the common review and approval type workflows.



Note: * You can actually build quite complex workflows with the Transact collaboration framework. The focus, however, is on simplicity, robustness, and reliability. For this reason, there is no BPMN notation, and very few settings that can be configured by non-programmers. If you want any of these, purchase a BPM system.

Additional Documentation

Collaboration Job developers should also refer to the Where To Find Documentation (CollabJobs v4.3) for information relating to the specific version of Transaction Manager.

Form Bundles (CollabJobs v4.3)

Form Bundles are used to group a number

The features of form bundles have evolved added to Transact over many Transact Versions.

@since 4.1

Authenticated Form Bundles

Share Form Data

@since 4.2

Anonymous Form Bundle

Form Bundle Context

allFormsEditable – enables users to open previously complete forms in a step. This option allows users to reopen any previous forms of their application bundle…

@since 4.3

Dynamic Pre Conditions

Form Chaining

shareExtractData

Job Definition

```
{
 "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 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" }
      ]
    },
    {
      "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"
    }
  ]
}
```

Job Reuse (CollabJobs v4.3)

Important understanding this principle is key to configuring and developing Collaboration Jobs	

Collaboration Jobs & Reuse

http://en.wikipedia.org/wiki/Code_reuse

Action Services

These can be reused between jobs and within the the same job by providing different attributes and action properties.

Copying Services

Using common services.

Creating common

Balancing Isolation of services between organisations.

Form Reuse

A collaboration Job may be associated with many forms. The is achieved via the following.

Start Form Code

This technique uses the task

```
{ "name": "Task Form Code", "value": "$func.startFormCode()" }
```

Using Form Properties

The following example stores the name of the form in the

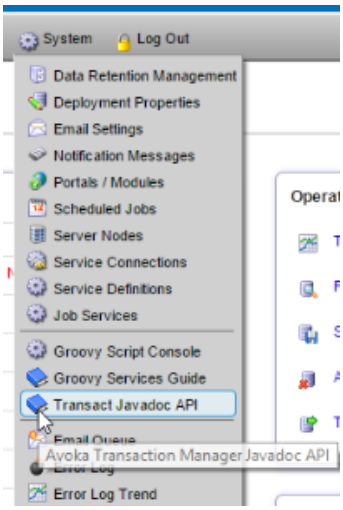
{ "name": "Task Assign Group", "value": "$func.formProperty('Initial Review')" },

Where To Find Documentation (CollabJobs v4.3)

Summary

There are 2 places in Transaction Manager that are useful references for developers of Collaboration Job. These are available from the System Menu and per the screen shot below.

- Transaction Manager JavaDoc API
- Groovy Services Guide



Transact Manager JavaDoc API

The JavaDoc API is a good way source of specific information for collaboration jobs in a version of Transact. The API is organised around java packages and there are 3 packages listed that are referred to in order of most frequently used. What to look for in these packages is detailed in the following sections.

- **com.avoka.fc.core.service.job.impl**
- **com.avoka.fc.core.service.job.config**
- **com.avoka.fc.core.entity**

Note: The Transaction Manager JavaDoc API is also a good general reference classes available to your Groovy Services.

Package: com.avoka.fc.core.service.job.impl

Job Action Property Functions

Class: **JobFunctions**

Provides Job Definition functions which can be called during the evaluation of a job action property value.

Job Action Properties

The job action properties for a predefined Job Action Service that has been implemented in Transaction Manager in a Java Class can be found in the JavaDocs for the Job Action Class.

Class Summary

Job Action Class	Description
JobDeliveryService	This is used in conjunction with the Job Delivery Wait. It is used to kick off the standard Transact Delivery process. The delivery process is completed by the Transaction Processing Scheduled Job not the job controller service.
JobDeliveryWaitService	See above. The Job Delivery Wait service is an asynchronous mechanism (TODO LINK HERE). When the delivery is completed by the Transaction Processing job it calls a callback method that allows the Job Delivery Wait to complete.
JobFormStartService	This action service associates the initial form submission with the initial step. It can add a submission processing status messages to the initial Submission. These appear in the user's submission history. It can also send a status update via email.
JobProcessMessageService	This service will add 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 an endpoint steps. It can also send a status update via email.

JobReceiptWaitService	This action **polls** the submissions until the receipts have completed. The generation of the receipts is completed by a separate Transact 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. Note: This is required as the existing Job Task Wait service does not wait for receipts to be created.

JobTaskAssignService	The Job Task Assign Service is used for creating / assigning tasks in Transaction Manager has a lot of configuration properties. These properties are either 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 categorised as:
	  • **Standard Tasks**: to an authenticated user or group.
	  • **Review Tasks**: these are the similar to the standard tasks but attachments are copied from a previous submission.
	  • **Anonymous Save**: A Submission record is saved but not assigned to a user in the TM 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.

JobTaskWaitService	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 (TODO Link) 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.

Utility Classes

class: **JobPropertyUtils**

Provides a utility class with common methods called by Job Action Services and JobFunction.

Example of a useful method.

`static` **`Submission`**	**getStepSubmission**(Job job, String stepName) Return the first submission for the given step name or null if not found.
`static` **`List<Submission>`**	**getSubmissionsForStep**(Job job, String stepName) Returns a List of submissions, in id order, for the job and given step name or and empty List

class: **ActionStepProperties**
Provides a class to resolve action step properties.

This contains the methods that get the property value by merging the velocity template with model elements. The following are the velocity model info

Velocity Model	Description
$formDataMap	This holds all the form Data Extracts. Note Route Name is automatically added to be available in the data extracts **$formDataMap.routeName**
$job	This holds the job instance
$jobAction	This holds the current JobAction instance. From this we can get the current step $jobAction.stepName
$func	This provides job functions see **JobFunctions** in the API

Package: com.avoka.fc.core.service.job.config

JSON attributes definitions used by the JobDef, StepDef and ActionDef classes

Package: com.avoka.fc.core.entity

Details fields and methods in the **Job**, **JobStep**, **JobAction** and Submission Entities. These are used in the Velocity template substitutions

Groovy Services Guide

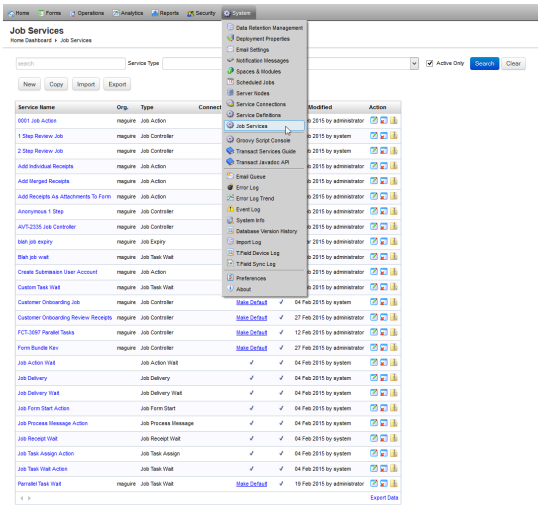
The guide show how to create custom groovy services including custom Groovy Job Actions, Groovy services that are referenced by Collaboration Jobs.

Job Services And Groovy Services (CollabJobs v4.3)

Collaboration Jobs Services

These are developed by configuring Job Services in Transaction Manager.

To get to the services in Transaction Manager select the **System** menu -> **Job Services.**



Collaboration Jobs makes use of 3 categories of services which include Job Controllers, Action Services and Groovy Services.

Job Controller Service

The Job Controller is responsible for the processing of the collaboration job. It has a Job Definition (JSON) that describes when and how it calls the Action Services and Groovy Services.

Link Here to the Job Definition JSON or describe.

Action Services

List all the types and describe each in pairs with the Wait

Job Task Assign and Job Task Wait

Job Delivery and Job Delivery Wait

demo

Custom Groovy Job Action

Job Expiry

Advanced Examples (CollabJobs v4.3)

This page contains specific examples and techniques that can be adopted for your Collaboration Job

- Task Types - Anonymous, Form and Review (CollabJobs v4.3)
- Asynchronous Step Execution (CollabJobs v4.3)
- Anonymous Tasks Example 1 (CollabJobs v4.3)
- Step Expiry (CollabJobs v4.3)
- Customer Onboarding Job - Dynamic Form Bundles (CollabJobs v4.3)
- Task Cancellation Processor (CollabJobs v4.3)

Old Content below which will be removed after migration.

Forms

Form Name	Form Code	Description
Customer Onboarding	onboard-customer	Form Bundle Example. This is the initial form in that a new customer fills in. They select which products they are interested in Credit Cards and Insurance. This works with the Customer Onboarding Job below.w
Customer Onboarding - Credit Cards	onboard-credit-cards	Form Bundle for Credit Card
Customer Onboarding - Insurance	onboard-insurance	Form Bundle Similar to above
Customer Onboarding - Review	**onboard-review**	This form add an additional review modification to the Customer Onboarding form, see **Customer Onboarding Review Receipts** Job (Job Controller Services below). The receipts for the Credit card and Insurance are added as **inform attachments,** see **Add Individual Receipts**. (Job Action service below) The PDF receipts are concatenated together to form a Merged PDF file which is then added as an inform attachment, see **Add Merged Receipts** (Job Action Service below) Not provided in Transact 4.1. Download archive:
Job Anonymous 1	**job-anonymous-1**	This is a minimal form that works with the Anonymous 1 Step Job blow Not provided in Transact 4.1. Download archive:
Job Example 1	job-example-1	This is a minimal form that works with the **1 Step Review** Job below (Job Controller is created from a template) below. It is a very simple workflow where an application form is submitted. The form is next goes to the **Application Review** step where it is reviewed by a member of the **Job Reviewer**. The reviewer may **Accept** or **Reject** the application. If it is rejected the job moves to the **Application Rejected** endpoint For this demo to work the initial submitter can be anonymous or authenticated. The reviewer is authenticated. It uses **Job Example 1** which is a minimal form. A minimal form contains only the fields that are used in the job including. It contains data extract fields, which are values taken from the form that can appear in email message or task subject etc. These fields are set as mandatory. It contains a collaboration route drop down widget. In addition it contains The job can be used for other forms as long as the form has fields with the data extracts set. The form shows a simple use of the inform attachments.
Job Example 2	job-example-2	This is a minimal form that works with the 2 Step Review Job below.
Job Example 3	**job-example-3**	Not provided in Transact 4.1. Download archive: form-version-archive-Job_Example_3-all.zip
Parallel Bundle Triage		This form is used in a Form Bundle scenario and uses the **Parallel Form Bundle** Job service below. The scenario is a user at a household is filling out a survey for every person in the house. The enter details into the triage form including listing the names of everyone in the house. The **Parallel Form Bundle** Job then creates a new task using the **Parallel Bundle Individual** form for each individual listed. https://services.avoka.com/confluence/display/~lbunton/Anonymous+Form+Bundles+-+For+Kevin+Mortimer
Parallel Bundle Individual		The is the individual form associated with the triage form above.

Parallel Review		UWS use case but with Minimal For Associated with Job and related Input XML Merge Groovy Scripts

Job Services (Job Controllers) And Actions

Service Name	**Service Type**	**Description**
1 Step Review Job	Job Controller	
2 Step Review Job	Job Controller	
Add Individual Receipts	**Job Action**	Not provided in Transact 4.1 see application package below
Add Merged Receipts	**Job Action**	Not provided in Transact 4.1 see application package below
Anonymous 1 Step Job	**Job Controller**	Not provided in Transact 4.1 see application package below
Create Submission User Account	Job Action	Note this is an example for a Customer Onboarding Jobdemo. We should provide something that can work
Customer Onboarding Job	Job Controller	
Customer Onboarding Review Receipts Job	**Job Controller**	Not provided in Transact 4.1 see application package below
Parallel Bundle Job	**Job Action**	From UWS example
Parallel Review Job	**Job Action**	
Merge Parallel Review	**Groovy Action**	
Step Expiry Job	Job Controller	Step with a Task is set to Expiry is x hours / days
Step Expiry Action	Job Expiry	Runs on the Expiry of the step. The purpose is to clean the
Delay Job	Job Controller	This and the following show an implementation of action (below) that expires x minutes after its created . This work was done for AVT-2400
Delay Action	Job Action	as above

Task Types - Anonymous, Form and Review (CollabJobs v4.3)

Summary

This page describe:

- Location of the documentation for the Action Properties used when Tasks are assigned.
- List differences in creation of the task type (Anonymous, Form or Review) and what action properties are required for each type.
  - User, Group, email and portal
  - that get the XML data from a previous submission.
  - It also shows how attachments are copied from a previous submission.
- Provides examples for Anonymous an Form Tasks. Most of the examples in the documentation use Review tasks.

Background

Action Properties

The following image of the Java Doc API documentation is taken from the JobTaskAssignService in the Transact Core API.

Configuration: Task Assignment

Below is a list of the Job Action Properties which are associated with e use to configure this step action.

Name	Description	Examples
Task Type	Specify what type of task to create [Anonymous | Form | Review].	{ "name": "Task Type", "value": "Review" }
Task Form XML Data	the form XML data (schema seed)	{ "name": "Task Form XML Data", "value": "$func.startSubmissionXml()" }
Task Input XML Prefill	the input pre-fill data XML which is mapped into the form XML data using prefill input XPath mappings. This property value is ignored for 'Anonymous' Task Types.	{ "name": "Task Input XML Prefill", "value": "$func.invoke('Job Prefill Servi
Task Form Code	the form code of the form to assign in this Task	{ "name": "Task Form Code", "value": "ABC123" } { "name": "Task Form Code", "value": "$func.startFormCode()" }
Task Review Previous Step	This property is only valid with 'Review' Task Type. Specify whether to review a form submission of the previous step [true | false]. If true this will override the value in **Task Review Submission Step**	{ "name": "Task Review Previous Step", "value": "true" }
Task Review Submission Step	This property is only valid with 'Review' Task Type. Specify the name of the form submission step to review. Ignored if **Task Review Previous Step"** is set to true	{ "name": "Task Review Submission Step", "value": "Application Start" }
Task Attachments Previous Step	Specify whether to copy attachments from the submission of the previous step [true | false]. If true this will override the value in **Task Attachments Submission Step**. This property is ignored if a 'Review' Task which will automatically copy attachments form the reviewed submission.	{ "name": "Task Attachments Previous Step", "value": "true" }
Task Attachments Submission Step	Specify the name of the form submission step to copy the attachments from. Ignored if **Task Attachments Previous Step"** is set to true. This property is also ignored if a 'Review' Task which will automatically copy attachments form the reviewed submission.	{ "name": "Task Attachments Submission Step", "value": "Application Start" }
Task Assign Group	This property is only valid with 'Form' or 'Review' Task Types. The assigned task group name. A function can also specify the assigned group.	{ "name": "Task Assign Group", "value": "Initial Review Group" } { "name": "Task Assign Group", "value": "$func.formProperty('Manager Review G
Task Assign Groups	This property is only valid with 'Form' or 'Review' Task Types. The assigned task groups. A function can also specify the assigned groups.	{ "name": "Task Assign Groups", "value": "Blue-Reviewers, Red-Reviewers" } { "name": "Task Assign Groups", "value": "$func.formProperty('Reviewer Groups
Task Assign Group Create	Enable form groups to be automatically created if they don't already exist, and create submission group association.	{ "name": "Task Assign Groups", "value": "$formDataMap.assignGroups" } { "name": "Task Assign Group Create", "value": "true" } { "name": "Task Assig
Task Assign User	This property is only valid with 'Form' or 'Review' Task Types. The assigned task user's login name. The user has to be an active user in the databse.	{ "name": "Task Assign User", "value": "johnsmith" } { "name": "Task Assign User", "value": "$func.startUser()" } { "name": "Task Assign User", "value": "$formDataMap.managerLoginName" } { "name": "Task Assign User", "value": "$func.userForEmail($formDataMap.manag { "name": "Task Assign User", "value": "$func.formProperty('Manager Login')"
Task Assign Email	This property is only valid with 'Anonymous' Task Type. The assigned task email for anonymous user email assignment	{ "name": "Task Assign Email", "value": "john.smith@drdaves.com" } { "name": "Task Assign Email", "value": "$formDataMap.doctorEmailAddress" }
Task Assign Repeating	This property specifies whether the Task Assignment is repeating, and a task assignments should be performed to each assignment member, either Email, Group or User. The example below will assign separate Tasks to the 'App QA' group and the 'App Reviewers' group.	{ "name": "Task Assign Group", "value": "App QA|App Reviewers" } { "name": "Task Assign Repeating", "value": "true" }
Task Assign Repeat Item	This property specifies the Task Assign Repeat Item, and its value is made available during the evaluation of other properties as ${assignRepeatItem} and is stored in the JobAction record. This value can be used to differentiate task assignments during repeating assignment. This data is also populated into the XML form data element: `//SystemProfile/Job/AssignRepeatItem`	{ "name": "Task Assign Group", "value": "App QA|App Reviewers" } { "name": "Task Assign Repeating", "value": "true" } { "name": "Task Assign Repeat Item", "value": "$formDataMap.divisionChoice" }
Task Enable Claiming	Enable group tasks to be claimed/assigned to individual users belonging to the task's groups.	{ "name": "Task Enable Claiming", "value": "true" }
Task Save Challenge	This property is only valid with 'Anonymous' Task Type and is is associated with Anonymous save and the **Task Assign Email** property. The save challenge question is configured in the form version.	{ "name": "Task Save Challenge", "value": "${formDataMap.email}" }
Task Assign Portal	the space name to be used for the task assignment, for anonymous user email assignment (manditory) for user or group assignment (optional). If not specified then it uses the first space that is assigned to the form.	{ "name": "Task Assign Portal", "value": "Maguire" } { "name": "Task Assign Portal", "value": "$func.formProperty('Review Portal')

Job Functions

Ref to the Transact Core API JobFunctions API on how to use **$func** in the Action Property value.

Task Types

The action properties **Task Type** a user must specify either **Anonymous**, **Form** or **Review** task.

Form and **Review** tasks are both require a user to be authenticated user, a user **Task Assign User** and / or group(s) **Task Assign Group or Task Assign Groups** must be specified.

Anonymous tasks do not require a user to authenticate. They require an email address **Task Assign Email** and a portal **Task Assign Portal** to be specified.

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 don't know about previous tasks. 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 specified.
- The attachments can be copied from a previos submission using **Task Attachments Previous Step** or **Task Attachments Submission Step**

The following table summarises the action property required by task type. Where a single letter is repeated in 2 cells 1 of the action property should be used.

Action Prop 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.

Example Jobs

There are a number of examples of a review tasks in the documentation and examples. Eg the Job Example 1 and Job Example 2 forms.

Form Task

```
{
  "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"
    }
  ]
}
```

Anonymous Task

```
{
  "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 ${submission.
contactEmailAddress}." },
            { "name": "Task Assign Portal",  "value": "Maguire" },
            { "name": "Task Form XML Data",  "value": "$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"
    }
  ]
}
```

Asynchronous Step Execution (CollabJobs v4.3)

This documentation demonstrates the principal in calling an external service from a collaboration job that asynchronously calls back to have the job continue proceed.

The process is similar to the Job Task Assign and Job Task Wait actions see Job Basics - Wait Action Processing

The working sample is attached in an Application Package at the bottom of the page.

Background

Job

The job used in this demo has 3 steps and looks like this. We will be looking at the middle step.

Aysnc Invoke Wait - Step

The code block below shows the step definition for our example job which can be found in the application package.

Job Step Definition - Async with Callback

```
{
  "name": "Async Invoke Wait",
  "type": "",
  "actions": [
    {
      "name": "AVT-4097 Job Action Invoke",
      "type": "Job Action"
    },
    {
      "name": "Wait For Callback",
      "type": "Job Action Wait"
    }
  ],
  "routes": [
    { "name": "Default",  "nextStep": "Application Completed" }
  ]
}
```

Job Action State Transitions

The table below summarises Action state from the example service attached. We can see 2 actions for the **Async Invoke** step.

Action Service	State Initial Run	State After Callback
Job Action	Invoked	Completed
Job Action Wait	Pending	Completed

Job Action - Calling an external service

The following code show the scaffolding required in the groovy Job Action service. It can be found in the **AVT-4097 Job Action Invoke.**

Make call to the rest service

if call was successful then set the Action Result to **Invoked**

If the call fails then set the value to **In Progress**. The rest call will be tried again in X minutes.

Job Action to Call Rest Service

```
def restCallOk = true

ActionResult actionResult = new ActionResult(Status.INVOKED)

// Call rest service

// hand rest failure
if(!restCallOk) {
    actionResult = new ActionResult(Status.IN_PROGRESS)
    // retry in 15 min (service Parameter)
}

return actionResult
```

Callback

The external system may call back to a groovy service rest endpoint. It could be generated by a new schedule task or triggered by a submission.

In any case it will need to get the job action key for the job action that called the external server. eg **AVT-4097 Job Action Invoke**

`jobController.completedJobAsyncAction(jobActionKey)`

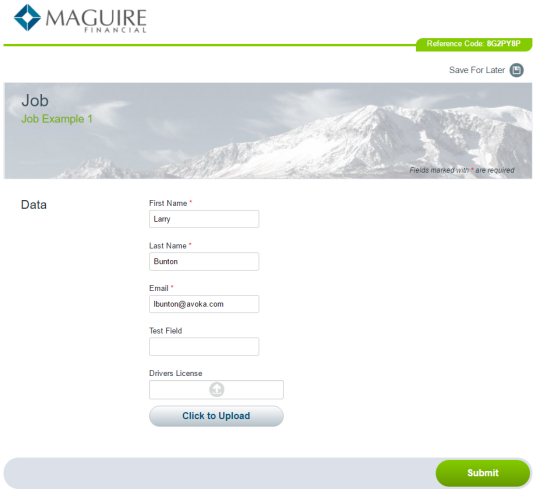
The following code can be run in the Groovy console. I have included it in the example **AVT-4097 Fake Callback.**

- The code requires us to manually enter the job reference code.
- It finds the job
- From there it looks up the job action key.
  I know that the action is the first one in the list (0), the wait is the second (1)
- if finds the job controller service and executes the completedJobAsyncAction(..) method

```
import com.avoka.core.groovy.GroovyLogger as logger
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.service.ServiceLocator
import com.avoka.fc.core.service.job.IJobController

String jobRef = "9QC2234"
Job existingJob = DaoFactory.getJobDao().getJobForReferenceNumber(jobRef)
JobStep currentStep = existingJob.getCurrentStep()
JobAction action = currentStep.getJobActions().get(0)
String jobActionKey = action.getJobActionKey()
IJobController controller = ServiceLocator.getJobController(existingJob)
boolean result = controller.completedJobAsyncAction(jobActionKey)

println 'result = ' + result
```

End To End

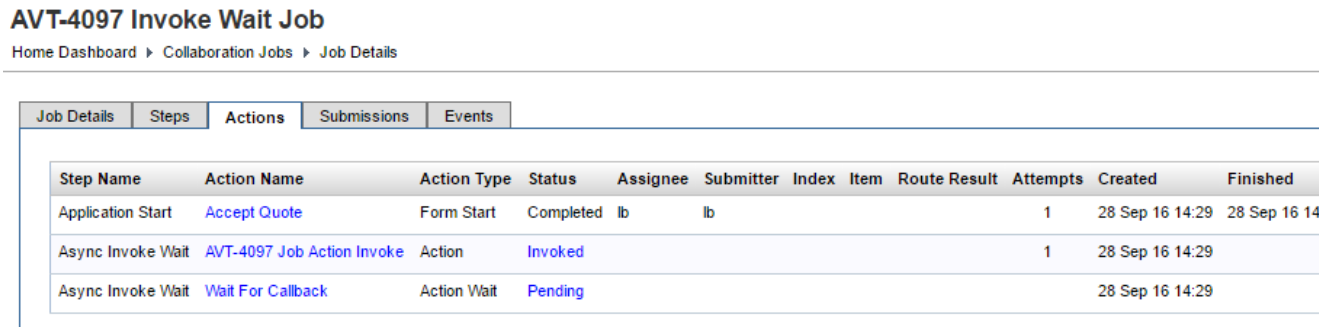
1. Run Form AVT-4097... Record the Reference code



2. menu Operations -> Collaboration Jobs
   Find Entry Press play

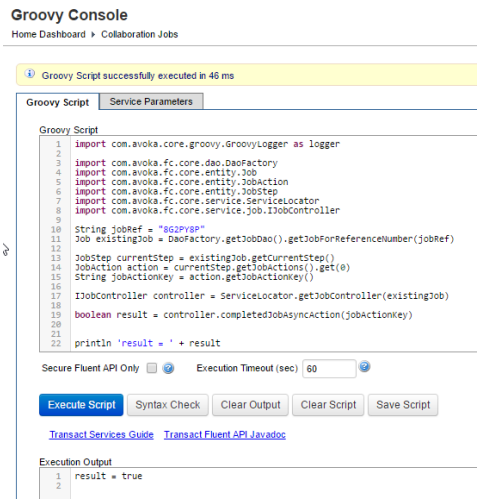
ID	Job Number	Job Name	Org.	Created	Job Status	Current Step	Current Action	Action Type
127	8G2PY8P	AVT-4097 Invoke Wait Job	maguire	28 Sep 14:29	In Progress	Async Invoke Wait	Wait For Callback	Action Wait

   Here we can compare the job Action Status



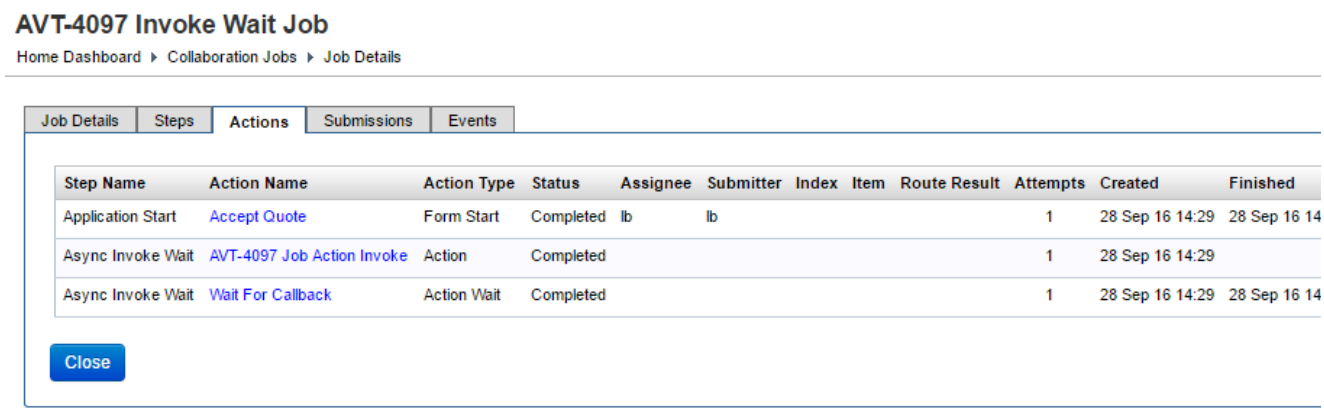
Step Name	Action Name	Action Type	Status	Assignee	Submitter	Index	Item	Route Result	Attempts	Created	Finished
Application Start	Accept Quote	Form Start	Completed	lb	lb				1	28 Sep 16 14:29	28 Sep 16 14
Async Invoke Wait	AVT-4097 Job Action Invoke	Action	Invoked						1	28 Sep 16 14:29	
Async Invoke Wait	Wait For Callback	Action Wait	Pending							28 Sep 16 14:29	

3. Run Call Back - note replace jobRef with code from step 1 or 2



4. menu - Operations -> Collaboration Job
   Find Entry Press Pay.

ID	Job Number	Job Name	Org.	Created	Job Status	Current Step	Current Action	Action Type
127	8G2PY8P	AVT-4097 Invoke Wait Job	maguire	28 Sep 14:29	Completed	Application Completed		



Application Package

Anonymous Tasks Example 1 (CollabJobs v4.3)

Summary

The following contains a flow similar to the 1 Step Review Job (CollabJobs v4.3) but with an anonymous tasks.

Application Package

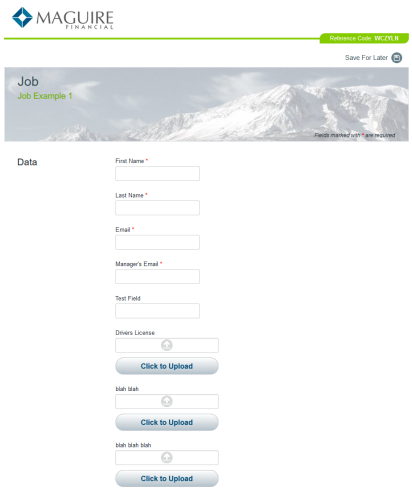
Error rendering macro 'view-file'

For input string: " 250 "

Components

Form

Job Example 1 - Anonymous



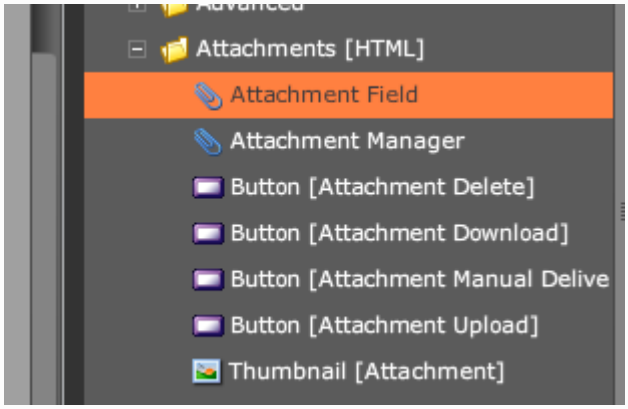
The extra fields are Managers email. This is who the Job Step Application Review is assigned to.

The first inform attachment field drivers licence is read only at the review step.

The other 2 inform attachment fields have no rules applied.

Note

The inform attachment field is located here in the composer palette.



Job

1 Step Review - Anonymous

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

Step Expiry (CollabJobs v4.3)

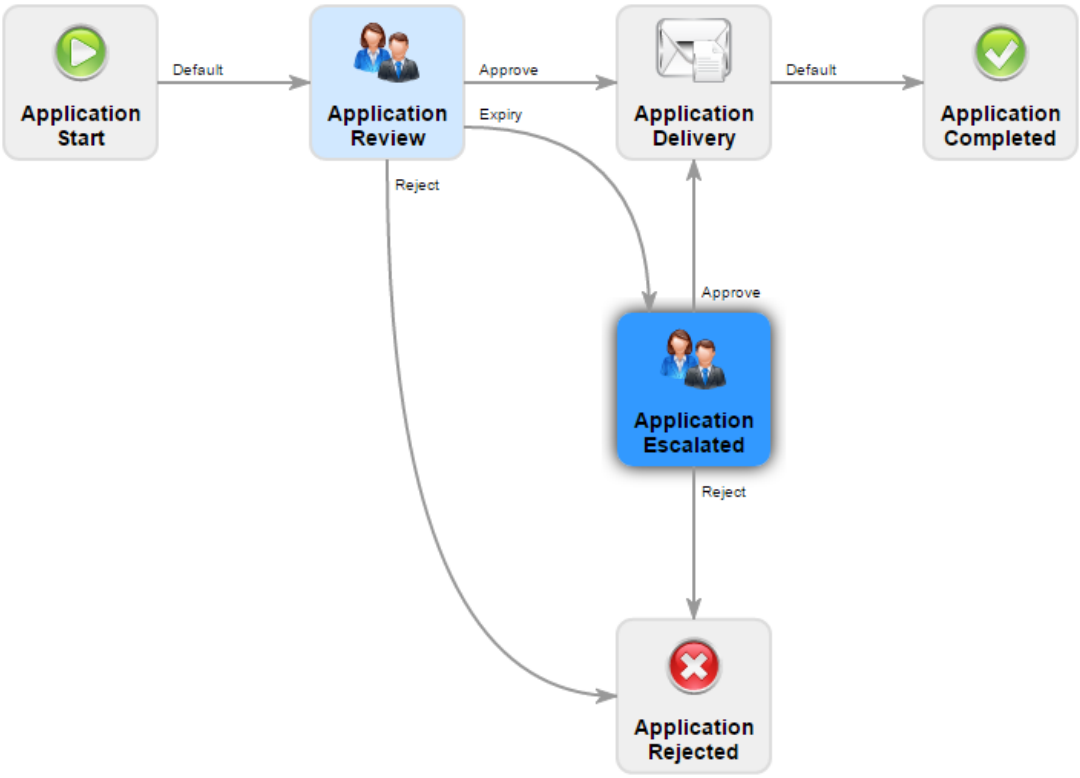
This example demonstrates how to setup expiry for a Step.

The original post was made using a TM 4.3.1 it used a modified Job Example 1. The application package can be found under Previous Versions.

Update July 2016 - Tested with TM 4.3.4

I have update the job by adding an Application Escalated step as per the job diagram below.

Now when the Application Review step expires it will be routed to the Application Escalated step.

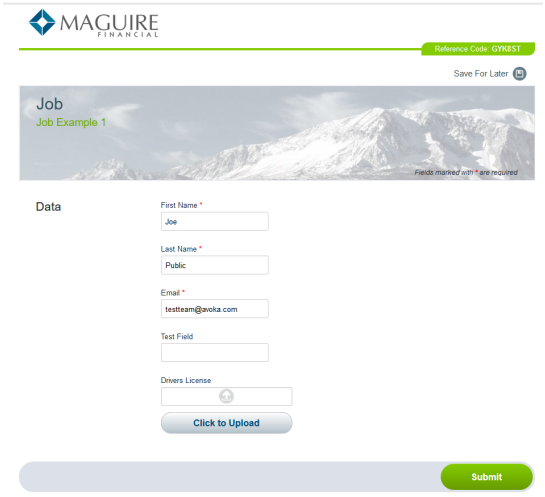


Note: This was modified to test of this scenario detailed here: 29360137

End to End Test Case

1. Application Start

Joe click on the AVT-3547 Step Expiry Form which opens



Joe's detail are prefilled he clicks submit.

2. Action Review Expiry

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.

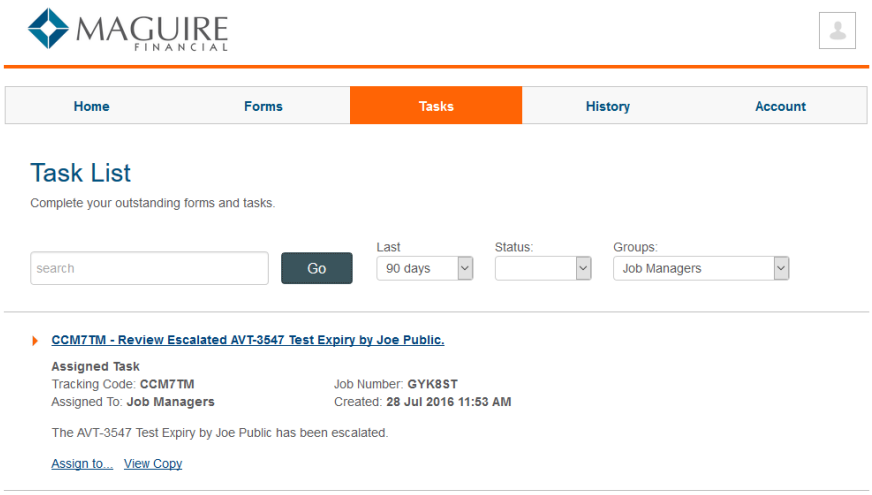
When the step expires these 2 actions are expired. The Application Review - Expiry action (Custom Groovy Service) runs to clean up the submissions for this step. For more details see the Job Expiry Service below.

3. Application Escalated

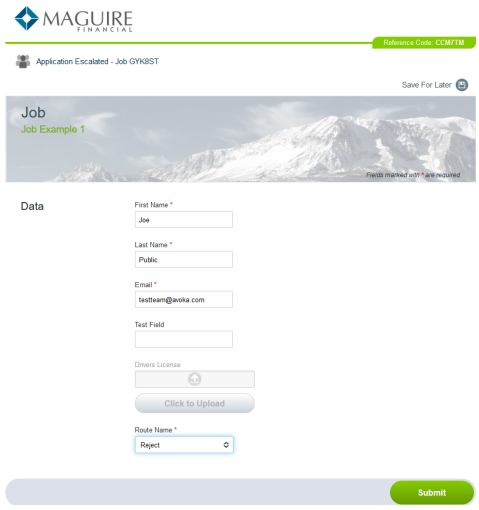
Log in as Mike Manager (Job Manager)

Goto Tasks page and select Groups: Job Manager... Press Go.

The tasks appears as follows



Click on the CCM7TM Task heading.



Application Package TM 4.3.4

Form

AVT-3547 Test Expiry

This is just a copy of Job Example 1 that kicks off the job controller below.

Job Controller

AVT-3547 Test Expiry B

This is a modified copy of the 1 Step Review Job. Version B introduces a new Group Task Step

The Application Review Step has been modified to

- expire after 1 minute* and to call the job Expiry service detailed below.

> **Note**
>
> The minutes settings is only available on transaction manager 4.3.1. Prior to the version hours were the minimum expiry rule time out was 1 hour eg "expiryRule" : "+1h"

- Call the **expiryServiceName** service below
- Expiry routes to the net step **Application Escalated**
- **Expiry** route added with display false.

Application Review and Application Escalated Steps

```
    {
      "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"
        }
      ],
```

Job Expiry Service

Job Expiry - Expire Step Tasks

This cleans up an task that hasn't been submitted.

It also returns the route name **Expiry**.

Job Expiry - Expire Step Tasks

```
 /* 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(submission);
            }
        }
    }
}
ActionResult actionResult = new ActionResult(Status.COMPLETED)
actionResult.setRoute(EXPIRY_ROUTE_NAME)
return actionResult
```

Previous Versions

TM 4.3.1

Customer Onboarding Job - Dynamic Form Bundles (CollabJobs v4.3)

Please review the Cusomer Onboarding Example that comes with the Maguire Aplication Package examples by clicking on the link below.

Customer Onboarding Job - Form Bundle (CollabJobs v4.3)

Coming soon

This page will discuss enhancing the standard Customer Onboarding form bundle with Dynamic Pre-conditions.

The standard form bundle will has a user fill a triage form. On submission of the triage form data extracts are created. These triage form data extracts are 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 has been submitted.

Task Cancellation Processor (CollabJobs v4.3)

Summary

I would like to cancel/abandon a task submission and/or a task that is part of a collaboration job using API.

Background

JobController 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) {
```

Groovy Service Usage

```
Job job = submission.getJob()

def jobControllerService = ServiceLocator.getJobController(job)

jobControllerService.cancelJob(job)
```

Working Implementation

The following service archive contains a submission completed processor that cancels a job.

Step By Step Examples (CollabJobs v4.3)

The following use the Maguire forms, form spaces and services provided with Transaction Manager installer for installation on development environments.

- Customer Onboarding Job - Form Bundle (CollabJobs v4.3)
- Multi Step Group Review Job (CollabJobs v4.3)
- 1 Step Review Job (CollabJobs v4.3)

User Setup

The 2 Review Jobs above use specific actors to fulfil user roles. The Groovy Script can be copied to the Groovy Console to create these Users for these examples.

> **Note**
>
> Before running you should update the following lines for your test email and password.
>
> ```
> final EMAIL_TESTTEAM = "testemailgroup@yourcompany.com"
> final PASSWORD_COMMON = "replace with password"
> ```

Create Test Users

```
/**
 * 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 = ServiceFactory.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) userMap.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) userMap.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"
    }
}
```

Customer Onboarding Job - Form Bundle (CollabJobs v4.3)

The Customer Onboarding Job comes with Transact 4.3.0 Maguire Examples.

Form Bundles are a number of separate forms that are filled out by a single user in the one step.

The example demonstrates the following features:

- Anonymous use case which suits onboarding type forms.
- Pre Conditions to determine which forms to use.
- Maguire Form Bundle Navigation
- Redirect Next Functionality (Form Chaining)
- Confirmation Signature Page
- Consolidation of receipts

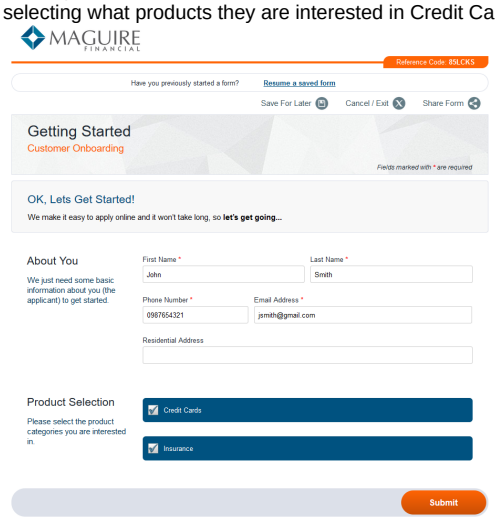
A separate advanced example demonstrates

- Dynamic Pre Conditions
- Shared form modal using
    - entire form xml (shareFormData)
    - publish / subscribe of Data Extracts - using shareExtractData

Overview

A user opens the triage form Customer Onboarding and fills out

- name, phone number and email,
- selecting what products they are interested in Credit Cards and / or Insurance.

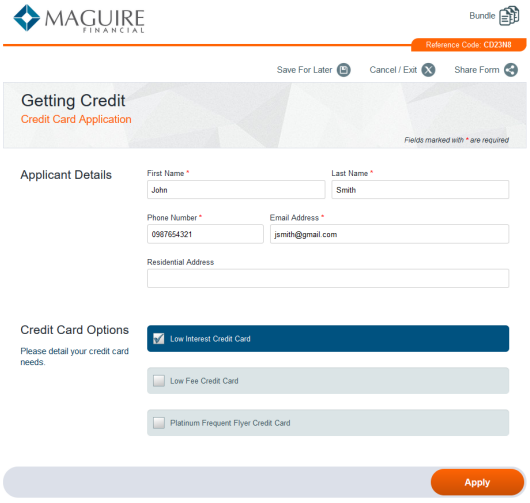


Click Submit

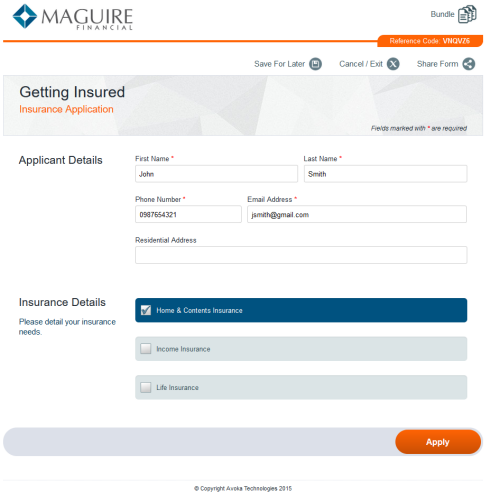
A collaboration job is started and the processing immediately progresses to the Additional Products step (ref Job Definition below).

Here the following Tasks are created.

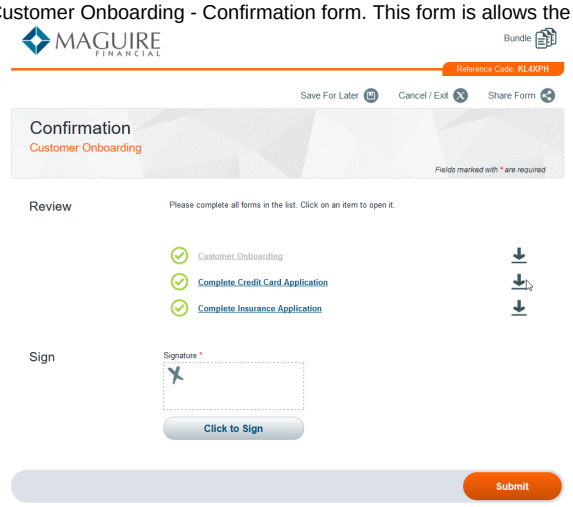
1. Customer Onboarding - Credit Card form (created if Credit Card was checked: "preCondition": "$formDataMap.productCreditCards == 'true'",).



2. Customer Onboarding - Insurance form (created if Insurance was checked: "preCondition": "$formDataMap.productInsurance == 'true'",).



3. Customer Onboarding - Confirmation form. This form is allows the user

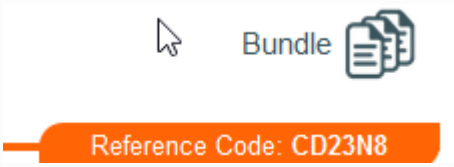


The Redirect Next functionality takes the user to the next form / task that has not be completed.

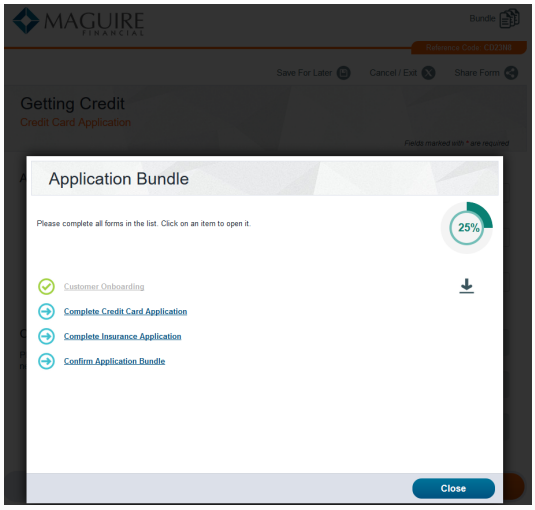
A happy day scenario would open the credit card. Submitting would take them to the insurance form then the confirmation page.

Note

The Maguire template has a form bundle option. When turned on you get a Bundle icon at the top right of the form.



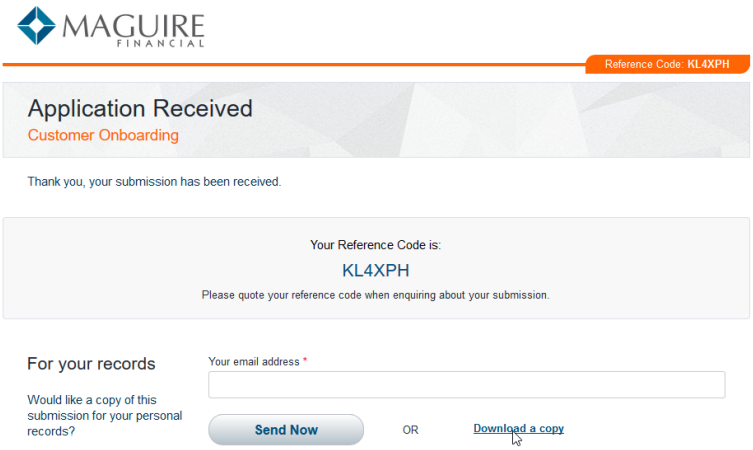
Clicking on this brings up the Application Bundle navigation screen where its possible to jump to other forms in the bundle in a different order to the Redirect Next.



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 (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.



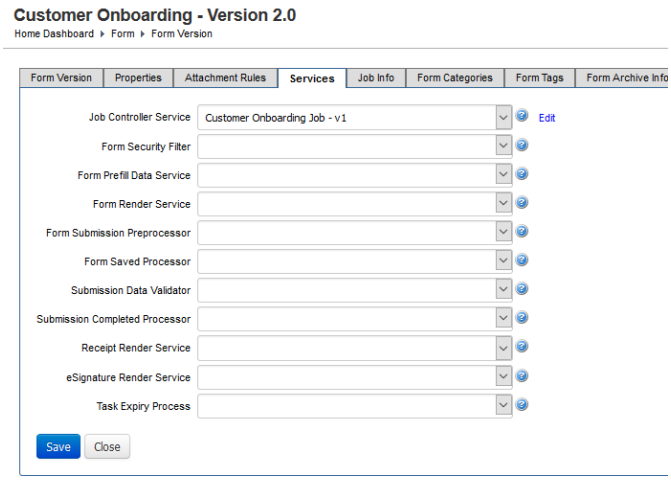
Setup

Forms

Form Name	Org.	Form Code	Cur
Customer Onboarding	maguire	onboard-customer	
Customer Onboarding - Confirmation	maguire	onboard-confirmation	
Customer Onboarding - Credit Cards	maguire	onboard-credit-cards	
Customer Onboarding - Insurance	maguire	onboard-insurance	

Collaboration Job

The **Customer Onboarding** form is the starting or Triage form. This is linked to the **collaboration job**:



Job Definition

```
{
  "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 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" }
      ]
    },
    {
      "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"
    }
  ]
}
```

Multi Step Group Review Job (CollabJobs v4.3)

? Unknown Attachment

This example uses the Form Job Example 2 (see form Setup below) combined with the **Review and Approval - Multi Step Group Review** Job Controller (Job Template).

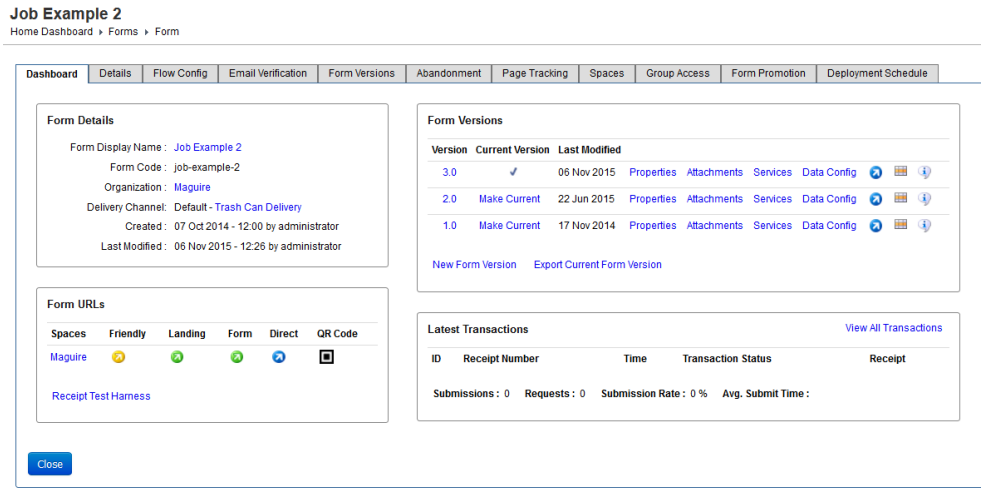
The Setup of the Form, Job Controller and Actors are discussed below.

> **Note**
>
> This example assumes you are familiar with concepts introduced in 1 Step Review Job (CollabJobs v4.3) example.
>
> Instructions or ideas presented in the previous example may be omitted.

Example Setup

Form

The Multi Step Group Review example uses the **Job Example 2** form as per below.



Note:

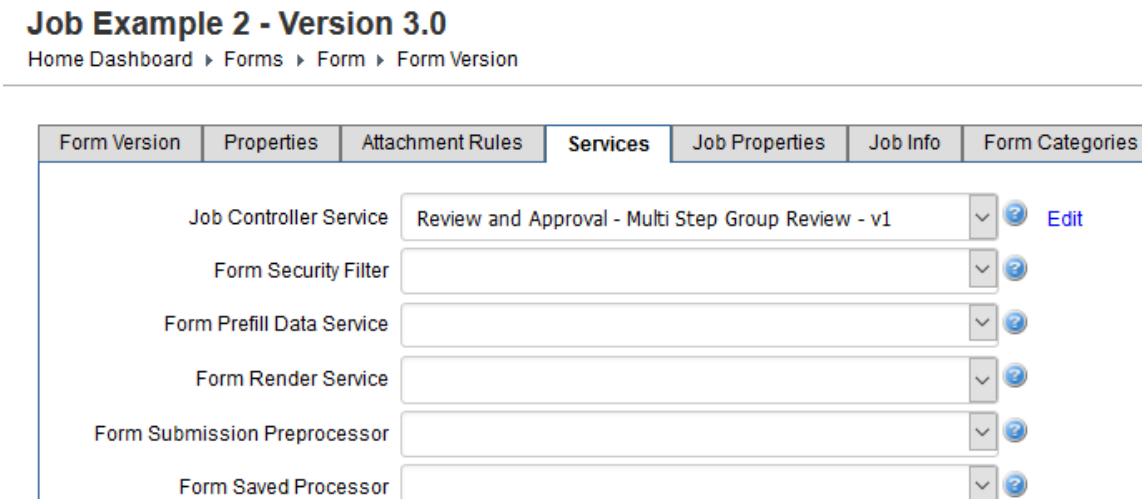
Form versions 1 and 2 were part of previous releases. They are both minimal forms like Job Example 1.

Form Version 3 released with Transact 4.3.0 and is a more fully featured form that has:

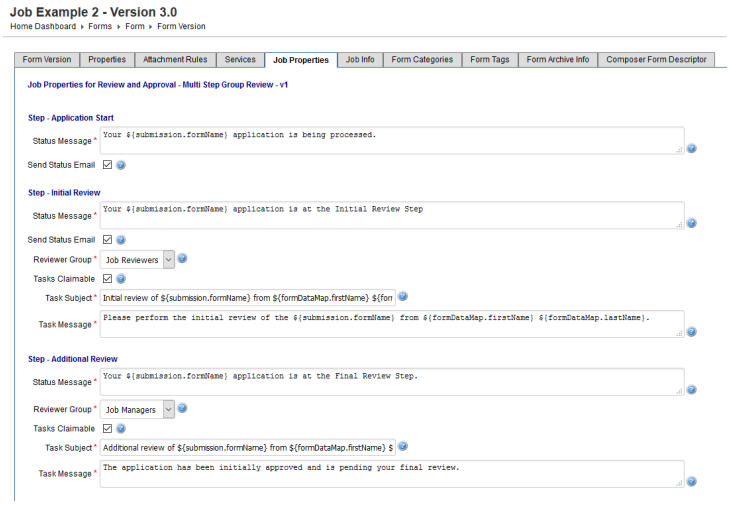
visibility and editability rules based upon the Job Step.

uses Radio Buttons

It works with this Job Template



After selecting the template please make the changes to the Reviewer Groups as per the screenshot below.



Job Controller

The following Job Definition is from the Job Controller Template **Review and Approval - Multi Step Group Review**

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

How It Works

Example Actors

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 Coordinator	cat	Cat Coordinator	Cat looks after the Job as a whole and needs to track where jobs are up to. She does this using the Reviews interface in the Maguire Portal.

Job Diagram

? Unknown Attachment

User Step

Please refer to the Applicant steps (White) and Reviewer steps (blue) in the Job Diagram above. The table discusses the route options at each user step.

Step	Group	login	Description
Start	Job Applicants	anne	Anne starts the 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** **Decline**: Routed to the **Applicant Update** Step **Terminate**: Routed to the **Terminated Initial** endpoint
Additional Review	Job Managers	mike	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 and **Update**: Anne can update the form and the route it back to the review step that declined the application.** **Terminal**: Routed to the **Terminated Applicant** endpoint.

Note

Please review the section Job Basics (CollabJobs v4.3)#Step Definition on routes for the documentation on these highlighted features above.

Conditional Route Name*

The following Action Property is from the Task Wait Service. It defines a rule that is applied on the Server Side using collaboration Jobs.

```
{ "name": "Conditional Route Name",  "value": "#if ( $formDataMap.routeName == 'Approve' && $formDataMap.
loanAmount >= 20000 ) Exceeds Threshold #end" }
```

The following is the routes listed. Note the **Exceeds Threshold** has the **display** attribute set to **false.** This means this route is not put in the Available Routes see Form Design (CollabJobs v4.3)#Job Context Information.

```
"routes": [
  { "name": "Approve",  "nextStep": "Application Delivery" },
  { "name": "Decline",  "nextStep": "Applicant Update" },
  { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display": "false" },
  { "name": "Terminate",  "nextStep": "Terminated Initial" }
]
```

##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" }
]
```

Note Form Design

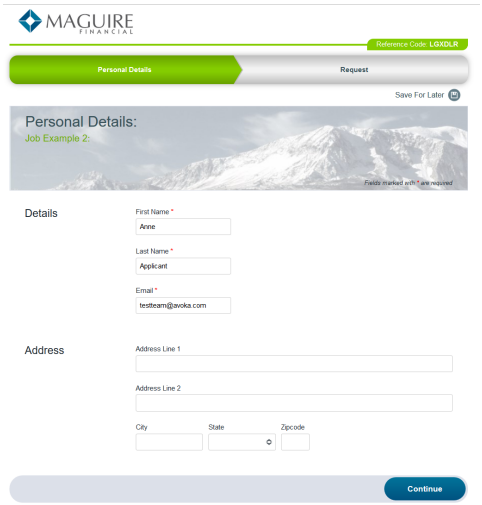
The routes in the form below use radio buttons.

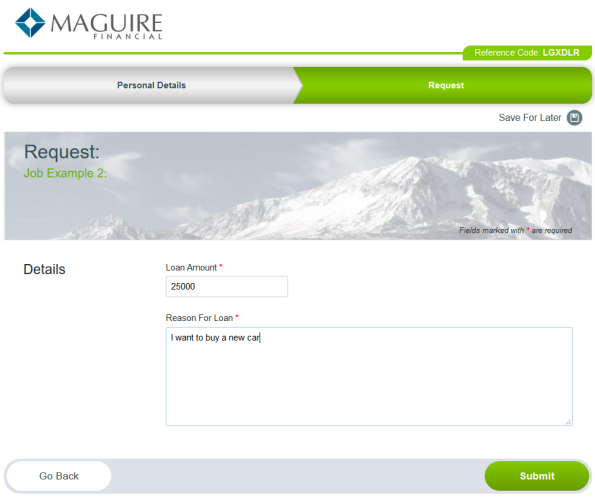
Please see Composer (CollabJobs v4.3)#Routeing Using the Standard Radio Button Group

Example Execution

Anne- Start

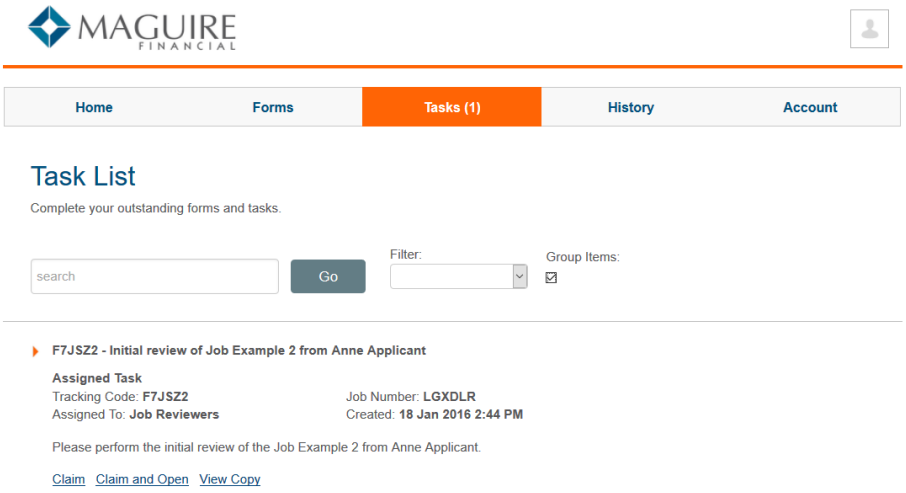
Fills in the 2 sections and submits.



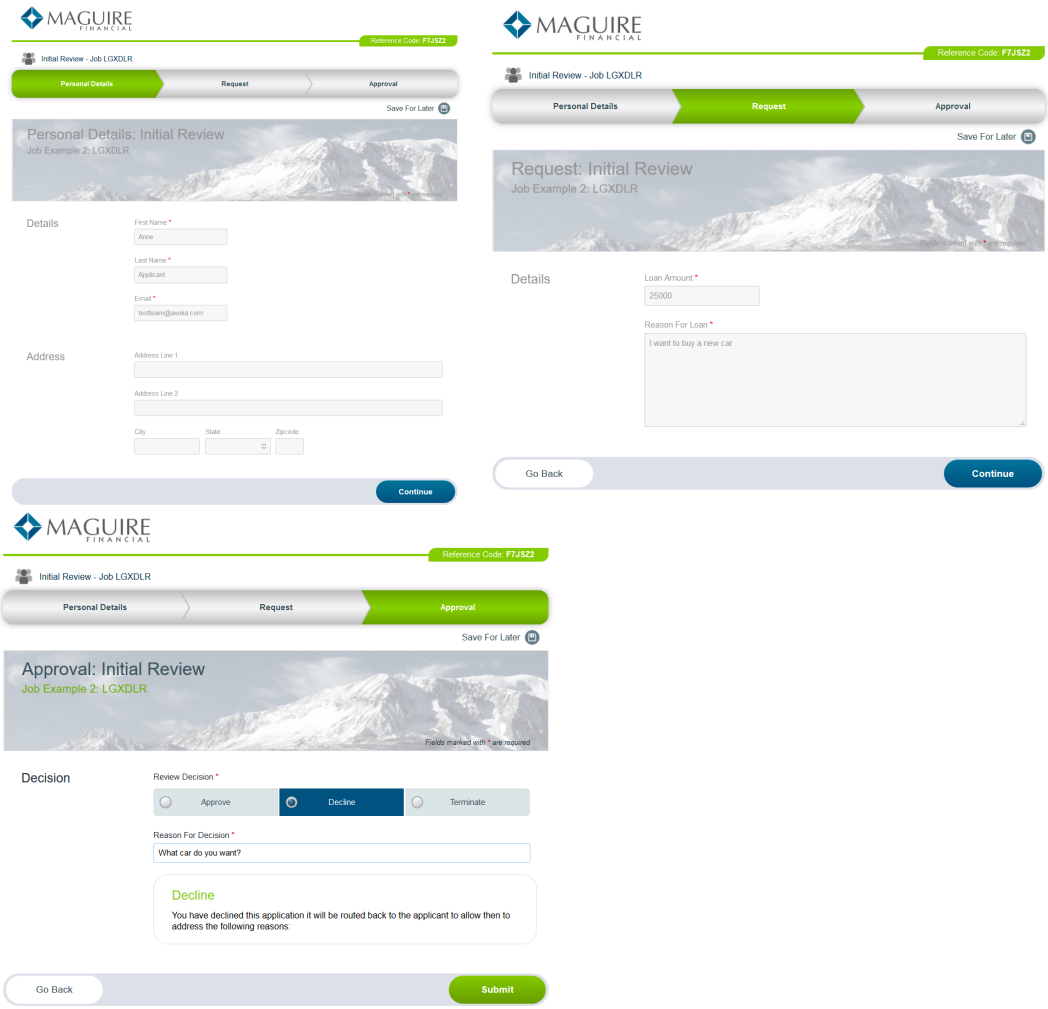


Roger - Initial Review

sees the group task and claims and opens



Roger reviews the first 2 sections. He then selects the Decline radio button* asking Anne "What type of car?"

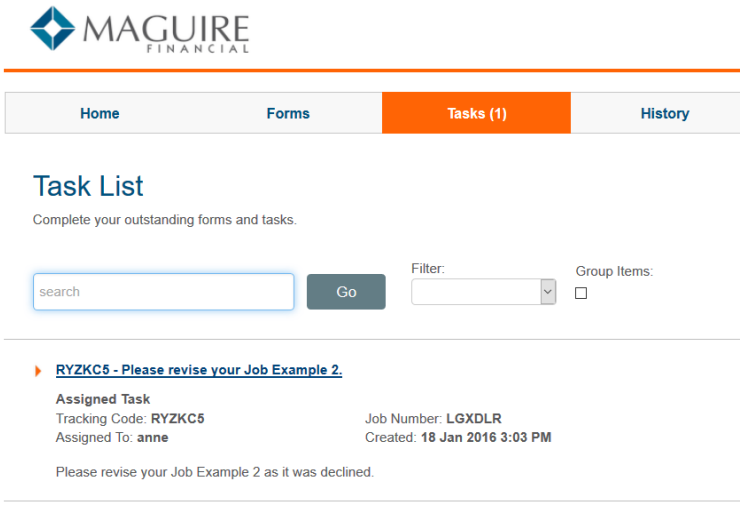


Note

See Composer (CollabJobs v4.3)#Setting Route with Business Rule when using more that 1 radio button groups to control the Route Name. *

Anne - Applicant Update

Sees the tasks assigned to her

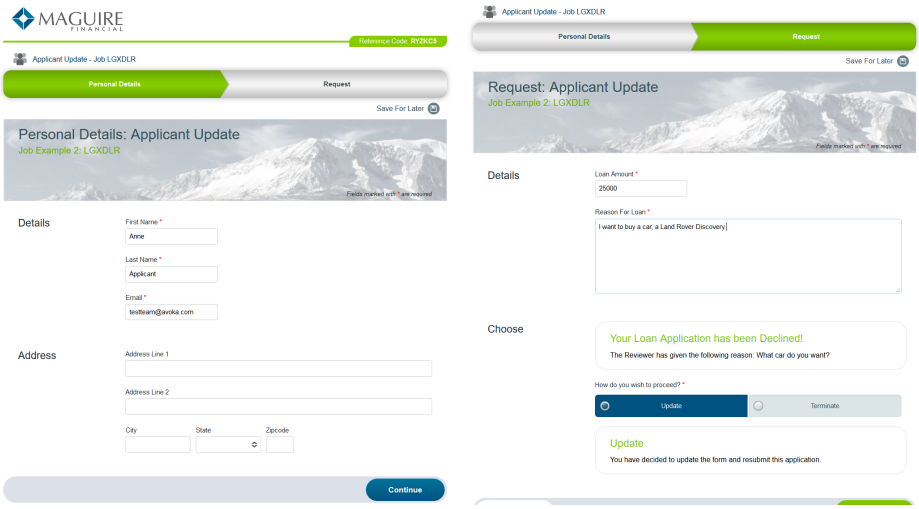


Open Form, Note the Request now has an extra section Choose.

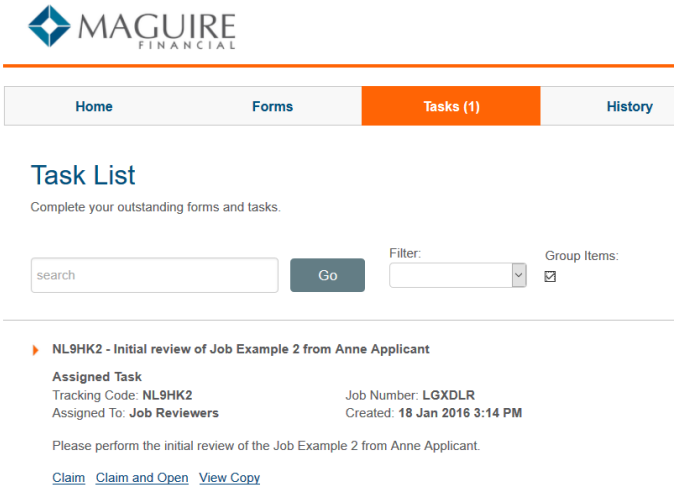
>Select the Update radio button*

>Type the new car make into the details.

>Click Submit



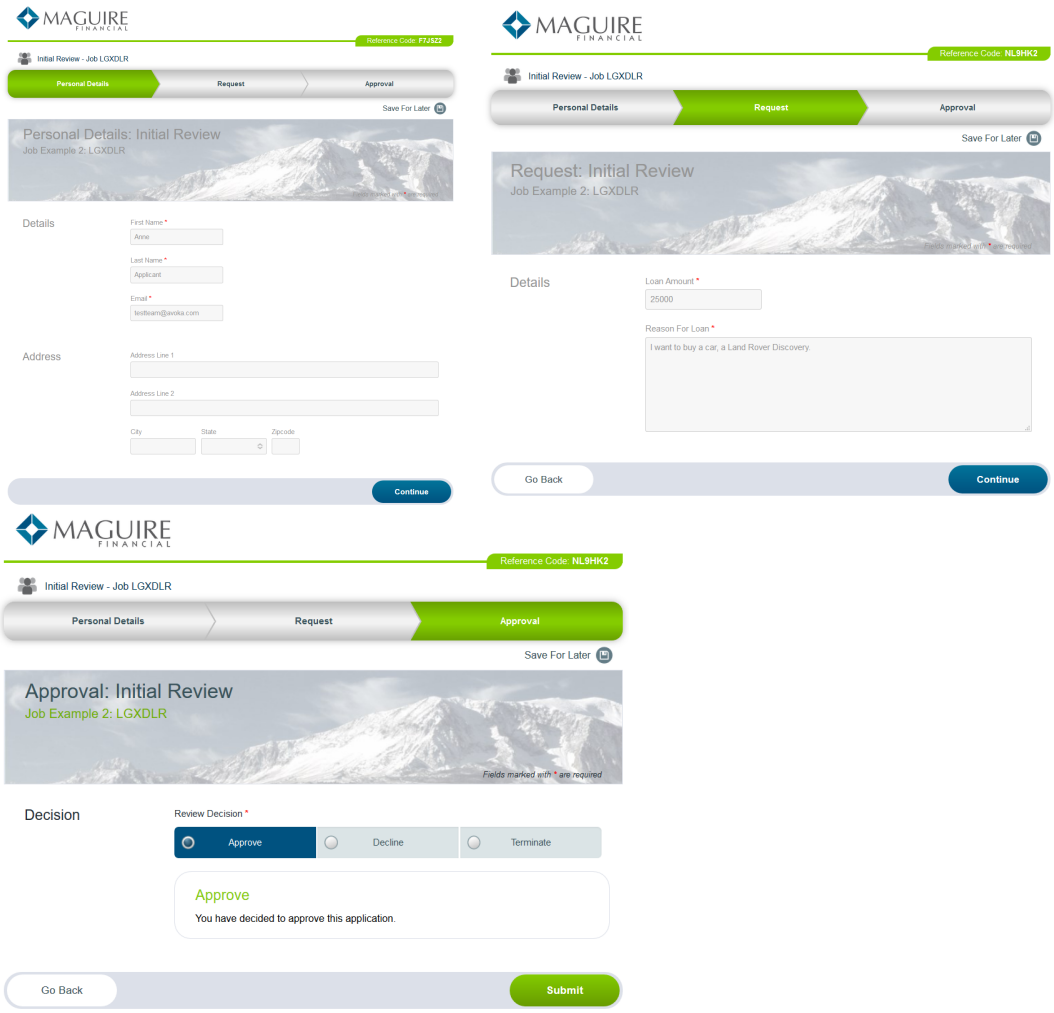
Roger - Initial Review



> Select Claim and Open

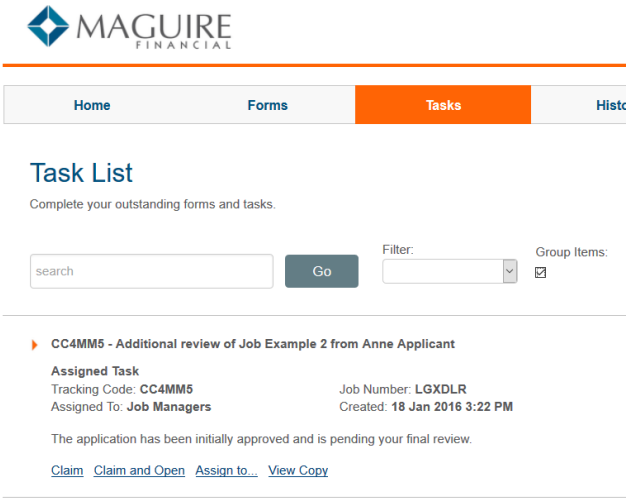
> Review Request - Reason For Loan

> In Approval section select the Approve Radio Button.



Mike - Additional Review

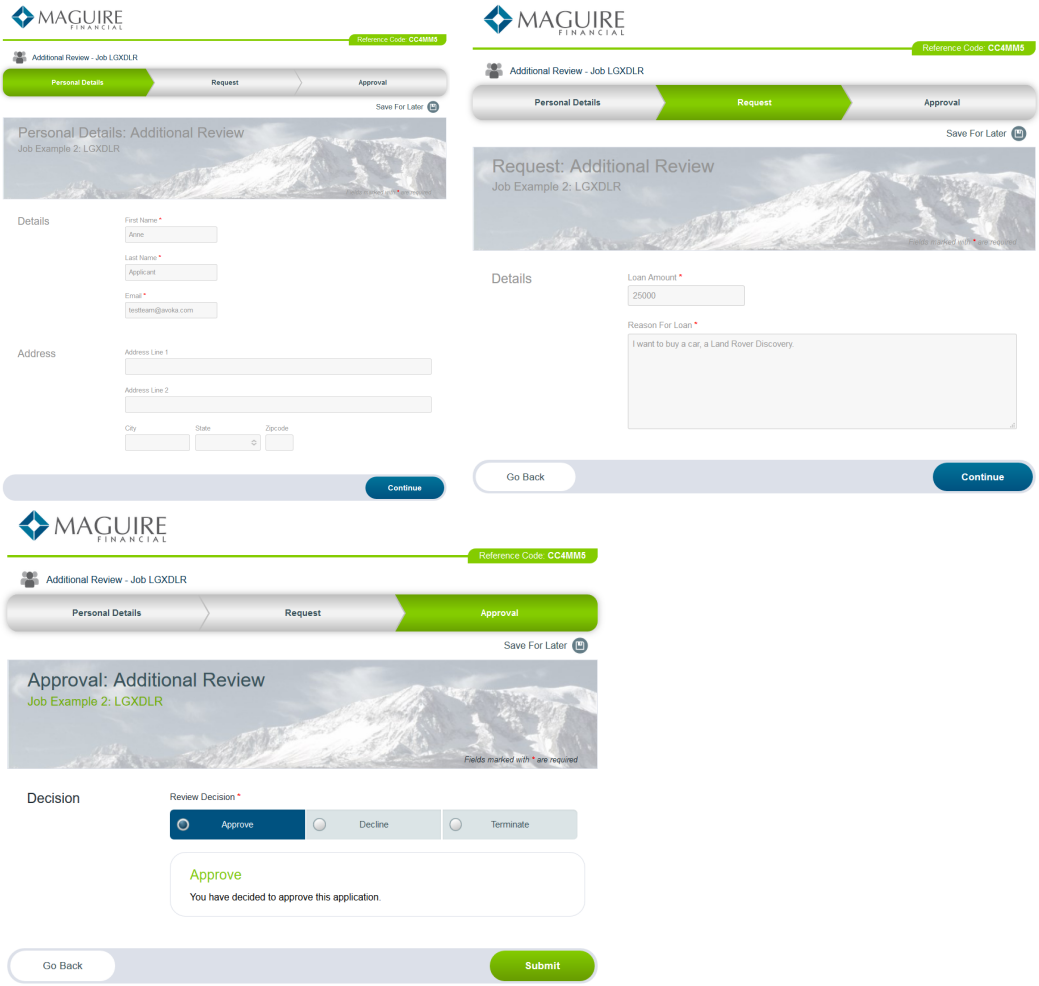
As the loan amount was greater that $20,000 the job is routed to Mike



> Select Claim and Open

> Review Personal Details and Request Section

> In Approval section select the Approve Radio Button.



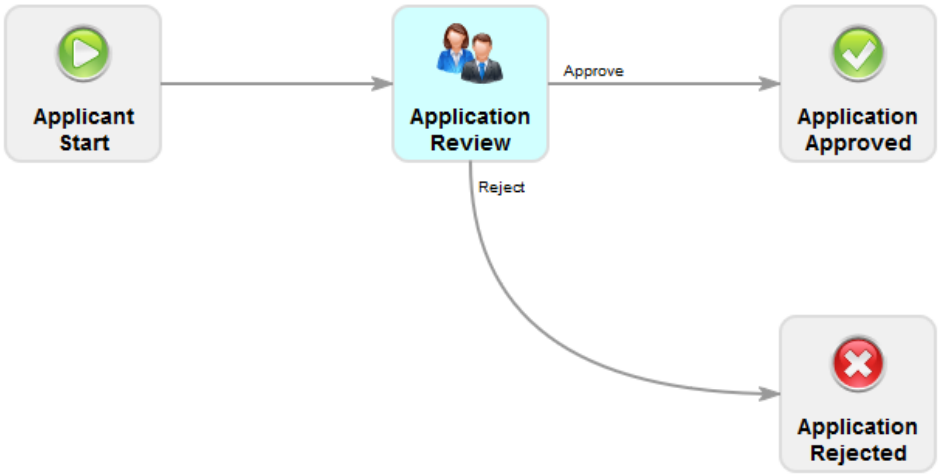
Summary of Sections Visibility and Editability

Section	Start	Initial Review	Additional Review	Applicant 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

To set visibility and editability rules please review the Form Design Composer page under the **Composer (CollabJobs v4.3)#Visibility and Editability Rules** section.

1 Step Review Job (CollabJobs v4.3)

Summary



This example uses **Job Example 1** which is a minimal form (see Form - Job Example 1 below). A minimal form contains only the fields that are used in the job .

It is a very simple workflow where:

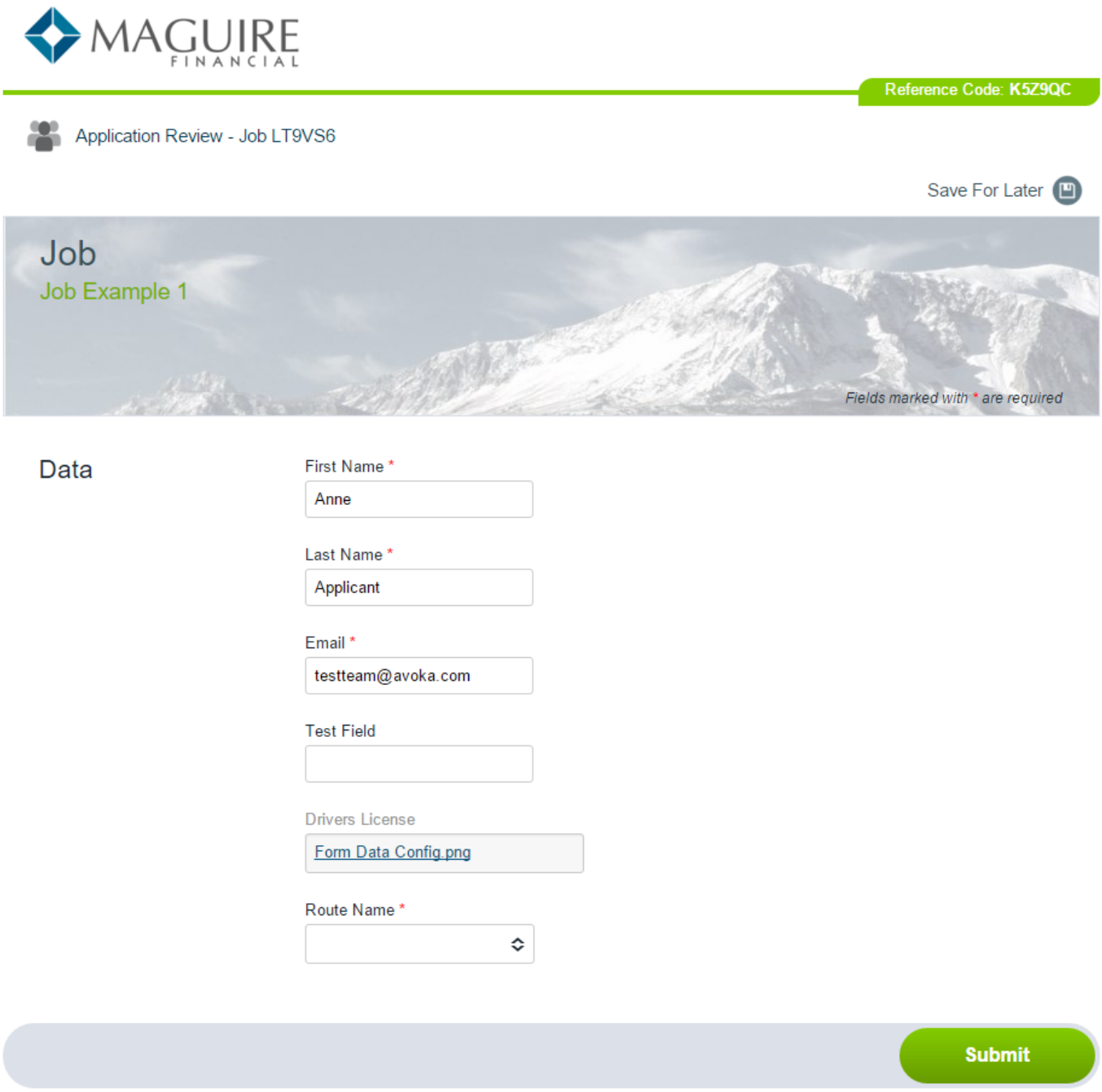
- An application form is filled in and submitted by a user.
- The form next goes to the **Application Review** step where it is reviewed by a member of the **Job Reviewer** group.
- The reviewer may **Accept** or **Reject** the application.
    - If it is rejected the job moves to the **Application Rejected** endpoint.
    - If it is accepted the form is delivered (Step not shown) then progresses to the **Application Approved** endpoint.

The example also shows how attachments can be reviewed using the In Form Attachment widget.

Setup

Form - Job Example 1

Job Example 1 is a form built using Composer below is a screenshot of the form.



Fields

Mandatory:

First Name (Text Field)

Surname (Text Field)

Email (Email Field)

> **Note**
>
> These fields will be pre-populated when an authenticated user is logged into the form space.
>
> These are set up as Data Extracts which are documented in Form Design (CollabJobs v4.3). and for Composer (CollabJobs v4.3).
>
> Other forms can use the same Job Controller as long as they have these same Data Extract field.

Other:

Test Field (Text Field)

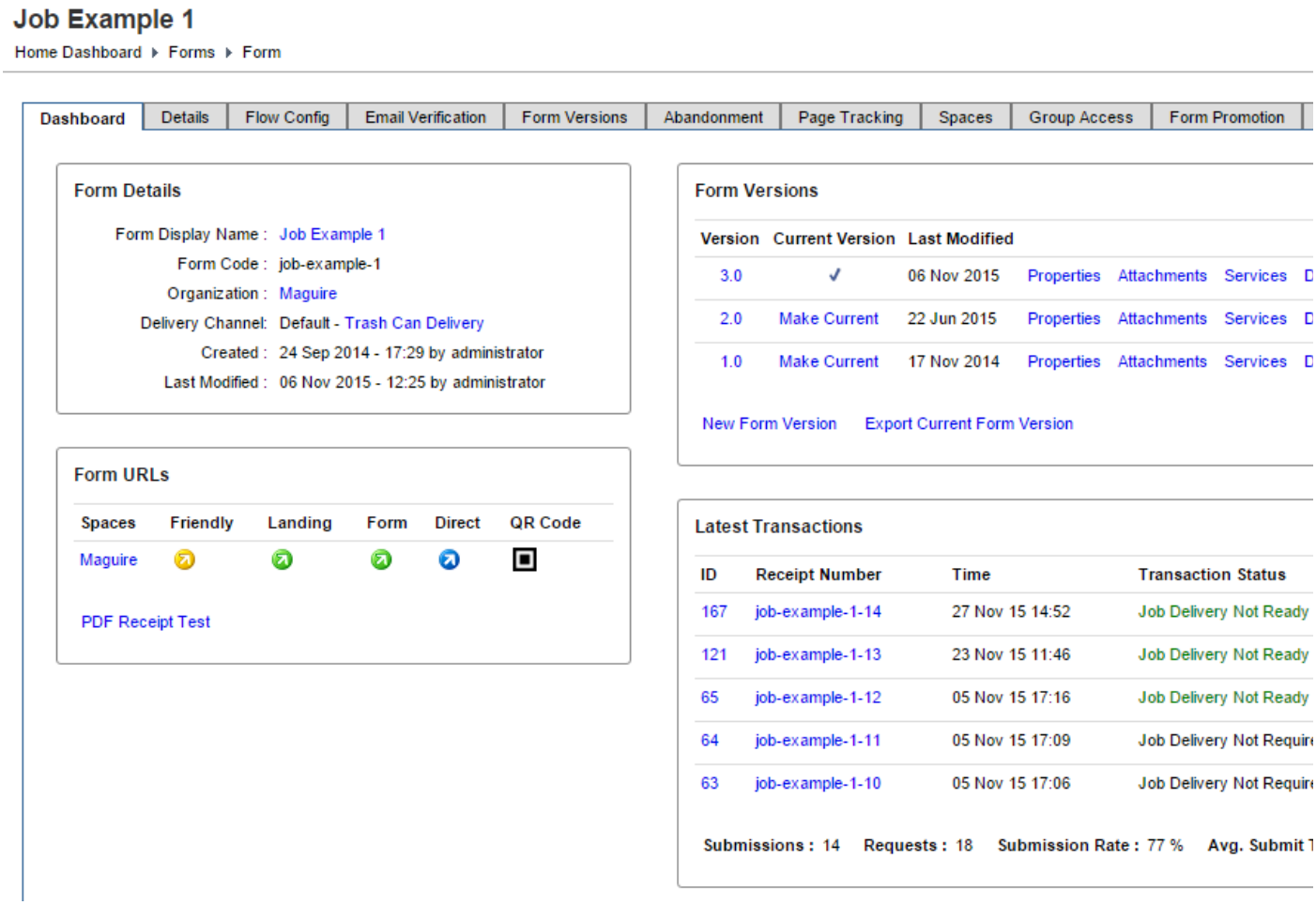
Drivers License (Inform Attachment Field). This is setup so it's read only at the Application Review step.

Route Name (Collaboration Routes Dropdown).

> **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 see Form Design (CollabJobs v4.3)#Job Context Information.
>
> 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.

Setup

Open the Form Dashboard for Job Example 1



Confirm that the Form Details Delivery Channel for this form is set to Trash Can Delivery as per the screenshot below.

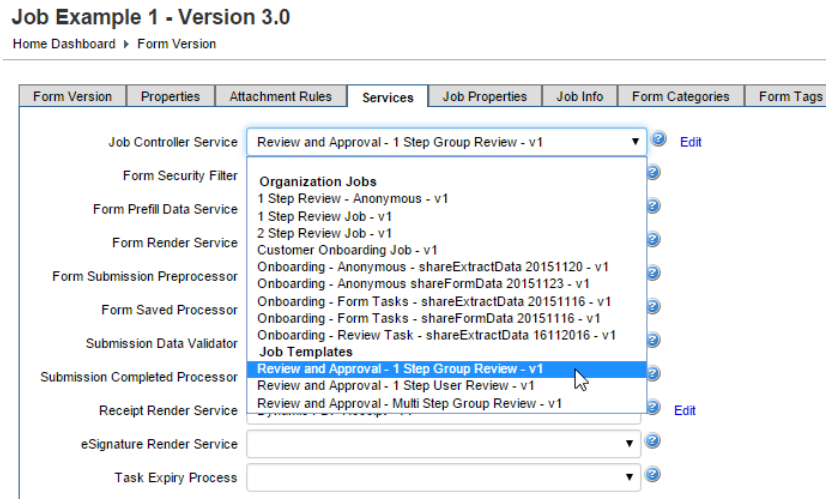
Set the Job Controller Service

From the Dashboard click on the **Form Version - Services** link for the latest form version.

This will open the Form Version - Services Tab.

> Select the Job Basics (CollabJobs v4.3)#Job Templates : **Review and Approval - 1 Step Group Review**

> Select Save.



Note

The Job Example 1 form also works with the Job Template: **Review and Approval - 1 Step User Review** Job Controller

This then opens the **Job Properties** tab, which is available for Job Templates.

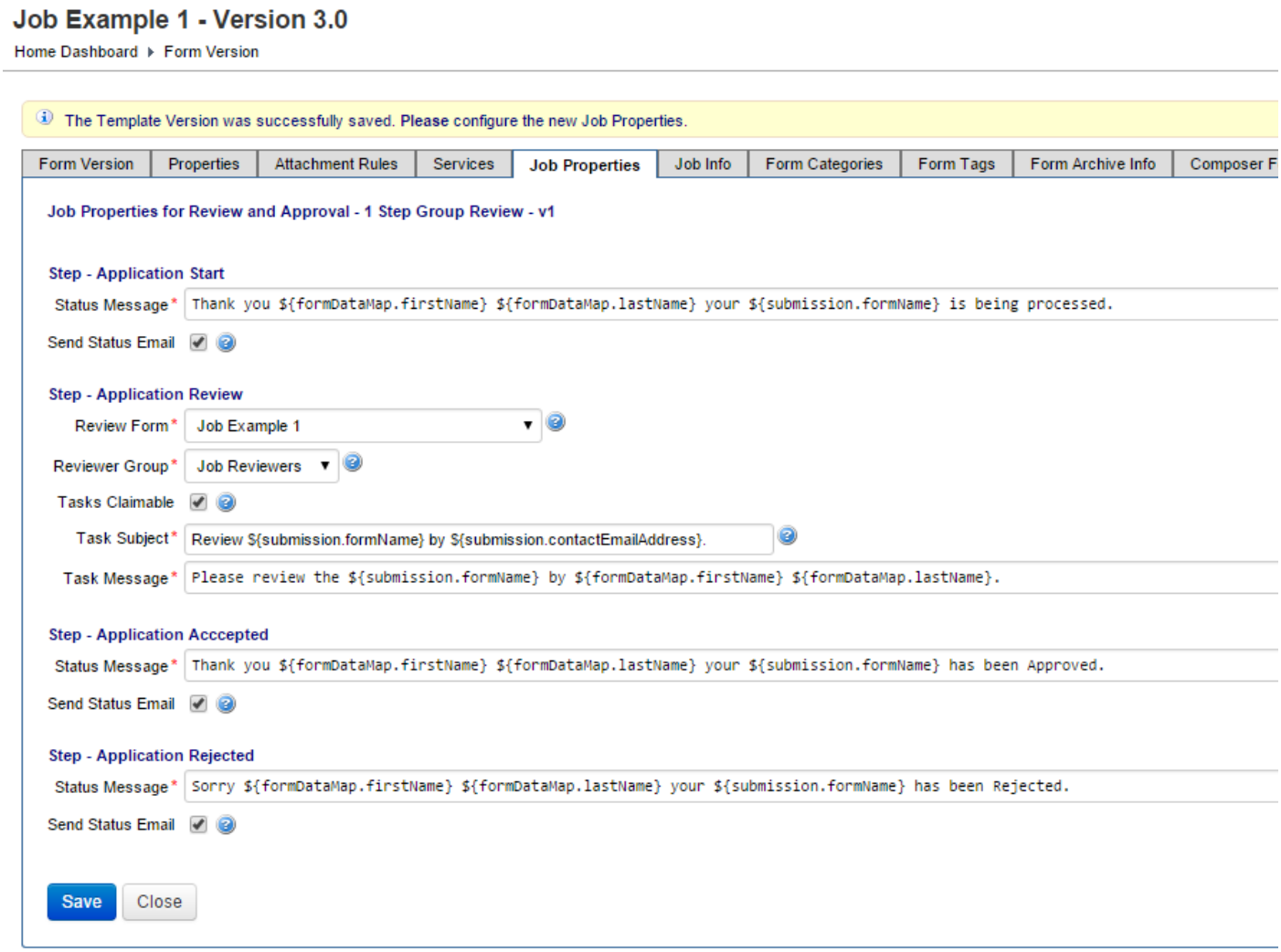
Under the Step - Application Review

> Fill the following as per the screenshot below.

Review Form: Job Example 1

Reviewer Group: Job Reviewers

> Select Save



Note

This is where the Job Controller can be customised on a this form.

Note the use of Data Extracts and other model elements in the screenshot above. See Job Basics (CollabJobs v4.3)#Action Properties for details.

Actors: Users and Groups

For the screenshots in my Example I created 2 users that have access to the Maguire Portal and are in the Maguire organisation..

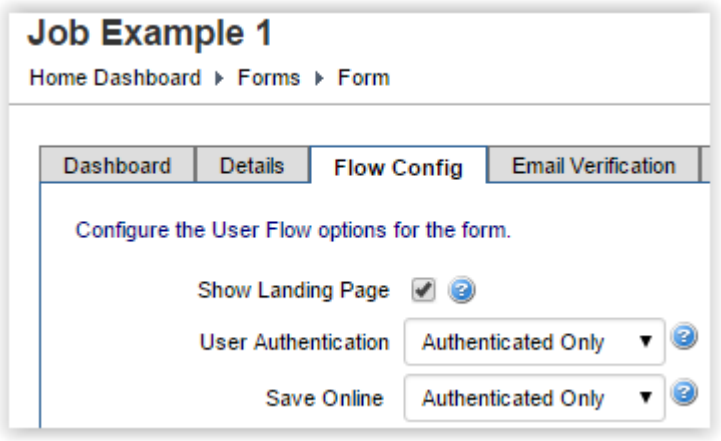
1. Anne Applicant who does the Application Submission
2. Roger Reviewer who does the Application Review
   We will add Roger to the **Job Reviewers** group.

Note

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. For more detail see Form Space (CollabJobs v4.3)#Public and Authenticate Users

Note: The Job Example 1 form pre-population only works for an Authentication User. The example works better with this settting Form > Flow Config > User Authentication: **Authenticated only**.



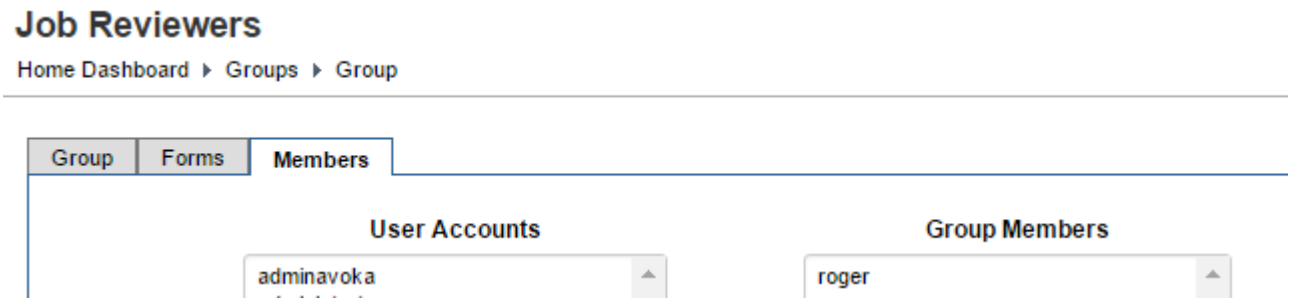
Set Job Reviews Group

> Select Menu **Security -> Group**

> Edit **Job Reviewers**

> In the Members tab add Roger to the Group Members.

> Select Save



Running the Example

Maguire Form Space - Anne

> Log into the Maguire Form Space as **Anne** **

> Click on the Forms Tab.

> Scroll down and select **Job Example 1**

> Click **Open New Form**

This will display the form

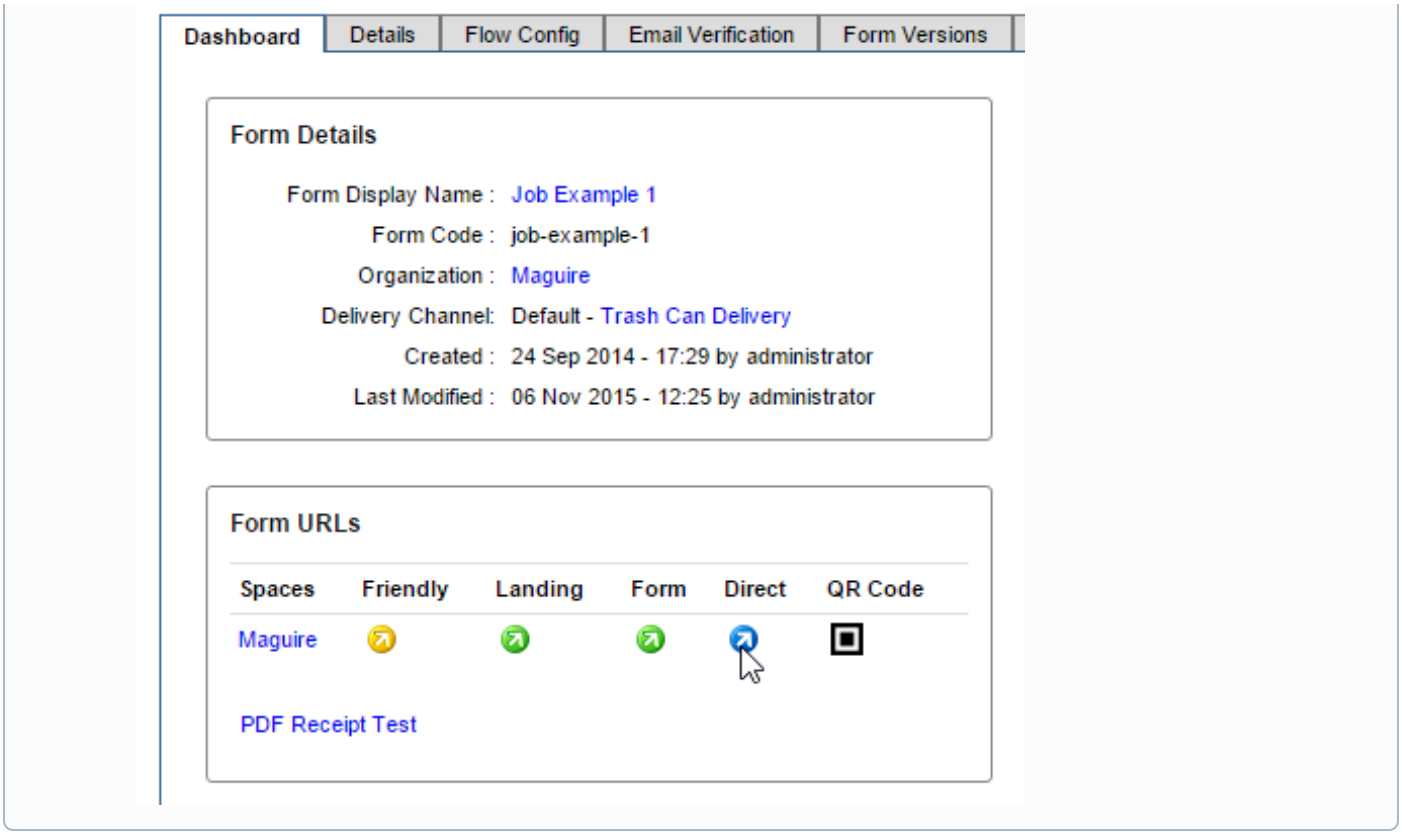
Note

To Run the example you should be familiar with concepts covered in Form Space (CollabJobs v4.3)#Secured Form Space

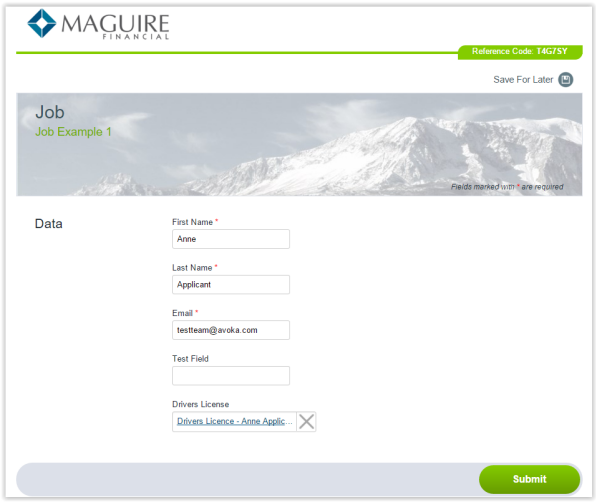
** **Tip for Development and Testing**

Click on the direct url link (blue arrow button) within the Form Dashboard - Form URLs as per screenshot below. This will open the 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.



> Make sure that the First Name, Last Name and Email are populated as per screenshot below.

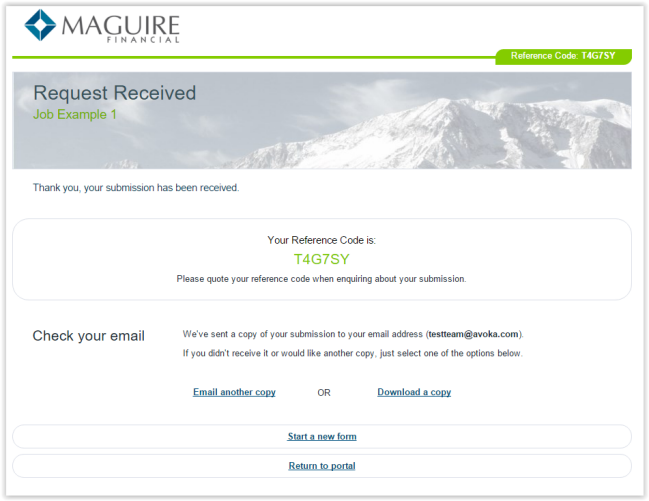


> Upload an image file to the Drivers Licence Field.

> Select Submit

The Job Example Confirmation page is shown

> Note the Reference Code for this form submission which is used as the Job Number.



> Click on the return to Portal (Form Space) link

This should take you to the Maguire Form Space as per the screenshot.

Transaction Manager - Administrator Role

The submission creates the Collaboration Job

> Go to Transaction Manager in a separate browser tab.

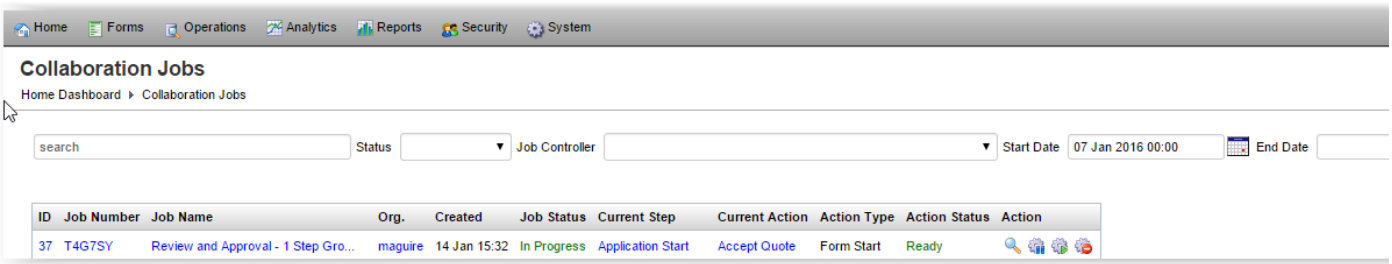
> Menu Operations > Collaboration Jobs

You will see the Job has been created and has the following state:

Current Step: **Application Start**

Current Action: **Accept Quote**

Action Status: **Ready** (This says that the Job is available and is waiting to be processed see Job Basics (CollabJobs v4.3)#Job Execution)

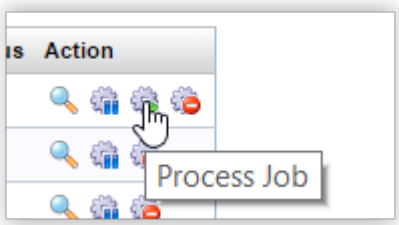


Note

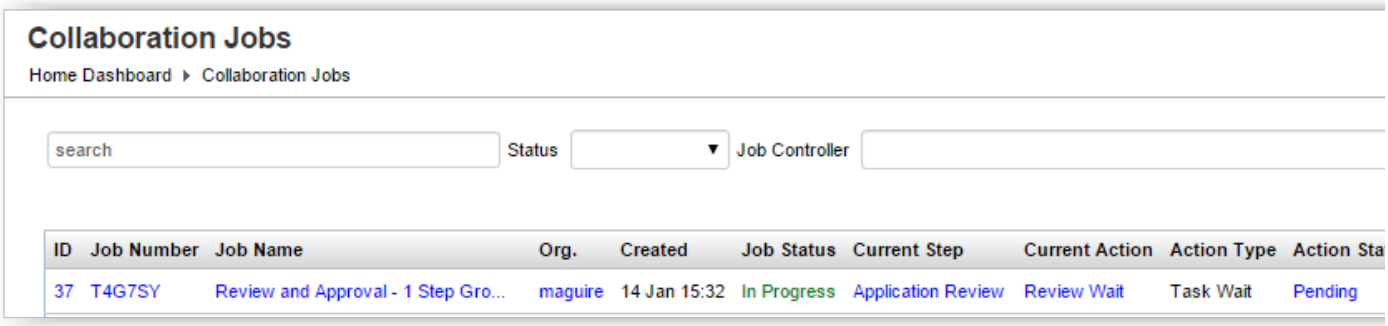
Be familiar with Admin Operations (CollabJobs v4.3)#Collaboration Job Management

To speed the testing of the example.

> In the Action column of the Job Entry Press the Process Job icon as per the sheet below.

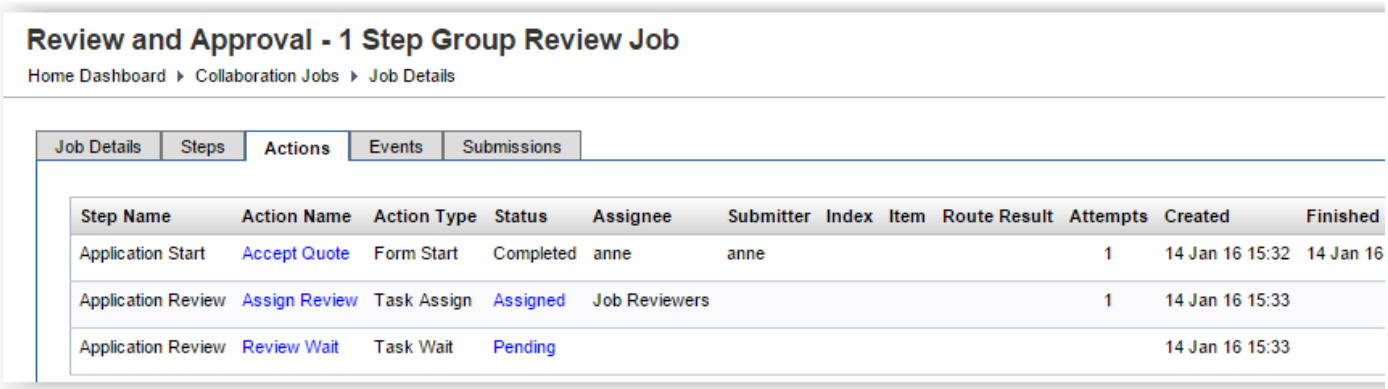


After the job has been processed it moves to the **Application Review** step



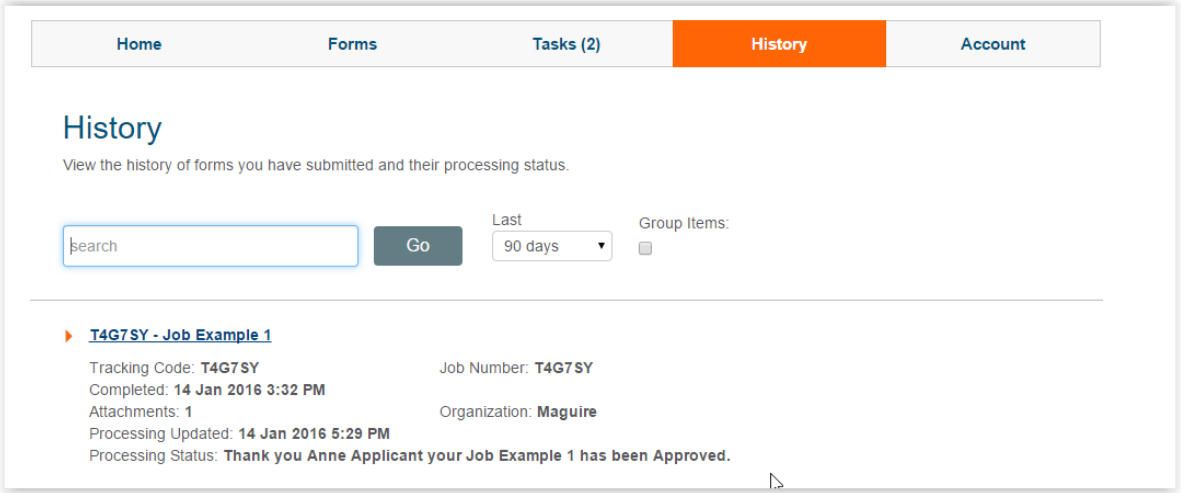
> Drill down into the Actions tab for this Job instance.

Note the **Application Review** - **Assign Review** entry is assigned to **Job Reviewers**.



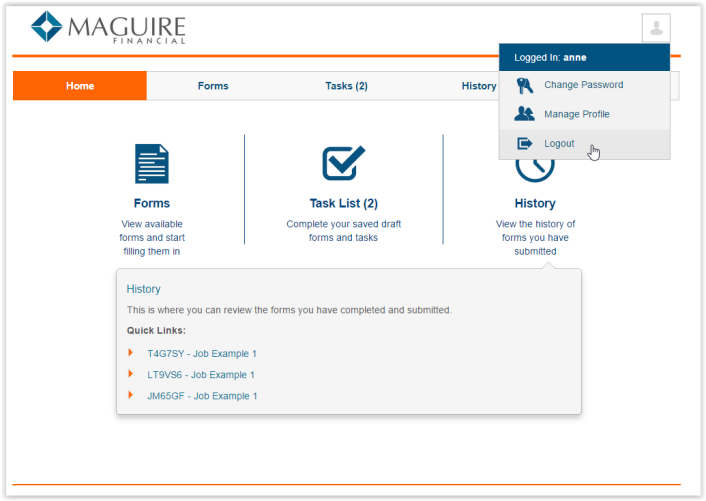
Maguire Form Space - Anne

> Click on History Tab



We see the Processing Status

> Logout as Anne



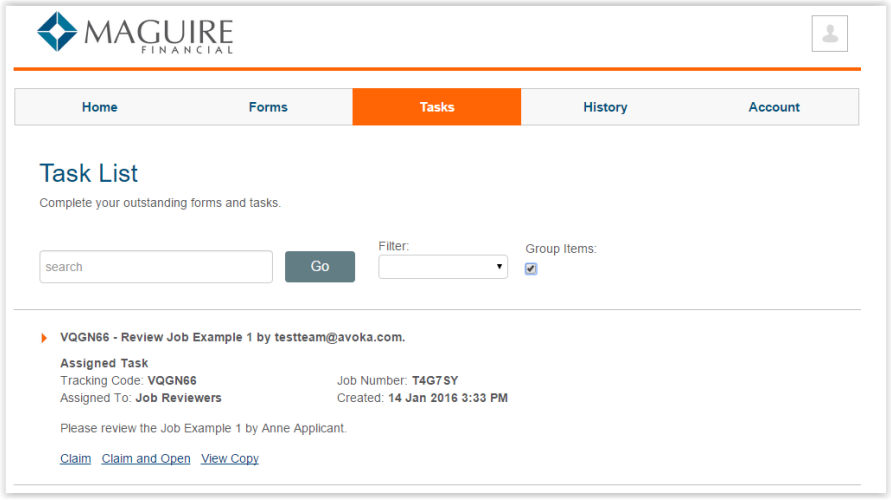
Maguire Form Space - Roger

> Log into the Maguire form space as Roger

> Click on the Tasks Tab

> Check the Group Items check box

You should see the Group Task details as per the screenshot below.



> Click Claim and Open, the form should open as per the screenshot below

Note

The Step Name and Job Number displayed in the top left corner under the Maguire logo

The Drivers Licence field (inform attachment) is read only, this is done with an editability rule as per Composer (CollabJobs v4.3)#Rules

The Route Name dropdown is now populated with the **Approve** and **Reject**.

> Select Route Name: Approve

>Select Submit

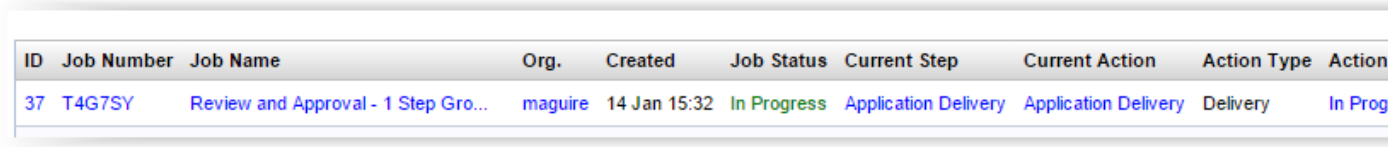
The Confirmation page will be displayed

> Select Return to Portal and then logout.

Transaction Manager - Administrator Role

> Find the Job, If its still at the Application Review step then Press Process Job

The Job moves to the Application Delivery Step, where it is waiting for the delivery to be completed.



ID	Job Number	Job Name	Org.	Created	Job Status	Current Step	Current Action	Action Type	Action
37	T4G7SY	Review and Approval - 1 Step Gro...	maguire	14 Jan 15:32	In Progress	Application Delivery	Application Delivery	Delivery	In Prog

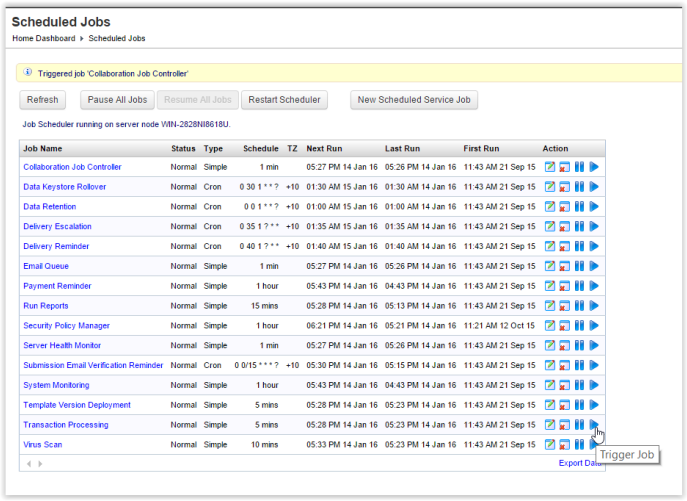
Note

The delivery is performed by the Transaction Processing - Scheduled Job which by default runs every 5 minutes.

To speed this example along you can do the following.

> select menu System > Schedule Jobs to get to the schedule jobs screen as per the screen shot below

> then press Trigger the Transaction Processing.

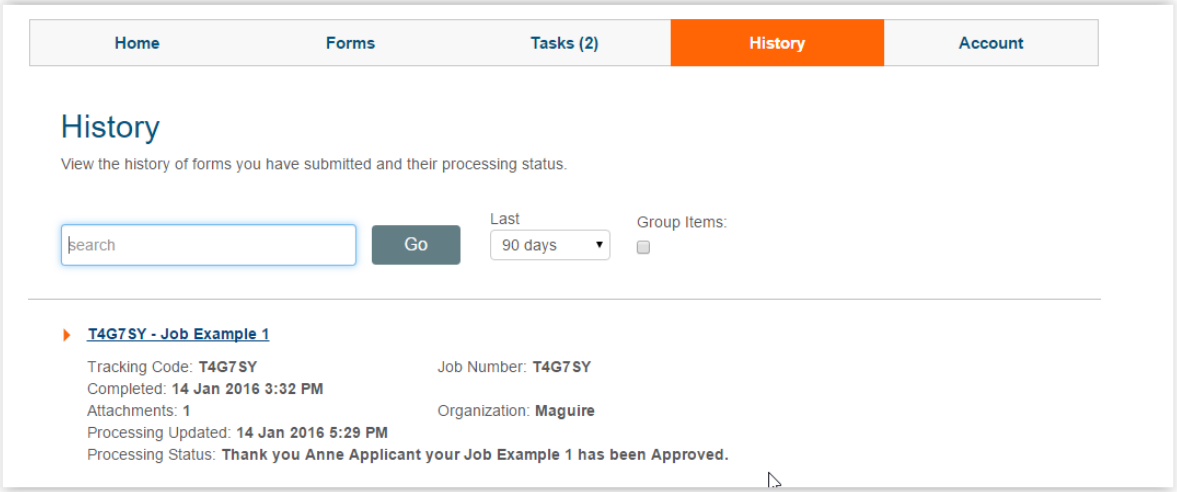


Maguire Form Space - Anne

> Login as Anne

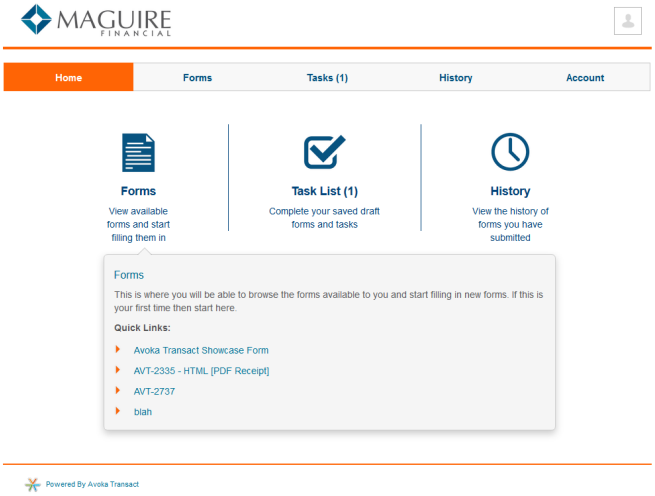
> Select the History Tab

We can see the processing status of the application has now been approved


Home
Forms
Tasks (2)
History
Account
History
View the history of forms you have submitted and their processing status.
search
Go
Last
90 days
Group Items:
T4G7SY - Job Example 1
Tracking Code: T4G7SY
Job Number: T4G7SY
Completed: 14 Jan 2016 3:32 PM
Attachments: 1
Organization: Maguire
Processing Updated: 14 Jan 2016 5:29 PM
Processing Status: Thank you Anne Applicant your Job Example 1 has been Approved.

Form Space (CollabJobs v4.3)

Transact uses Form Spaces a means to host the forms, list of the tasks assigned to users an show a users submission history.



Public and Authenticate Users

Authenticated User

Authenticated users either login to Transact Space using credentials that are delegates authentication and authorisation to 3rd party system. Transact integrates to a number of authentication and authorisation systems such as Active Directory, LDAP and via SSO.

- Business Users Internal to an Enterprise.
- Government Department.
- Staff and Contractors that work in the field.

Transaction Manager can also maintain local users. An example could is contractors that aren't held in the organisation LDAP system.

Authenticated users have access to features of Secured Form Space as per below.

Public Anonymous User

Members of the public click on a link a clients website which opens an unsecured form page in the portal. The available list of forms are manually maintained by the client on their website.

Using anonymous users has the following advantages:

- Removes the overhead of maintaining users and passwords.
- Storing public users can have issues around privacy.

However Anonymous users do not have access to the the advance features of the secured portal such as Form, Todo and History pages as described below.

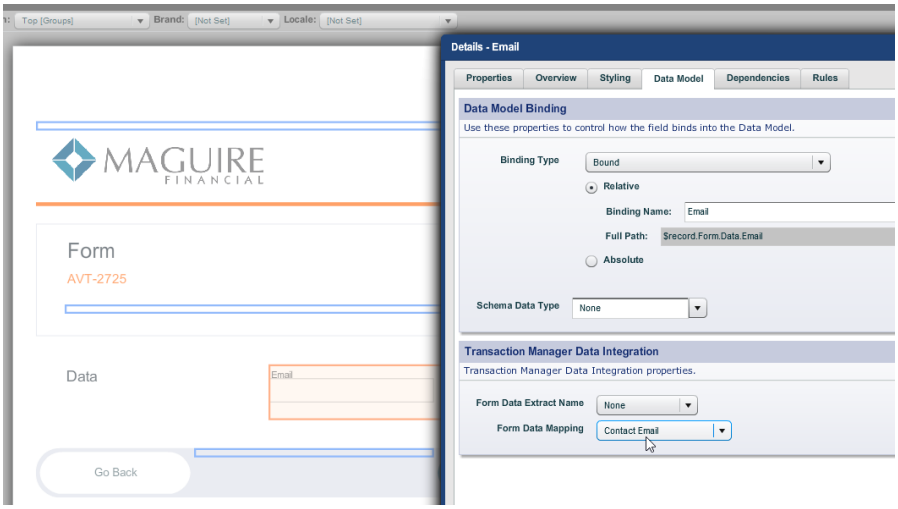
An anonymous user can still participate:

- 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 the user as the job progresses.

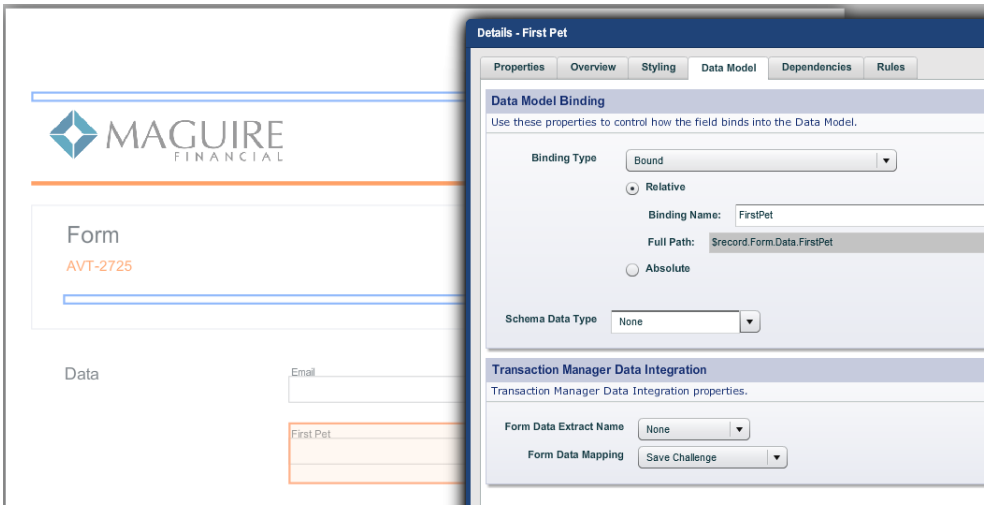
Anonymous users must provided their email address. Optionally a security question can be set to open saved forms and tasks.

Anonymous Form Setup

The form is hosted on an unsecured page in the space. At minimum the public user filling out the forms must provide their **email** address. This email is used to send status updates, saved transactions and can also be used to assign tasks back to the user. The following shows a composer form that has an email field configured.



In addition one of the form fields can be used to provide a security challenge question as shown below. The security challenge question is entered by the user when they open a saved form or anonymous task created by the job later.



Secured Form Space

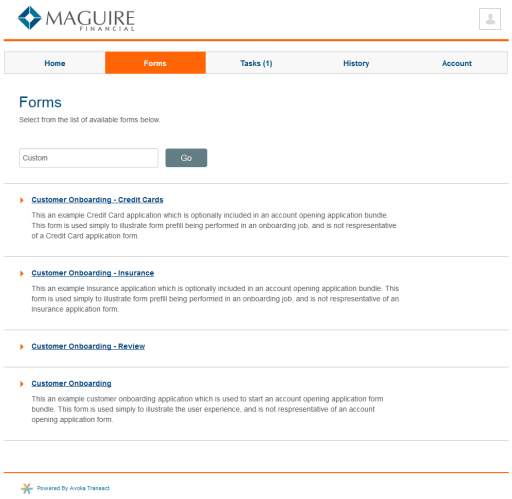
The following describes the section are only available to users that are authenticated.

Common Portal Features

These features are available to all users that are assigned to a space.

Form Page

This Form page list all the forms available for a user to start. The user can filter the results using a .



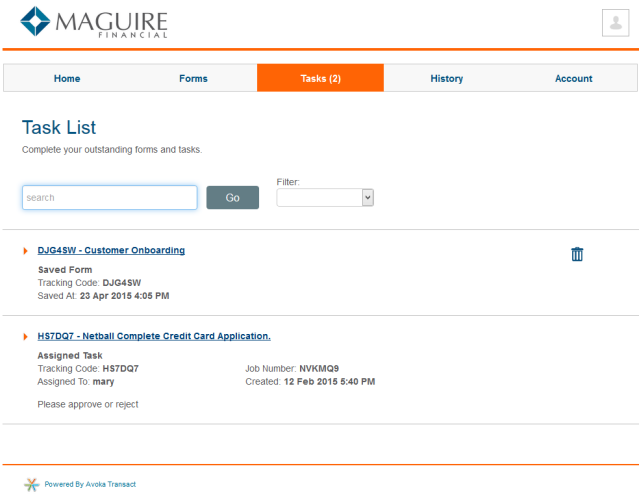
Note:

- Form Groups are used to restrict what forms are listed here reg Form Groups - User Access Control below. By default a form is not assigned to a form group an 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 visible in this list to members of the form group.

Tasks Page

This page list Saved Forms, Assigned Tasks and Submitted forms. Note submitted forms are ones that require additional attachments or payments and are not using most . Most Collaboration Jobs will use the inform transaction management features of Maguire which means the submitted forms will be empty.

The user can filter the results on a keyword; on Saved, Assigned or Submitted; or to show Group Items.



Assigned Task Types

A job create tasks which are assigned to a user, group or anonymous user (email).

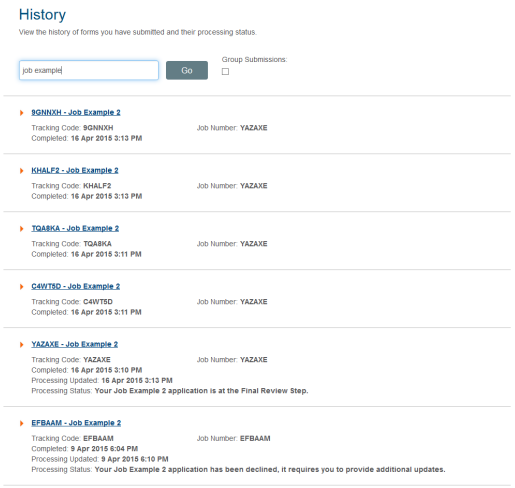
There are 3 task types:

- **Form Task**: The form xml can be pre-filled. Assigned to a User or a Group and displayed to secure users in the Todo Page of the portal.
- **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 Todo Page of the portal.
- **Anonymous Task**: The task can only be assigned to an email. The anonymous form xml can be pre-filled, .

History Page

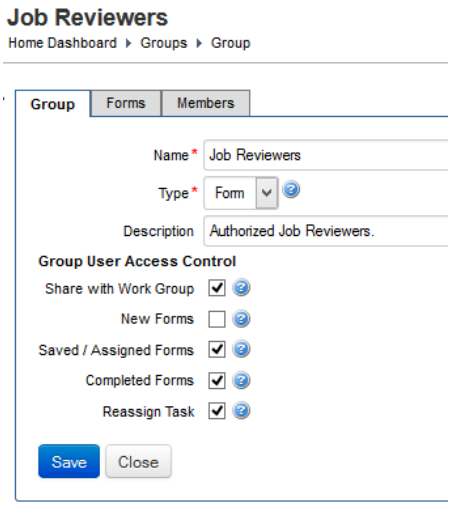
The history page shows a list of all submissions that a user had. A submission that starts a job can contain a processing status that is updated as the job progresses through its steps.

The user can filter the result using a Keyword search and to show all Group Submissions.



Form Groups - User Access Control

The permission to edit tasks, claim assign and return to group are based upon the Form Group - User Access Controls.



Share with Work Group: Minimum access control that allows a user to work with the forms in the form group. Also requires additional access below.

New Forms: Access control allowing users to open new forms. With this set the form becomes visible in the Forms Page.

Saved / Assigned Forms: Access control allowing users to open saved work group forms and group assigned form tasks in the Tasks page. If the Group tasks are claimable, it makes the **Claim**, **Claim and Open** and **View Copy** links available.

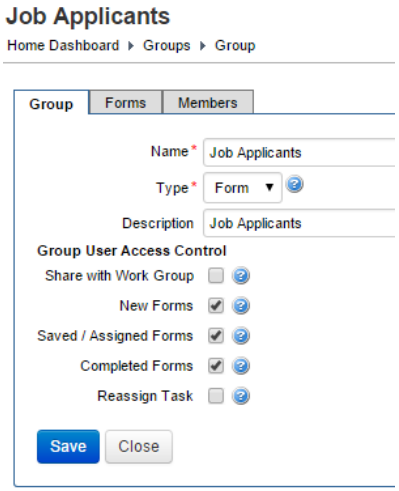
Completed Forms: Access control allowing users to view completed form group submissions in the history page.

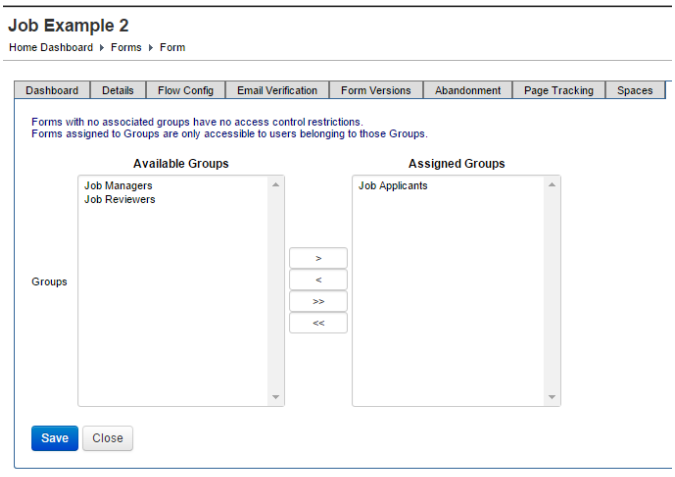
Reassign Task: Access control allowing users to perform Claim, Reassign tasks to other users.

Authenticated User Roles

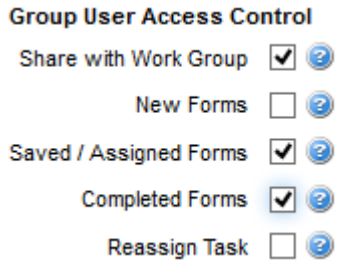
There are a number of authenticated user roles

- **Start User:** Such a user may
  - Submit a form that starts a Collaboration Job
  - They want to check on the status of the job and the may be asked for more information.
  - They may also want to participate in a workflow by updating their form details if requested.

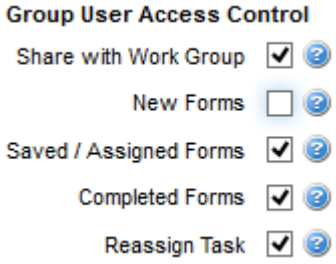




- **Reviewers**: These will be reviewing jobs as part of their job on a daily basis. They might be part of a group that looks after a particular step.



- **Managers**: These manage a group of reviewers for a particular step. They may also review tasks themselves. They can have the ability to assign or reassign tasks to another user.



- **Job Coordinator**: They are a Business Systems Coordinator that look after the job as a whole. Please see Job Coordinator section below.

Groups, Task Claiming Type

There are 2 models that can apply when working with tasks.

1. **Default**: Group tasks is and optimistic model.
   It allows a group task to be opened by 2 group members the first user to submit the form won. The second users submission is ignored.

2. **Task Enable Claiming**: (Transact Version 4.1+)
   Task claiming can be setup for Group tasks. Claiming allows a tasks to be edited by a user while preventing another user editing it.
   This requires specific additions to the Job Definition as described below.

Job Definition - Action Configuration

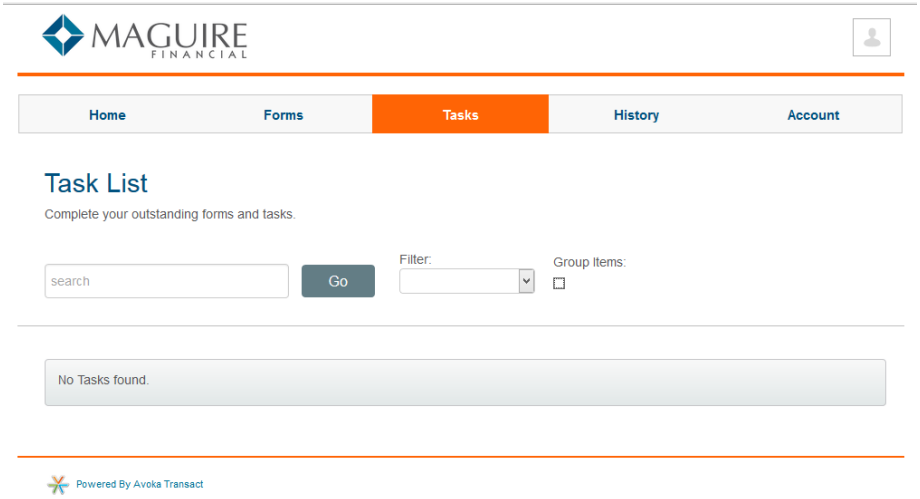
Group Task Claiming needs to be configure when the task is created. This is done the Job Task Assign - Action by the **Task Enable Claiming** action property as seen the extract from Job Example 2 below.

Job Task Assign - Action Definition

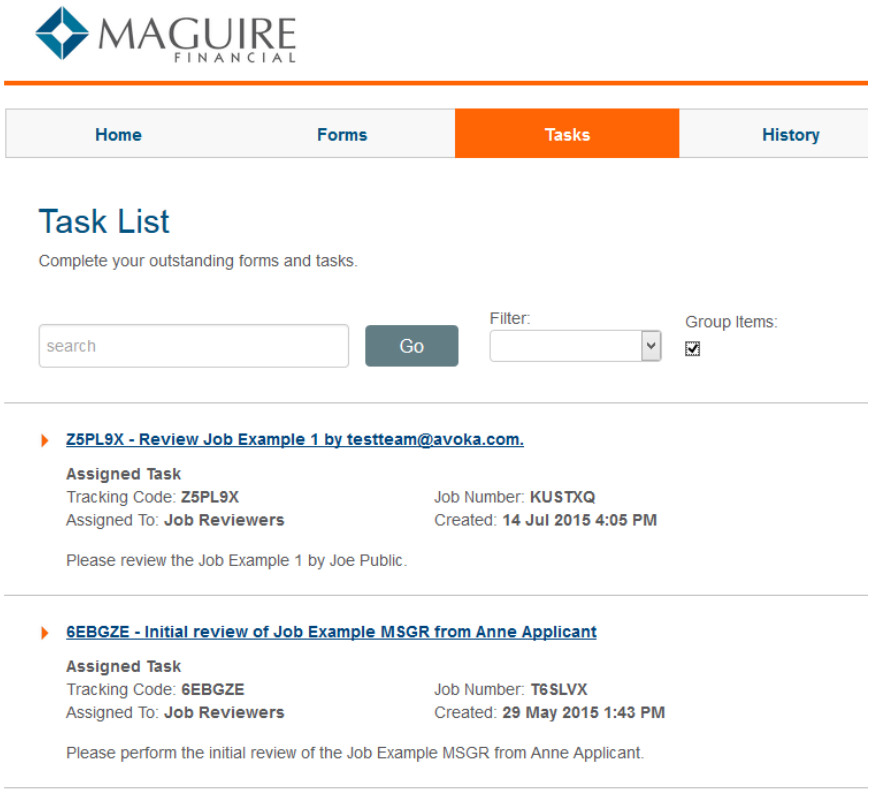
```
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "Your ${submission.formName} application is at the Final
Review Step." },
            { "name": "Task Assign Groups",  "value": "Job Reviewers,Job Managers" },
            { "name": "Task Enable Claiming", "value": "true" }
            { "name": "Task Message",  "value": "The application has been initially approved and is pending your
final review." },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "Additional review of ${submission.formName} from ${formDataMap.
firstName} ${formDataMap.lastName}" },
            { "name": "Task Type",  "value": "Review" }
          ]
        },
```

No Task Claiming Example

Below we have logged into the portal as Roger Reviewer and click on Tasks. Roger does not have any tasks assigned to him so hi list is empty.



Clicking on the Group Items checkbox shows the tasks assigned to the Job Reviewers Groupl

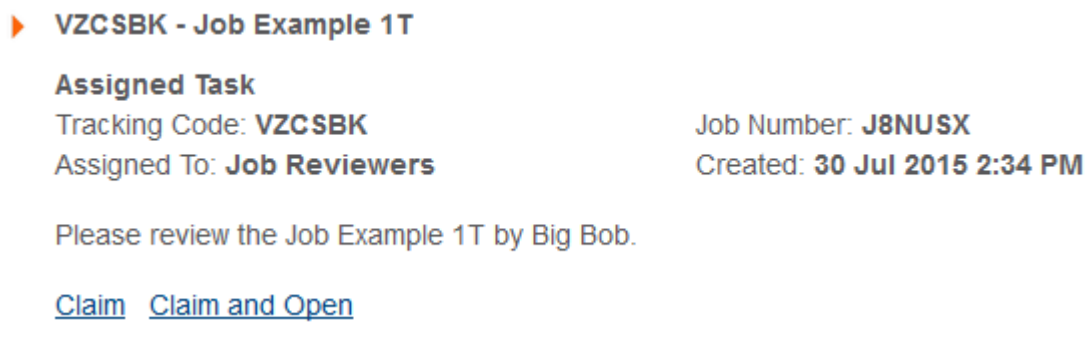


Roger can open the Job Example 1 Task by clicking on the link. This opens the form.

Mike Manager who is a member of Job Managers could open the task at the same time. The person that submitted the tasks **last** submission is ignored.

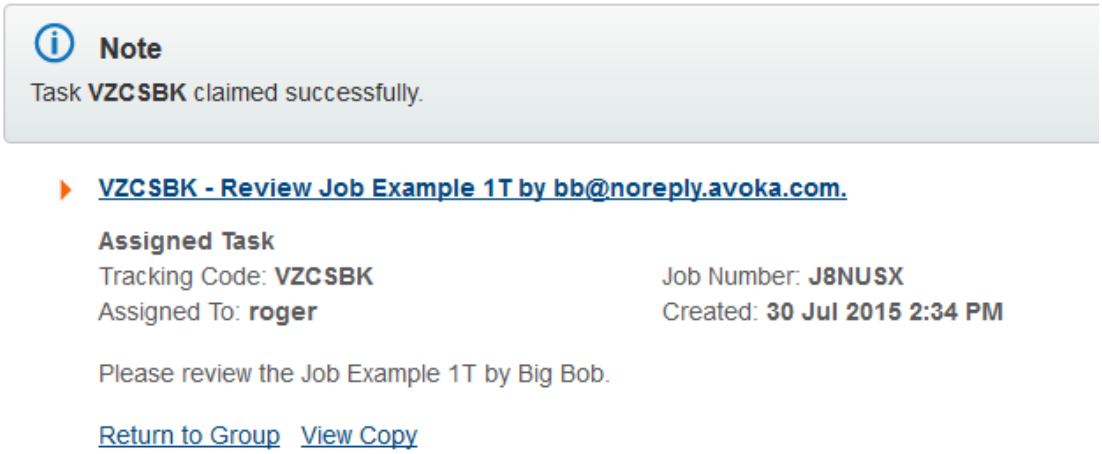
Example - Task Claiming Enabled

When Task Claiming is enable the Job Reviewer looks at his group items and can find the task as per below.

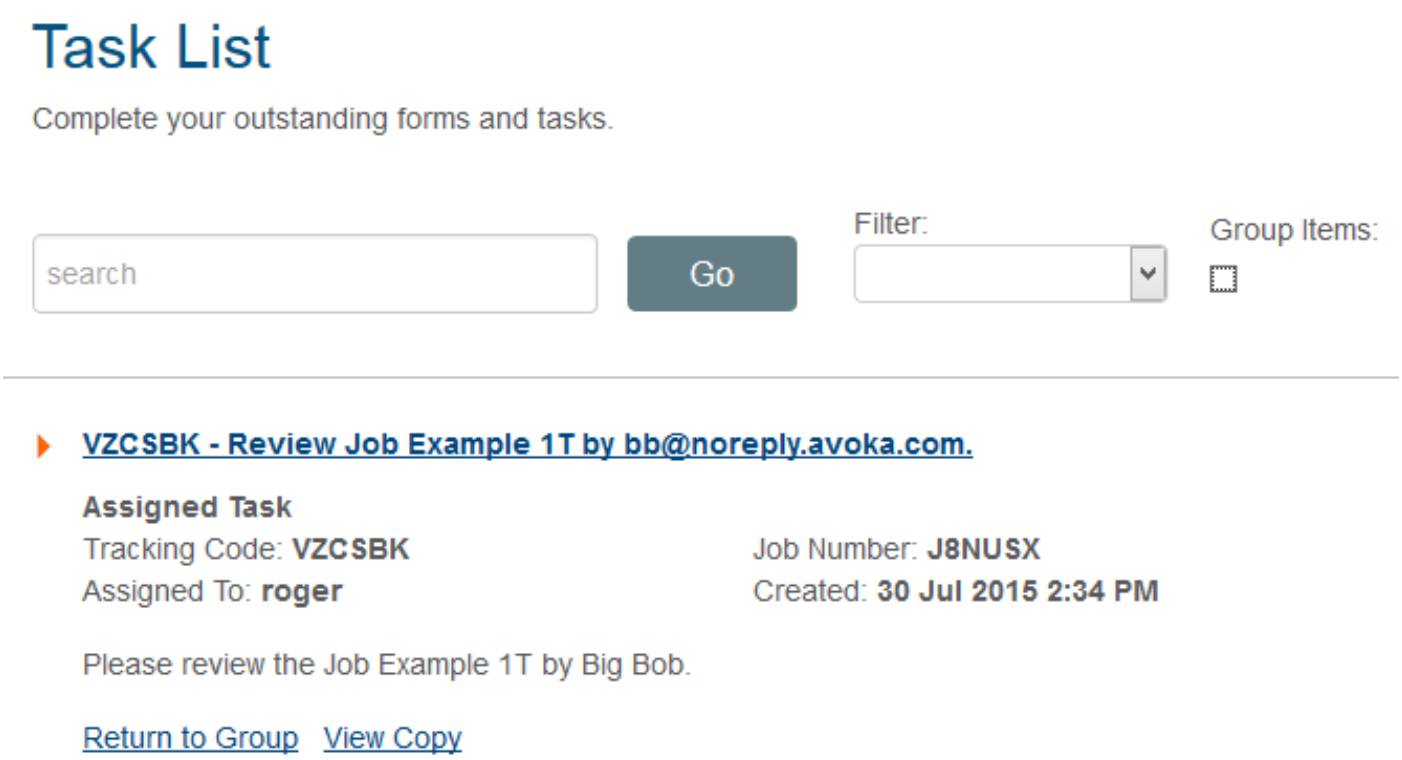


Clicking **Claim and Open** will open the form. Other users will be restricted from editing the form.

Clicking **Claim** assigns it to roger and we see the notification.



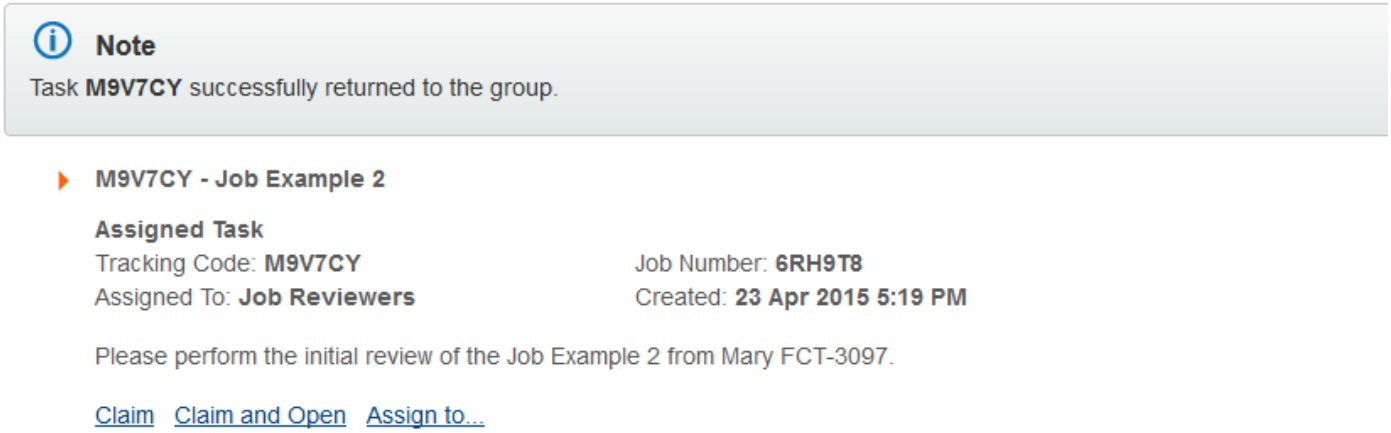
Tasks that have been assigned will appear in the users list without having to tick the Group Items



Actions:

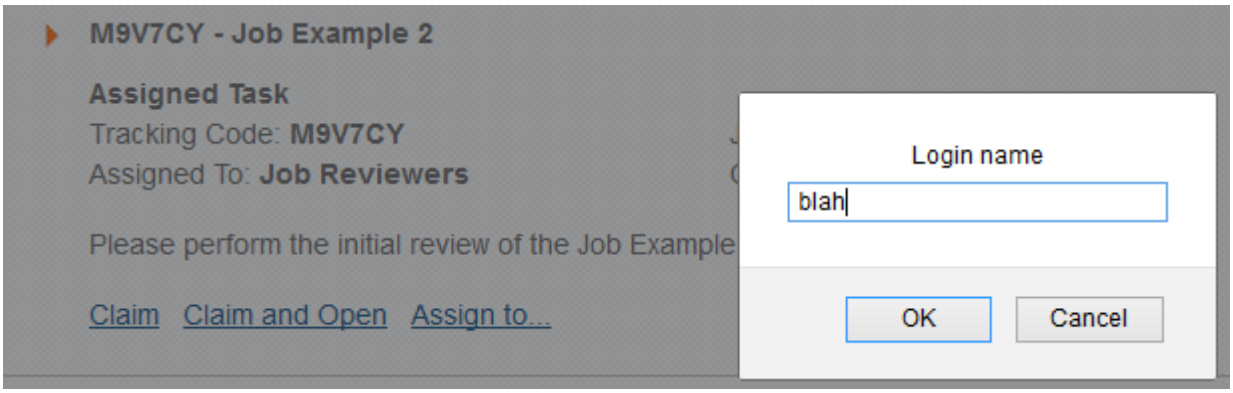
- **Return to Group** removes the claim from the task effectively returning it to Job Reviewers Group
- **View Copy** allows the user to open
- Clicking on the task title link opens the form.

Opening the form as a Job Manager

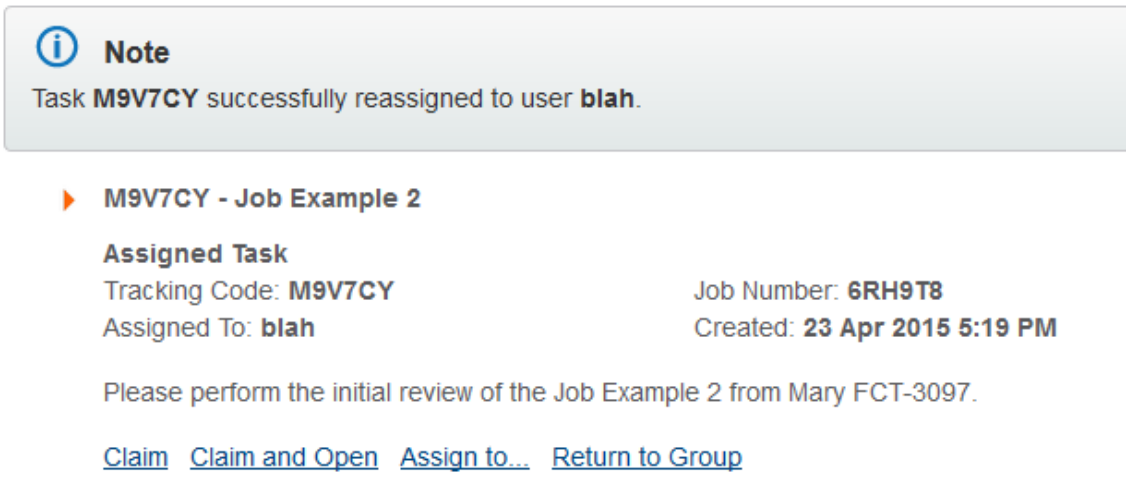


Claim and Open claims the form to (claire) and opens the form.

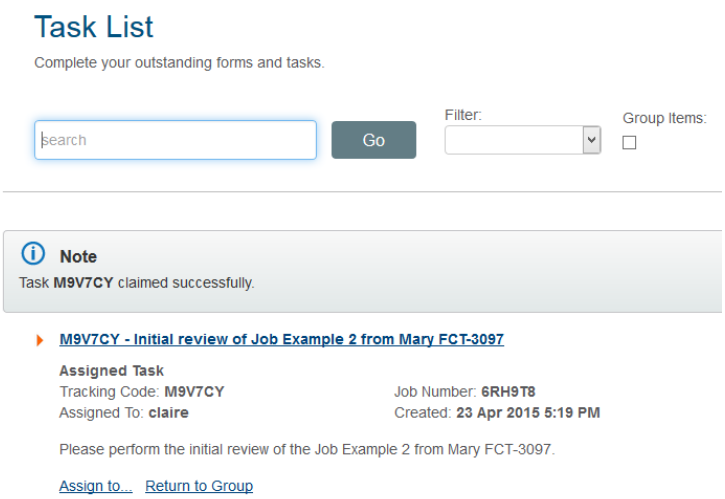
Assign To allows the form to be allocated to someone else in the group (Brian Lah) . We need to enter their login name (blah) as follows.



We can see the task is now assigned to blah



For Claire to open she can claim it back from Brian Lah.



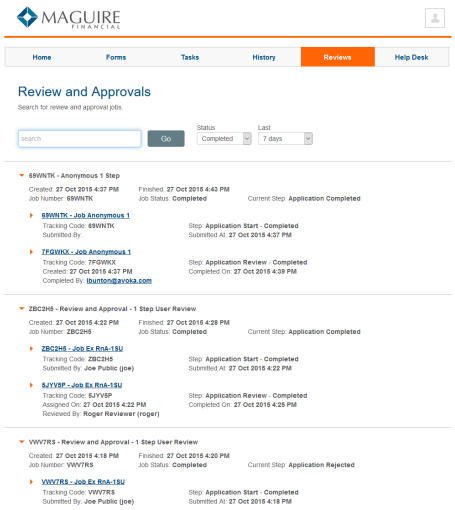
Job Coordinator

The Job Coordinator or Business System Coordinator looks after a job as a whole. They are interested in searching for a job, reviewing the Job status and the tasks in the job, viewing details of a Job.

Reviews

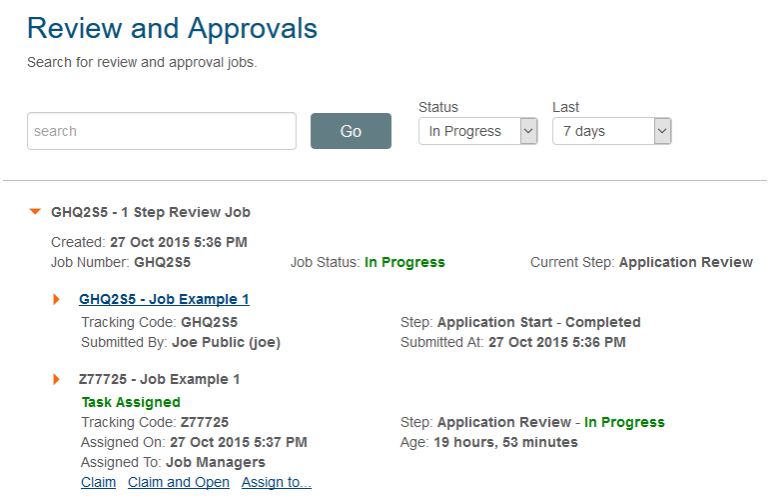
The Reviews section appears as a tab in the Form Space. Coordinators can use this feature to search for a job. They can then see the job details and find out what step is the job up to. It lists information about the current task information, a history of the job's previous submissions and allows the submission receipts to be viewed.

The Reviews section allows the Coordinator to search for a job by entering a filter and getting a search result list (ref screenshot below).



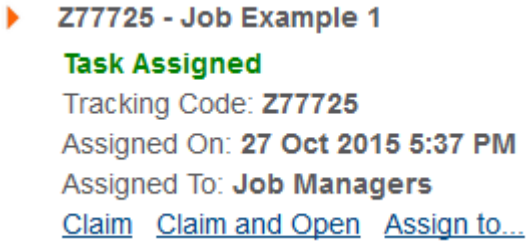
In Progress Jobs

The coordinator can select the jobs that in progress by selecting **In Progress** from the **Status** dropdown



The user can click on the heading of completed forms to download the Tasks receipt.

For Tasks that are Assigned or In Progress the coordinator can claim, claim and open, assign to and return to group (if claimed first)



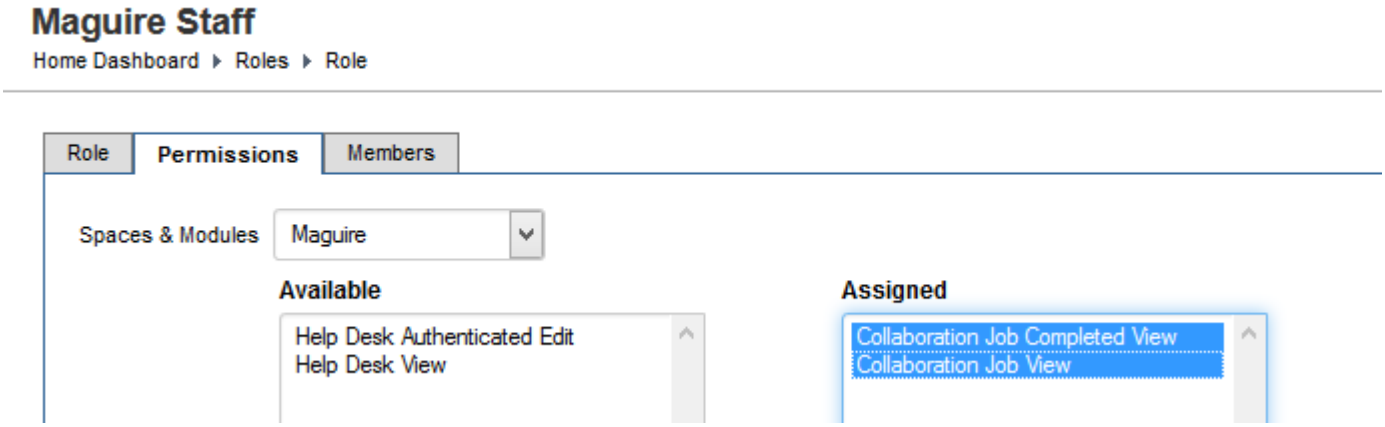
Organizations, Roles and Permission Assignment

Permissions

To see the Job View tab a Job Coordinator must be given the Collaboration Job View permission. Every Transact Web portal has a role created with the name <portal name> Staff which contains this permission as per the screenshot below.

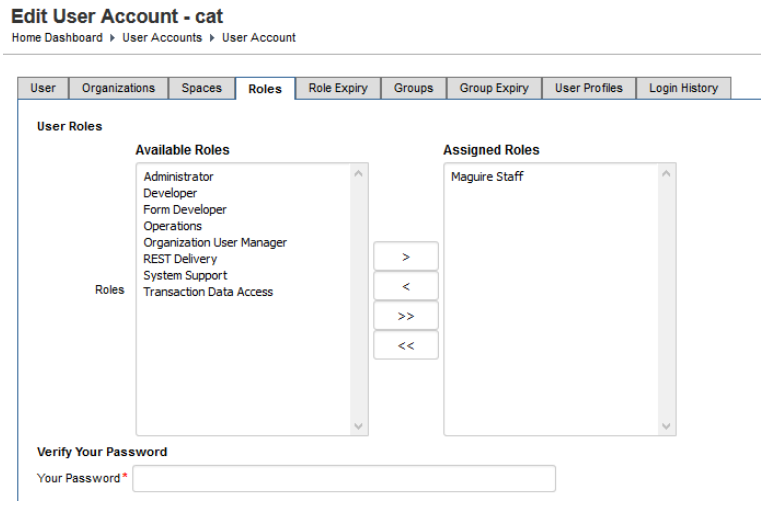
As portals are shared between They also have to belong to the organisation.

System -> **Roles**, select on **Maguire**, select the **Permissions** tab and select **Maguire** from the **Spaces and Modules** drop down.

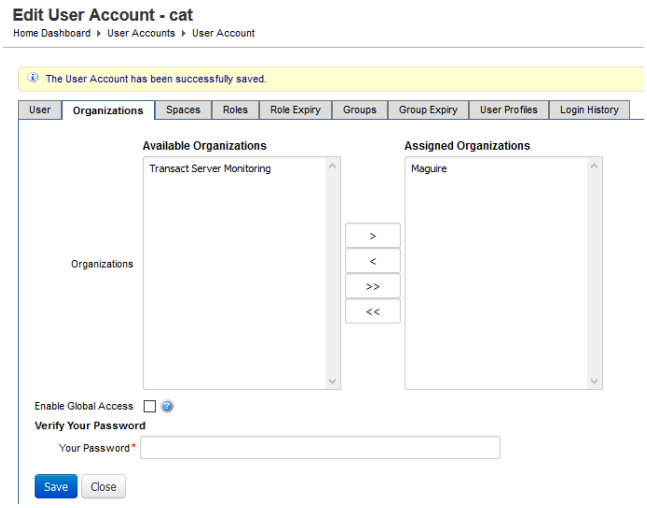


Users Account

The Job Coordinator needs to be assigned to the Role.



The Job Coordinator user must also belong to the organisation that the form belongs to so they can view all the receipts.


Edit User Account - cat
Home Dashboard ▸ User Accounts ▸ User Account
The User Account has been successfully saved.
User
Organizations
Spaces
Roles
Role Expiry
Groups
Group Expiry
User Profiles
Login History
Available Organizations
Assigned Organizations
Organizations
Transact Server Monitoring
Maguire
>
<
>>
<<
Enable Global Access
Verify Your Password
Your Password *
Save
Close

Admin Operations (CollabJobs v4.3)

Developers and Administrator

Developers

Developers will need access to the Organisations, Forms, Job Basics (CollabJobs v4.3)#Job Services, Service Definition, Security (Users, Groups and Roles).

They will also need Collaboration Jobs Management screen (section below) which is useful for testing their job. It enables them to search for the test job and then drill down to review the job, step and action state. The Event tab which is useful for viewing actions that have debug output and an errors tab to review Errors with and the context associated with them.

System Administrators

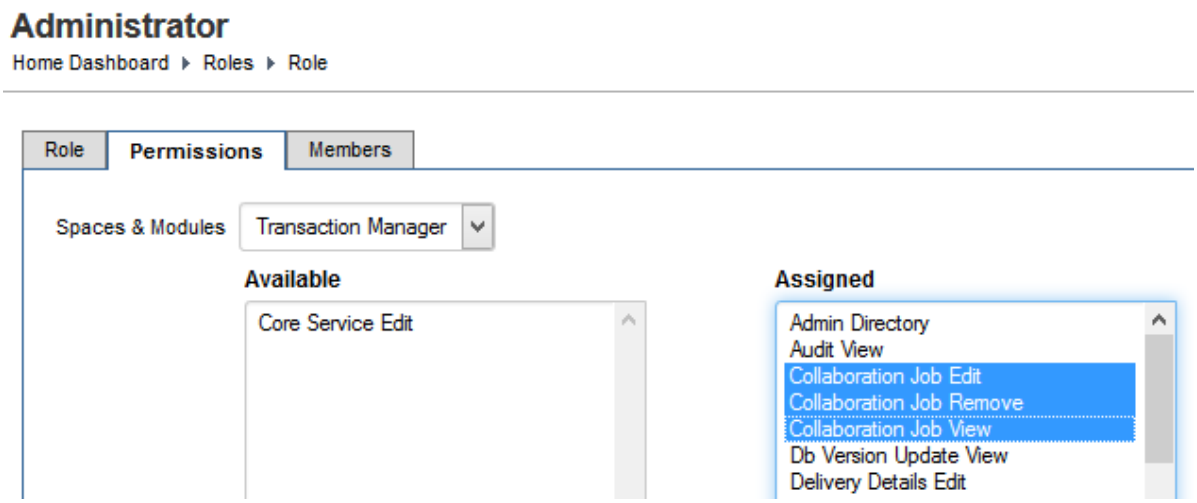
System Administrators will need access to the developers section above above roles to move forms and jobs between environments, to maintain and assign users to the correct groups and roles.

The Collaboration Jobs Management screen can be used in production they can:

- Search on a job and see "what step is my job up?" then drill down into it for more detail.
- Look at a action task and see who the task is assigned to.
- For completed submissions they view the receipt, Form XML and other details.
- Review error status.

Roles and Permissions

The permissions associated with the Collaboration Job Management are



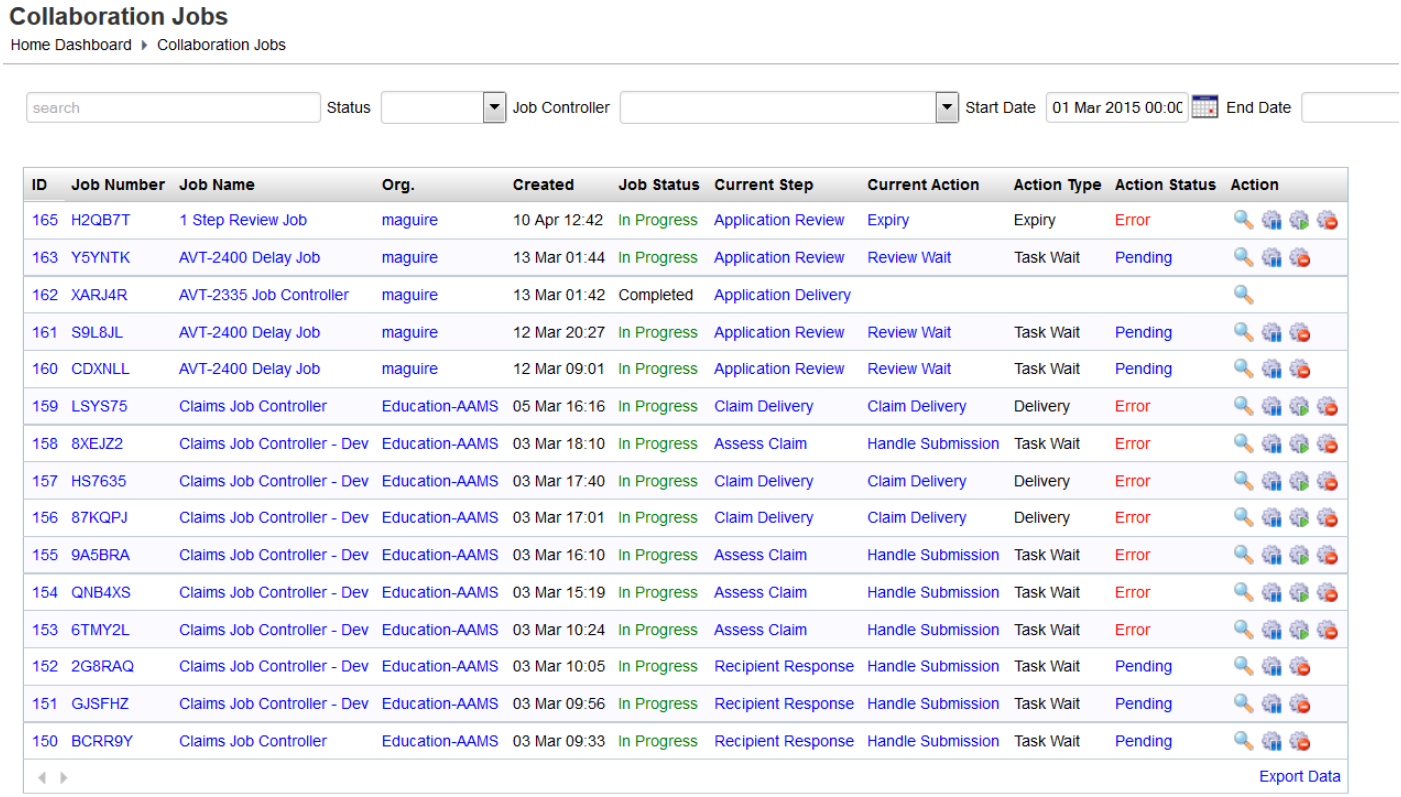
Collaboration Job View allows one to review the entries.

Collaboration Job Edit allows one to make changes to the status save and cancel a job

Collaboration Job Remove allows a developer to permanently remove the job from the Transact Manager database.

Collaboration Job Management

This located under the **Operations** menu -> **Collaboration Jobs**



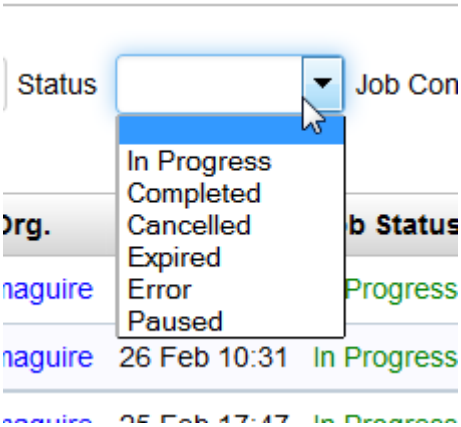
Job Search

The first page is a search page with a filter above and results table below

Search Filter

It allows an administrator or developer to locate job instance(s) based upon:

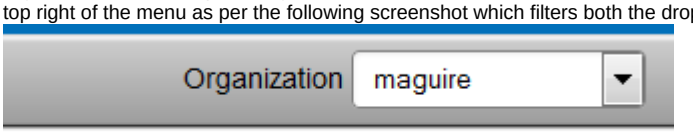
1. A **full word** search over a number of Job fields such as Job Number and Job Name.
2. The current job status can be selected from a drop down.



3. The Job Controller can be also be selected from a drop down.
4. A Date Range: Start and End Date

TIP:

- This Job Controller Drop Down only includes the Job Controllers for organisations that the administrator has access to. The search results only show the job instances linked to those organisations.
- A global administrator (or an organisational administrator that has been allocated to more than one organisation) can select an organisation on the top right of the menu as per the following screenshot which filters both the drop down and the search results.



Search Result Table

The Search Results are ordered in reverse chronological order based on the job creation date. Each row in the results shows a job instance.

The row entry values can be hyperlink

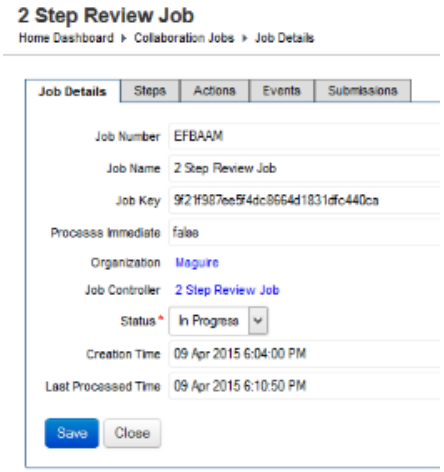
- (Job) **ID**, **Job Number**, **Job Name** selecting any of these will take you to the Job Detail page for the job instance (as described below)
- **Org.** (Organization) clicking on this link is a shortcut to the Organization Edit page.
- **Created Date:** The date the job was created
- **Job Status** valid status are listed in the Job Basics (CollabJobs v4.3)#Job Statuses page.
- **Current Step** selecting the link takes you to the Step Details page below for the current step instance.
- **Current Action** selecting the link takes you to the Action Details page below for the current step instance.
- **Action Status** is for the current Action. The valid action statuses are listed in the Job Basics (CollabJobs v4.3)#Action Execution page.

Action Column shows icons that when present perform some task associated with the rows job instance.

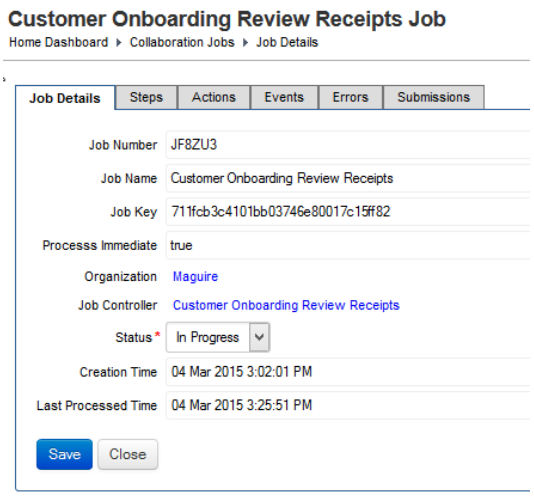
Icon	Details
	This icon takes you to the Job Detail page for the selected job instance
	This icon pauses the job instance.
	This executes the job instance immediately. This is useful for a jobs developer trouble shooting an action service. They can make a programming or configuration change then press this to execute. For an administrator a action may have failed due to a system being down. Once the system has been fixed they can press the button to test it.
	This cancels the job. The Job details are still available. Note it is only available when a job has not completed.
	This appears after a job is cancelled, selecting it will permanently remove the job from the Transact Manager database. This is useful for cleaning out test jobs created during developing.

Job Detail

The screen below contain the details for the job instance. The Steps, Actions, Events and Submissions tabs are discussed in the sections below.



In addition if custom action services are being developed that has a programming or runtime error there will be an Errors Tab.



Job Number, **Job Name** and **Job Key** are specific to this job instance.

Process Immediate setting see Job Basics (CollabJobs v4.3)#Job Execution.

Organization: Name of the organisation the job belongs to, its is also a link to the organisation edit page.

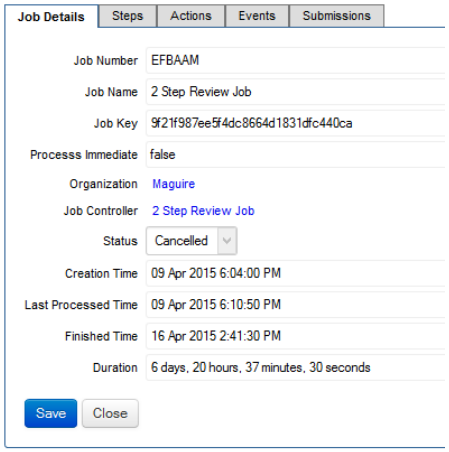
Job Controller: The name of the Job Controller which has a link to the Job Controller's Editor opening in the Job Definition page.

Status: The drop down displays the current job status. The status can be modified as follows.

- If the job is **In Progress** it can be changed to **Paused** or **Cancelled** and saved.
- A Job that is **Paused** can be changed to **In Progress** or **Cancelled**.
- Changing to **Cancelled** and saving cannot be undone.

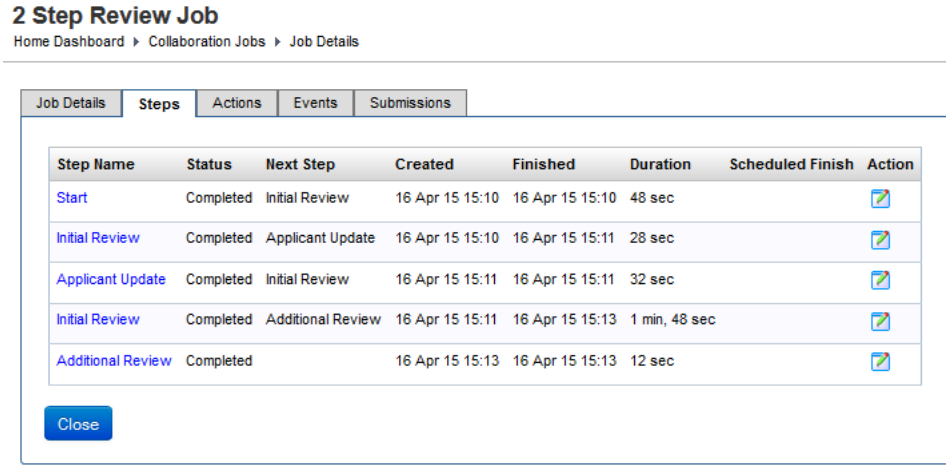
For **In Progress** and **Paused** Jobs it shows the Job Creation Time and Last Processed Time.

For Completed jobs it list the Finished Time and elapsed duration.



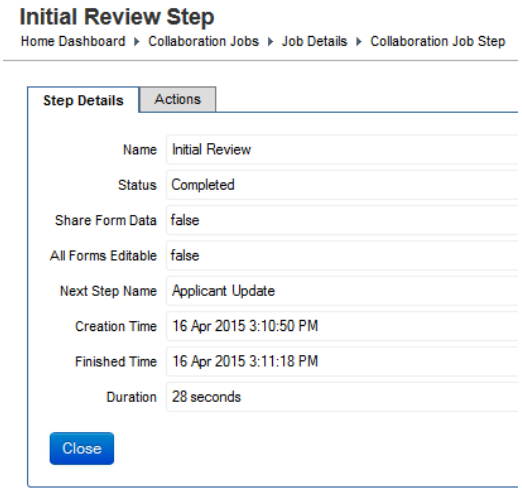
Steps

The steps tab lists the steps that have run or are in progress in a table as per the screenshot below

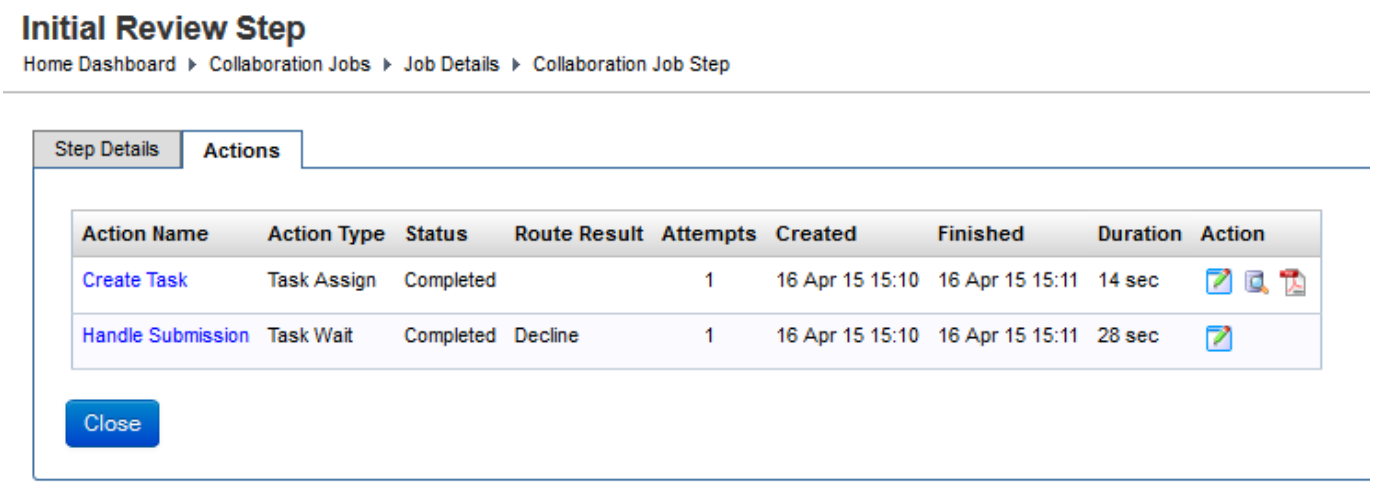


Clicking on the Action icon will take you to the steps details

Step Details



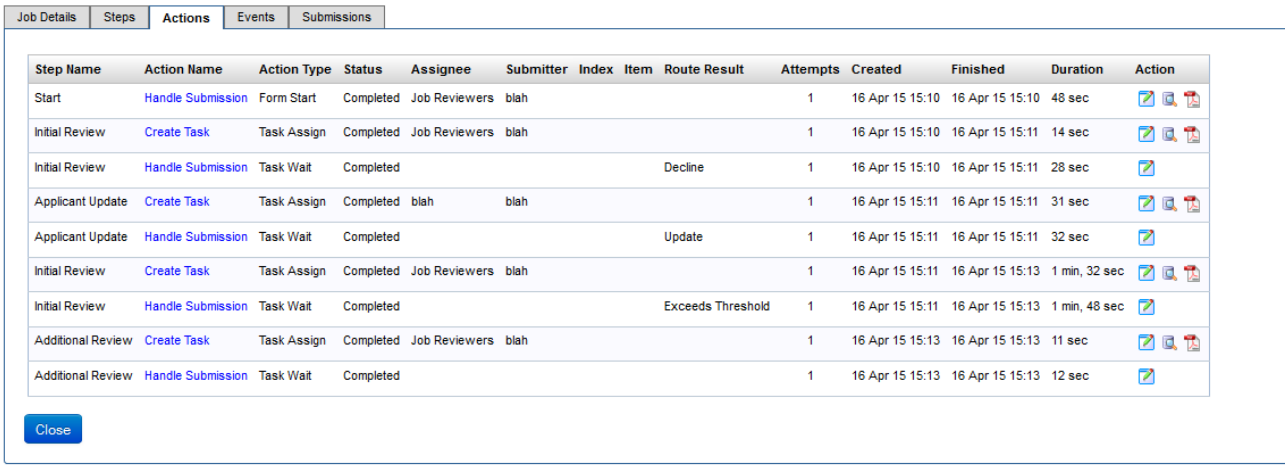
Clicking on the Actions Tab shows only the Actions that are run for that step. this screen is discussed in the Actions section.



Actions

If the Actions tab from the edit screen list all the steps and actions in order that they were created. each row details information about an action instance.

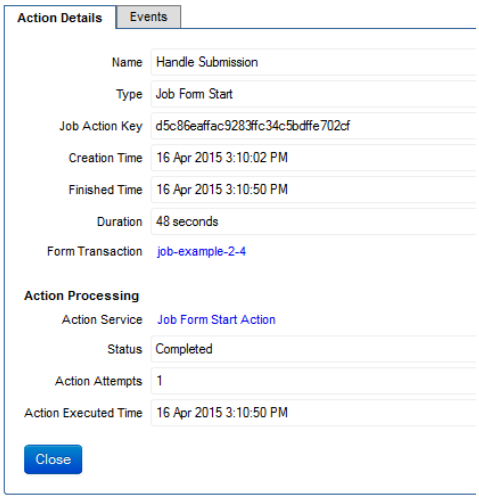
The Action Name is a link that can be clicked on to get to the Action Detail page. Clicking on the  icon will take you to the Action Detail page.



Action Column shows icons that when present perform some task associated with the rows action instance.

Icon	Details
	This icon takes you to the Action Details page for the selected job instance
	If the Action Type is a Task Assign or Form Start there will be a submission linked. This takes you to the Form Transaction view page where you can inspect the FormXml
	If the Action Type is a Task Assign or Form Start there will be a submission linked. It shows if the task after the task has been submitted and the receipt generated. Clicking on this will allow you to view the receipt.

Action Details

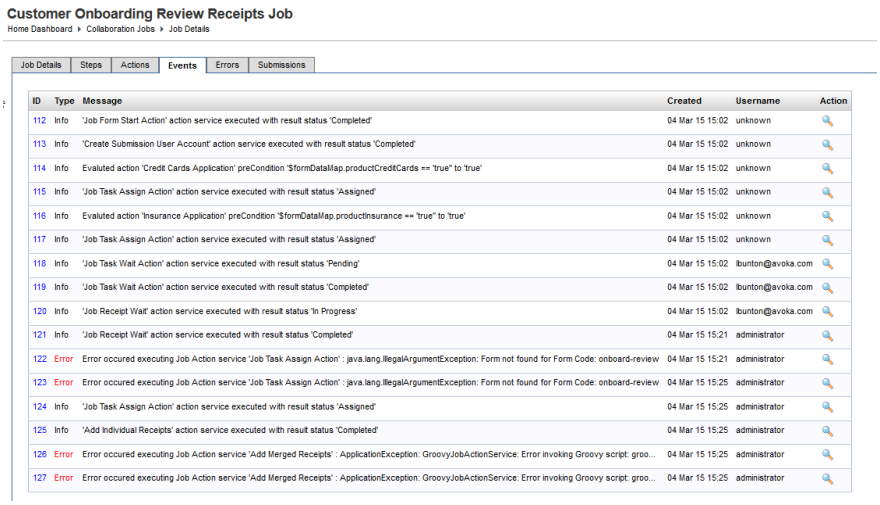


The Action Details page gives even more detail about the action instance.

The event tab only shows events that are associated with the action instance.

Event Tab

The Event Tab list of all the action events associated with a job.



Tip: The event Tab is useful for a developer debugging a Groovy Action Service. The following is used extensively in the advanced examples:

- A **logEvent** closure is created to write to the event log. (line 37)
- This is used to write debug output which can be seen in the Event Tab

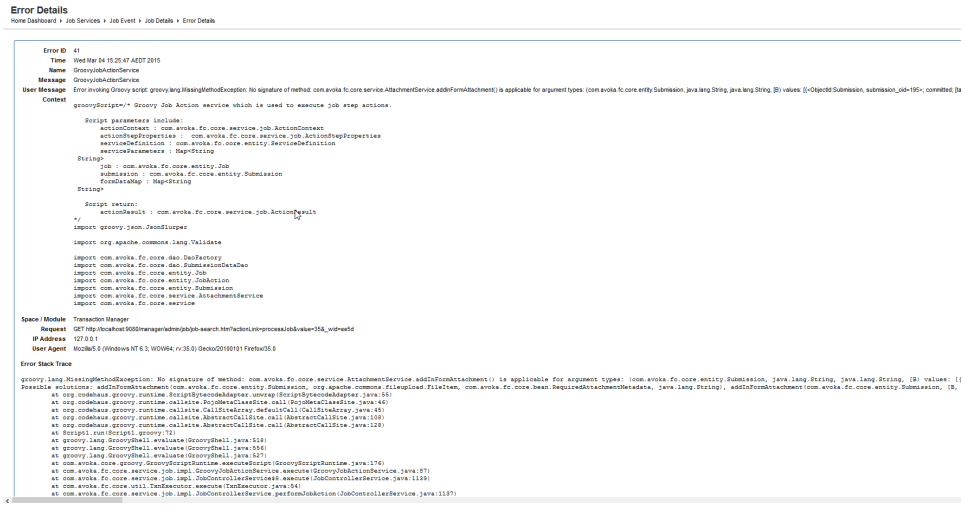
Value

```
final ACTION_PROPERTY_WAIT_TIME_MINUTES = "Wait Time Minutes"

JobEventLogService jobEventLogService = new JobEventLogService()

boolean isLogEvents = serviceParameters["Log Events"]

def actionResult = new ActionResult(Status.IN_PROGRESS)
def jobAction = actionContext.getJobAction()

// logEvent closure
def logEvent = { msg ->
    if ( isLogEvents ) {
        // Logs to Job Action Event Tab
        jobEventLogService.logInfoEventWithAction((String) msg, jobAction)
    }
}

String waitTimeString = actionStepProperties.getProperty(ACTION_PROPERTY_WAIT_TIME_MINUTES)
Validate.notEmpty(waitTimeString, "The Action Property - " + ACTION_PROPERTY_WAIT_TIME_MINUTES + " is empty")

def waitTimeMinutes = waitTimeString.isInteger() ? waitTimeString.toInteger() : 0;

Validate.isTrue(waitTimeMinutes > 0, "The Action Property - " + ACTION_PROPERTY_WAIT_TIME_MINUTES + " is less than 1 or i

def dateStart = jobAction.getTimeCreated()
def dateLater = DateUtils.addMinutes(dateStart, waitTimeMinutes)

logEvent( "dateStart: "+ dateStart + ", waitTimeMinutes: " + waitTimeMinutes + " , dateLater: " + dateLater )

```

Errors Tab

The errors are the Error Log entries associated with the Job Instance.

Drilling down give more information including Context and Stack Trace if available.



Form Design (CollabJobs v4.3)

Collaboration Jobs requires the configuration in Transaction Manager and Forms.

This page gives background on designing forms so they can use Collaboration Jobs, how to check in Transaction Manager if the forms have been configured correctly and how the job definition references

- Data Extracts
- Standard Job Context Information

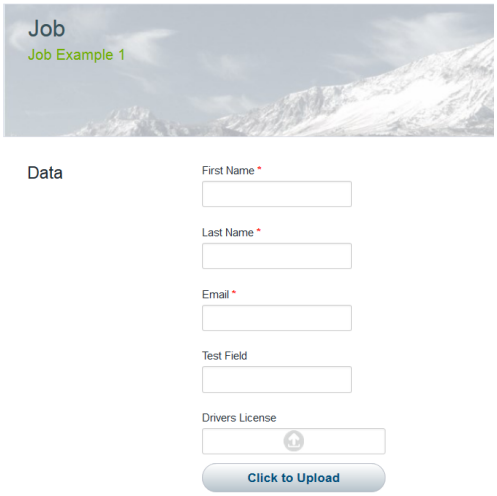
More information can be found in the child pages:

Submission Data Extracts

Collaboration Jobs uses Data Extracts to reference form data from a particular form. They can be setup when designing the form (preference) or can be mapped manually in Transaction Manager.

> **Note**
>
> Data extracts that are setup in the form design will **overwrite** data extracts with the same name in Transaction Manager. This happens every time the form is imported into Transaction Manager
> **Best Practice** is to setup the data extracts in the Form.

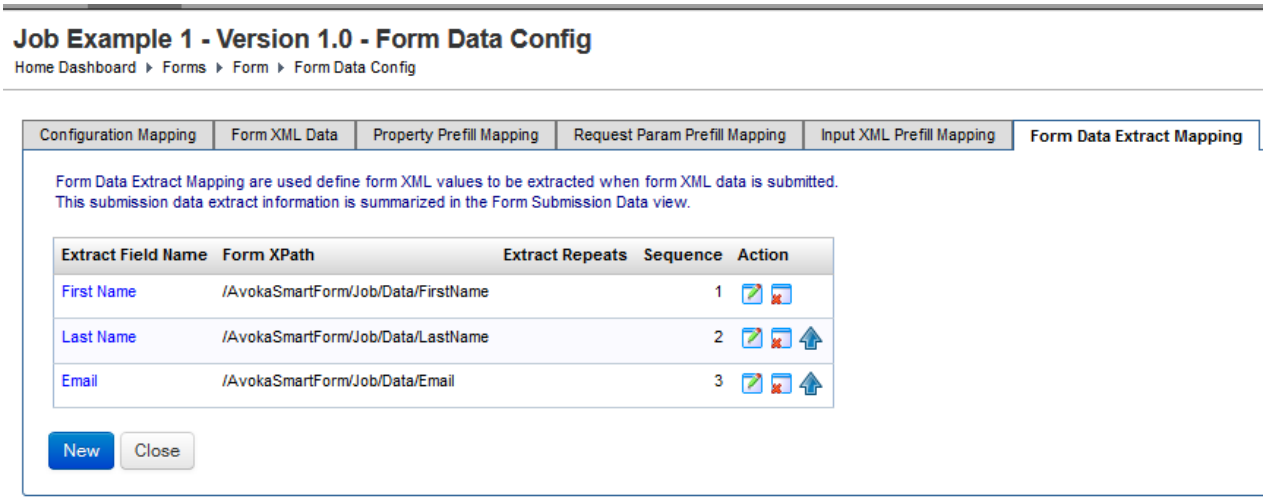
In this section we will use the Data Extracts setup in the Job Example 1 form below.



Transaction Manager

To see where the Data Extracts have been automatically populated in Transaction Manager

Form Dashboard -> **Data Config** -> **Form Data Extract Mapping**.

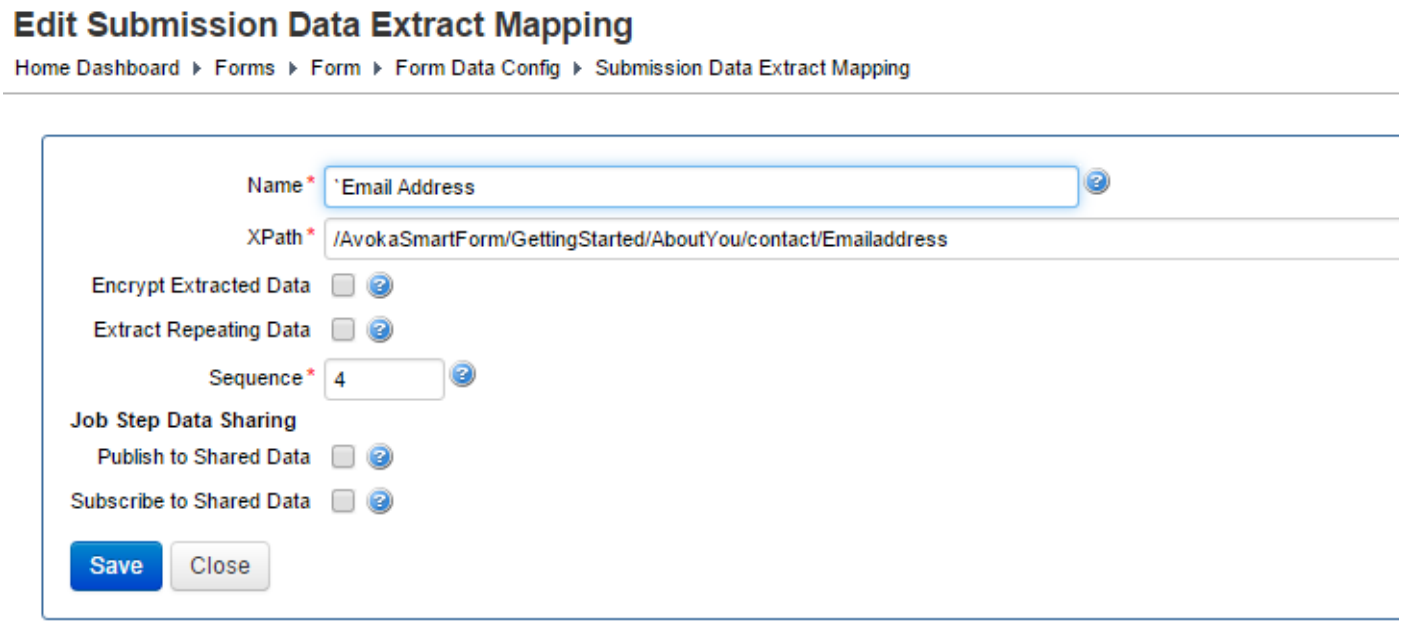


The following table shows the data extract field name, how the data extract is referenced in the Action Property Value and gives an example of the action property from the job definition (JSON). Note the use on camel case in when referring to the data extract **First Name** is reference as **$formDataMap.firstName.**

Extract Field Name	Extract Reference	Example
First Name	$formDataMap.firstName	{ "name": "Task Subject", "value": "Job Number: $job.referenceNumber $submission.form.formName From $formDataMap.firstName $formDataMap.lastName" }
Last Name	$formDataMap.lastName	{ "name": "Task Subject", "value": "Job Number: $job.referenceNumber $submission.form.formName From $formDataMap.firstName $formDataMap.lastName" }
Email	$formDataMap.email	{ "name": "Task Subject", "value": "Job Number: $job.referenceNumber From $formDataMap.email" }

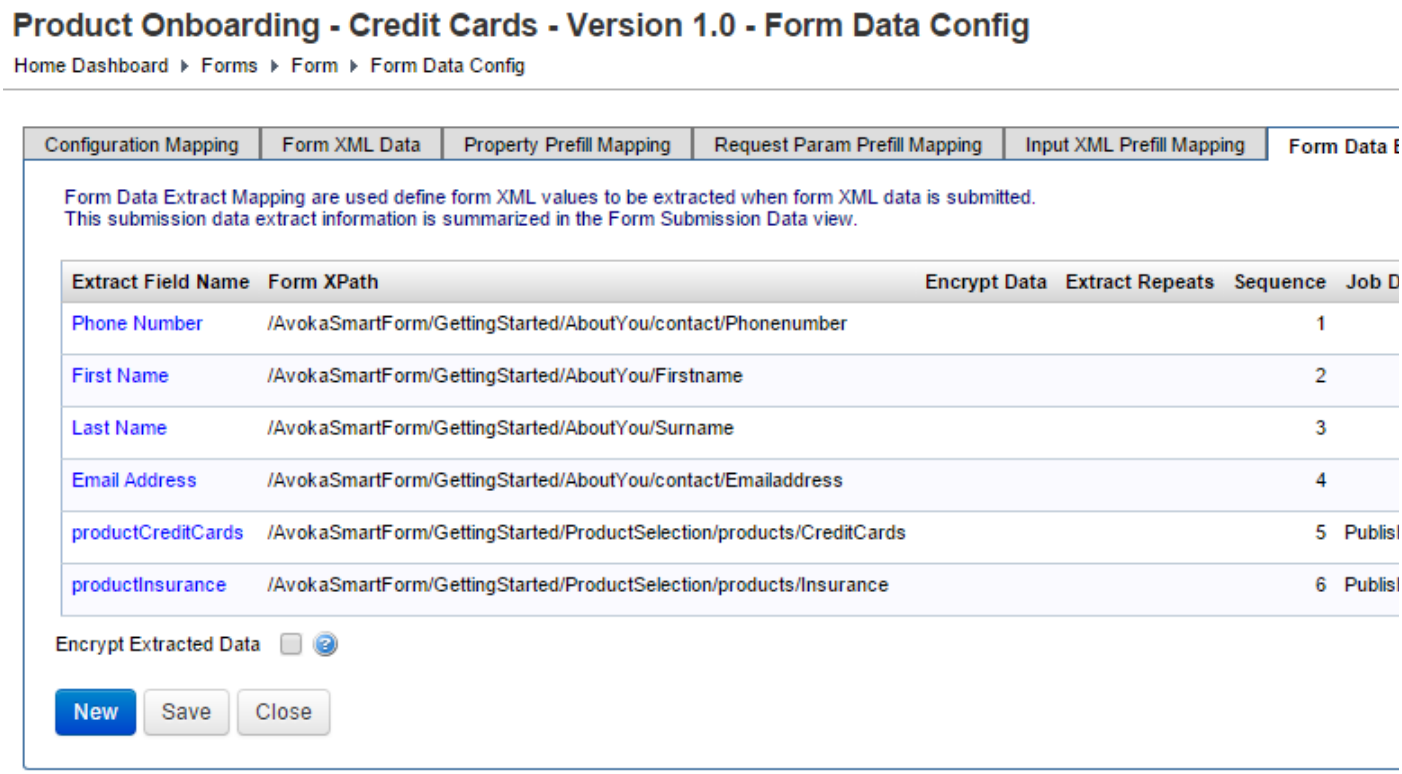
Data Extract Features

The following screenshot shows the data extract mapping and its fields for a 4.3.0 system. The Table below summarises the fields



Field Name	Version	Description
Name		The data extract name.
XPath		The XPath location in the formXml that is used to get the value for. Note this is used in the Publish / Subscribe to update the formXml

Encrypt Extract Data	Since 4.3.0	Prior to 4.3.0 the data extract data was stored in the database tables as clear text which may be an issue if data privacy is kept. Note: • Checking this option will mean that the value is encrypted and won't be able to be read from the database. The value will be unencrypted when accessing via the TM admin console, rest api, as well as from the entities in groovy scripts like delivery services. • Turning this on may mean that you will no longer be able to search on these fields. • If the Encrypt Extract Data Checkbox is checked (Screenshot below) it effectively will set all the Data Extracts to be encrypted. • Selecting it will only encrypt new data, old data extracts will remain unencrypted
Extract Repeating Data	Since 4.1	Please see Repeating submission data extracts V4.1
Sequence		Orders the extract data for reporting. See TM Admin Console Operation -> Form Submission Data
Publish to Shared Data	4.3	This is used by Collaboration Jobs in conjunction with the Subscribe to Share data. It is configured in the Job Definition JSON in the Step by setting the **shareExtractData** attribute to **true**. Publish Subscribe are an alternative mechanism to share data between tasks in a Form Bundle - Job Step. A form bundle Job Step is made up of 2 or more separate tasks (Submissions). When one task is submitted the processing loops over all the data extracts marked as publish and put the values in a Map. It then loops over 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.
Subscribe to Shared Data	4.3	Used by Collaboration Jobs in conjunction with the Publish to Share Data. Subscribe consumes the publish event using the published value as describe in Publish to Shared Data above. Note: • To use subscribe the XPath location needs to be set to a single node. You would not be able to use the double slash //Emailaddress to set the value using XPath. • Data extracts specified when designing you form do not have the full XPath location.



Job Context Information

Transaction Manager passes Job Context information to the form via the **SFMData.SystemProfile.Job** xml node when the form is rendered. Below is an example form data from TM 4.2

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

Standard Nodes

The following table explains the use of each of the **SFMData.SystemProfile.Job** nodes.

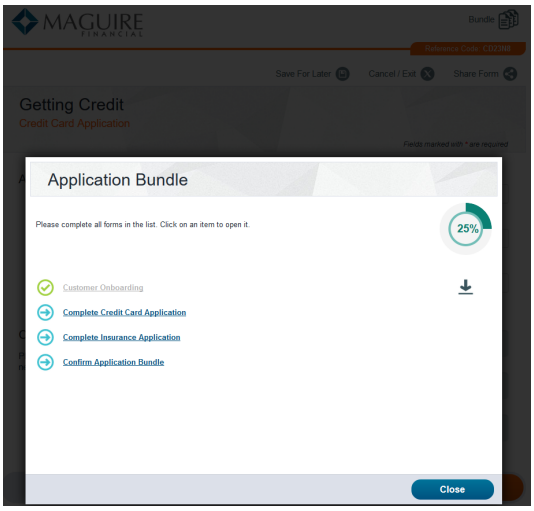
Node Name	Description	Form Use	Since	
AvailableRoutes	Collaboration jobs will pass through the Available Routes for the tasks created at a particular Step. The Job Definition below is taken from the 2 Step Review Job. When the job gets to the Initial Review step it will populate the form data as follows: <AvailableRoutes>Approve|Decline|Terminate</AvailableRoutes> Note the Exceeds Threshold is route is not included due to the display Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'	Read Only	TM 4.0	
ReferenceNumber	The Job Number field. This is displayed on Maguire forms with the Step Number in the heading. See Composer Documentation	Read Only	TM 4.0	
StepName	The name of the current step that the job is currently at. The form can use this value to hide / show or set editability of fields and form sections.	Read Only	TM 4.0	
RouteName	This is set in the form by the user via drop downs or radio button. If this is not set the Job will follow the default route.	Set by Form	TM 4.0	

The following topics are covered in their respective child pages

- Task Assign Repeat Context (Advanced Topic) (CollabJobs v4.3): discuss the Assignee, AssignRepeatIndex and AssignRepeatItem nodes.
- Form Bundle Context:

Form Bundle Context (Advanced Topic) (CollabJobs v4.3)

Collaboration Jobs provides additional context information to Form Bundles created using the Maguire Template. This information is automatically used by Maguire form to provide a list of all the other forms in the bundle, there completion status and a way to navagate to the the chosen form. To get to the list a user can select as per the screen shot below.....



The form xml shown below is from the Customer Onboarding Job - Form Bundle (CollabJobs v4.3). The data is created in the "//SystemProfile/Job /StepTasks" xml node. It is a list of the current tasks for the form bundle.

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

Task Assign Repeat Context (Advanced Topic) (CollabJobs v4.3)

The task assign repeat functionality allows for the following to be done in a single step

1. Form Bundles with multiple tasks assigned to the same person to be created from a single form.
2. Reviews and Approval workflow with multiple tasks to be assigned to different users for parallel review.

It is important for the job to provide context to the form to differentiate tasks. This is provided in the following tables.

<table>
<tr><th>Node Name</th><th>Description</th><th>Form Use</th><th>Since</th><th></th></tr>
<tr><td><code>Assignee</code></td><td>Used by Parallel tasks which is an advanced topic.

The repeat processing iterates over a pipe delimited list of assignees. This shows the current assignee which can be a Username, Group or Email Address (Anonymous Tasks).

The example Job Definition definition - Task Assign - Job Action.

Collaboration Jobs Task Assign Action

<pre>
{
  "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": "${assignRepeatItem} Complete
Credit Card Application." },
        { "name": "Task Input XML Prefill", "value": "$func.
startSubmissionXml()" },
        { "name": "Task Type", "value": "Form" }
      ]
    },
</pre>
To specify Users: use the action property 'Task Assign User' specify a pipe delimited list of usernames

In the above example Task Assign User = $formDataMap.assignDelimited. This is a data extract it may evaluates to "peter|paul|mary"

The first task is created for peter and the Assignee will be peter, the second task will be created for paul and the assignee will be pault etc.

To specify Groups: use the action property 'Task Assign Groups' specify a pipe delimited list of groups

To specify Anonymous Users: user the action or Email Address (Anonymous Tasks) that an individual item is assigned to.</td><td>Read Only</td><td>TM 4.1.8</td><td></td></tr>
<tr><td><code>AssignRepeatIndex</code></td><td>Used by Parallel tasks which is an advanced topic.

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)</td><td>Read Only</td><td>TM 4.2.0</td><td></td></tr>
<tr><td><code>AssignRepeatItem</code></td><td>Used by Parallel tasks which is an advanced topic.

A repeating section in the form may be broken down by something other than a user or group. An example may be a student fills out what sports they want to play. Cricket, Basketball and Rugby

<table><tr><th>Sport</th><th>Coach</th><th>Season</th></tr><tr><td>Cricket</td><td>John</td><td>Summer</td></tr><tr><td>Basketball</td><td>Al</td><td>All Year</td></tr><tr><td>Rugby</td><td>John</td><td>Winter</td></tr></table>
Say 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</td><td>Read Only</td><td>TM 4.2.0</td><td></td></tr>
</table>

Repeat form Form XML

```
<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"

Transaction Manager Form Configuration (CollabJobs v4.3)

Prior to discussing the form configuration we need to first create a new Job Controller.

The following are configured in Transaction Manager after a form has been imported:

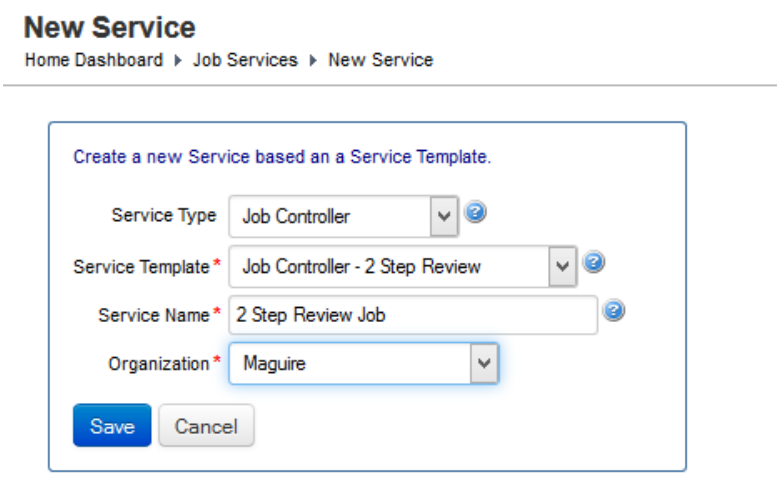
- Form Properties: can be reference in the job definition to provide form specific setting.
- Form Groups: control access to the form
- Form Delivery Method: this has to be setup for the job controller to complete.

Create Job Controller

For the purpose of this documentation we have created one from an existing service template.

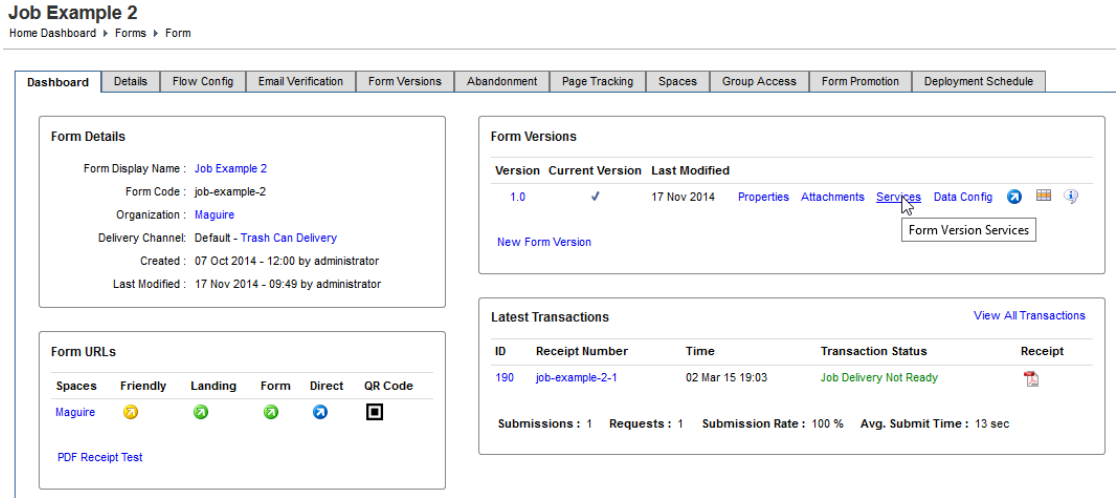
Before we configure the Form, we first have to create a new **Job Controller** service instance from the service template **Job Controller - 2 Step Review**. Please refer to the 2 Step Review example here.

System -> **Job Services** , click the **New** button, fill out as follows



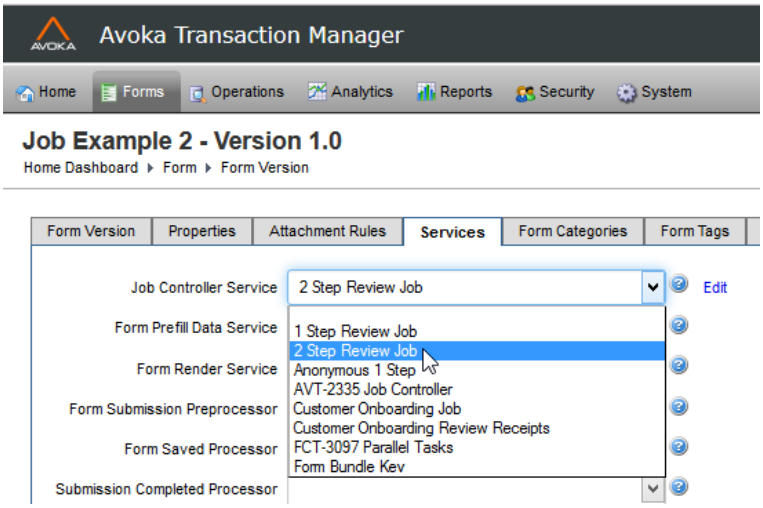
Form Imported into Transaction Manager

In the Form Dashboard



Assign the Job Controller service

Click on the services link as shown in the form Dashboard above.

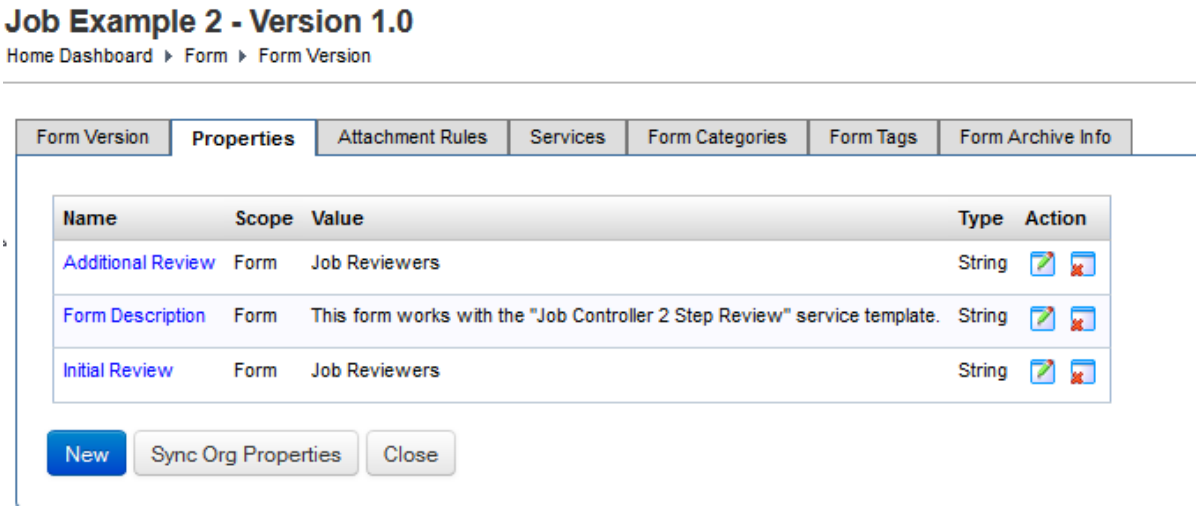


Form Properties

Form Properties are used by the job to link the reviewer for Initial Review and Additional Review Steps to be set to a Form Group. See form groups below.

Next go to the **Form Version** -> **Property tab**. Set the **Additional Review** and Initial **Review property** values.

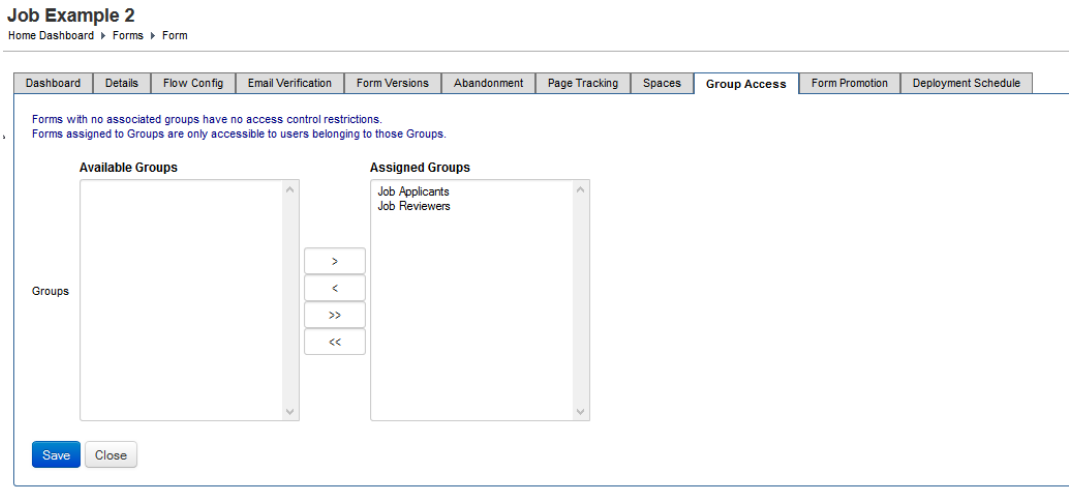
NOTE: By default both will be mapped to the same Job Reviewers form group.



Form Groups

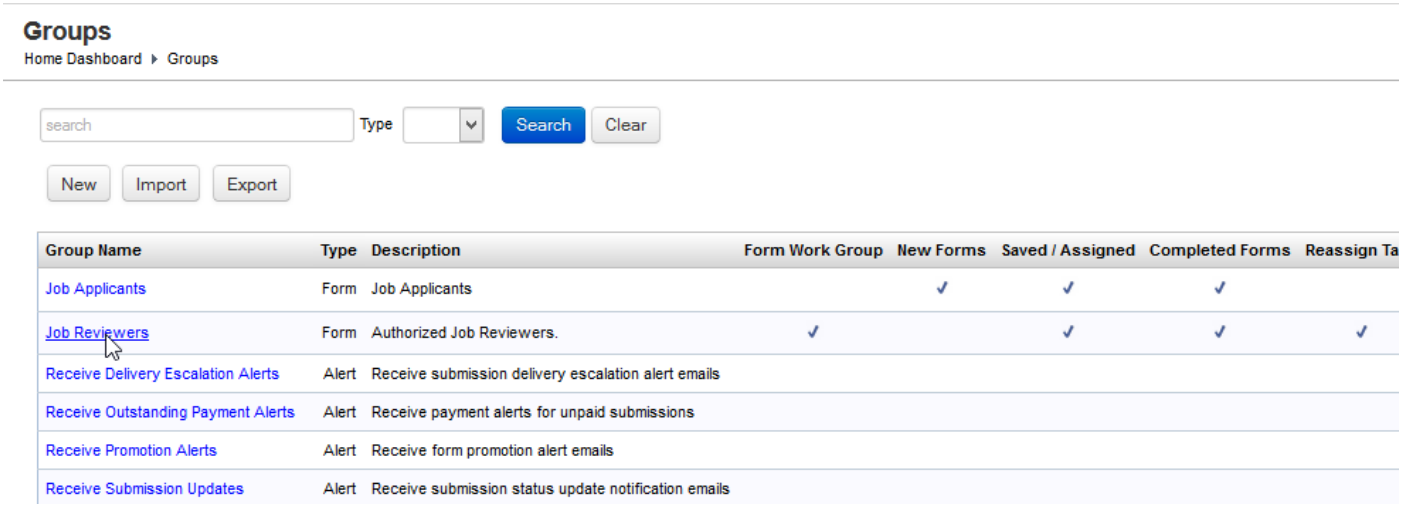
Form groups are use to control access the following access to a form. By default if a form group is not assigned to a form it will be available to any user in the organisation to start.

From the **Form Dashboard** if we select the **Group Access** tab. We can set the groups as per below.



To review the jobs at the Initial and Additional Review steps we add a user that you log in with to the **Job Reviewers** form group.

Click on the **Security** -> **Groups** menu item.

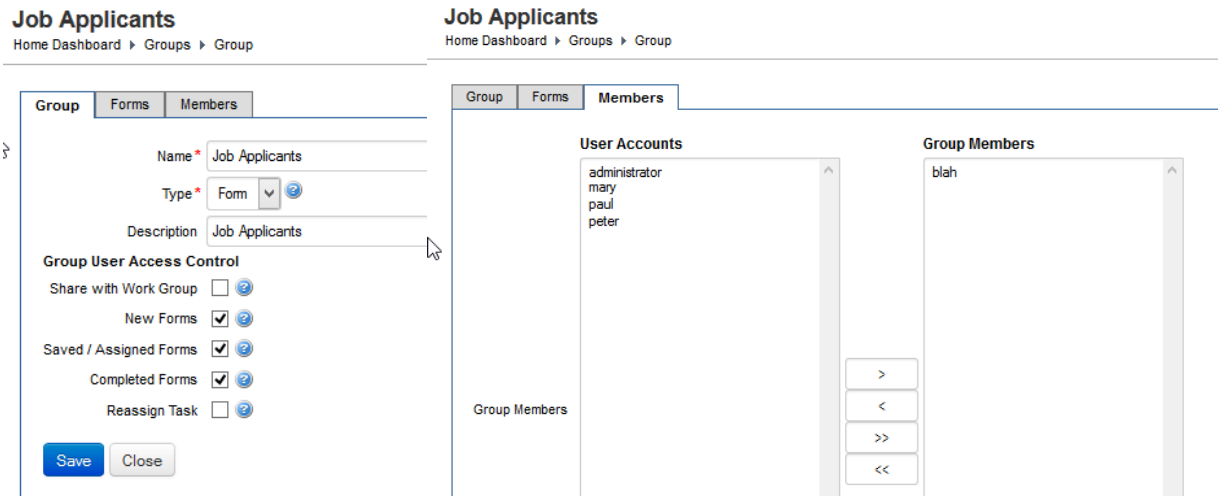


The purpose of the **Job Applicants** form groups is to restrict who can start the form. The **Job Reviewers** cannot initiate the form but can participate in reviewing the submitted forms.

Select the **Job Applicants** group.

The **New Forms** check box allows members of this group to start new forms.

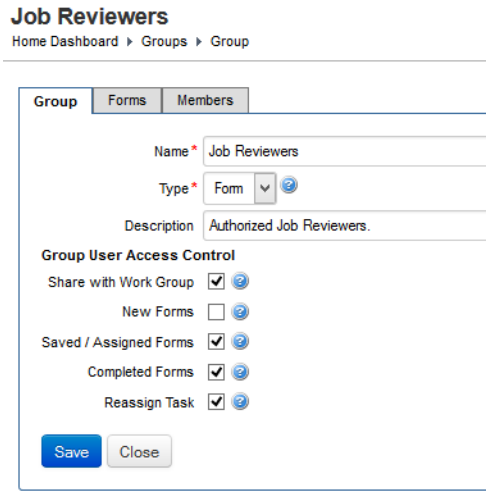
Select the **Members** tab, add your user to this tab.



Select the **Job Reviewers** group.

The **Share with Work Group** check box allows members of this group to participate in tasks but not start new forms.

As per above select the **Members** tab, add your user to this tab.



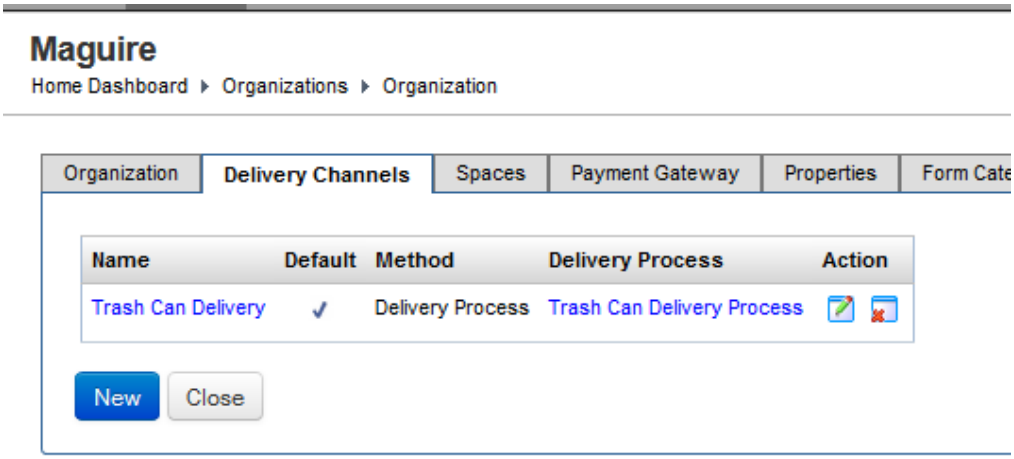
Delivery Method

Before the Collaboration Job will run end to end, either a default Delivery Method should be set for the Maguire organisation or a delivery method should be set for the form.

Note by default **Trash Can Delivery** is set by default so the Collaboration Job will run. However you may want to choose to user email delivery.

To set the default for an Organization.

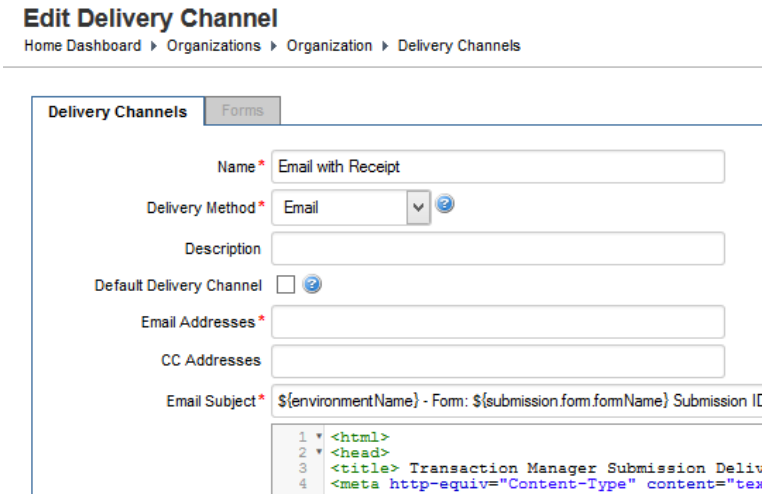
Forms -> **Organizations**, select **Maguire** to edit.



To use Email

Select the **New** Button

Fill out the form as follows be sure to enter an Email Address so you can receive these.



Maestro (CollabJobs v4.3)

This page covers preparing Maestro forms for use with Collaboration Jobs, including:

- How to set up Transact Integration: Data Extracts and Form Data Mappings
- Describe Collaboration Jobs support widgets that come with Maestro
- Using collaboration widgets included in Standard Maestro Templates.
- Setting Rules (such as Visibility and Editability) based on collaboration job step
- Design-tool support for previewing forms with Collaboration Job context.

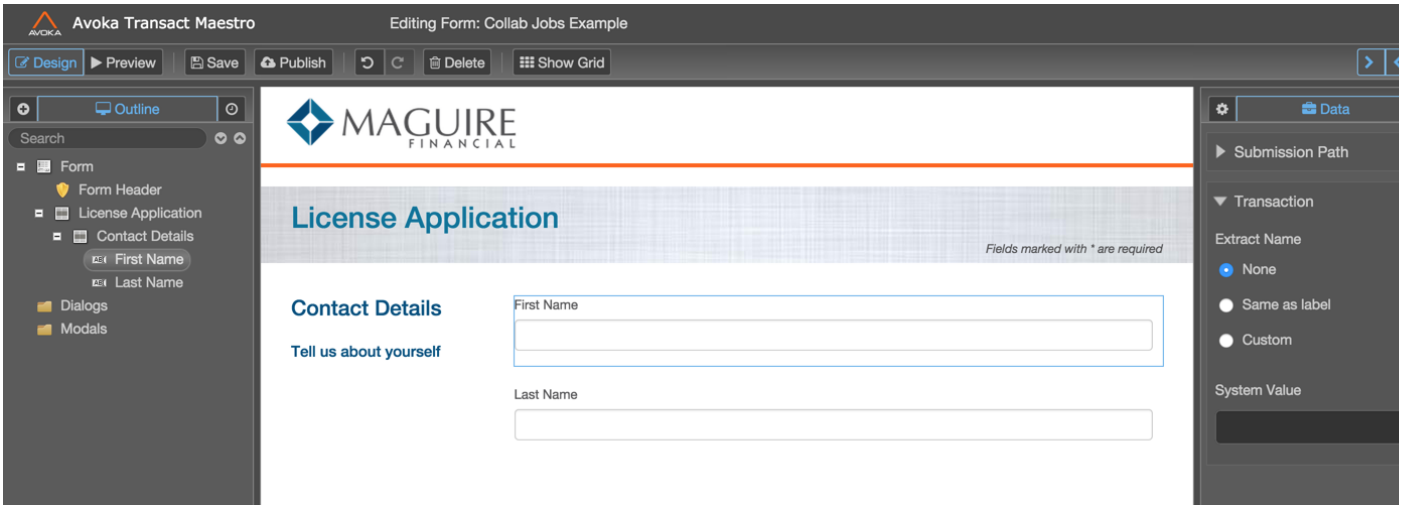
More advanced composer techniques are documented here:

Integration

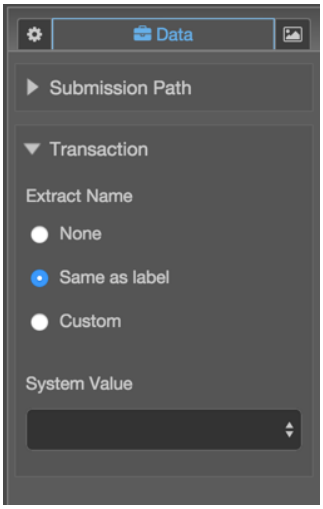
Data Extracts

To use the values of individual form fields within the job definition, assign them a Data Extract name, in the Transaction area of the form designer's Data tab. You can choose to use the field's label as its Extract name, or to set a custom name.

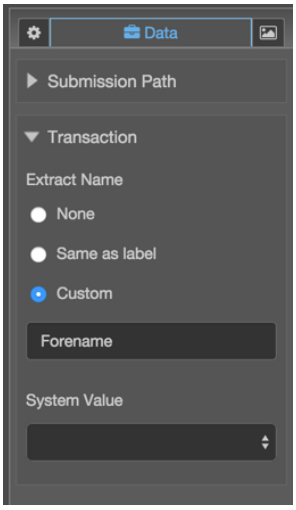
- Open the Transaction area of the Data tab



- Set the extract name to be the same as the label

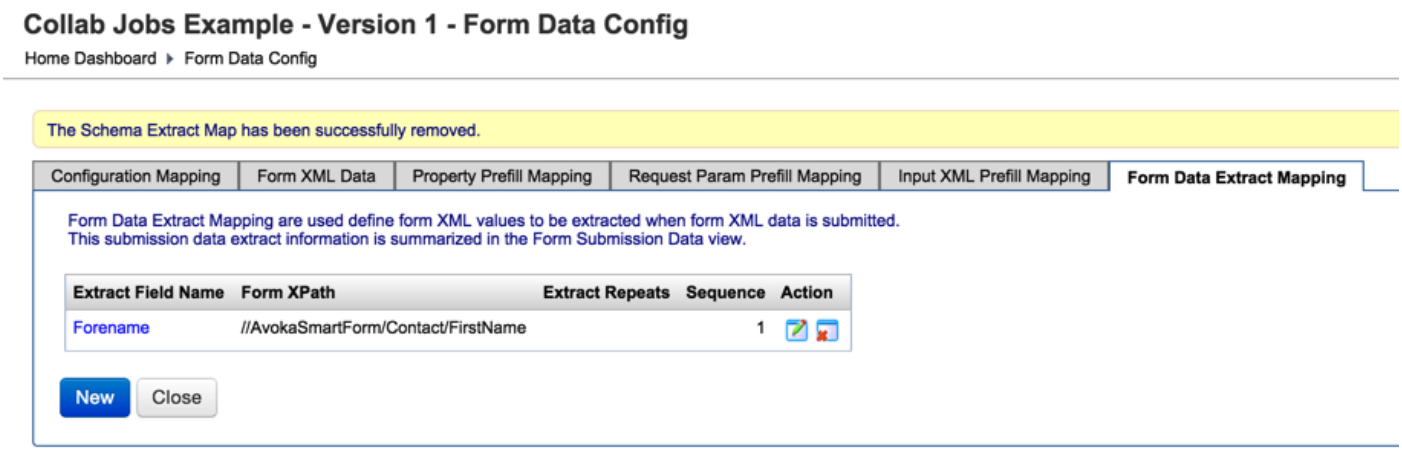


- Or choose a custom extract name



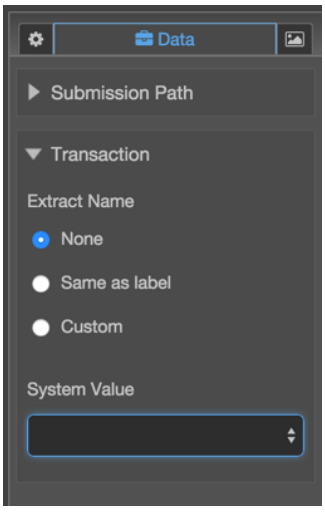
- The published form displays the mapping in TM

Need to show Encrypted, Publish and Subscribe



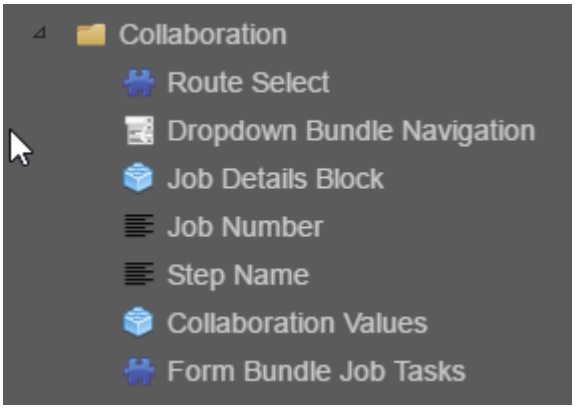
Form Data Mappings

You can assign fields in the form to the standard system values used within Transact (Contact Email, Contact Phone or Save Challenge) using the System Value dropdown, in the Transaction area of the form designer's Data tab.



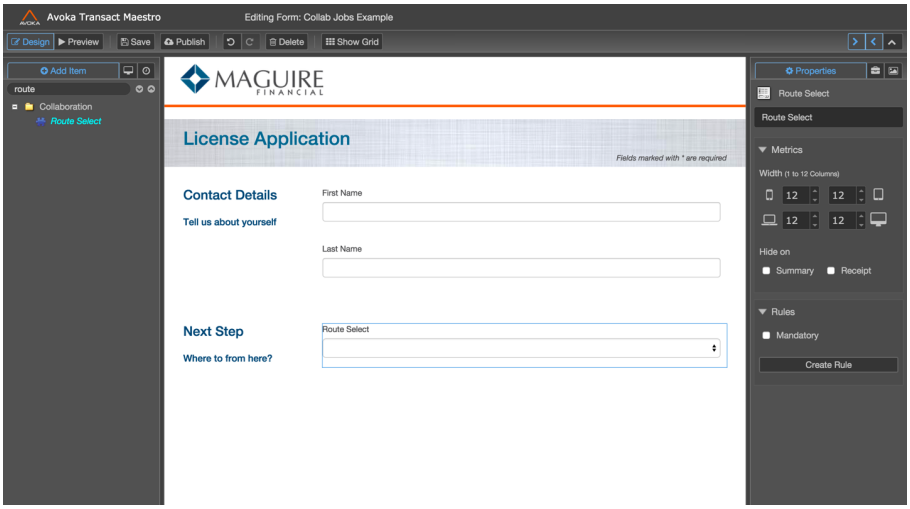
Collaboration Widgets

A number of Collaboration widgets assist in building Collaboration Jobs forms:



Routes Select

The Route Select widget in the Collaboration category lets the form applicant select the route for the next stage of the Collaboration Job, taking its value from the currently available routes.



Composer (CollabJobs v4.3)

This page gives background on the design of Composer forms for Collaboration Jobs.
• How to setup Data Extracts
• Describe Collaboration Jobs widgets that come with Composer 4.1
• Setting Rules the Visibility and Editabilty of sections.

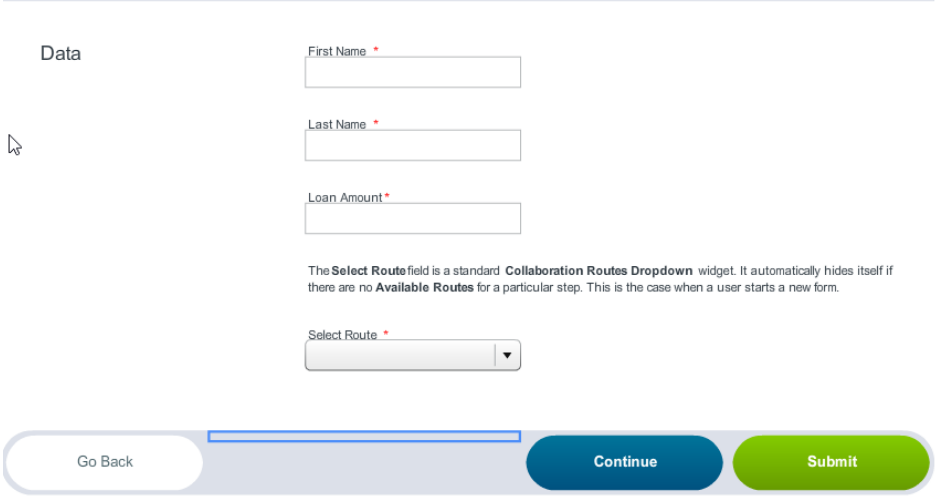
More advanced composer techniques are documented here:

Error rendering macro 'children'

Must specify a valid sort type ('creation', 'title' or 'modified').

Data Extracts

These are used by Collaboration Job to access form data as described Form Design (CollabJobs v4.3). The following screenshot of Job Example 2 Minimal form to describe how Data Extracts are configured. The first 3 fields that are setup as data extracts.



Note

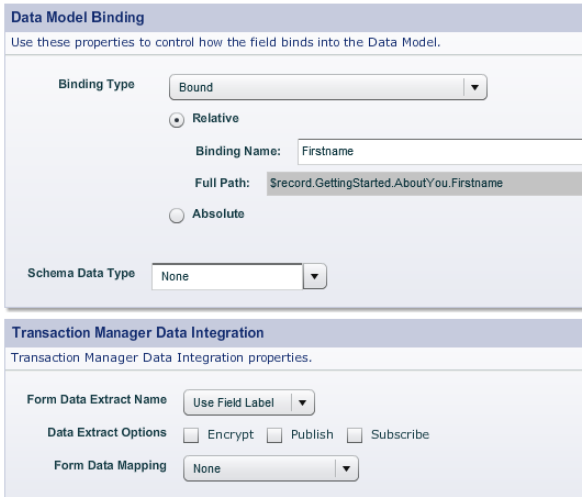
Collaboration jobs will put this into the $formDataMap.routeName values automatically.

Please see the Collaboration Routes Dropdown section below for the correct use.

Data Extracts are setup in Composer in the **Data Model** tab of the field's properties dialogue see screenshot below.

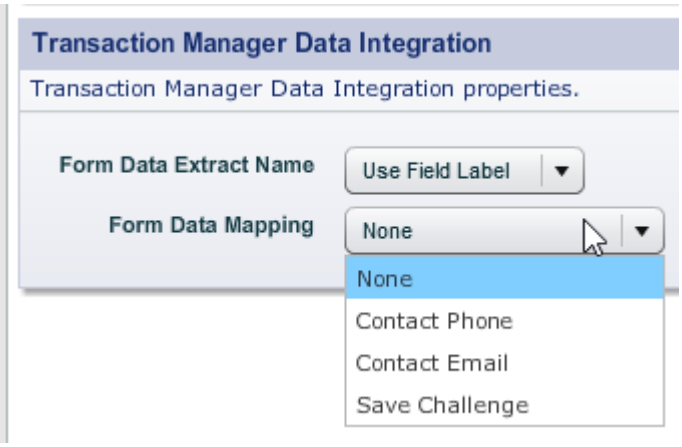
The easiest way to setup the Data Extract is to use the field label, however you can use an alternative name.

From Composer 4.3.0 there are check boxes to specify data extract options Encrypt, Publish and Subscribe.



Form Data Mappings

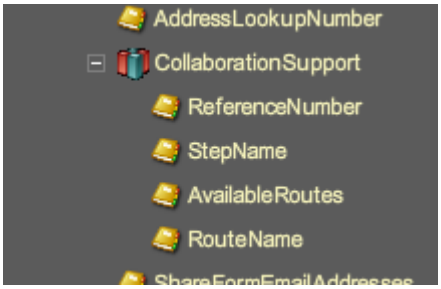
Fields can be mapped to predefined form data mappings as described here.



Collaboration Jobs Data Elements - Phase Variables.

The collaboration components described in the section below rely on the 4 Job data elements that are read into form phase variables. These can be found in

Nuts & Bolts -> **Transaction Manager Support** -> **Collaboration Support** as per the screenshot below.



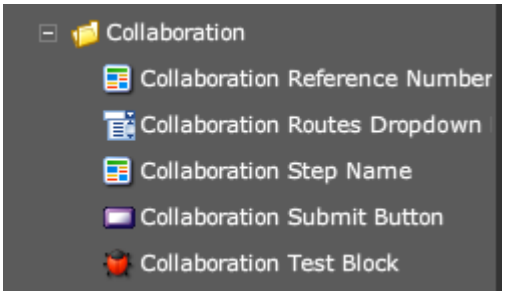
As describe the TM Job Controller sets the following XML data elements for each Task:
• **ReferenceNumber**: aka Job Number
• **StepName**:
• **AvailableRoutes**: This is a pipe delimited list of the current steps route names.

The Composer Form via a widget or business rule sets the RouteName on Task Submission.

Collaboration Widgets and other Components

Composer has a number of widgets that can be used to build forms for Collaboration Jobs as shown in the screenshot below.

The Maguire Template has additional collaboration support to display the step name and job number in the form heading.

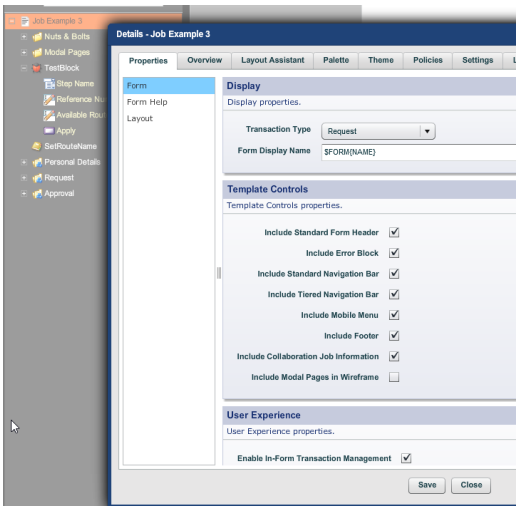


Collaboration Reference Number and Collaboration Step Name

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.

NOTE: We do not recommend that these fields be used as they are now both incorporated into standard Maguire 4.1 Template as per screenshot 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.



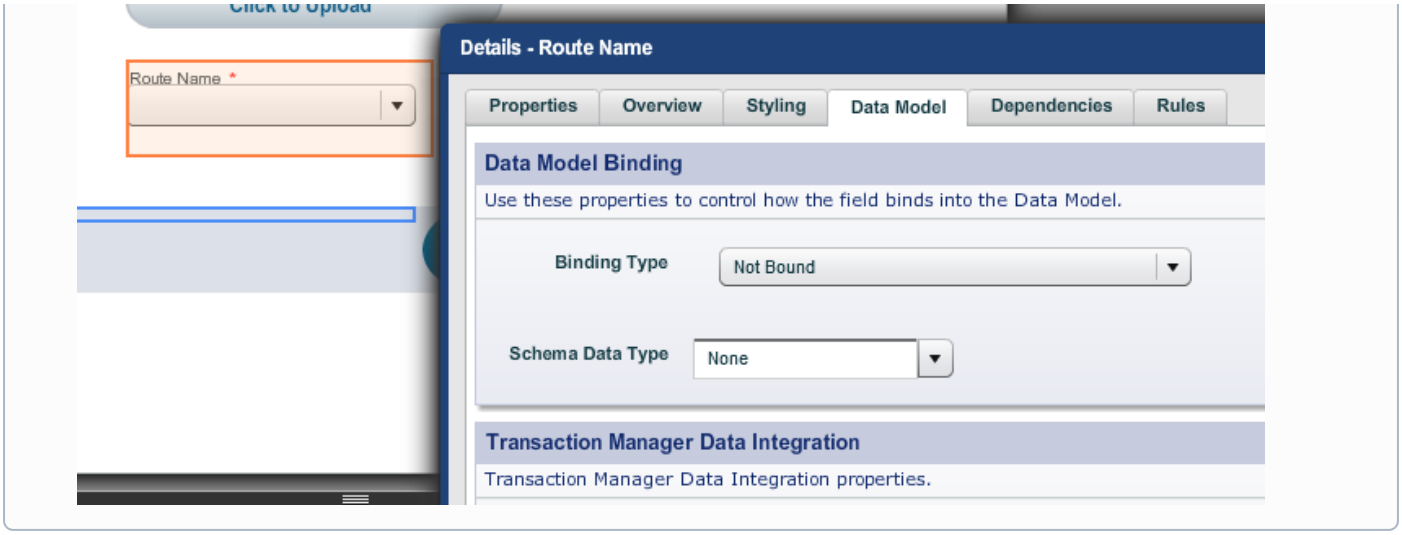
Collaboration Routes Dropdown List

The Collaboration Routes Dropdown List is the easiest way to make a user select the route to the next step.

Note

To ensure this item does not pick up the route of the previous step its strongly recommended that the binding be changed to **Not Bound**

After positioning it on the form you should select Properties > Data Model > Binding Type select Not Bound



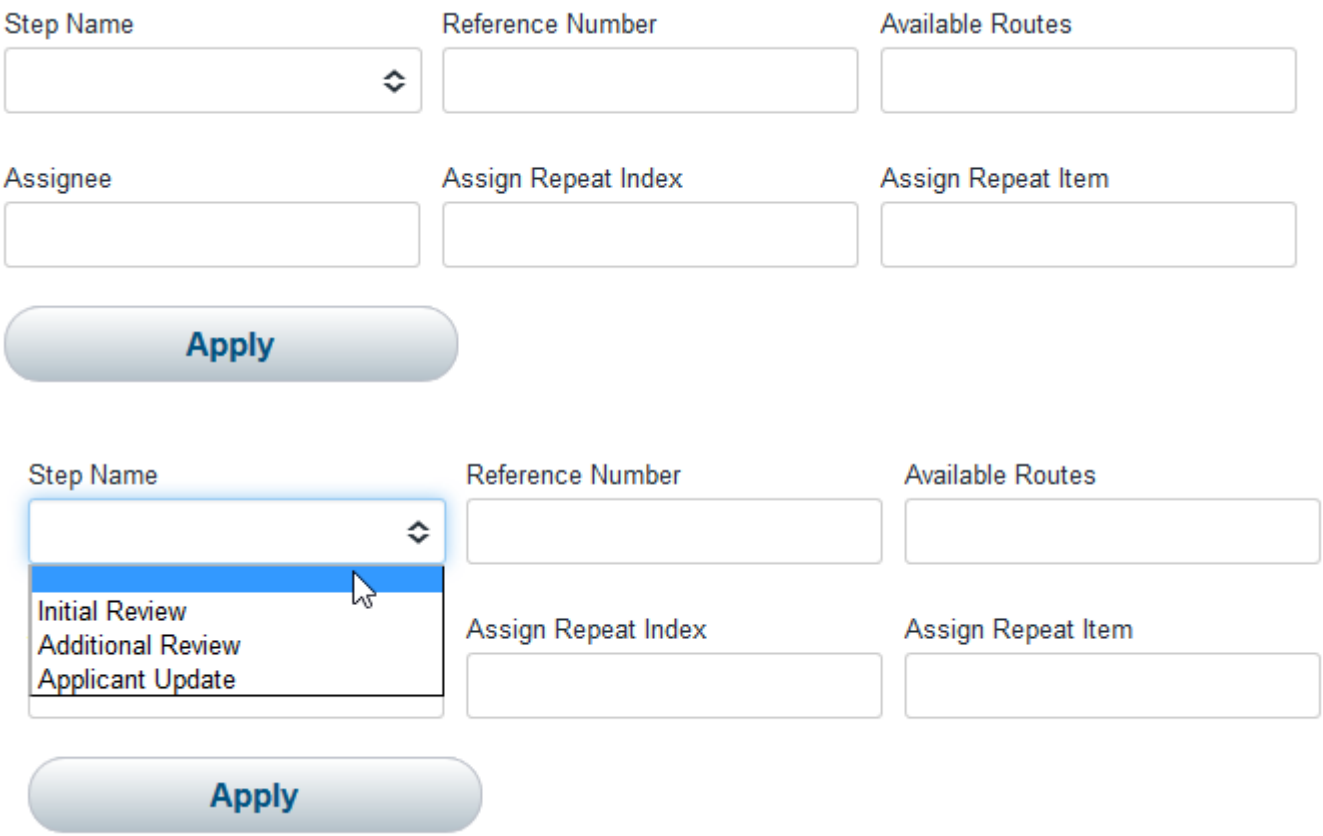
Collaboration Submit Button (Deprecated)

Do **NOT** use the Collaboration Submit Button widget. There have been some cases where the form using this button has failed to return the route.

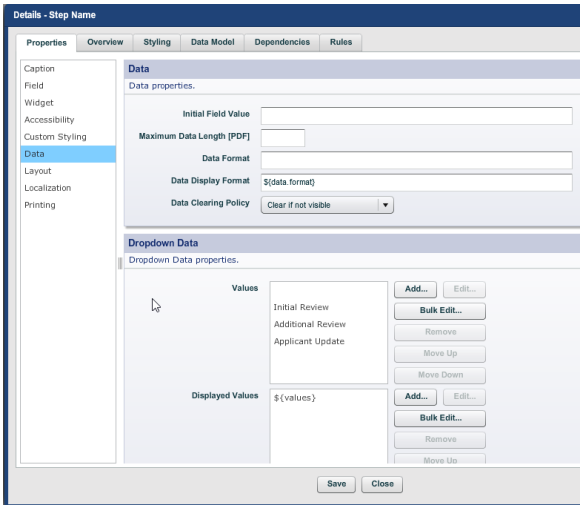
It was initially prototyped to create a submit button that sets it's label as a route.
We recommend that you either use the Collaboration Routes Dropdown or the standard Radio Button Group please see the **Routeing Using the Standard Radio Button Group** section below

Collaboration Test Block

The Collaboration Test Block is used to Test your form behaviour when previewing in Composer. It has the Step Name, Job Number and the Available Routes as per the screenshot below. From Composer 4.2.0 it has 3 fields that are used to test **Task Assign Repeat** tasks.



The Step Name field is a dropdown, you can select the Collaboration Jobs step names into the dropdown data values. Include an empty row to test the field's behavior in the form before its submitted.



Routeing Using the Standard Radio Button Group

This is the preferred method over using the collaboration submit button. Note this technique does not dynamically create the radio buttons. You will have to does hard code the values in you form.

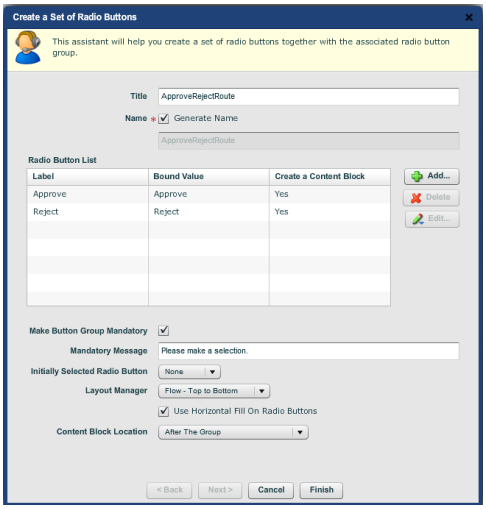
The advantages in using radio button groups:

- 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 contain an optional content block that gives additional instruction to the user. Useful for new users of a form. The content block can also hold addition fields that give feedback to the next review step.

The following instructions will create a single radio button group to use for approval.

In Composer drag the **Radio Button Assistant** widget into you form hierarchy. This will open up the **Create a Set of Radio Buttons** wizard dialog. Add the route names as Radio Button Labels.

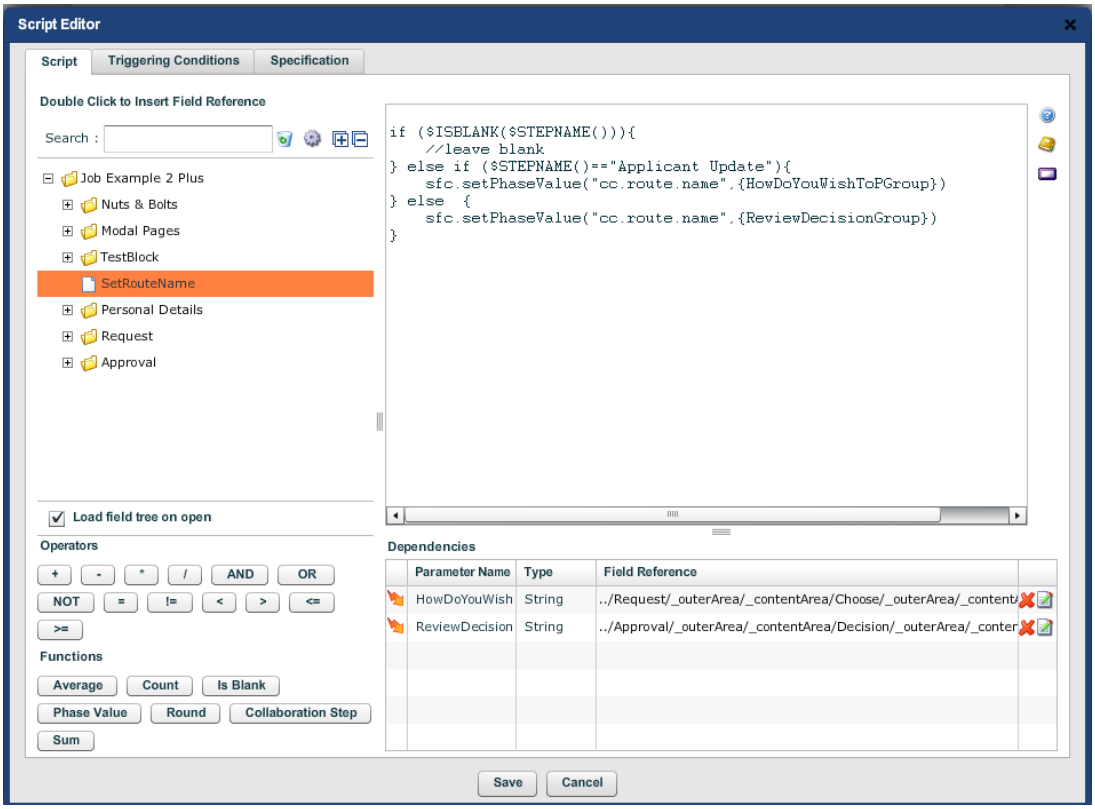
Note: You should make the button group mandatory.



Business rules can be used to set the route as per the section below.

You form has steps with different route choices like the Job Example 2 (Multi Step Review Job). The business rule may look like this.

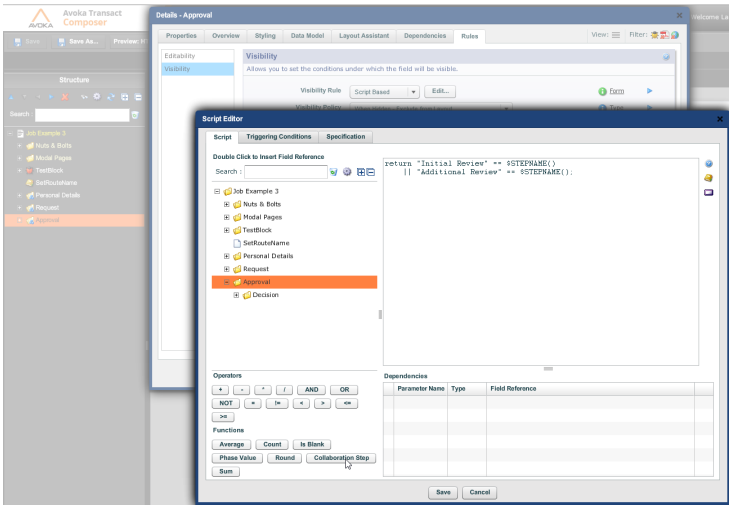
Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'



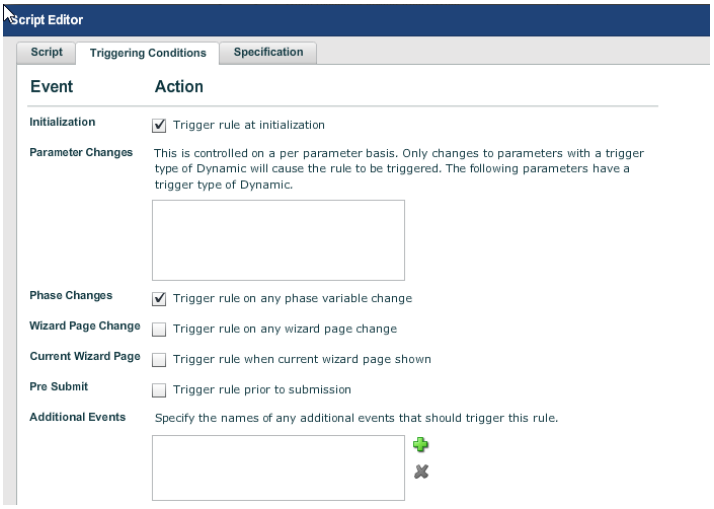
Visibility and Editability Rules

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 setting up a script rule there is a collaboration step button that will insert a reference to the step name ($STEPNAME()) into your script (screenshot below).



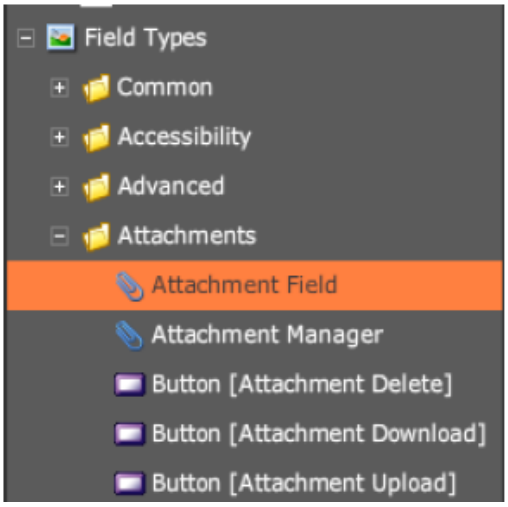
IMPORTANT! Before you save your rule go to the **Triggering Conditions** tab and click the Phase Changes check box (screenshot below).



As discussed above in the Collaboration Test Widget 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 Transact 4.1 with the Maguire Template 4.1 releases and is shown in the screenshot below.

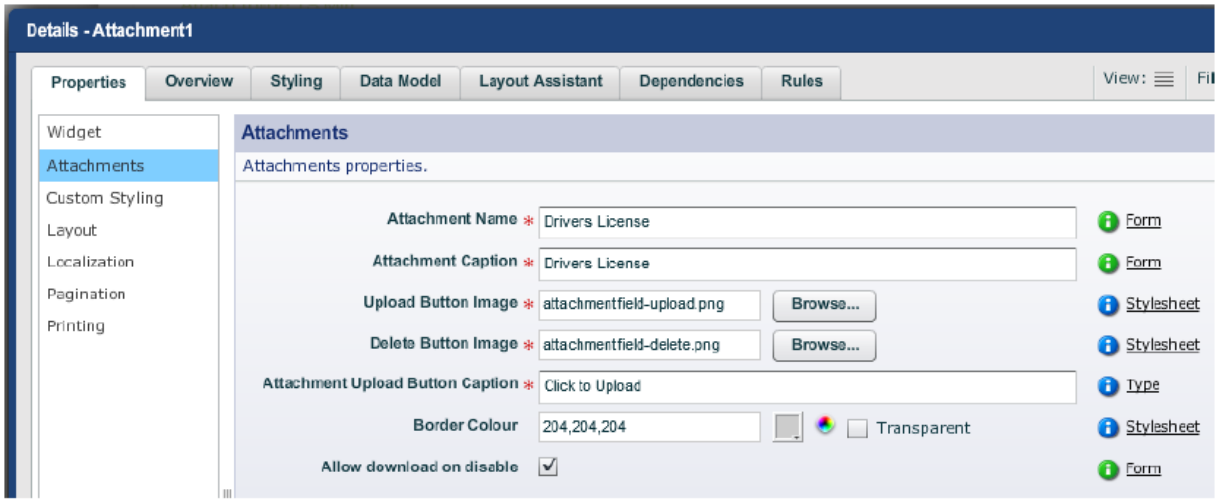


This widget will allow a file to be uploaded. A user may want to confirm the file is correct and can download the form by clicking on the file link. They can delete the file by clicking on the X.

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 per the section above.

But if the Editability rules determine that the widget is not editable at a particular step, the attached file cannot be uploaded, downloaded or deleted. There is an **Allow download on disabled** check box as per the screenshot 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.

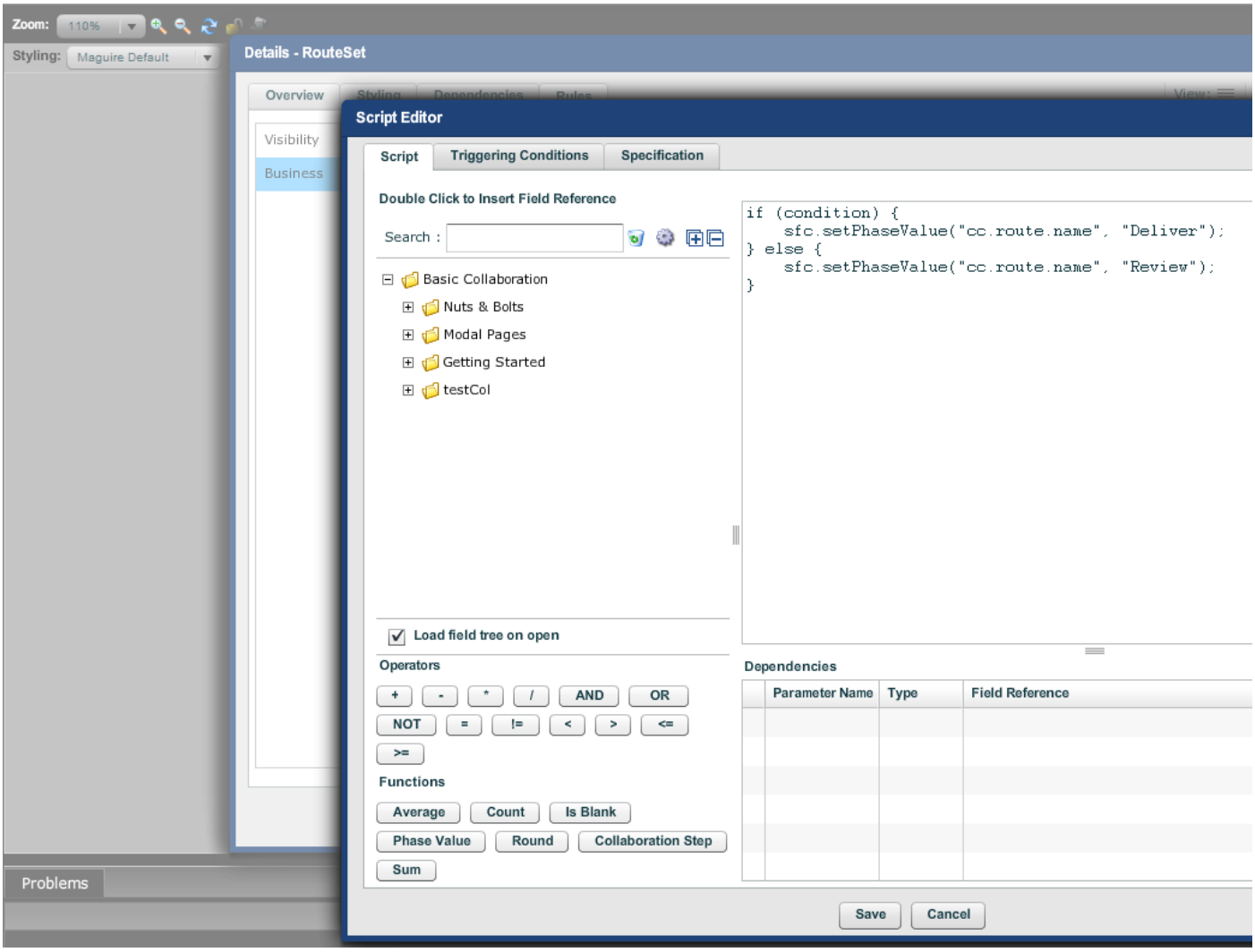


NOTE: The widget has an internal field Attachment Manager, which is pre-named "AttachmentManager" that must not be altered when using with TM task assignments and the job controller.

Setting Route with Business Rule

It is possible to set the route from within the form using a business rule. This functionality is how the prebuilt Collaboration Widgets and Components work. Using a small amount of code you can set the route phase value, this allows for logical changes to the collaboration route based on in-form information.

To set the route add a business rule to your form, and edit the business rule script with the following code.




```
if (condition) {
    sfc.setPhaseValue("cc.route.name", "Deliver");
} else {
    sfc.setPhaseValue("cc.route.name", "Review");
}
```

Using this method it is possible to set a number of the values asscoiated 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).

Anonymous Forms - Composer Save Challenge (CollabJobs v4.3)

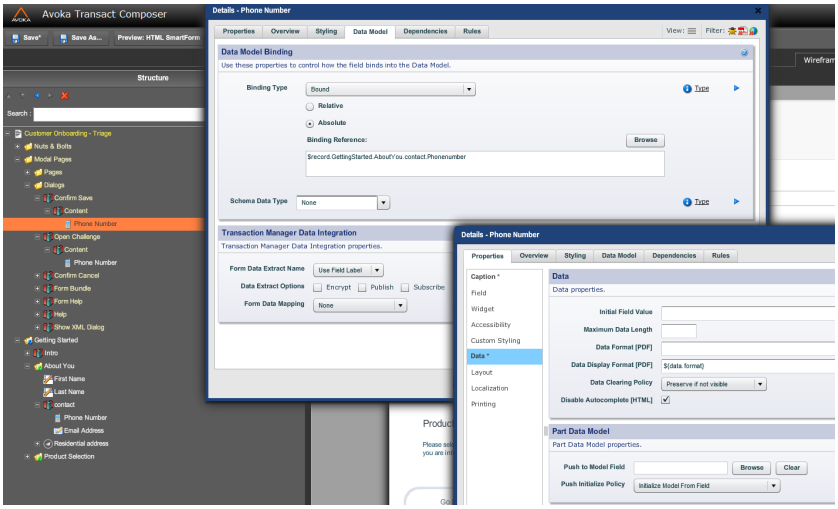
Setting up a Security Question or Save Challenge is a security feature that allows an anonymous user to save their form and retrieve it later.

It is used by Form Bundles (CollabJobs v4.3) since Transact 4.2.

How to configure in Composer

The steps can be found here KB:4.1 Adding a security question in Maguire forms

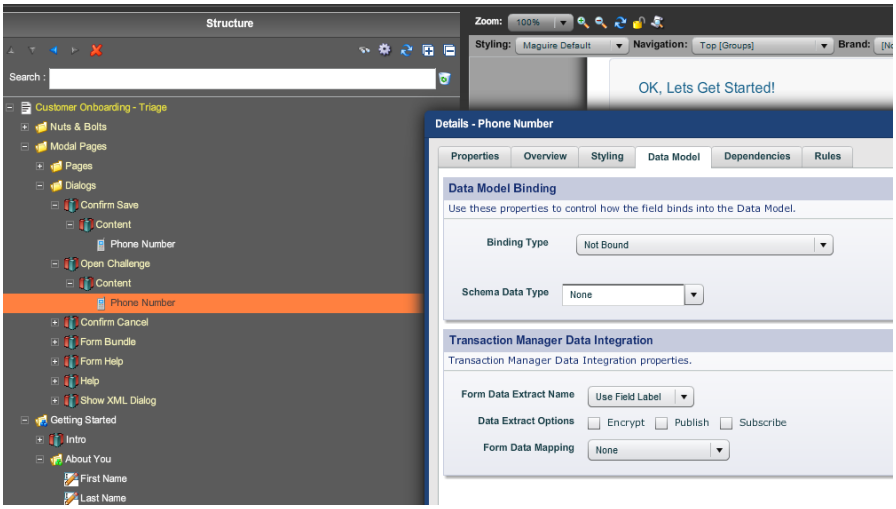
This is what the Confirm save dialog setting should look like. *Note the **Form Data Mapping is set to None**



> **Note**
>
> *** Form Data Mapping**
>
> The **Data Model -> Form Data Mapping** should always be set to None on all form fields.
>
> This is done by the Maguire Template in its hidden fields under the **Modal Pages -> Dialogs -> Confirm Save**

Additional Step

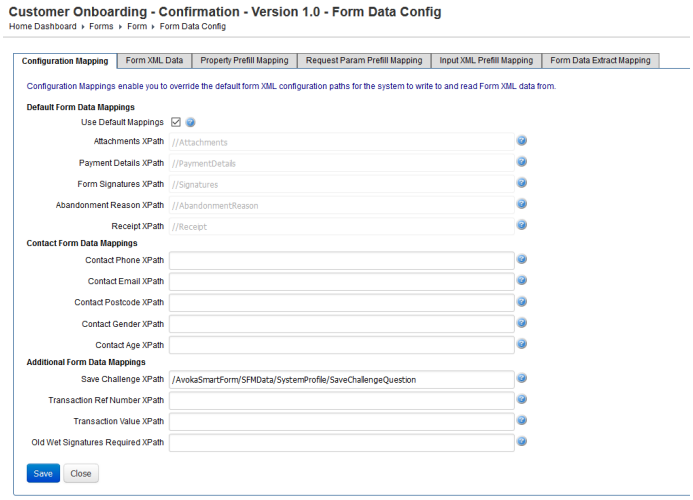
On the Open Challenge -> Content -> Field that was copied set the Binding Type to **Not Bound**.



> Export to Transaction Manager

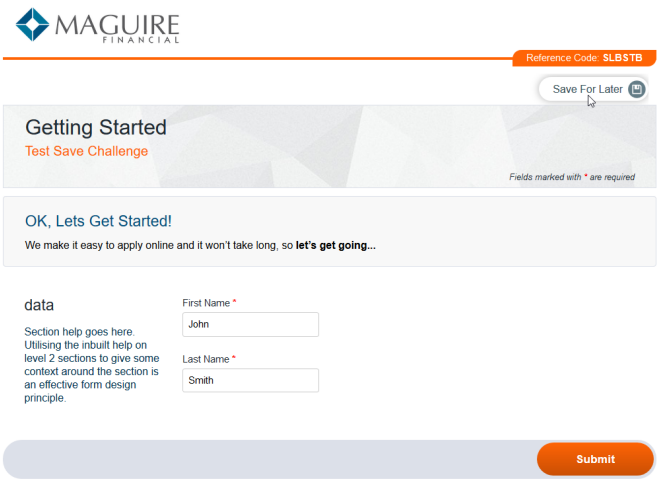
Verify Transaction Manager

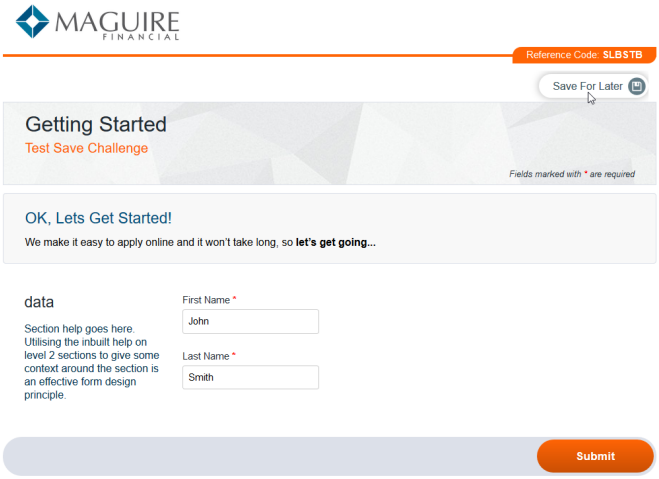
Form -> Form Version -> Form Data Config

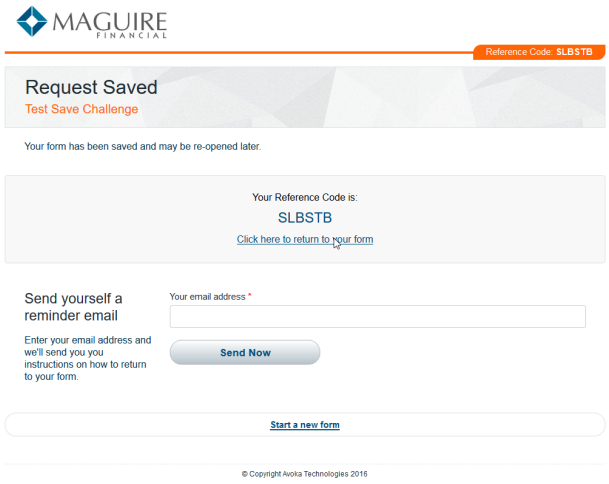


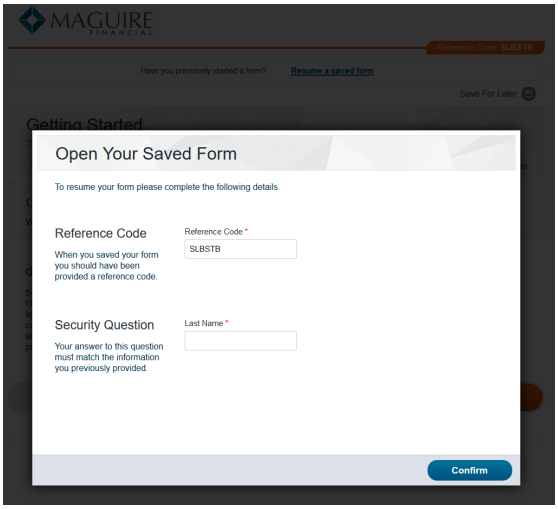
> **Note**
>
> The Save Challenge XPath should ALWAYS be
>
> **/AvokaSmartForm/SFMData/SystemProfile/SaveChallengeQuestion**

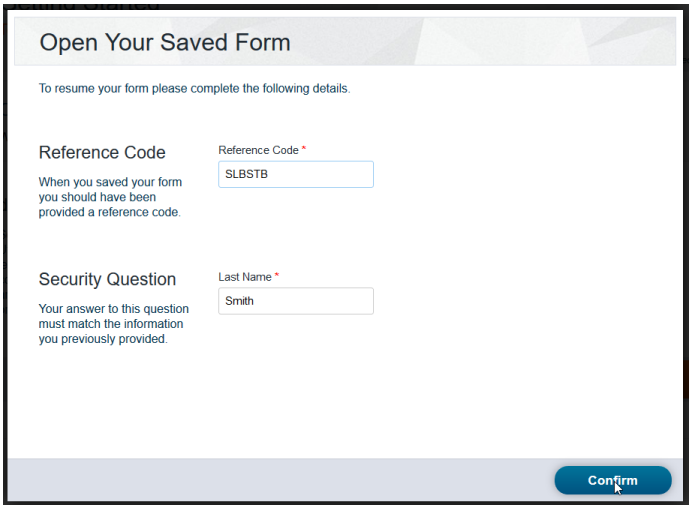
Example Flow

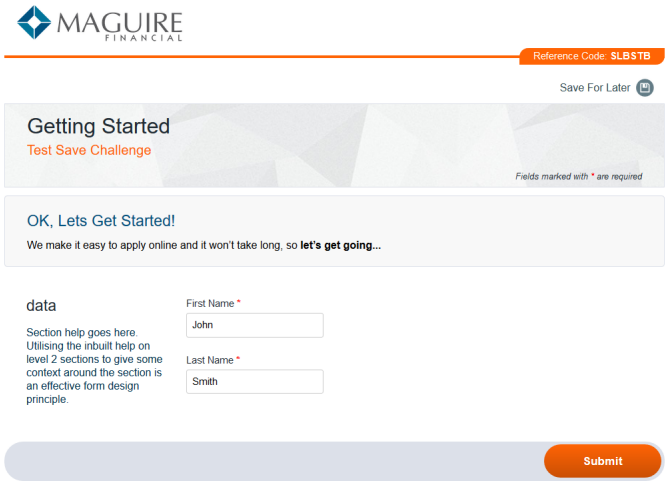



MAGUIRE
FINANCIAL
Reference Code: SLBSTB
Save For Later
Getting Started
Test Save Challenge
Fields marked with * are required
OK, Lets Get Started!
We make it easy to apply online and it won't take long, so let's get going...
data
Section help goes here. Utilising the inbuilt help on level 2 sections to give some context around the section is an effective form design principle.
First Name *
John
Last Name *
Smith
Submit


MAGUIRE
FINANCIAL
Reference Code: SLBSTB
Request Saved
Test Save Challenge
Your form has been saved and may be re-opened later.
Your Reference Code is:
SLBSTB
Click here to return to your form
Send yourself a reminder email
Enter your email address and we'll send you you instructions on how to return to your form.
Your email address *
Send Now
Start a new form
© Copyright Avoka Technologies 2016


Open Your Saved Form
To resume your form please complete the following details.
Reference Code
When you saved your form you should have been provided a reference code.
Reference Code *
SLBSTB
Security Question
Your answer to this question must match the information you previously provided.
Last Name *
Confirm


Open Your Saved Form
To resume your form please complete the following details.
Reference Code
When you saved your form you should have been provided a reference code.
Reference Code *
SLBSTB
Security Question
Your answer to this question must match the information you previously provided.
Last Name *
Smith
Confirm


MAGUIRE
FINANCIAL
Reference Code: SLBSTB
Save For Later
Getting Started
Test Save Challenge
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...
data
Section help goes here. Utilising the inbuilt help on level 2 sections to give some context around the section is an effective form design principle.
First Name *
John
Last Name *
Smith
Submit

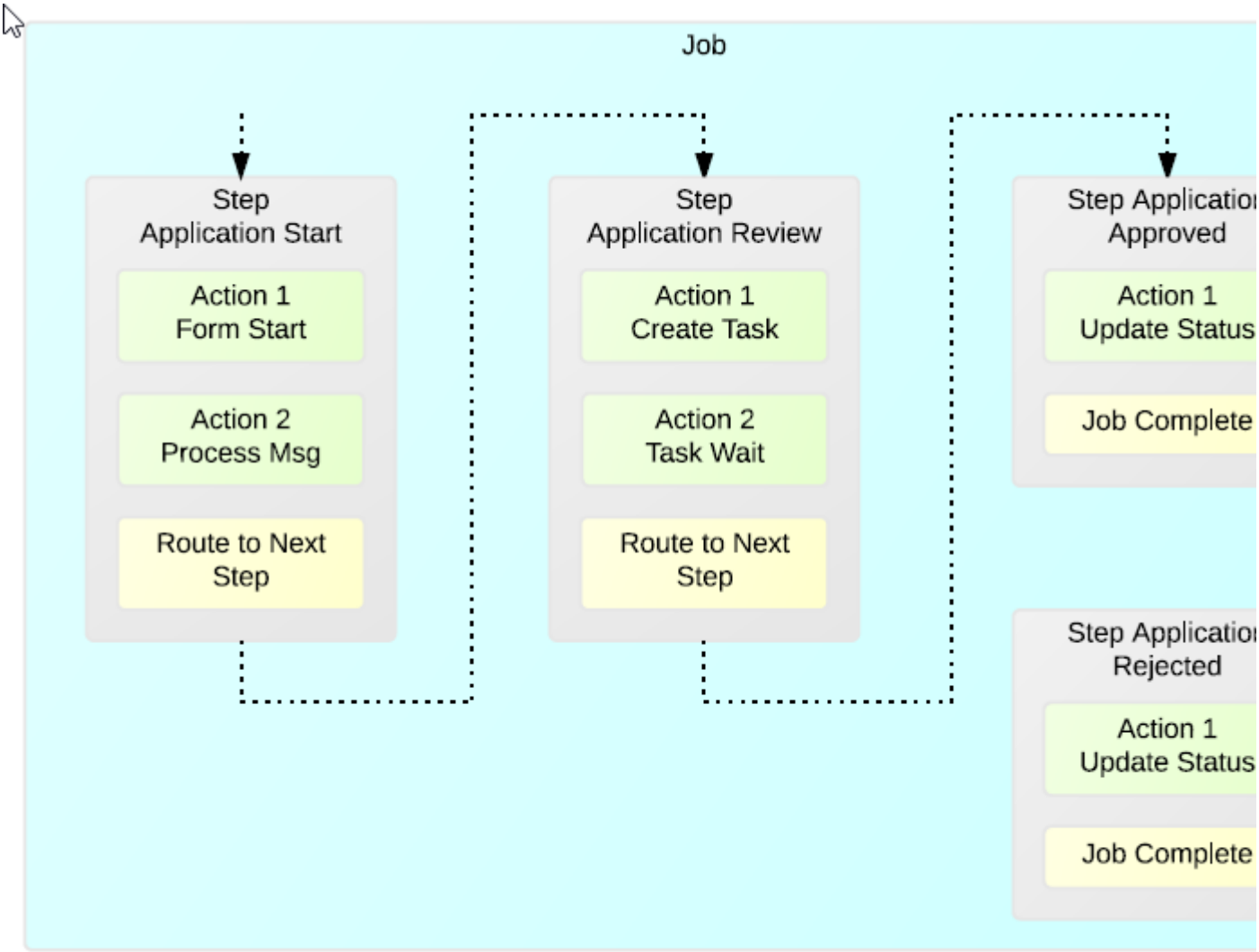
Job Basics (CollabJobs v4.3)

Building Blocks

Job Composition

Job, Step, Action Hierarchy

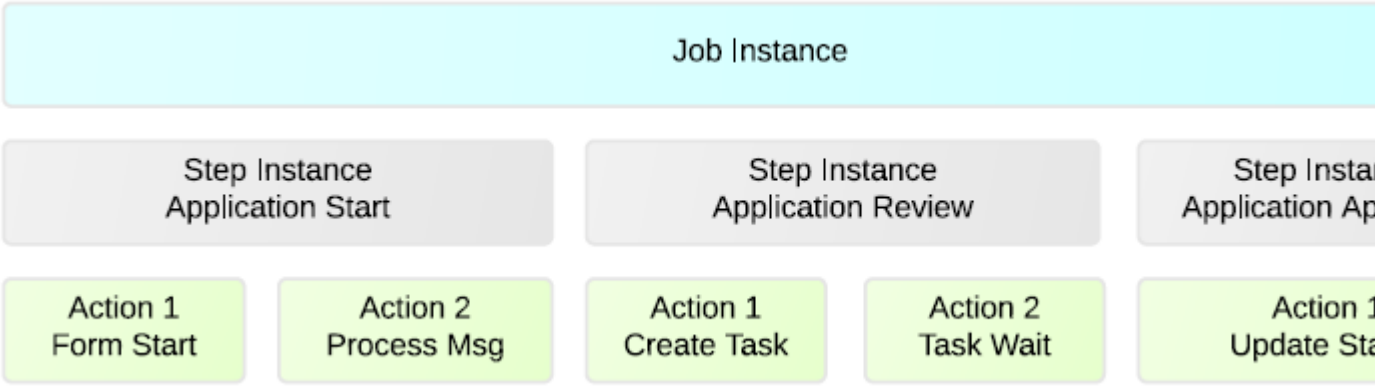
A **Job** contains 2 or More **Steps**, each **Step** contains one or more **Actions** as shown in the the diagram below. This hierarchy is important to understand as it flows down into the design of the Job Database Entities and the structure of the Job Definition (JSON).



Job Instances

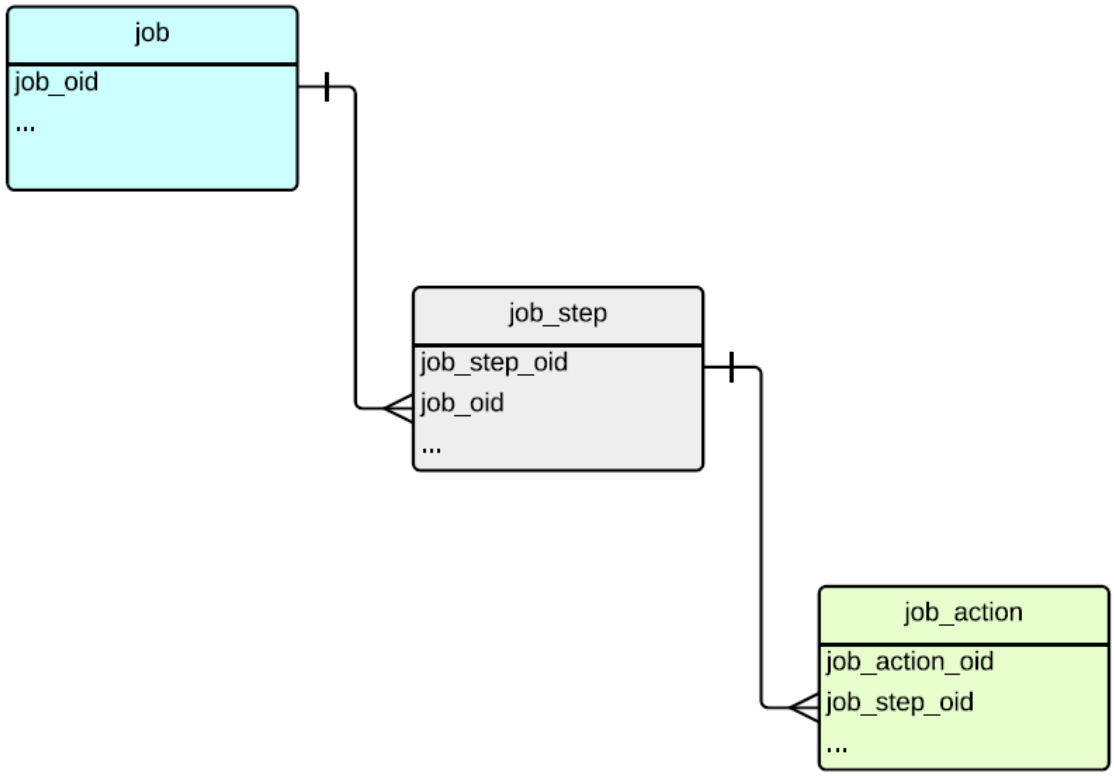
Lets look at a happy day scenario where the application is approved. This follows the dotted line in the diagram in the Job, Step, Action Hierarchy section above.

The Diagram below shows the job execution from left to right of a single job instance. The Job is started when a form is initially submitted. The job runs its start step **Application Start**, it completes its actions **Form Start** and **Process Msg** in order, The job is routed to the **Application Review** step which completes the **Create Task.** The **Task Wait** actions complete and then the step select the **Application Approved** as the next step. The **Update Status** action is run. As the step is an endpoint the step completes and the Job completes.



Job Database Entities

The Entity Relationship diagram below is a simple representation of the database tables job instances are stored in the **job** table, step instances are stored in the **job_step** table, action instances are stored in the **job_action** table.



Job Definition

Jobs are defined in a single text JSON format text structure. The block below is a sample job definition.

We can see that in the job definition has a first level property **steps** which holds an array of step objects. The individual step objects have a property **actions** which contains a list of the step actions in an array**.**

The attributes are described in detail in the Job Definition Structure section.

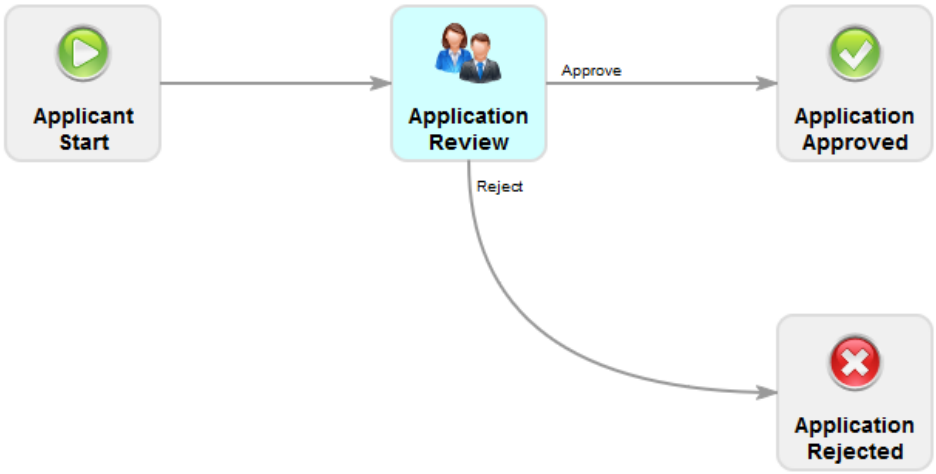
Note:

- The order that the actions appear is the order that they are processed within the step.
- The order in which the steps are listed does not matter.

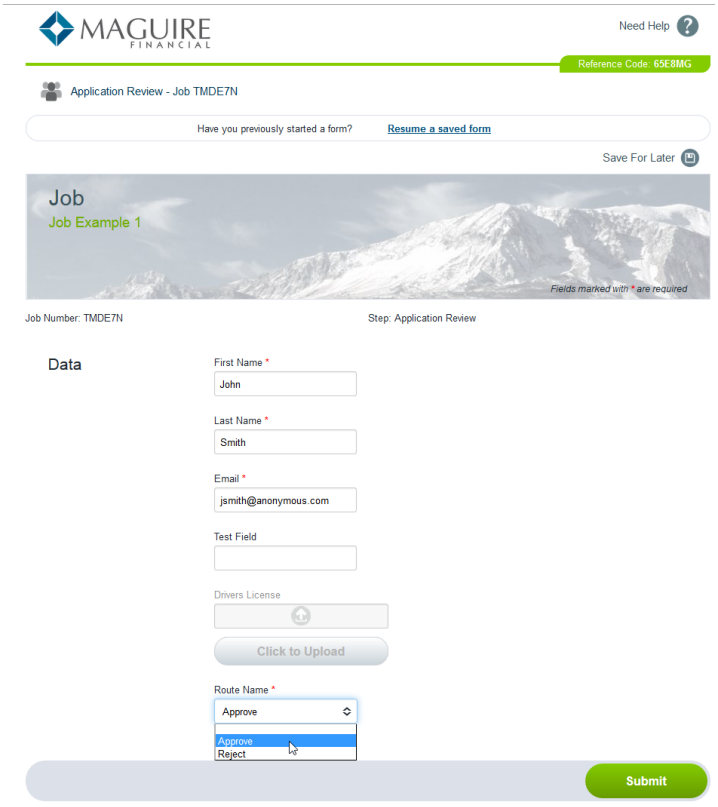
Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

Steps and Routes

A **Step** completes by executing all its **Actions** in order. The last **Action** may return a **Route Name** result. A lot of the time this **Route Name** is selected by a user when they fill in a form. Steps define **Routes** which are mappings between route name and the **Next Step** that will be run. The job processing is a finite state machine, with the steps being the states.



If we look at the Application Review step in the diagram above. The first action creates and assigns a Group Task. A user who is a member of that group can open the form associated with the task see form below. They are required to select a route name and submit the form.



If the user selects **Approve** it is routed to Application Approve, If they select **Reject** it is routed to Application Rejected.

Job

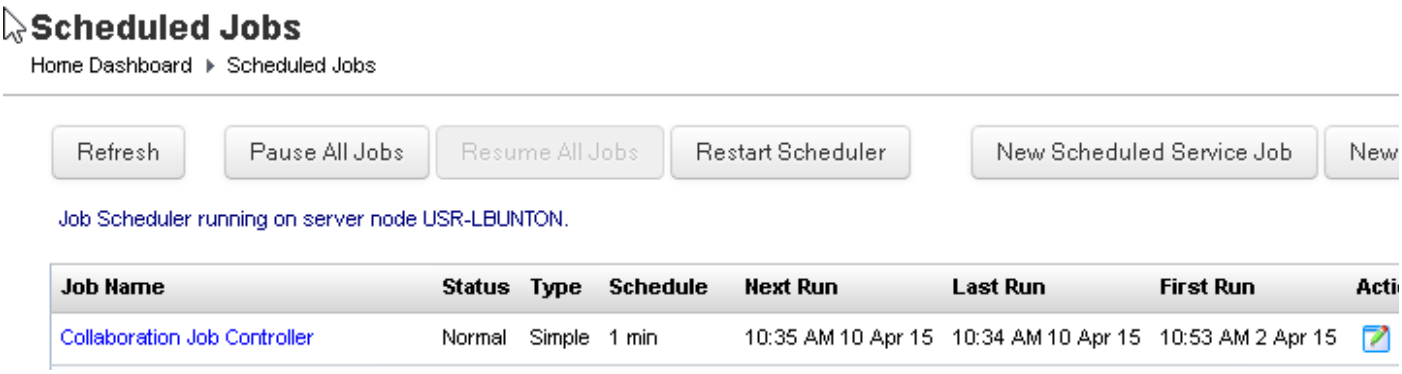
Job Statuses

- **In Progress** - the job is active and in progress
- **Completed** - the job has been completed. This happens when a endpoint step has completed.
- **Expired** - this happens when an Endpoint expiry job has been automatically expired by the system
- **Cancelled** - the job has been manually cancelled by an administrator.
- **Error** - there is an critical / unrecoverable error with the job, it requires operator intervention before it can be run

Job Execution

This is important to know the execution modes that are used by Collaboration Jobs as they differ for Form Bundles and Review And Approval jobs. This difference has licensing implications.

A Scheduled Job - **Collaboration Job Controller** runs every minute see screenshot below. Note the schedule period is configurable.



The Schedule Job kicks off a single thread (**Job Thread**) that loops over each job instance that is available for processing. It loads the associated Job Controller Service (Job Definition) and executes the Job processing. Whether a job is available for processing depends upon the job, step and action status, please see actionservices section below for more details.

Once the Job Thread has finished processing all the jobs it closes. Note the scheduler will only start 1 Job Thread at a time. A large backlog of work could take the Job Thread more than a minute to complete. A new Job Thread will not be started until the previous one has completed.

Standard Mode

License: 1 job counts as 2 transactions irrespective of the number of submissions in a job instance.

Use: Review and Approval jobs.

User submission completes:

1. The standard form submission processing
2. Calls a job and runs a callback that makes it ready for processing.

Job Thread runs (up to 60 seconds later) and executes the job processing.

Considerations

- Pausing the scheduler job **Collaboration Job Controller** stops these jobs processing.
- The standard mode of processing separates submissions from processing. The Submission is not slowed down by having to run the job processing.
- There is a delay between when the submission is completed and the processing begins this can be up to a minute. Issues can occur for:
    - Form Bundles scenario where tasks are immediately assigned back to the same user.
    - Demonstrating Review and Approval Job (non production)
- Developing and debugging is easier in this mode. Jobs can be failed at certain actions. This action programming or configuration can be fixed and the job retried and the action completed.
- Automatic retry and recovery.
- Best for where there are many participants in the job.

Process Immediate Mode

License: Per individual submission.

The Process Immediate Mode is used by Form Bundles such as customer on-boarding forms. For example a new banking customer fills out an initial form which has some common fields and a list of products which they select those they are interested in. On submission the user is then presented with a list of forms to complete, one for each product.

The process immediate mode works as follows

User Thread completes:

1. the standard submission processing
2. the callback to make the job ready for processing
3. the next part of the collaboration job instead of the Collaboration Job Controller

Considerations

- Pausing the scheduler job **Collaboration Job Controller** stops these jobs processing but does not stop Jobs from completing.
- Recommended Uses
    - Form Bundles where there is a single participant for completing all tasks.
    - Demonstrating Review and Approval Job (non production)
- Debugging is harder in this mode. Note there is a transaction associated with the whole form submission - user thread execution. If an error occurs all 3 items will be rolled back. Including steps and actions executed as well as job event logs.

- The job processing adds time to the submission completion, the end user can be waiting some time before the user is presented with the submission completed page. For Form Bundles it nay take an extra 5 - 10 seconds to run the Job that creates the form tasks. However we need to create the tasks before we can present the results back to the same user with the list of forms on the submission completed page.

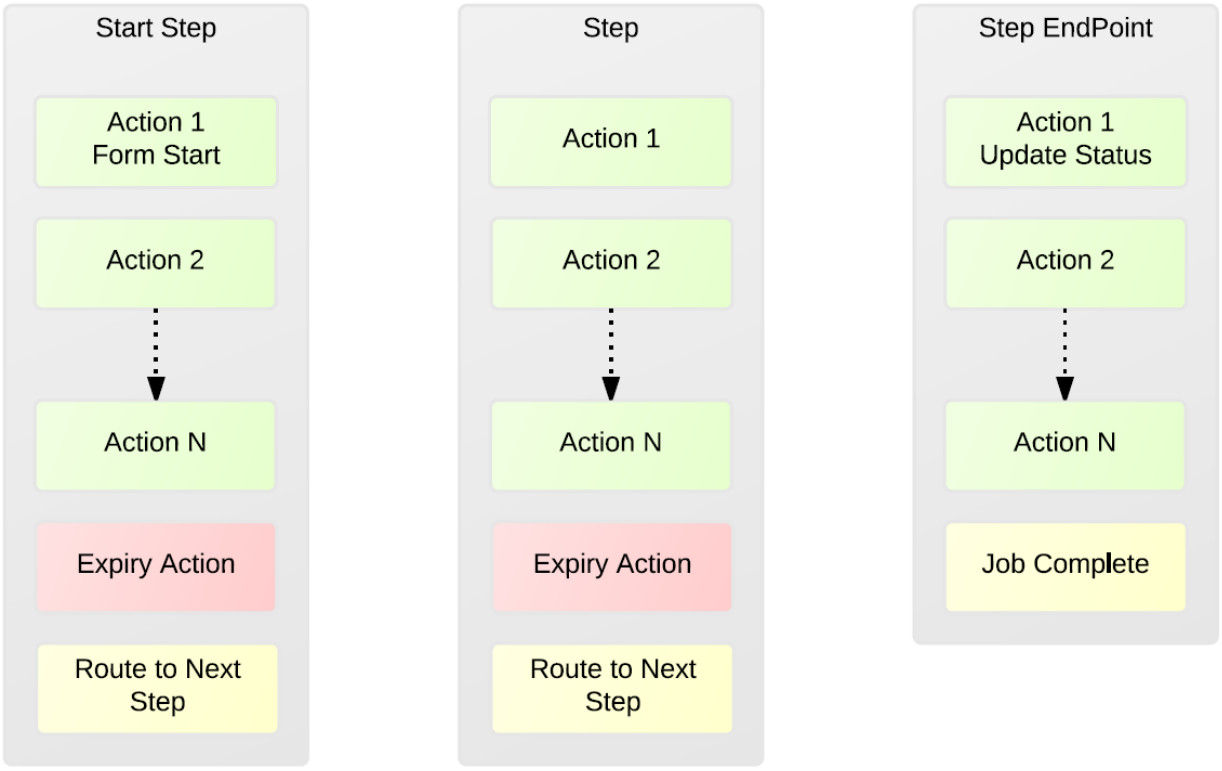
Step

Step Status

- **In Progress** - the step is active and in progress has actions that have not completed.
- **Completed** - the step is completed
- **Expired** - See the Step Expiry below.
- **Cancelled** - the step has been manually cancelled by an administrator

Step Processing

A Standard Step is structured as per the center "Step" in the following diagram. The step contains a list of 1 to "n" Actions which are executed in order. They can optionally define an Expiry Action which is discussed in the Expiry section below. They contain 1 or more routes.



One of the Steps in the job must be defined as type **Start**. The Start step has the same structure as the standard step. Usually Jobs are started by a form submission so the first Action **Start** Steps is usually a **Job Form Start** Action services (see Job Action Services below).

A job can have 1 or more **Endpoint** Steps. Endpoints can execute 0 or more actions, these actions might update processing status or send an email. When the actions complete, the step completes and the job completes. There are 2 endpoint Steps in Job Diagram from the Step Routing section above; Application Completed and Application Rejected. They represent the State of the job when it completes.

Step Expiry

There are 2 mechanisms that can be used to expire a step

- The step can have an attribute "expiryRule" that can set the step to expire after X hours or days or weeks after from its creation date. This is useful for setting an SLA. For example a Review step allocates a task to a group (reviewers). They must have 2 days to action a task, otherwise it will expire and be routed to an Escalation step for a manager to review.
- Any action in a step can control the step expiry by returning an action result that sets an action expiry or returns a status of Expired.

An Expiry Action (Job Expiry Service) may be set for a step. This allows clean up code to be run.

Steps that expire require an **Expiry** route name set. When a step expires its routed to the next step defined in the **Expiry** Route Name

Actions

Action Services

Each action will run an instance of its associated Job Action Service. There are a number of predefined actions that are packaged with Transaction Manager as shown in the table below. Groovy Action and Expiry services can be created to perform a custom task.

Action Execution

The following are the valid Action Statuses. Below we will discuss the use of these statuses in the following

- **Ready** - will be executed next time job runs
- **Assigned** - step process has completed successfully, but awaiting the user submission to be fully completed
- **Pending** (paused) - waiting for an external event to move it forward
- **Error** potentially (looping) - will be executed next time job runs.
- **In Progress** (looping) - will be executed next time job runs
- **Completed** - will never run again
- **Expired** - will never run again

Simple Action Processing

The job controller will process a simple action as follows.

- In a happy day scenario the action service completes and the Action Status gets set as **Completed**. The job controller proceeds to the next action or step.
- If an error occurs during processing may stop if **Error** is selected. There are different classification of errors as follows:
    - **Transient Errors:** some errors that we may want to retry automatically (web service down). For custom job actions programmed in groovy can set a time in the future to next run the job / Action Service.
    - **Hard Errors:** Errors of this type cannot be fixed by trying again, they often occur when testing newly written code. We may want to stop the job until the error is fixed. The job instance can be manually retried by an administrator.
- A Groovy Action Service can implement **Polling**. It can check to see if an event has happened such as - has a receipt been generated. If the event has:
    - happened then it will set the Action Status to **Completed**. The job controller proceeds to the next action or step.
    - not occurred then it can set the Action Status to **In Progress**.
- If an expiry is set on the containing step or the expiry time has been reached or the expiry time on the action is reached, the Action Status gets set to **Expired**.

Auto Retry Behaviour

Actions set to **Error** and **In Progress** statuses can have retry behaviour (looping). By default the action will be retried the next time the job controller runs after a 5 minute delay. This is a Job Controller service parameter **actionRetryDelayMins** see Job Controller Service section below.

eg. Lets say an Action calls a web service. If the web service is down for a couple of minutes the action instance status can be set to **Error** and the next time to run is set to now + 5 min. The Web services come up next time the action is run it connects to the web service and processing completes setting the action status to **Completed**.

Custom job actions written in Groovy (see Custom Action Services section below) can be programmed to specify a the next run time e.g. now + 20 min.

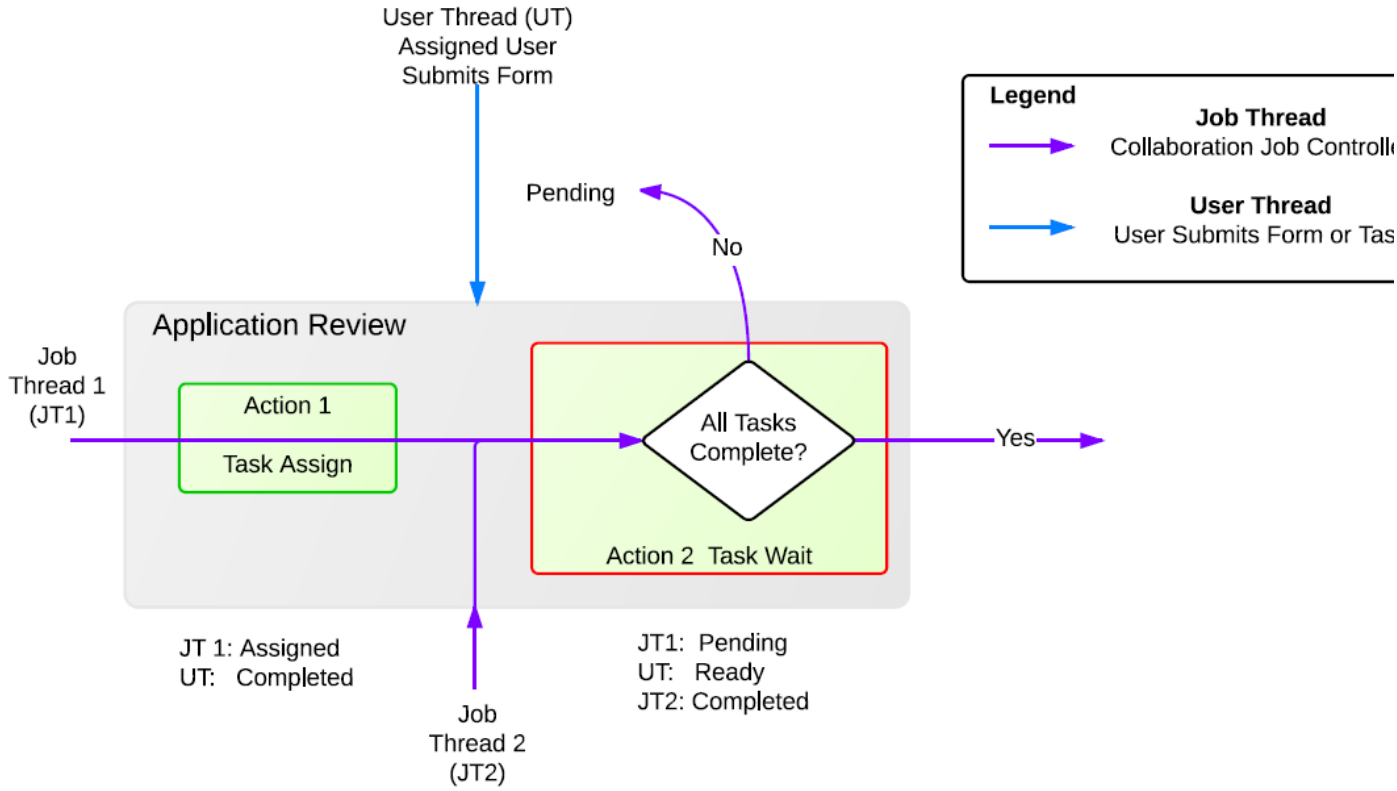
Wait Action Processing

A Wait action implements an efficient callback mechanism to deal with asynchronous event that occur outside of the job processing. The most common use case is a user completing a **Task**.

A **Job Task Wait Service** is paired in a single **Step** with one or more **Job Task Assign Action**. The Job Task Assign Action creates a Task and assigns it to a user or group. The user completes the task via a submission, this could be within hours or maybe days or even weeks after the Task was assigned. The submission triggers a callback which enables the job to complete.

Example 1 Task Step - Standard Mode Processing

The diagram below is used to describe the state changes associated with the actions in a task step. Please refer to the processing modes from the Job Execution section above. The **Job Thread** instances are in purple, the **User Thread** associated with submissions are shown in blue.



Job Thread 1

immediately runs the Job Task Assign action.

- It creates and assigns the Task (which is a Submission). The Task submission is linked to the Action Instance. The Action for the Task Assign Action is set to **Assigned**.
- Job Thread 1 then runs the Task Wait processing. As the Action 1 task isn't complete the Action Status of the Task Wait to **Pending** and J ob Thread 1 terminates. No processing is done on a job where the current action is in **Pending** status.

Assigned User Action 1 - Task Submission Thread.

This generates a call back that:

- Changes the Task Assign action status from **Assigned** to **Completed**.
- Changes the Task Wait action status from **Pending** to **Ready.** This makes the Task Wait to be available for processing.

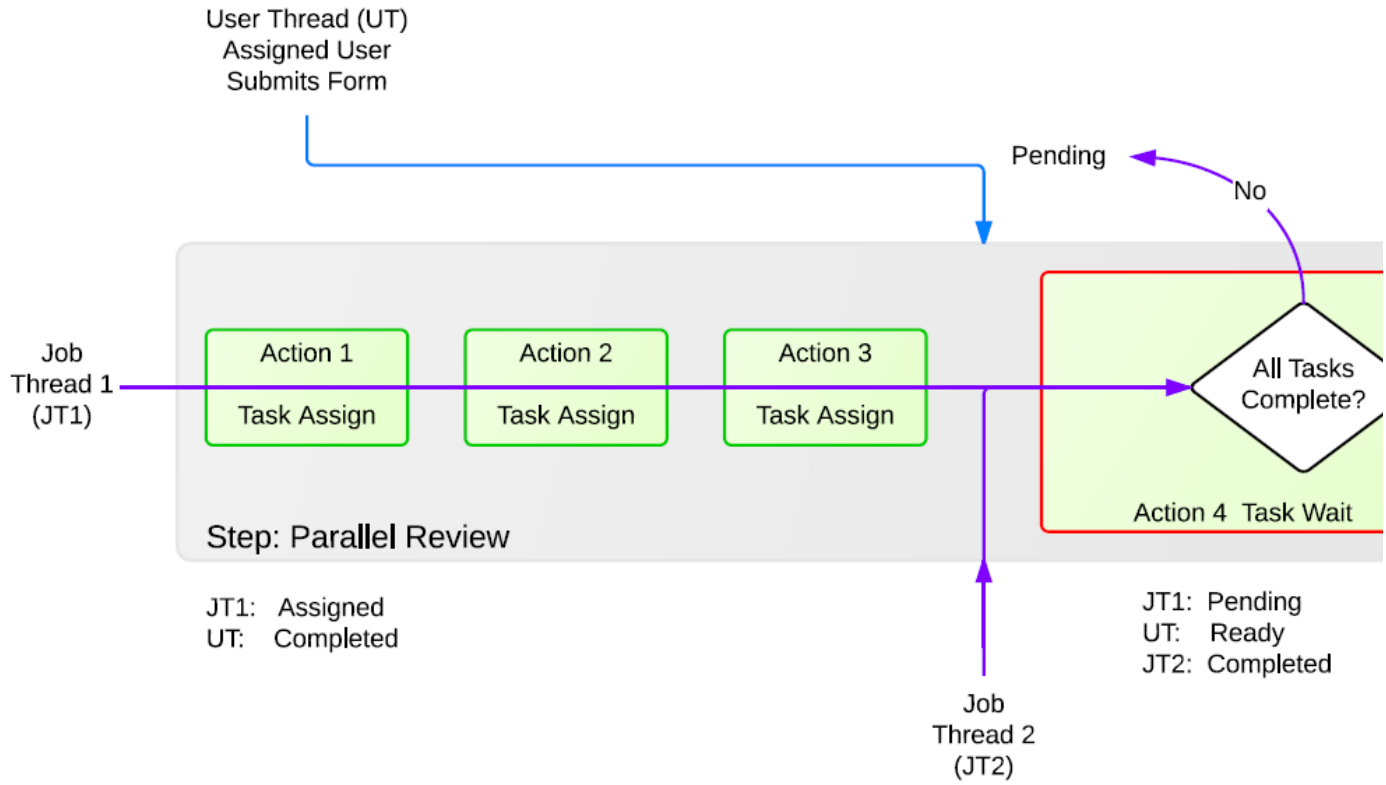
Job Thread 2

- Completes the Task Wait processing and the Status of the Task Wait is changed to **Completed**.
- Then completes the application review step and routes to the next step for processing..

Example 2 Step with Multiple Tasks - Standard Mode Processing

The Job Task Wait Action service can wait on more than 1 Job Task Assign action services. The diagram below shows a step that has 3 tasks assigned. This is a parallel tasks example which could be created with the same form assigned to separate users or groups.

Note: There are several configurations that are covered in advanced examples.



Job Thread 1

- Creates all of the action instances and the 3 associated tasks are created and are given an action status of **Assigned**.
- Stops when it gets to the Task Wait and sets the status to **Pending**.

Assigned User Action 1 - Task Submission Thread:

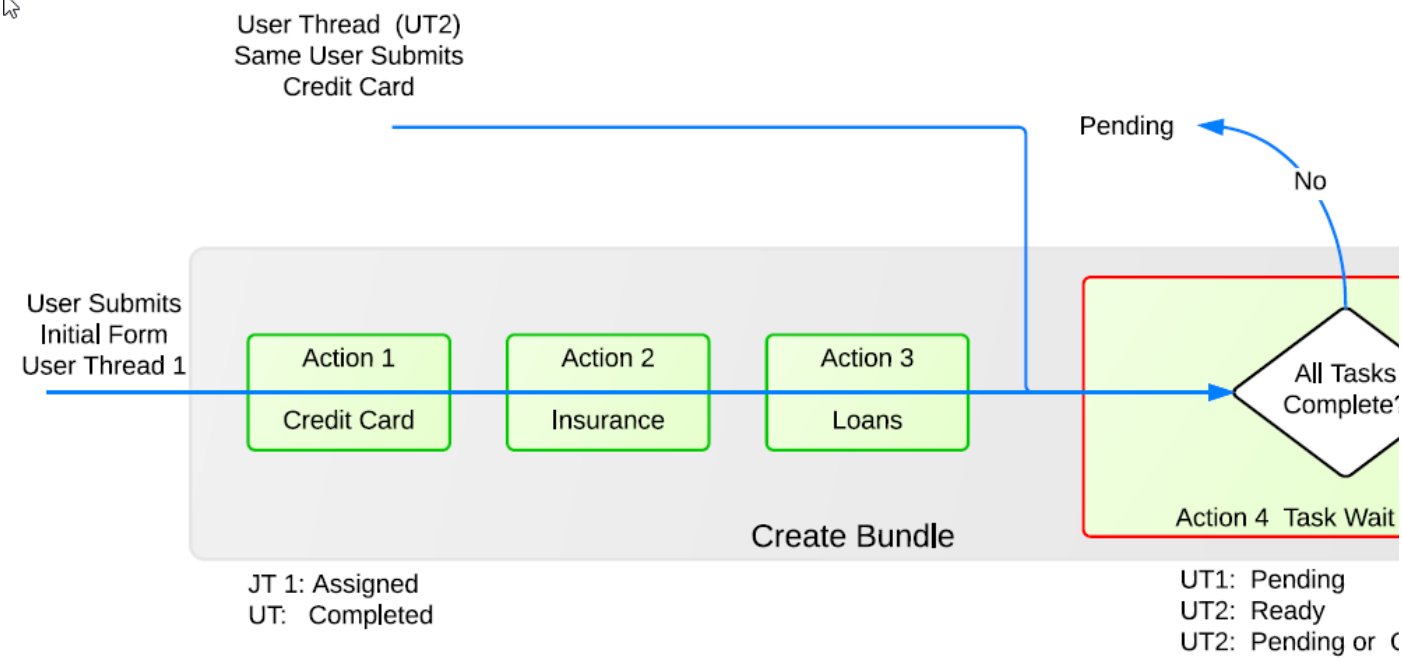
- Callback triggered that changes the Action 1 status to **Completed** and Changes the Task Wait status to **Ready**

Job Thread 2

- The Task Wait Processing is run which check to see Are Tasks Complete:
  - No: The action status for the Task Wait is set to **Pending**.
  - Yes: The action status for the Task Wait is set to **Completed**. The step is completed and processing the next step is done.

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

- 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**.
- Stops when it gets to the Task Wait and sets the status to **Pending**.

User Thread 2 - Same User Submits Credit Card:

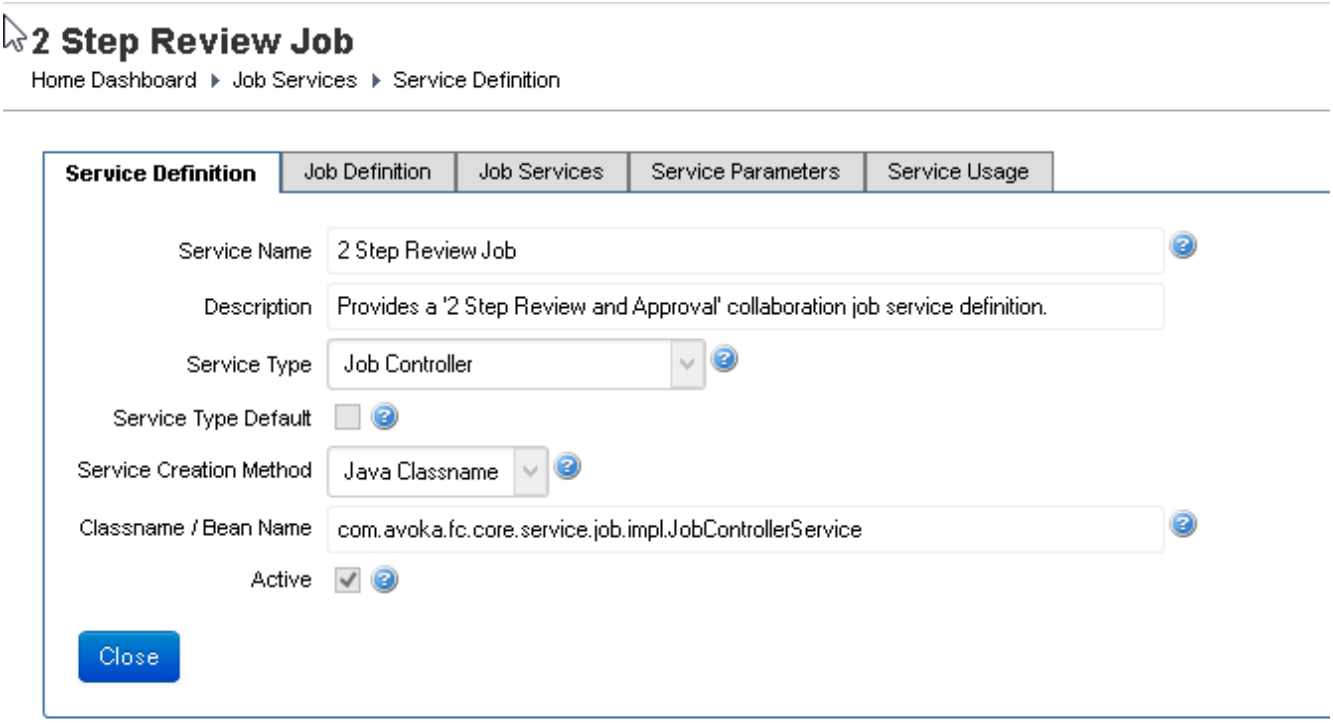
- Callback triggered that changes the Action 1 status to **Completed** and Changes the Task Wait status to **Ready**
- Immediately Runs Task Wait processing is run which check to see Are Tasks Complete:
  - No: The action status for the Task Wait is set to **Pending**.
  - Yes: The action status for the Task Wait is set to **Completed**. The step is completed and processing the next step starts.

Job Services

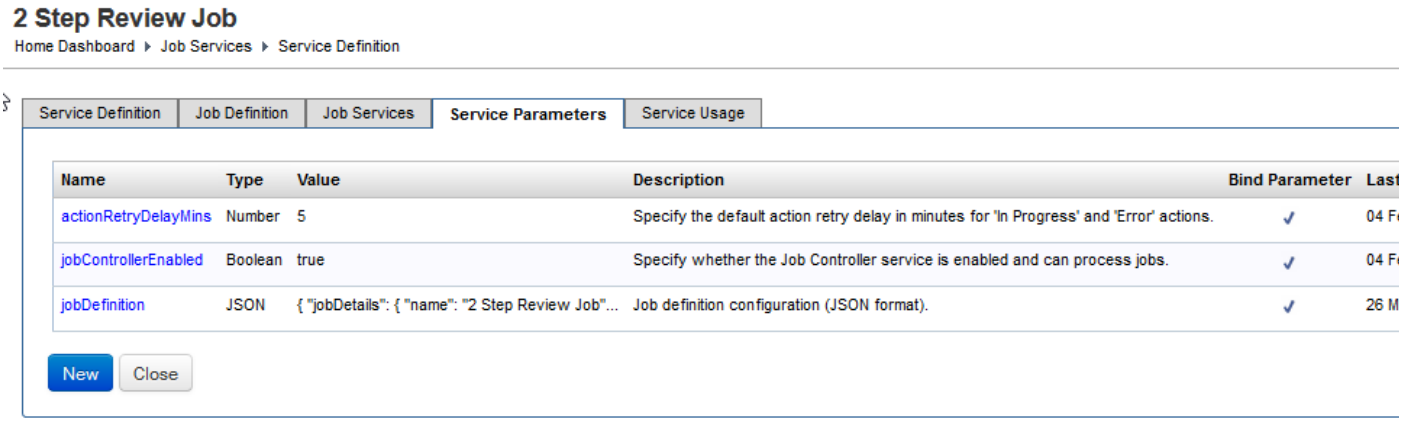
Job Controller

The Job Controller service is responsible for holding the Job Definition and the Job processing. For a job controller to run it needs to be associated with one or more forms see Transaction Manager Form Configuration.

Below is an example of the a job Controller that has been created from an inbuilt 2 Step Review template.



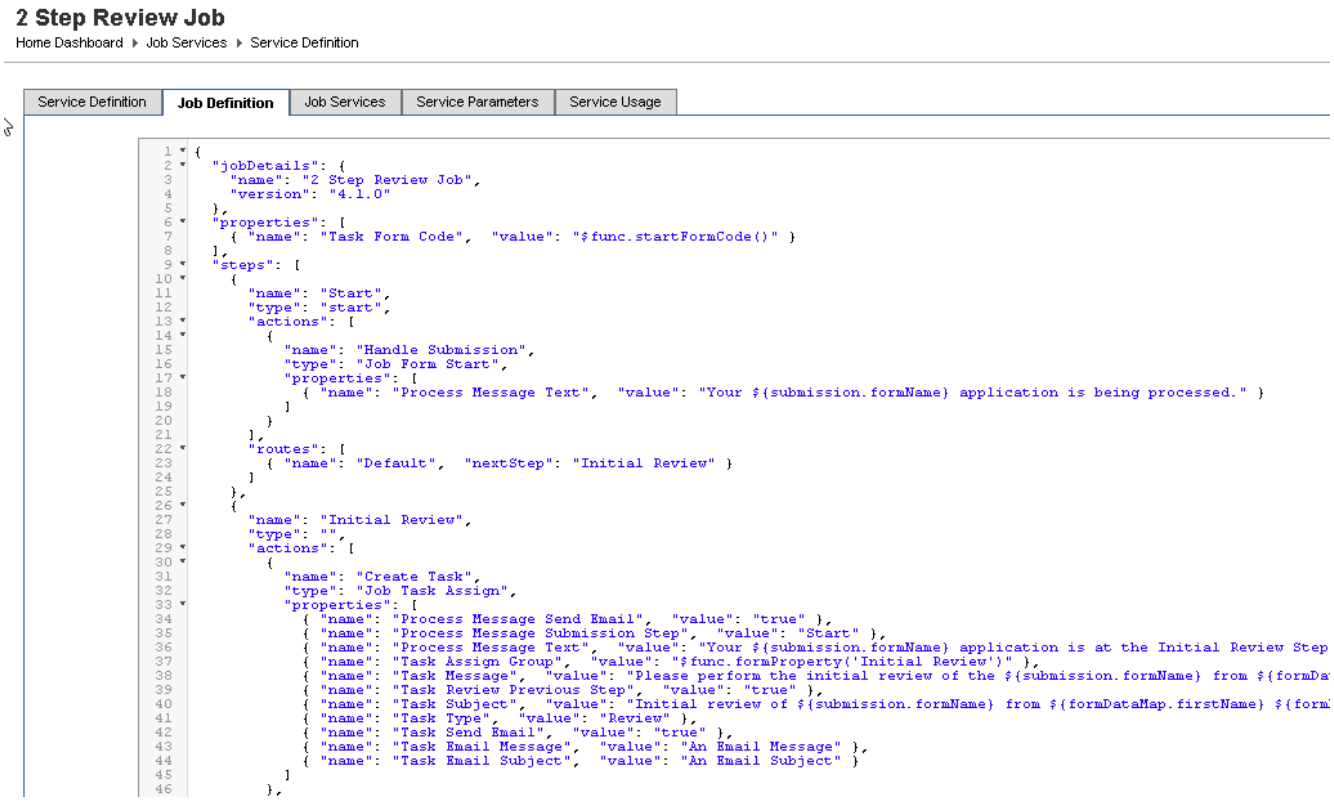
There are 3 default service parameters that are specified.



actionRetryDelayMins: This is the default delay in minutes for Actions that are have a result status of **In Progress** and **Error.** By Default this is set to 5 minutes.

jobControllerEnabled: Setting this to **false** turns off processing for any job instances that use this controller. This is useful for debugging test systems and selectively enabling certain jobs in production.

jobDefinition: Can be edited by clicking on the Job Definition Tab (Below). Previous versions of the job definition can be accessed via the **Service Parameters** tab. Edit the jobDefinition service parameter and there is a **Parameter History** tab.



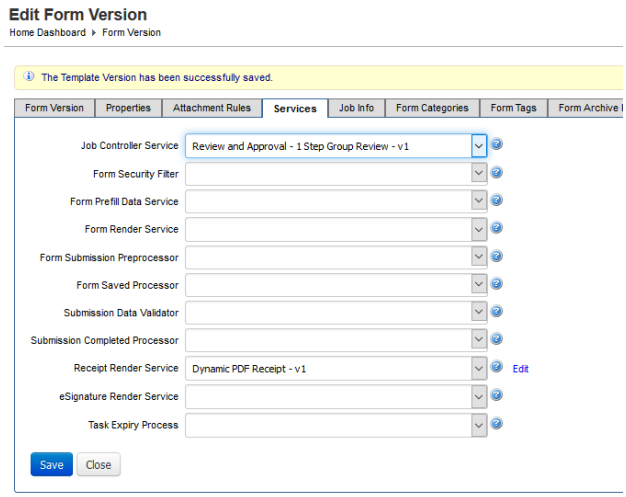
Job Templates

Since TM 4.2.0.

Job Templates are an enhanced form of Job Controller Service that allows a user select and customise a Collaboration Job without having to modify the JSON definition. This allows for:

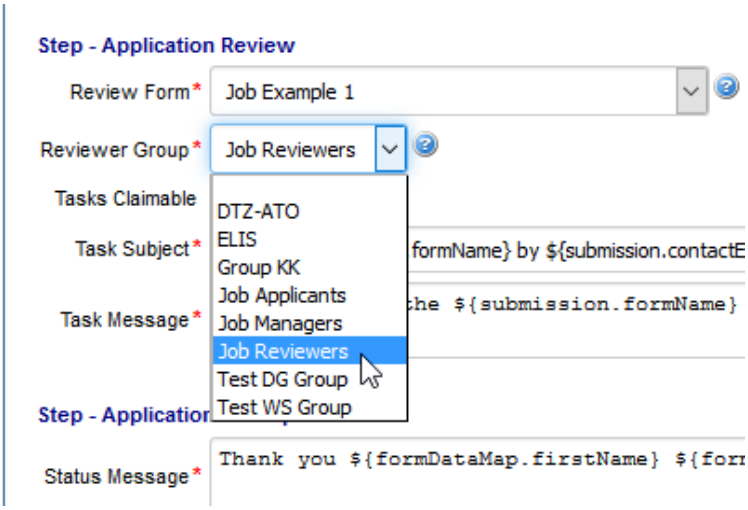
- Business Users and Forms Developers to configure their forms without a developer.
- Greater reuse of a single Job Controller.

A user select the Job template in the Form Version services tab and then clicks Save

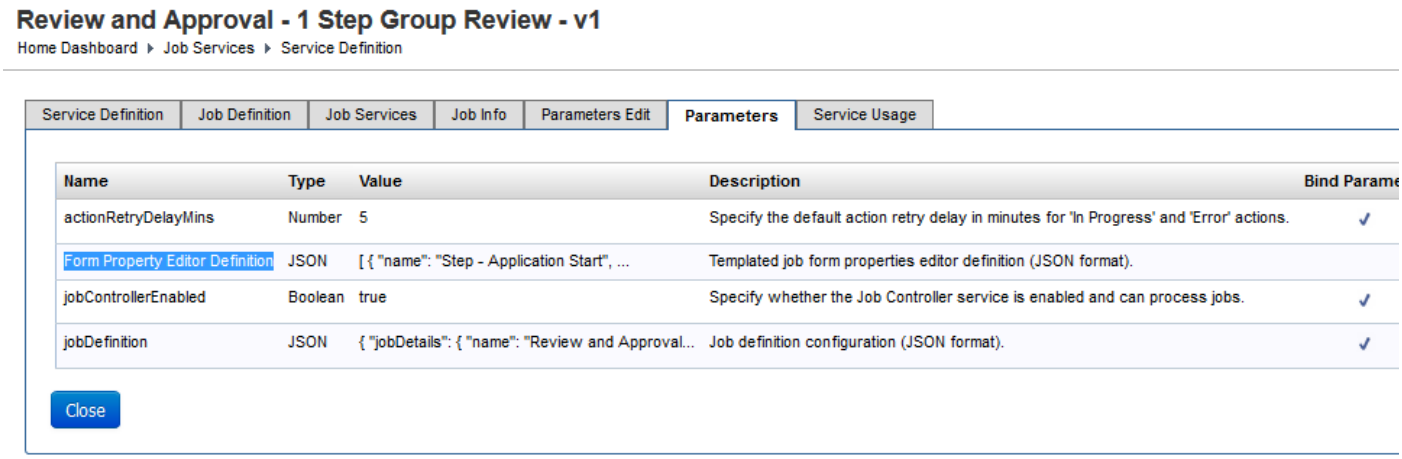


The Job Properties page is displayed which allows the user to customise values.

In this case I will keep all the defaults bug set the Application Review: Review Form and Reviewer Group by selecting an vales in the dropdowns.



A Job Template has an extra service parameter "Form Property Editor Definition" which is a JSON file containing meta data setting for the form properties.



Actions Services

When an action is processed an action service is run. The Job will usually specify the action service from one of the collaboration Jobs Inbuilt Action Services. The framework allows a developer to create their own Job Action and Job Expiry services in Groovy.

Standard Actions Services

The following table lists the Standard Action Services built into Transact Manager. It lists the Job Action Name, the associated Java Class and gives a summary description of what it does.

Action Type	Job Action Class	Description
Job Delivery	**JobDeliveryService**	This is used in conjunction with the Job Delivery Wait service. It is used to kick off the standard Transact Delivery process. The delivery process is completed by the Transaction Processing Scheduled Job not the job controller service.
Job Delivery Wait	**JobDeliveryWaitService**	See above. The Job Delivery Wait service is an asynchronous mechanism (see link below). When the delivery is completed by the Transaction Processing job it calls a callback method that allows the Job Delivery Wait to complete.

Job Form Start	**JobFormStartService**	This action service associates the initial form submission with the initial step. It can add a submission processing status messages to the initial Submission. These appear in the user's submission history. It can also send a status update via email.
Job Process Message	**JobProcessMessageService**	This service will add 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.
Job Receipt Wait	**JobReceiptWaitService**	This action polls the submissions until the receipts have completed. The generation of the receipts is completed by a separate Transact 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. Note: This is required as the existing Job Task Wait service does not wait for receipts to be created.
Job Task Assign	**JobTaskAssignService**	The Job Task Assign Service is used for creating / assigning tasks in Transaction Manager has a lot of configuration properties. These properties are either 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 categorised as: ◦ Standard Tasks: to an authenticated user or 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 TM 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.
Job Task Wait	**JobTaskWaitService**	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.

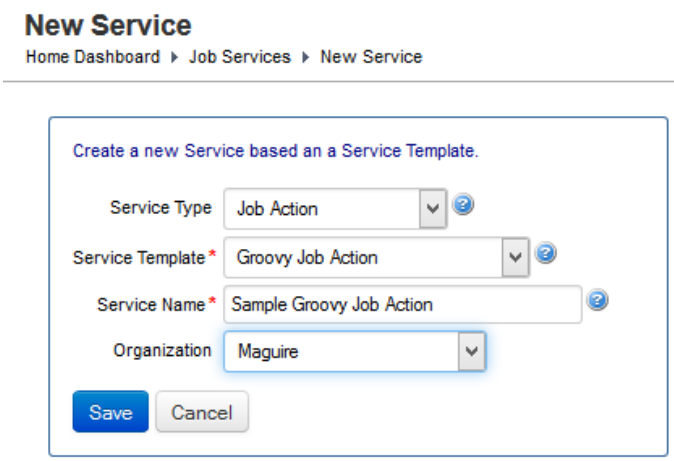
Custom Action Services

The framework allows a developer to define a Job Action and Job Expiry services as a Groovy Script that perform some custom processing.

Job Action

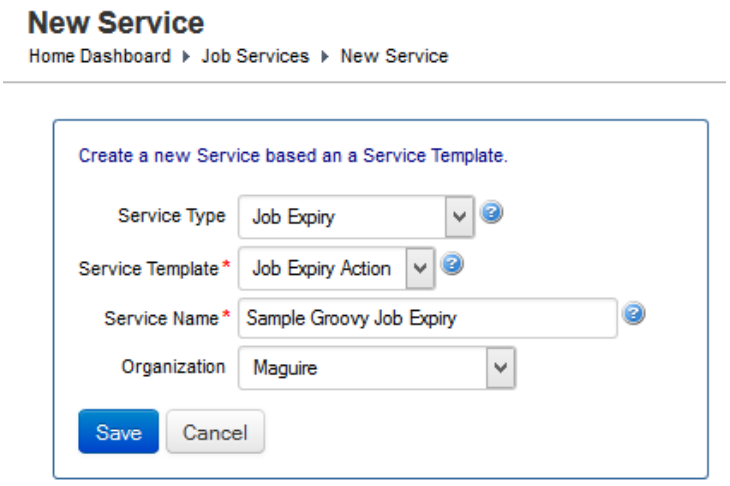
The majority of Jobs will be able to be created using the standard action services. The framework allows you to create a job action in Groovy using API's available to transaction manager. An example where a Groovy Job Action was created to show receipts from the previously submitted Form Bundle Task as inline attachments in the form. The step created the Task using the standard Job Task Assign service. It then runs a Groovy job action that retrieves the previous submissions, gets the receipts and attached them to the form.

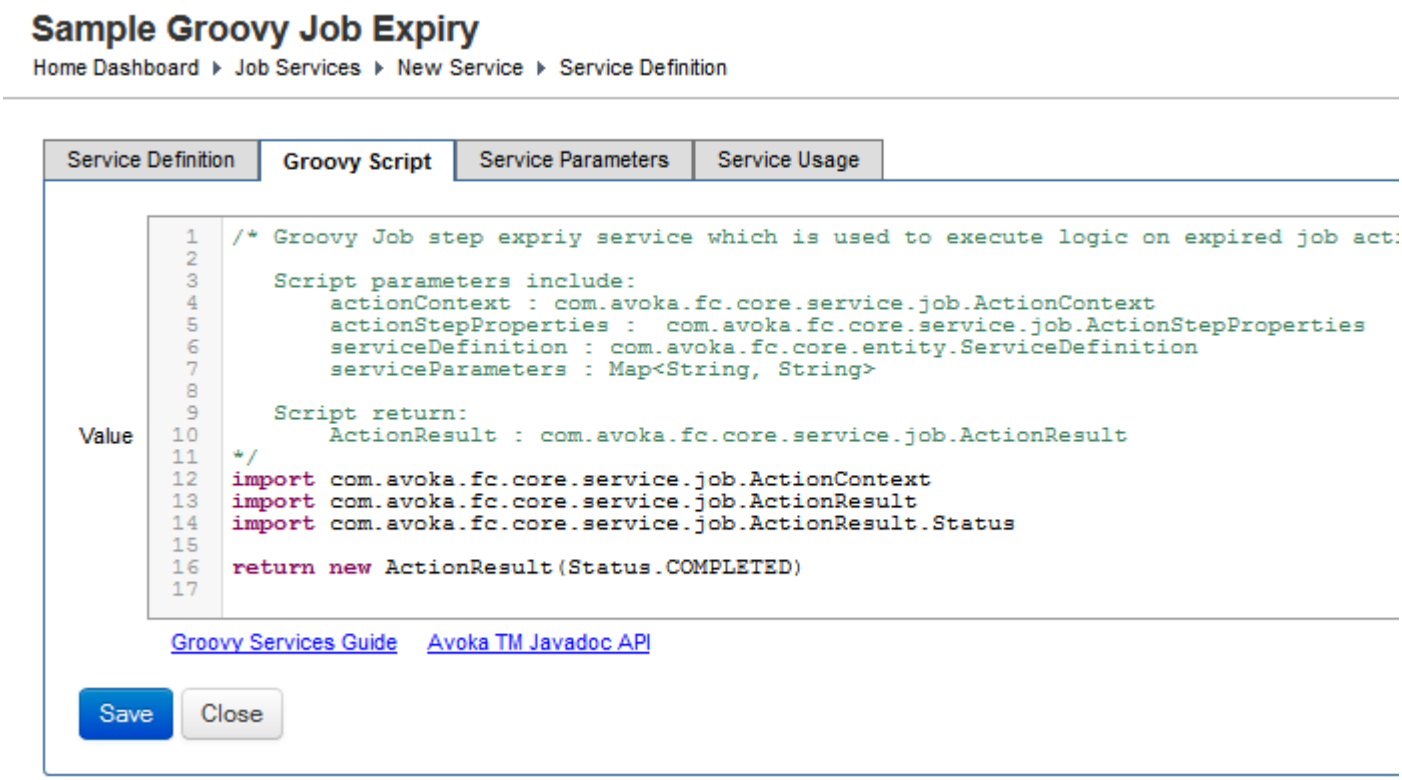
A new Groovy Job Action has been created below.



Job Expiry

A Job Expiry service is used to perform some clean-up action on a step before it routes the job to the expiry step.

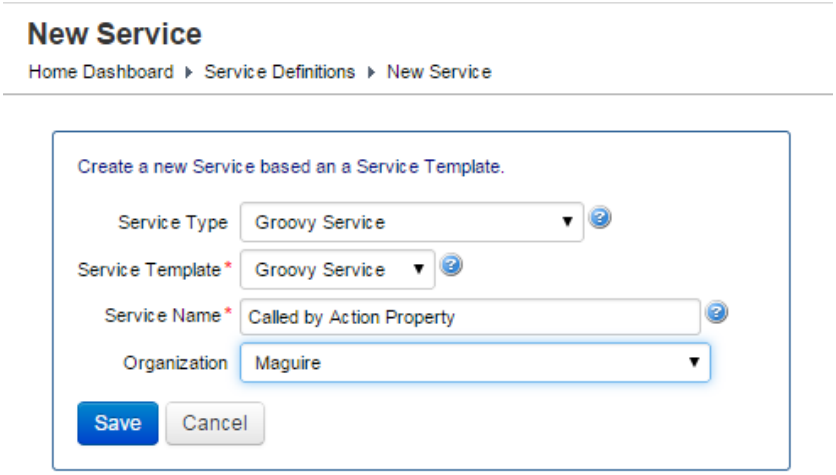




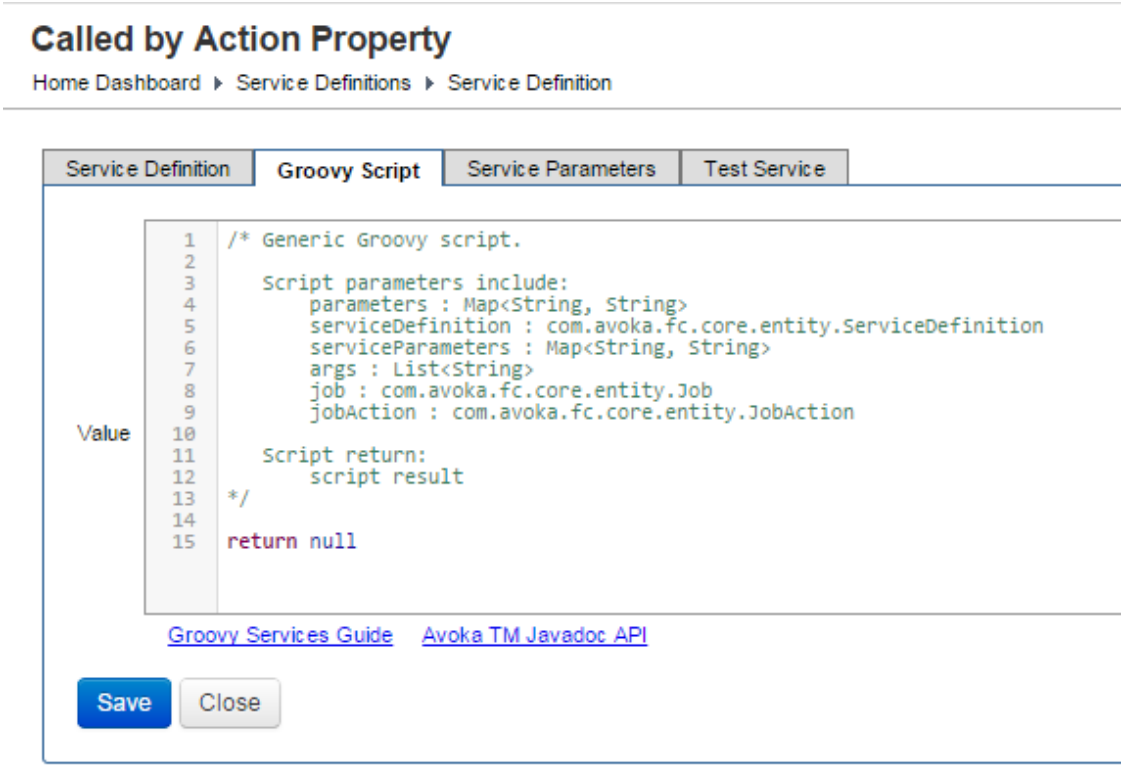
Action Property - Groovy Services

Action property can invoke a Groovy Services when they are evaluated. This is detailed in the Action Properties section below.

The screenshot below shows how to create a Groovy Service.



The default template for a Groovy Service is shown here. Note Groovy Services are a generic service used in other parts of Transaction Manager. If they are called by an Action Property the args, job an jobAction object are set.



There are a number of these services in the Advanced Example page

Job Definition Structure

Jobs are defined in a single text JSON format text structure. This section of the document details the JSON attributes at the Job Definition, Step Definition and Action Definition. It also describes the Action Properties and how they are used.

Job Definition

The following is an example of the structure of a job with the detail for the step definition removed.

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

The following describes the first level job attributes:.

jobDetails

This **jobDetails** is where information about a job and its processing are contain. It has the following sub attributes:

name:

This is the name of the job, it is set by the Job Controller service when the job has been saved.

processSubmitImmediate: (Optional)

if this attribute is omitted or false the Job will use Standard Mode processing as described in the Job Execution section

If the value is set to to "true" it will use the Process Immediate Mode

version:

The version of Transaction Manager that the job was created in.

properties

(Optional) Please see the Action Properties section below for more details

steps

The steps is an JSON array that holds the step objects defined in the next Step Definition below

Step Definition

The following JSON shows the job definition for a Review step with the Action Definition removed.

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

The following describes the first level job attributes:

name

The name of the step. Step Names are required to by unique within a Job.

type

Valid Values for the type are :

- **start** which indicates the first step for the job
- **endpoint** when the processing reaches a endpoint step the processing stops and the job completes. Endpoints do not specify actions, routes or customProperties.
- otherwise empty string.

example valuess

"type":"start"
"type":"endpoint"
"type":""

expiryRule

(Optional) The step scheduled completion time (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

example values

"expiryRule" : "+4h" **(hours)**

"expiryRule" : "+5d" **(days)**

"expiryRule" : "+4w **(weeks)**

Since 4.3.1 you can do minutes
"expiryRule" : "+4m"

If an expiryRule is used but and exprityServiceName is not specified then **Expiry** route must be specified.

expiryServiceName

(Optional) Allows the step to specify a **Job Expiry** service for the purpose of running clean up up any items such as incomplete submissions.

An **Job Expiry** Service should specify a route. This route name can be, but doesn't have to be **Expiry.**

If you are expiring a Step that has tasks that have not been completed them you will need to run a Job Expiry Action that cleans these up. The code below is an example that will clean up tasks that were not submitted.

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

actions

Required for all steps except **endpoints**.

It is a JSON array that holds the action objects defined in the Action Definition section below. The step processing runs the actions in sequence.

routes

Required for all steps except **endpoints**.

It is a JSON array that holds the route mapping between the route name and the next step. A route has the following attribute:

name:

This is the route name.

Note there are some special route names:

Default: If the step processing does not find a match with a route name or if the route name is empty, it will use the Default route.

Expiry: This is the route taken if the step has expired.

nextStep:

This has to be set to either:

- A valid step name contained in the job.
- A special case is **##PREVIOUS_STEP** the job processing assign the next step as the previous step name.

display: (Optional)

The Collaboration Job provides a list of available routes to the form please see Form Design (CollabJobs v4.3). Setting "display": "false" removes the route from the available routes.

properties

(Optional) Please see the Action Properties section below for more details

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

Step Definition - Bundle Specific

The following JSON shows the job definition for a Bundle Step with the Actions removed

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

The following describes the first level job attributes specific to form bundles:

shareFormData

since 4.1

If true

shareExtractData

since 4.3

> **Note**

allFormsEditable

since 4.3

showPreviousForm

since 4.3

redirectNext

since 4.3

Action Definition

The block below shows and Job Task assign.

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

The action attributes are discussed in the sections below.

name

The name of the action.

The following actions show in the JSON block below are used to demonstrate how the Job Controller loads the correct Action Service.

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

The Job Controller first try's to load the action service with the same name as the action.

The **Create Submission User Account** is a Groovy Job Action so that service that will be loaded and executed.

If a match was not found it will use the default **Job Action Service** for the type (attribute below).

> The Customer Submission loads the standard action service **Job Form Start**. This will run the **JobFormStartService** as per the Standard Actions Services above.

type

The Action Service Type

Valid action types are:

- listed above in the Standard Actions Services section in the table under **Action Service Type** heading.
- groovy action services can have a type of **Job Action**

properties

(Optional) Please see the Action Properties section below for more details

The actions defined in the block below show the definition of a standard action

Action Properties

There are name value pairs that are used to pass information into an instance of the Action Service.

Action Properties can be defined at the Action, Step and Job level, refer to the code job definition below. 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.

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

The Action properties have the following attributes.

name:

> The predefined Action Services like the Job Task Assign service have a set number of Action Properties (TODO Please refer here)

value:

> The value is an Apache Velocity Template. Velocity templates are used extensively throughout Transact to 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**
>
> The action properties are templates that are evaluated when an Action Service is run. The templates can be a static text value like below:
>
> ```
> { "name": "Task Email Message", "value": "An Email Message" },
> ```
>
> Or they can substitute values from the model like below:

```
    { "name": "Process Message Text", "value": "Your ${job.submission.formName} application is at the Initial
Review Step" }
```

> The Model Elements section below details the model elements that are associated Action Properties. Both properties and methods can be evaluated on an object.

template: (Optional, Advanced, Boolean)

> If **True** it evaluates the value as a velocity template which forces the return value to return a String. Using this can evaluate nested function calls.
>
> This is can be used with property values that start with "$func", these can return an object. eg $func.startUser() will return a user.
>
> ```
> { "name": "Task Email Message", "value": "$func.getSubmission().formXml()", "template": "true" },
> ```

Model Elements

$formDataMap

This holds all the Form Data Extracts, for more detail please refer to Form Design (CollabJobs v4.3) page.

To access the first name of a user entered into a form you could us **$formDataMap.firstName**. An example of a

Note:

- The collaboration job processing automatically adds **Route Name** is automatically added to be available in the data extracts: **$formDataMap.routeName**
- Where a submission is not defined for an Action Service, it will will use the data extracts from the previous or the start submission.

Example:

```
   { "name": "Task Subject", "value": "Initial review of ${job.referenceNumber} from ${formDataMap.firstName}
${formDataMap.lastName}" },
```

$job

This holds the Job entity instance. see Job in the API:

${job.name} is the job Name
${job.referenceNumber} is the Job Number
${job.submission.formName} gets the form name of the first submission

Examples:

```
    { "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 holds the current JobAction entity instance.

From this we can get the current step name $jobAction.step.name

Example:

```
    { "name": "Task Subject", "value": "${jobAction.name} of ${job.referenceNumber} from ${formDataMap.firstName}
${formDataMap.lastName}" },
```

$func

This provides various functions for there is a list **JobFunctions** in the API .

$func.formProperty('Initial Review') gets the the form or organisation property value initial.

$func.invoke('Groovy Service Name', arg1, arg2) can invoke a groovy service

Examples

```
    { "name": "Task Form Code", "value": "$func.startFormCode()" }

    { "name": "Task Assign Group", "value": "$func.formProperty('Initial Review')" }
```

Process Immediate Mode