Avoka Transact

Collaboration Jobs - Current Release Documentation

Collaboration Jobs v17.10

This documentation is related to the latest Collaboration Jobs version.

Looking for an earlier Collaboration Jobs version? Please see Collaboration Jobs - Previous Release Documentation for more information.

Search Collaboration Jobs Documentation

Documentation Content

Recently Updated

Transact Manager Form Configuration
Mar 20, 2019 • commented by Chad Bakeman

Maestro
Mar 20, 2019 • commented by Chad Bakeman

Maestro
Mar 20, 2019 • commented by Chad Bakeman

Chaining and Form Bundles
Oct 19, 2018 • commented by Anonymous

Customer Onboarding Job - Form Bundle
Aug 21, 2018 • commented by Sindy Tang

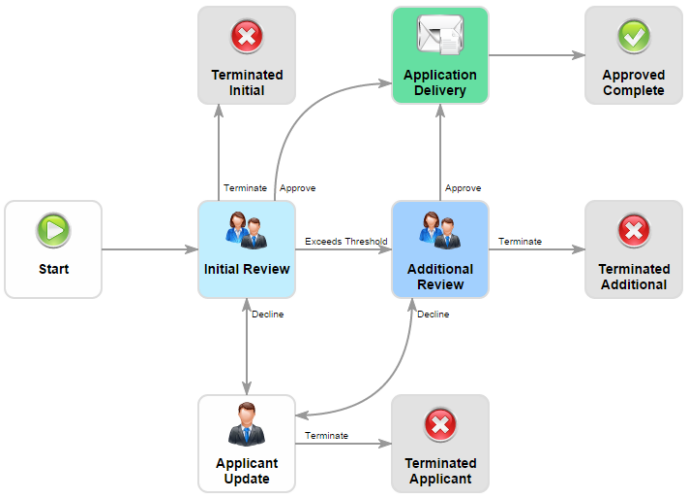
Avoka Transact Collaboration Jobs User Guide

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 Documentation for information relating to the specific version of Transaction Manager.

Job Basics

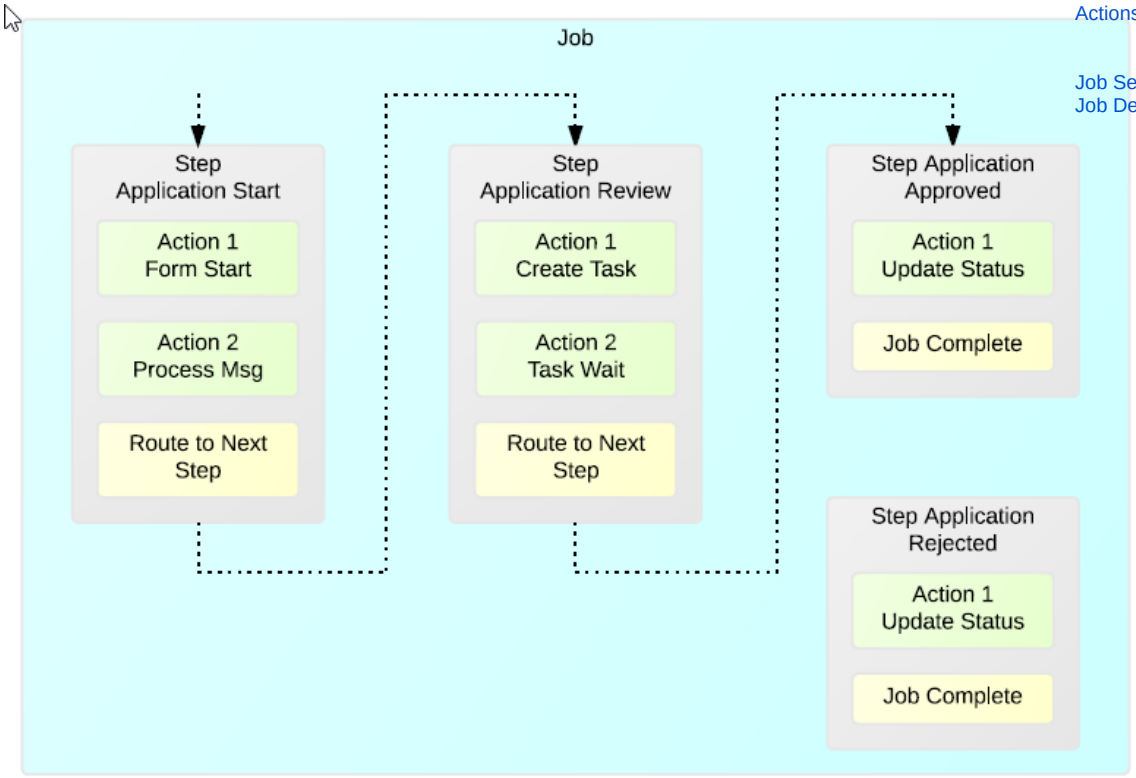
Building Blocks

Job Composition

Page Contents:

Job, Step, Action Hierarchy

A **Job** contains 2 or More **Steps**, each **Step** contains one or more **Actions** as shown in 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).



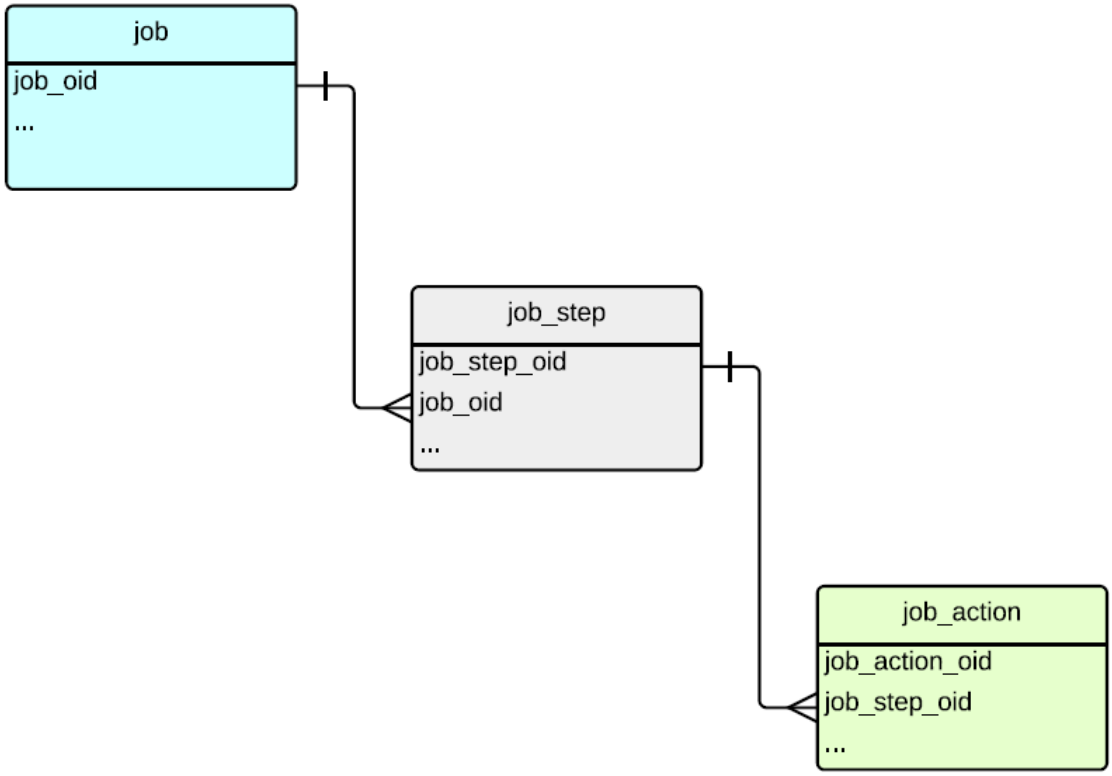
Job Instances

Let's 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 **For m 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.

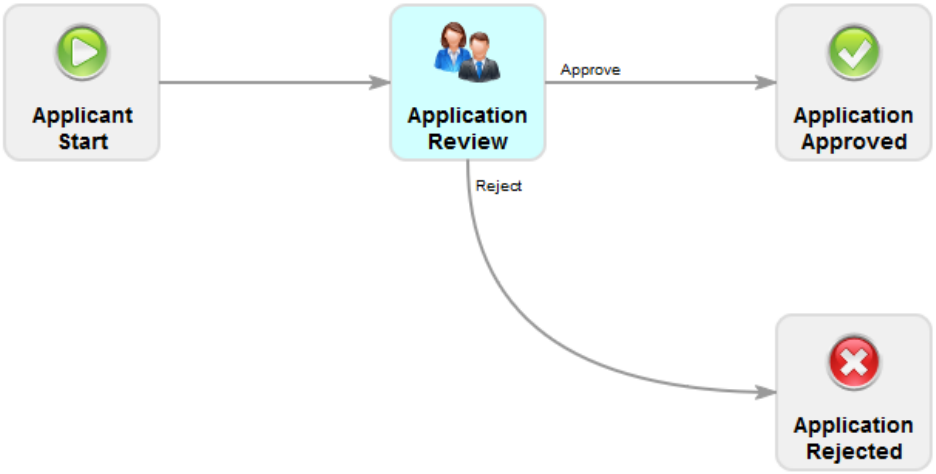
Job Definition (JSON)

```
{
  "jobDetails": {
    "name": "1 Step Job",
    "processSubmitImmediate": "false",
    "version": "4.0.7"
  },
  "steps": [
    {
      "name": "Applicant Start",
      "type": "start",
      "actions": [
        {
          "name": "Accept Quote",
          "type": "Job Form Start",
          "properties": [
            { "name": "Process Message Send Email",  "value": "true" },
```

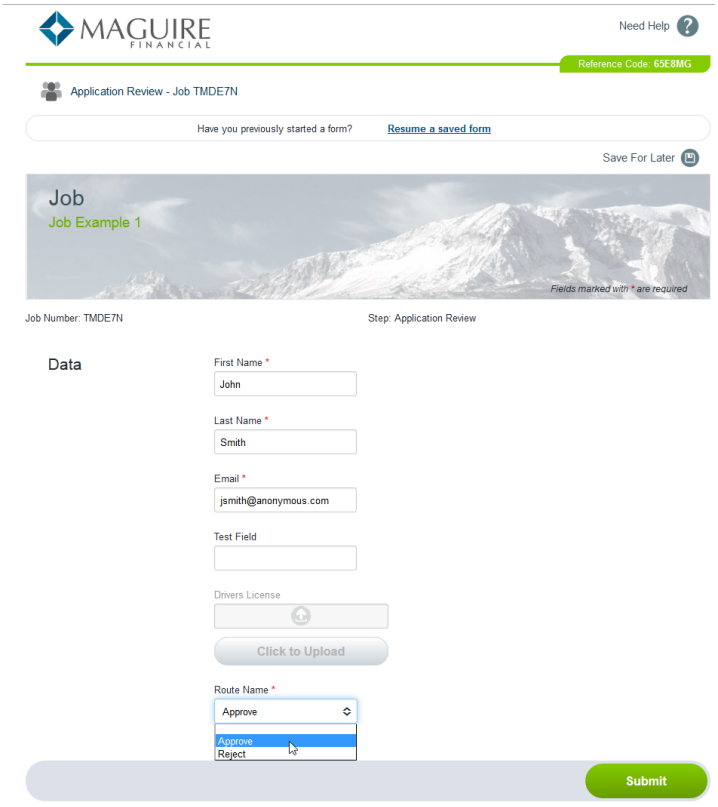
```
            { "name": "Process Message Text",  "value": "Thank you
${formDataMap.firstName} ${formDataMap.lastName} your ${submission.
formName} is being processed." }
          ]
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Review" }
      ]
    },
    {
      "name": "Application Review",
      "type": "",
      "actions": [
        {
          "name": "Assign Review",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Task Assign Group",  "value": "Job Reviewers" },
            { "name": "Task Form Code",  "value": "insurance-review" },
            { "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 ${submission.contactEmailAddress}." }
          ]
        },
        {
          "name": "Review Wait",
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Approved" },
        { "name": "Reject",  "nextStep": "Application Rejected" }
      ]
    },
    {
      "name": "Application Approved",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "true" },
            { "name": "Process Message Submission Step",  "value":
"Application Start" },
            { "name": "Process Message Text",  "value": "Thank you
${formDataMap.firstName} ${formDataMap.lastName} your ${submission.
formName} has been Approved." }
          ]
        }
      ]
    },
    {
      "name": "Application Rejected",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "true" },
            { "name": "Process Message Submission Step",  "value":
"Application Start" },
            { "name": "Process Message Text",  "value": "Sorry
${formDataMap.firstName} ${formDataMap.lastName} your ${submission.
formName} has been declined." }
          ]
        }
      ]
    }
  ]
}
```

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

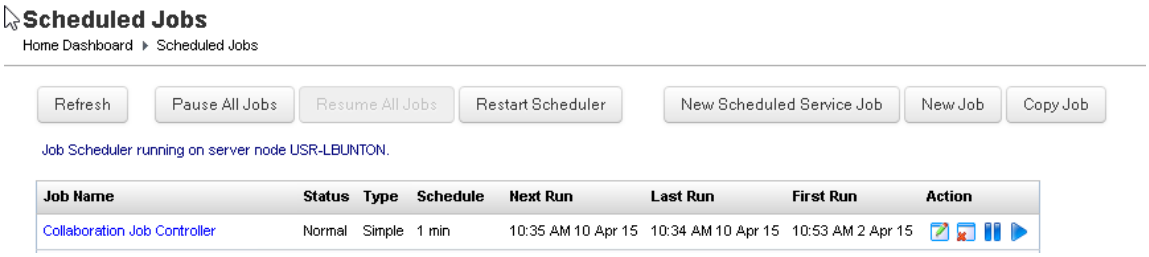
- **In Progress** - the job is active and in progress
- **Completed** - the job has been completed. This happens when a endpoint step has completed.
- **Expired** - this happens when an Endpoint expiry job has been automatically expired by the system

- **Cancelled** - the job has been manually cancelled by an administrator.
- **Error** - there is a 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 Action Services 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

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 (nonproduction)
- 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

The Process Immediate Mode is used by Form Bundles such as customer onboarding forms. For example, a new banking customer fills out an initial form which has some common fields and a list of products 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. Unknown macro: 'multiexcerpt'

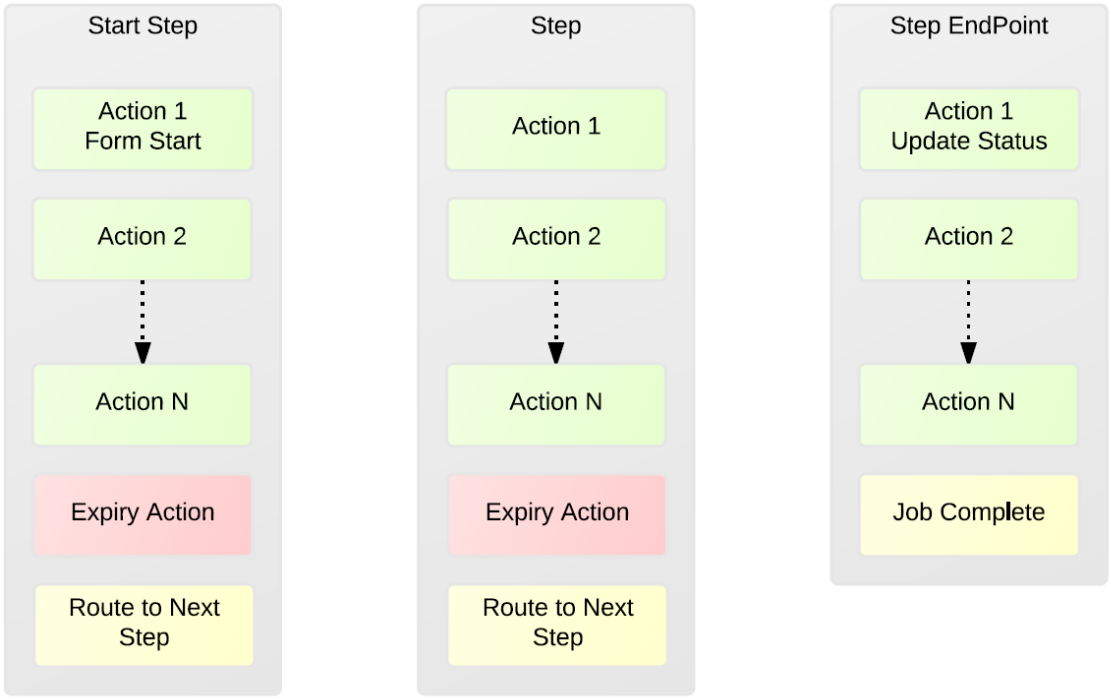
Step

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 testingnewly 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**.

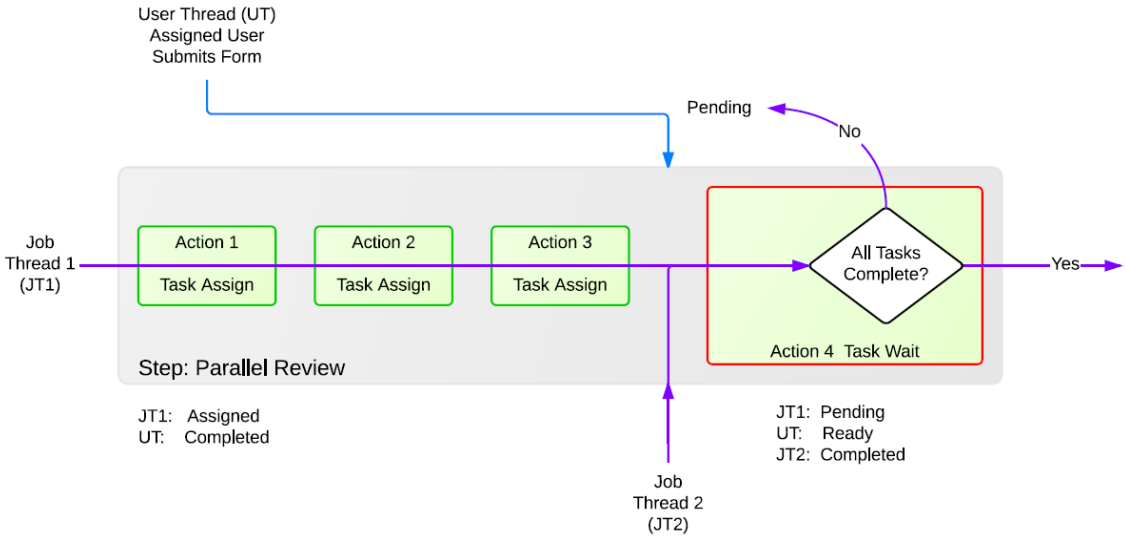
Custom job actions written in Groovy can be programmed to specify a the next run time e.g. now + 20 min.

Unknown macro: 'multiexcerpt'

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

Unknown macro: 'multiexcerpt'

Job Services

For information on Job Services please see Job Services - Standard and Custom Services.

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.

Review And Approval

```
{
  "jobDetails": {
    "name": "2 Step Review Job",
    "processSubmitImmediate": "true",
    "version": "4.1.0"
  },
  "steps":[
        { .. step definition ...},
    { .. step definition ...},
    ...
  ]
}
```

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 followingsub 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 Transact Manager that the job was created in.

jobGroups

The **jobGroups** lists a set of job groups allowing job coordinators to manage this job. For each applicable job group, the following sub attributes need to be defined:

name:

This is the name of the job group. This name, when applicable, will be displayed in the *Job Groups* dropdown from the *Reviews* tab in the form space.

description:

This is a suitable description for the job group.

Manage Job Groups:

There are two ways that job groups can be created:

- If a job group defined in a job definition, as in above, has not been created, then the job group will be automatically created first time that a user submits a form associated to this job. This is the preferred way to create job groups.
- Job groups may be manually created from the *Security* > *Groups* option. The job group is a group of type, *Job*. By manually creating the job group you must ensure that the job group names and descriptions match with the job definition. Any mismatch will not be subsequently updated to the job groups.

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.

Review Step

```
{
      "name": "Initial Review",
      "type": "",
      "expiryRule": "+5d"
      "expiryServiceName": "An Expiry Service"
      "actions": [
        { ... Action 1 Definition ... },
        { ... Action 2 Definition ... }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Exceeds Threshold",  "nextStep": "Additional Review",
"display": "false" },
        { "name": "Terminate",  "nextStep": "Terminated Initial" }
      ]
},
```

The 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.

Job Expiry Action Service

```
/* 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
```

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. Setting "display": "false" removes the route from the available routes.

properties

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

Normal Step

```
{
      "name": "Initial Review",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Send Email",  "value": "true" },
            { "name": "Process Message Submission Step",  "value": "Start"
},
            { "name": "Process Message Text",  "value": "Your ${submission.
formName} application is at the Initial Review Step" },
            { "name": "Task Assign Group",  "value": "$func.formProperty
('Initial Review')" },
            { "name": "Task Message",  "value": "Please perform the initial
review of the ${submission.formName} from ${formDataMap.firstName}
${formDataMap.lastName}." },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "Initial review of
${submission.formName} from ${formDataMap.firstName} ${formDataMap.
lastName}" },
            { "name": "Task Type",  "value": "Review" },
            { "name": "Task Send Email",  "value": "true" },
            { "name": "Task Email Message",  "value": "An Email Message" },
            { "name": "Task Email Subject",  "value": "An Email Subject" }
          ]
        },
        {
          "name": "Handle Submission",
          "type": "Job Task Wait",
          "properties": [
            { "name": "Conditional Route Name",  "value": "#if (
$formDataMap.routeName == 'Approve' && $formDataMap.loanAmount >= 20000 )
Exceeds Threshold #end" }
          ]
        }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Exceeds Threshold",  "nextStep": "Additional Review",
"display": "false" },
        { "name": "Terminate",  "nextStep": "Terminated Initial" }
      ]
    },
```

Action Definition

The block below shows and Job Task assign.

Task Action with Action Properties

```
{
      "name": "Create Task",
      "type": "Job Task Assign",
      "properties": [
        { "name": "Process Message Send Email",  "value": "true" },
        { "name": "Process Message Submission Step",  "value": "Start" },
        { "name": "Process Message Text",  "value": "Your ${submission.
formName} application is at the Initial Review Step" },
        { "name": "Task Assign Group",  "value": "$func.formProperty
('Initial Review')" },
        { "name": "Task Message",  "value": "Please perform the initial
review of the ${submission.formName} from ${formDataMap.firstName}
${formDataMap.lastName}." },
        { "name": "Task Review Previous Step",  "value": "true" },
        { "name": "Task Subject",  "value": "Initial review of ${submission.
formName} from ${formDataMap.firstName} ${formDataMap.lastName}" },
        { "name": "Task Type",  "value": "Review" },
        { "name": "Task Send Email",  "value": "true" },
        { "name": "Task Email Message",  "value": "An Email Message" },
        { "name": "Task Email Subject",  "value": "An Email Subject" }
      ]
    },
```

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.

Actions showing a call to a Standard Action and a call to a Groovy Script Job Action

```
  "actions": [
    {
      "name": "Customer Submission",
      "type": "Job Form Start"
    },
    {
      "name": "Create Submission User Account",
      "type": "Job Action"
    },
  ],
```

The Job Controller first try's to load theactionservice 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 **JobF ormStartService** 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 arename valuepairs 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.

Actions showing a call to a Standard Action and a call to a Groovy Script Job Action

```
{
  "jobDetails": { ... },
  "properties": [
    { "name": "Task Form Code", "value": "$func.startFormCode()" }
  ],
  "steps":[
    {
      "name": "Initial Review",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Text",  "value": "Your ${submission.
formName} application is at the Initial Review Step" },
            { "name": "Task Assign Group",  "value": "$func.formProperty
('Initial Review')" },
            { "name": "Task Message",  "value": "Please perform the initial
review of the ${submission.formName} from ${formDataMap.firstName}
${formDataMap.lastName}." },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "Initial review of
${submission.formName} from ${formDataMap.firstName} ${formDataMap.
lastName}" },
            { "name": "Task Type",  "value": "Review" },
          ]
        },
        { ... }
      ],
      "properties": [
        { "name": "Task Form Code", "value": "$func.startFormCode()" }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" }
      ],
    },
    { ... }
  ]
}
```

The 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 definehtmlpage 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 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')" }
```

Form Design

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:

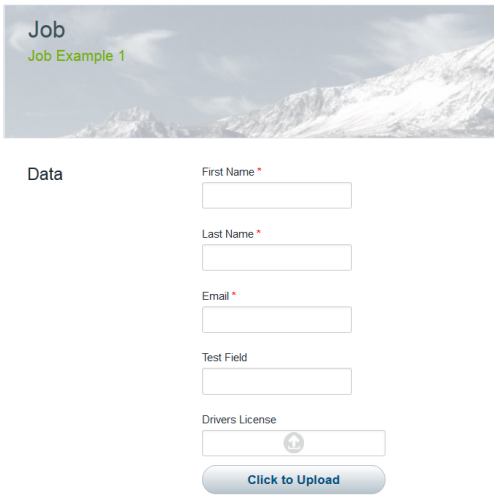
- Composer
- Maestro
- Transact Manager Form Configuration
- Task Assign Repeat Context (Advanced Topic)

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.

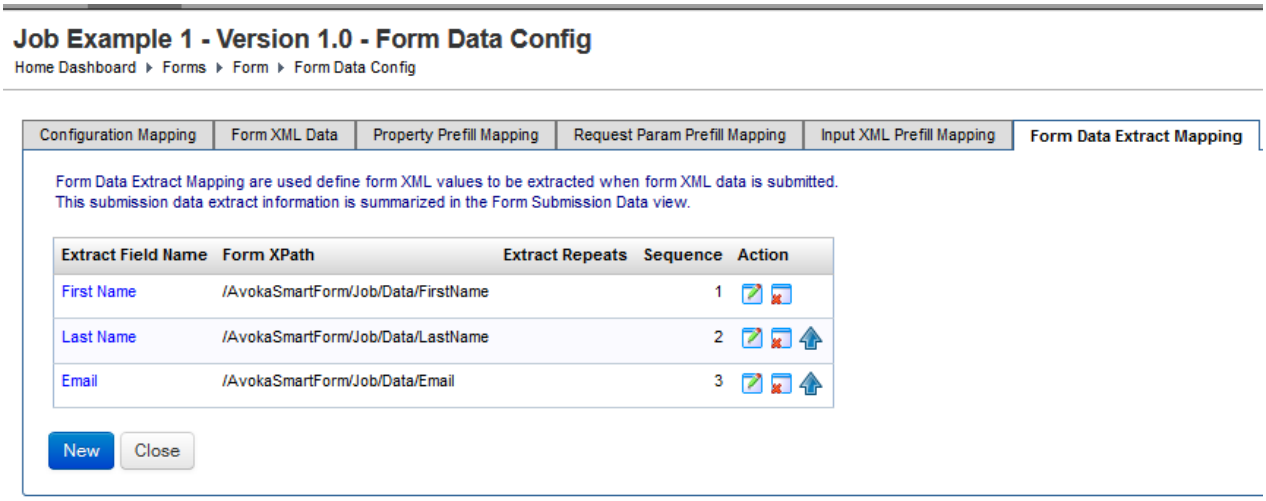
In this section we will use the Data Extracts setup in the Job Example 1 form below.



Transaction Manager

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

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

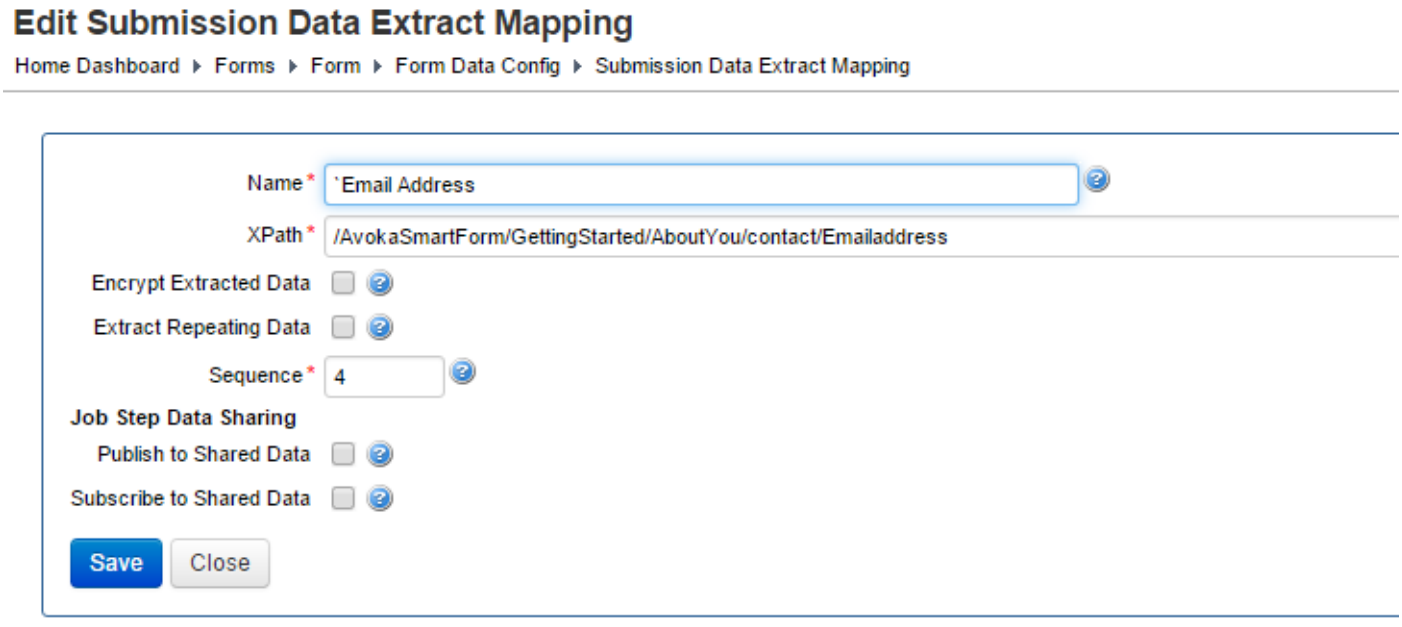


The following table shows the data extract field name, how the data extract is referenced in the Action Property Value and gives an example of the action property from the job definition (JSON). Note the use on camel case in when referring to the data extract **First Name** is reference as **$formDataMap.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

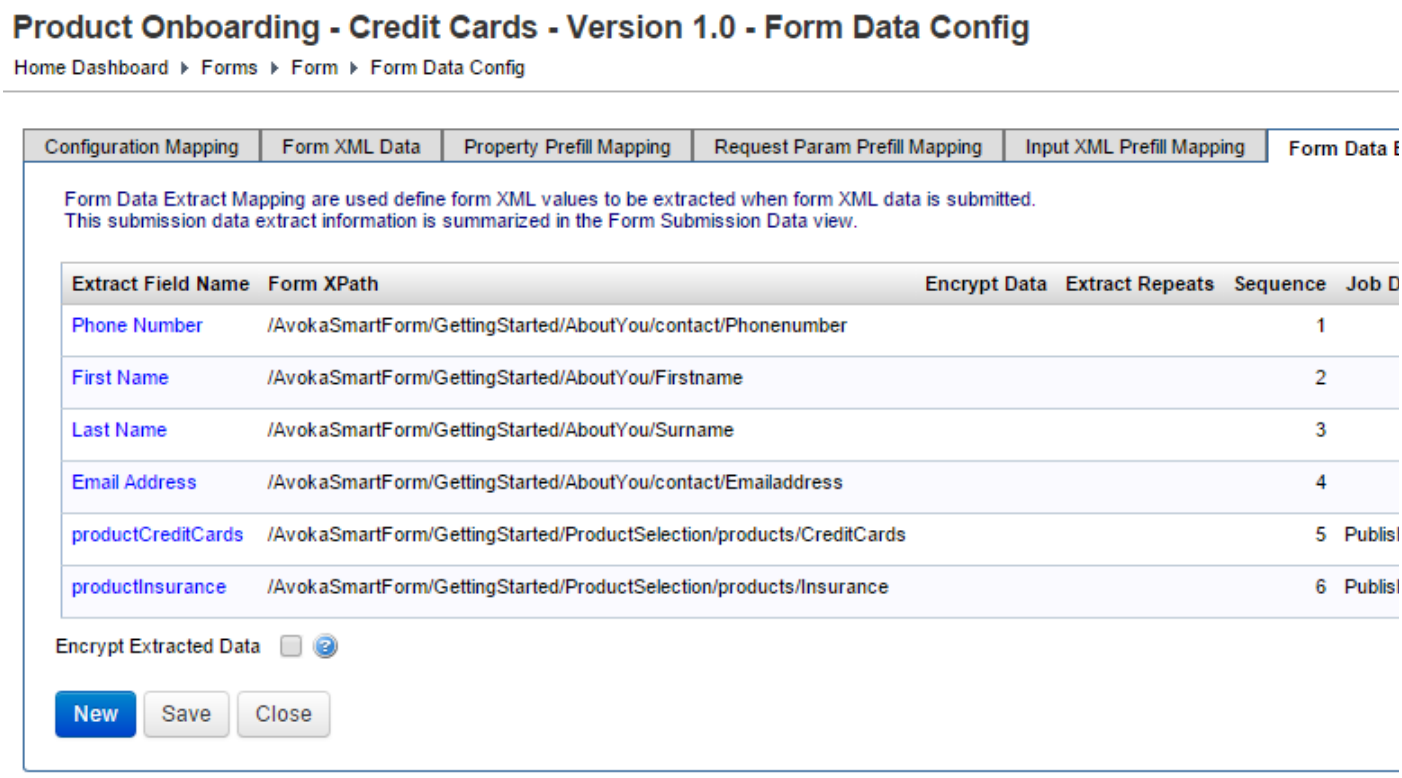
Data Extract Features

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 submisison data extracts
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

Form Xml Data

```
<?xml version="1.0" encoding="UTF-8"?>
<AvokaSmartForm>
    <SFMData>
        <SystemProfile>
            <DisplayMode>Receipt</DisplayMode>
            <HostContext>Page</HostContext>
            <ReceiptNumber>fct-3097-composer-3</ReceiptNumber>
            <SubmitDateString>12 Feb 2015 5:38:54 PM</SubmitDateString>
            <SubmissionMessage/>
            <FormDataServiceURL>http://localhost:9080/maguire/servlet/FormDynamicDataServlet</FormDataServiceURL>
            <RequestLogKey>9c1b9756b3958671c9b2471e23f91d08</RequestLogKey>
            <TrackingCode>NVKMQ9</TrackingCode>
            <Job>
                <AvailableRoutes/>
                <Assignee/>
                <AssignRepeatIndex/>
                <AssignRepeatItem/>
                <ReferenceNumber/>
                <StepName/>
                <RouteName/>
            </Job>
            <FormCode>fct-3097-composer</FormCode>
....
```

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 **Collaboration Jobs Step** `{ "name": "Initial Review", "type": "", "actions": [{ "name": "Create Task", "type": "Job Task Assign", "properties": [...] }, { "name": "Handle Submission", "type": "Job Task Wait", }], "routes": [{ "name": "Approve", "nextStep": "Application Delivery" }, { "name": "Decline", "nextStep": "Applicant Update" }, { "name": "Exceeds Threshold", "nextStep": "Additional Review", "display": "false" }, { "name": "Terminate", "nextStep": "Terminated Initial" }] }`	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): discuss the Assignee, AssignRepeatIndex and AssignRepeatItem nodes.
- Form Bundle Context:

Composer

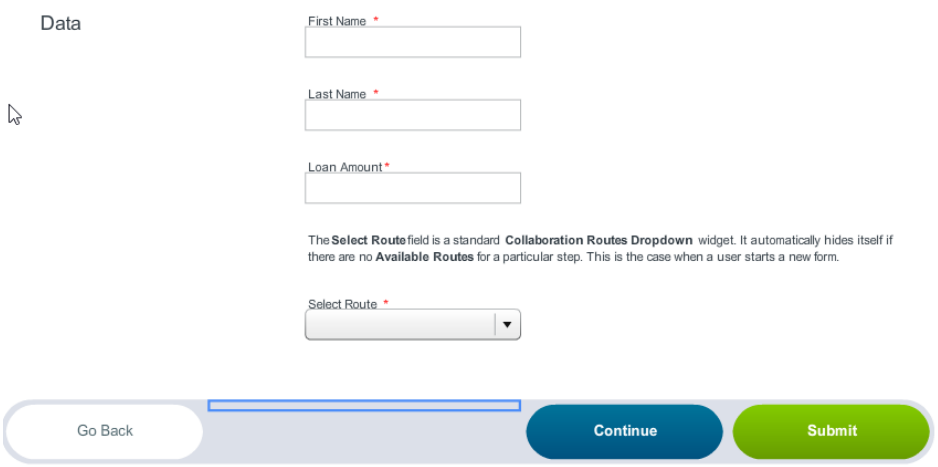
This page gives background on the design of Composer forms for Collaboration Jobs.
• How to setup Data Extracts
• Describe Collaboration Jobs widgets that come with Composer 4.1
• Setting Rules the Visibility and Editabilty of sections.

More advanced composer techniques are documented here:

- Anonymous Forms - Composer Save Challenge

Data Extracts

These are used by Collaboration Job to access form data as described Form Design. 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

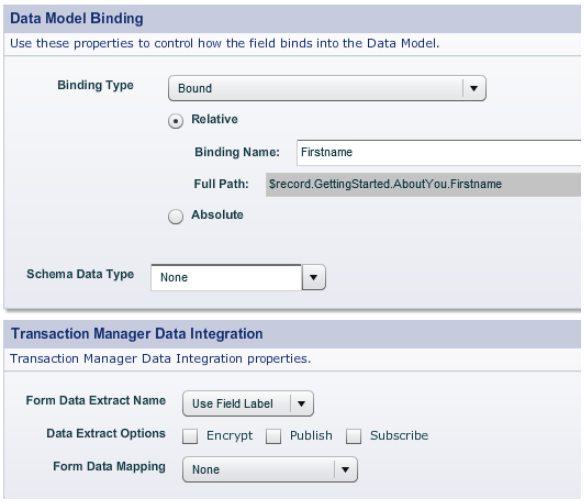
Collaboration jobs will put this into the $formDataMap.routeName values automatically.

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.

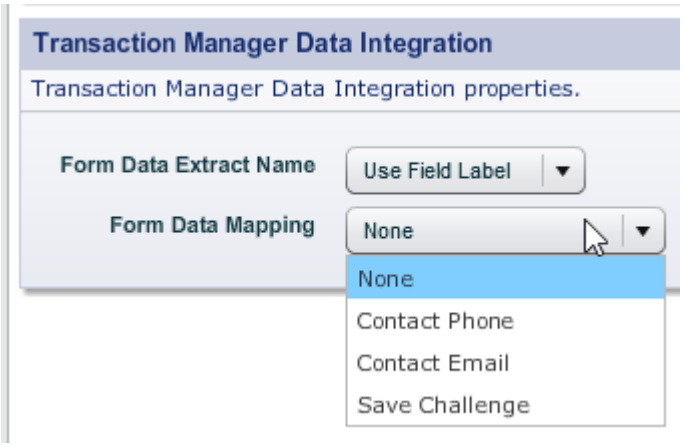
The easiest way to setup the Data Extract is to use the field label, however you can use an alternative name.

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



Form Data Mappings

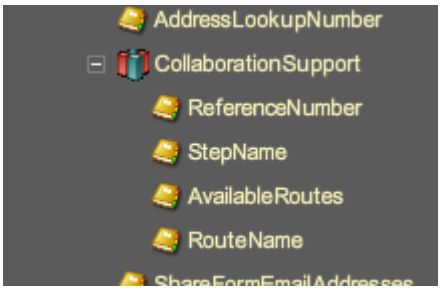
Fields can be mapped to predefined form data mappings as described here.



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.



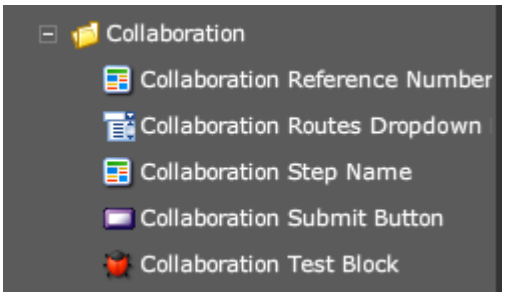
As describe the TM Job Controller sets the following XML data elements for each Task:
• **ReferenceNumber**: aka Job Number
• **StepName**:
• **AvailableRoutes**: This is a pipe delimited list of the current steps route names.

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.

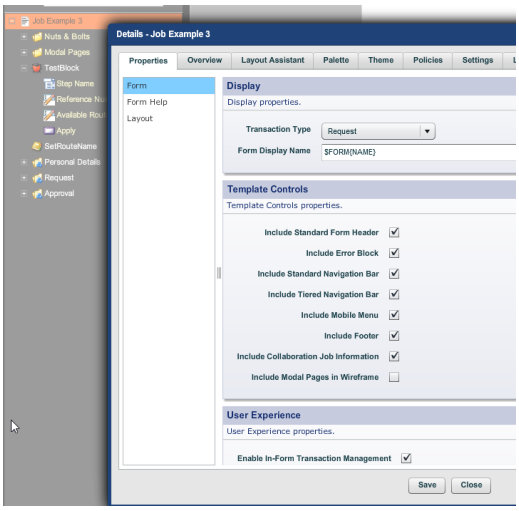


Collaboration Reference Number and Collaboration Step Name

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.



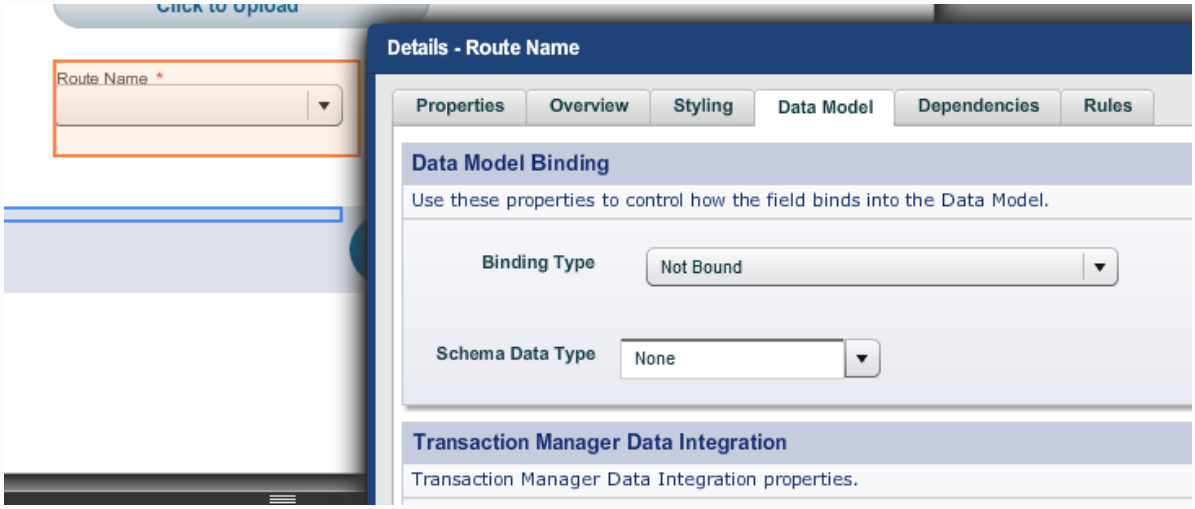
Collaboration Routes Dropdown List

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



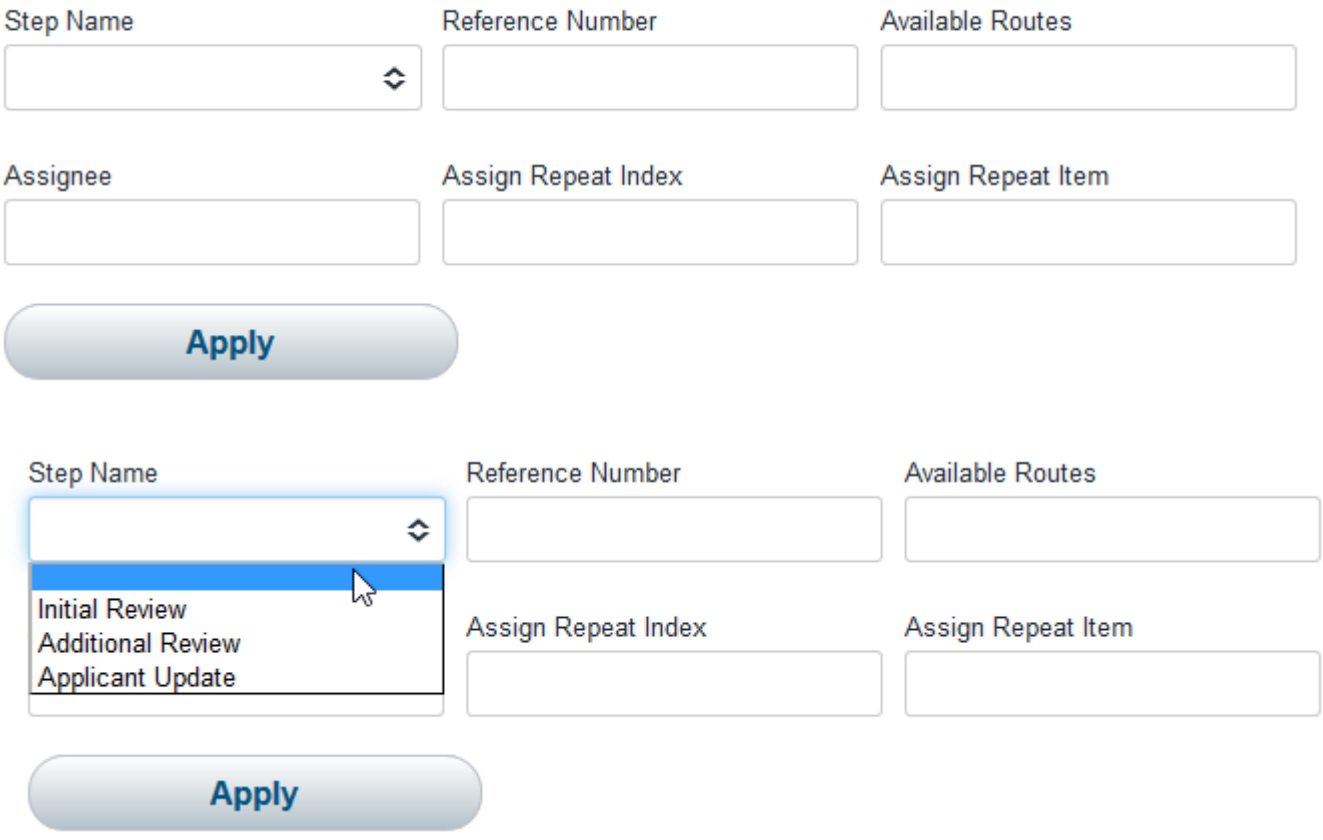
Collaboration Submit Button (Deprecated)

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.

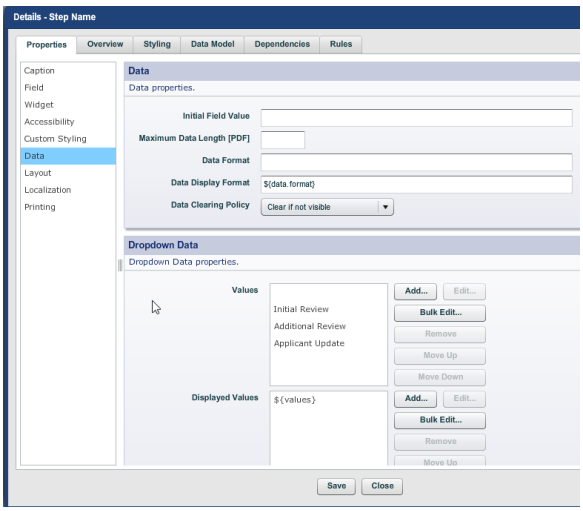
It was initially prototyped to create a submit button that sets it's label as a route.
We recommend that you either use the Collaboration Routes Dropdown or the standard Radio Button Group please see the **Routeing Using the Standard Radio Button Group** section below

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.



Routing 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.

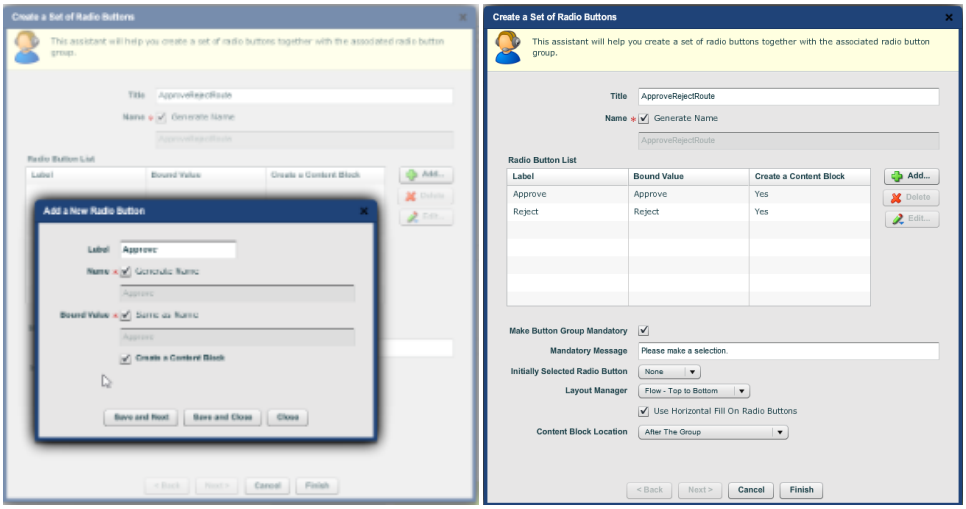
The advantages in using radio button groups:

- 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.

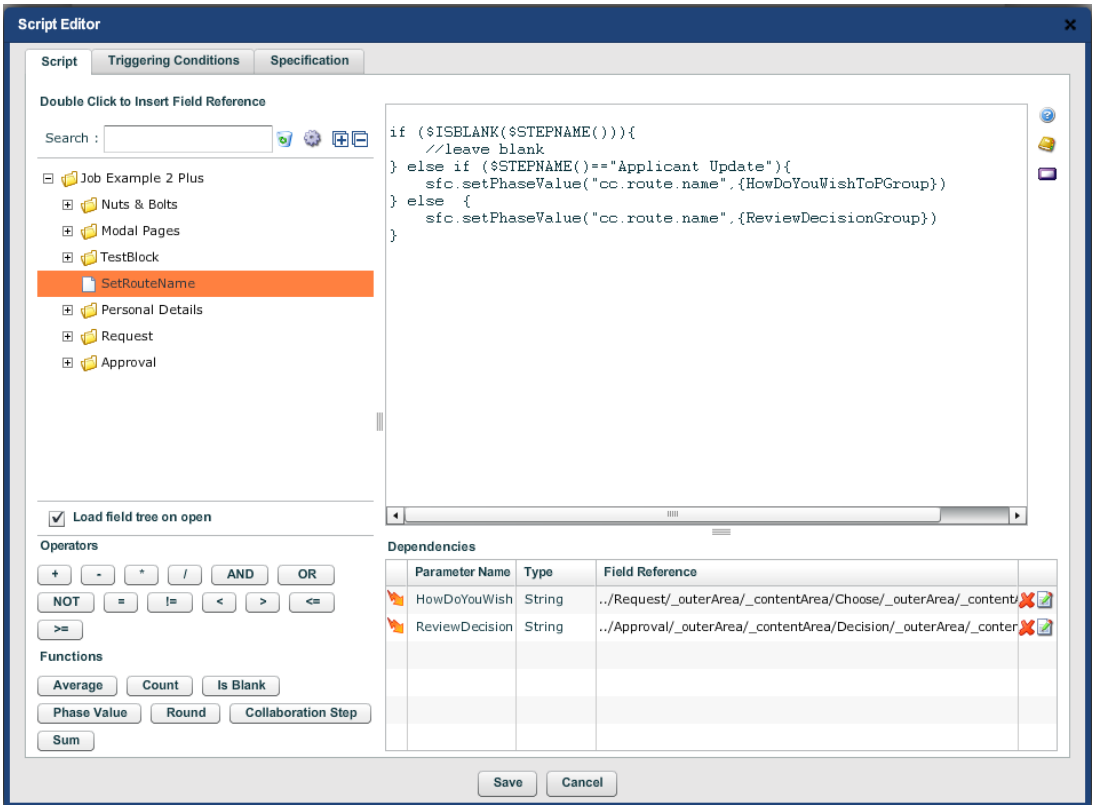


Business rules can be used to set the route as per the section below.

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

Job Example 2

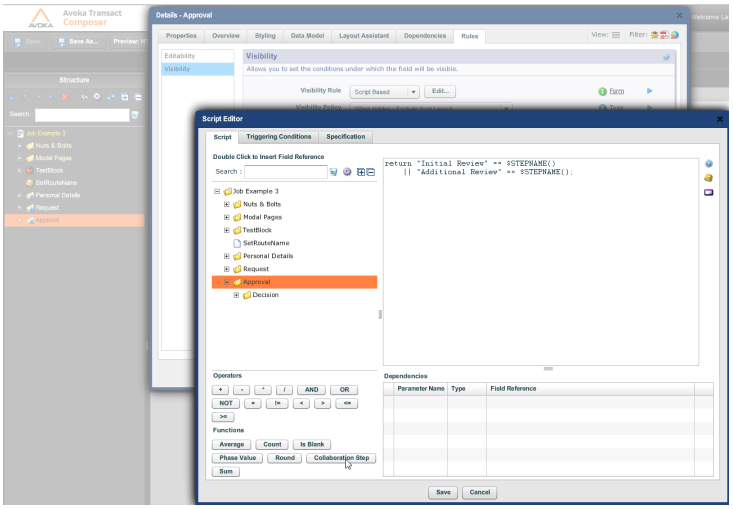
```
if ($ISBLANK($STEPNAME())){
    //leave blank
} else if ($STEPNAME()=="Applicant Update"){
    sfc.setPhaseValue("cc.route.name",{HowDoYouWishToPGroup})
} else  {
    sfc.setPhaseValue("cc.route.name",{ReviewDecisionGroup})
}
```



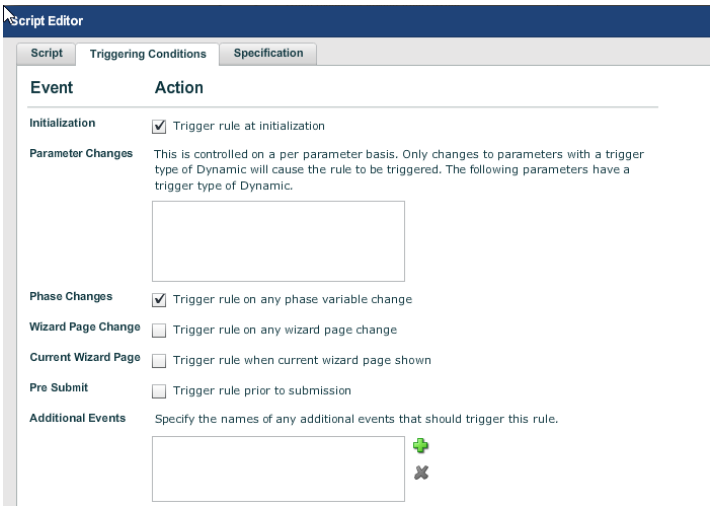
Visibility and Editability Rules

Scripts are used to setup Visibility and Editability Rules for sections based on the Step Name. This can be used to hide the office use only section from the applicant or make the section of form that the applicant fills out read only.

When 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).



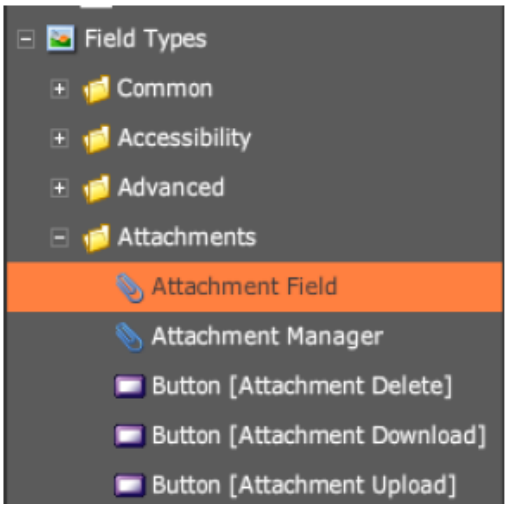
IMPORTANT! Before you save your rule go to the **Triggering Conditions** tab and click the Phase Changes check box (screenshot below).



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.

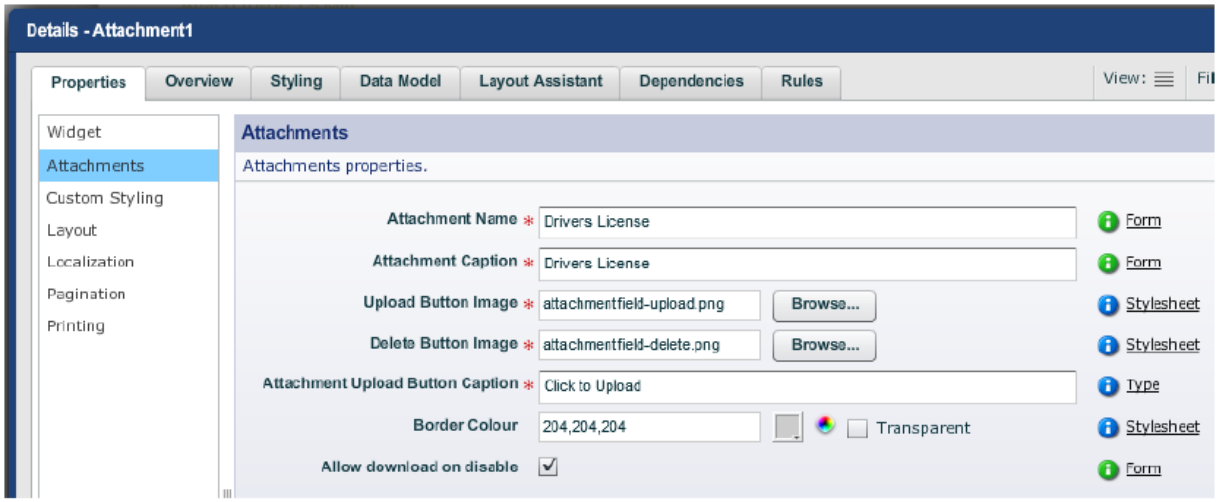


This widget will allow a file to be uploaded. A user may want to confirm the file is correct and can download the form by clicking on the file link. They can delete the file by clicking on the X.

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.

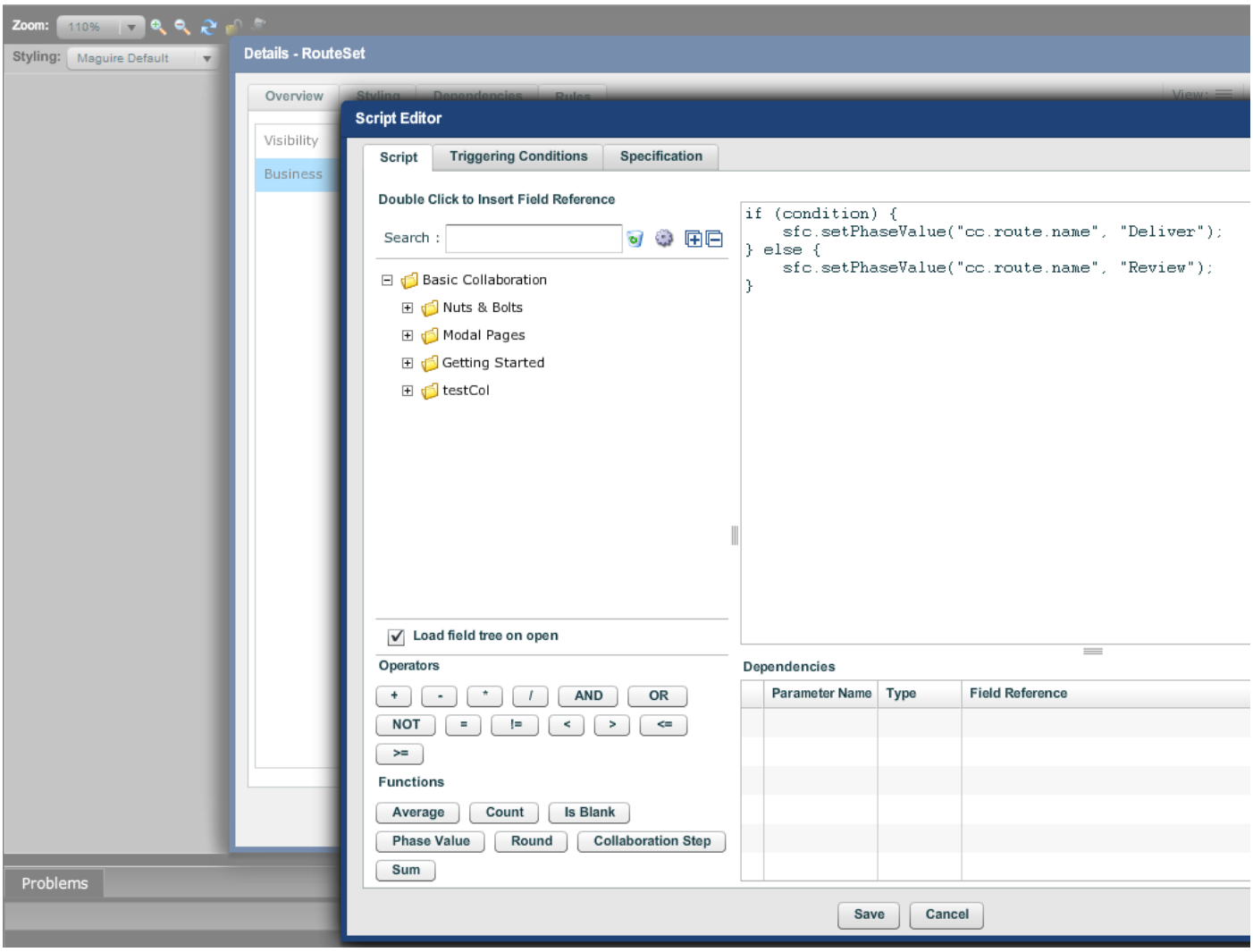


NOTE: The widget has an internal field Attachment Manager, which is pre-named "AttachmentManager" that must not be altered when using with TM task assignments and the job controller.

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

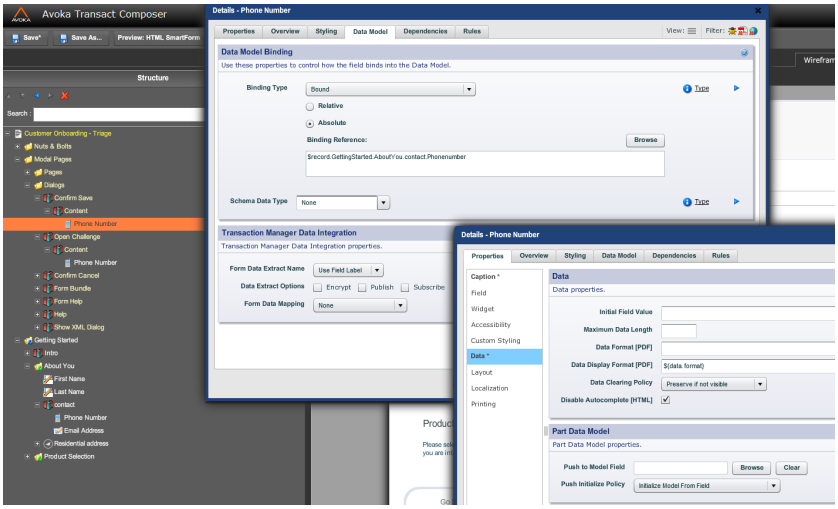
Setting up a Security Question or Save Challenge is a security feature that allows an anonymous user to save their form and retrieve it later.

It is used by Chaining and Form Bundles 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

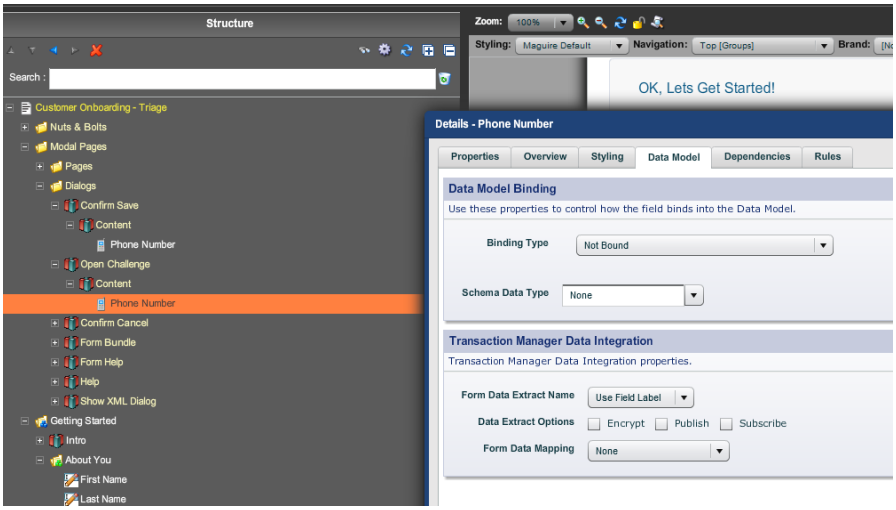
This is what the Confirm save dialog setting should look like. *Note the **Form Data Mapping is set to None**



> **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

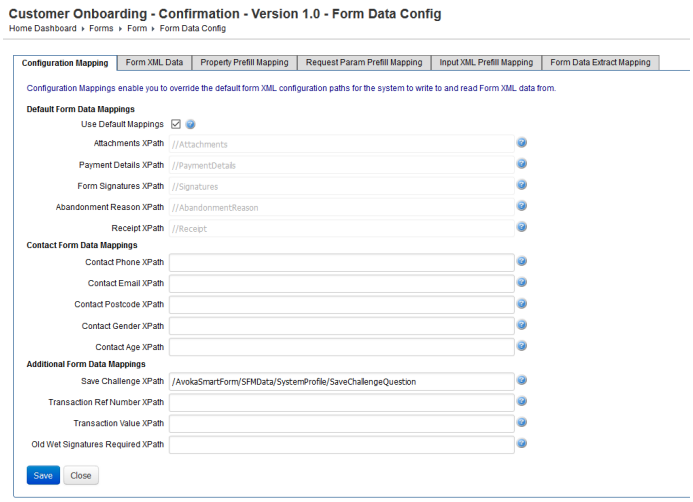
On the Open Challenge -> Content -> Field that was copied set the Binding Type to **Not Bound**.



> Export to Transaction Manager

Verify Transaction Manager

Form -> Form Version -> Form Data Config

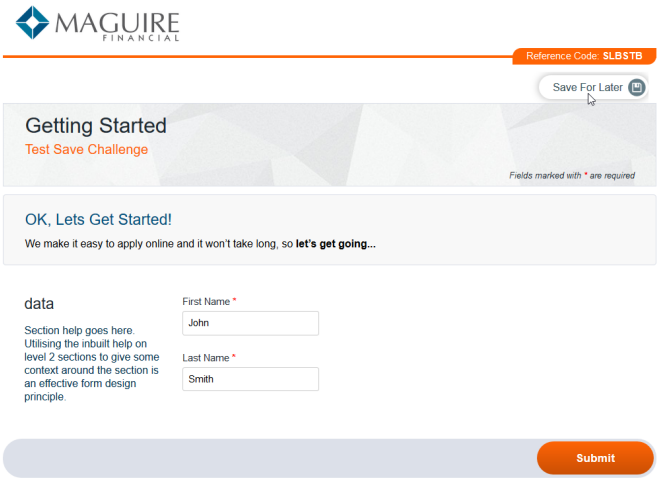


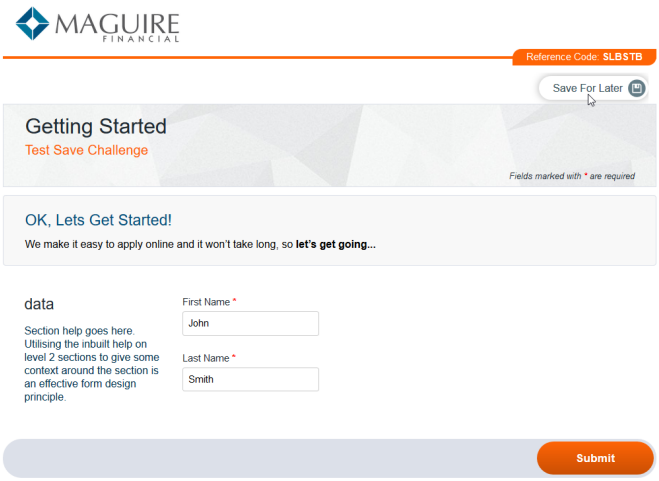
Note

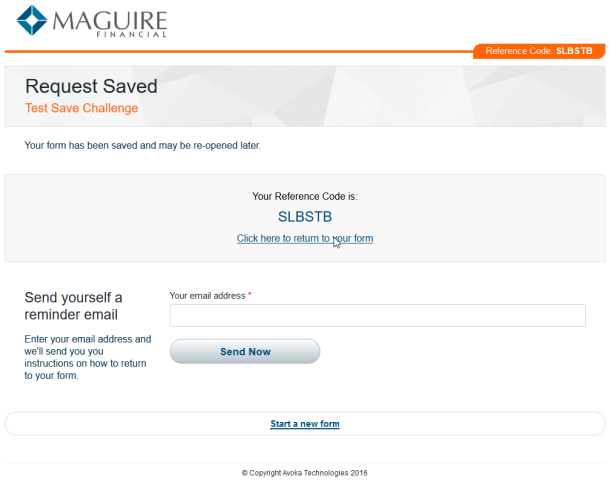
The Save Challenge XPath should ALWAYS be

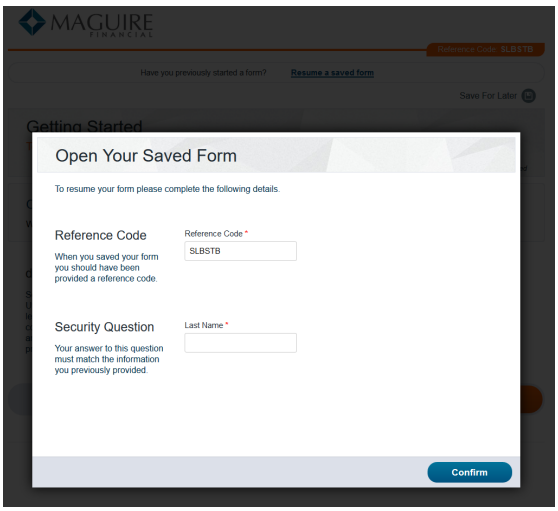
/AvokaSmartForm/SFMData/SystemProfile/SaveChallengeQuestion

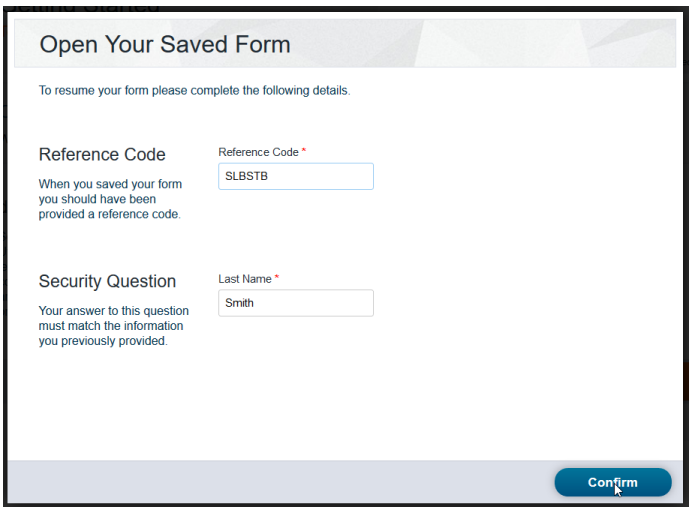
Example Flow

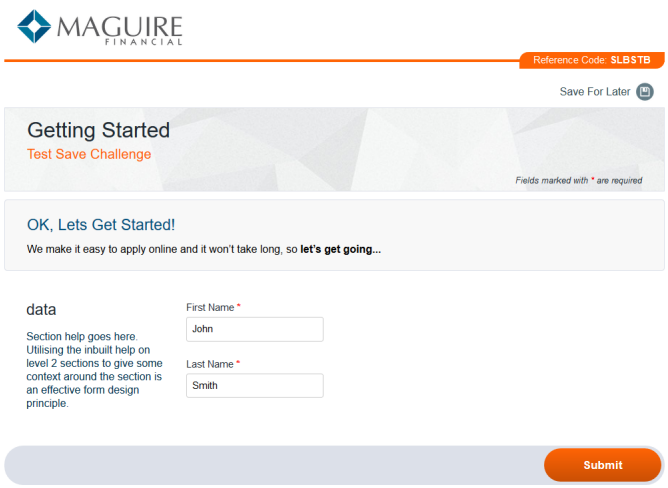



MAGUIRE
FINANCIAL
Reference Code: SLBSTB
Save For Later
Getting Started
Test Save Challenge
Fields marked with * are required
OK, Lets Get Started!
We make it easy to apply online and it won't take long, so let's get going...
data
Section help goes here. Utilising the inbuilt help on level 2 sections 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


MAGUIRE
FINANCIAL
Have you previously started a form?
Resume a saved form
Save For Later
Getting Started
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

Maestro

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

- How to set up data extracts
- How to use the collaboration job components that are included in Maestro by default
- How to set form rules (such as *Visibility* and *Editability*) based on collaboration job step
- How to preview your form in Maestro and simulate the collaboration job behavior

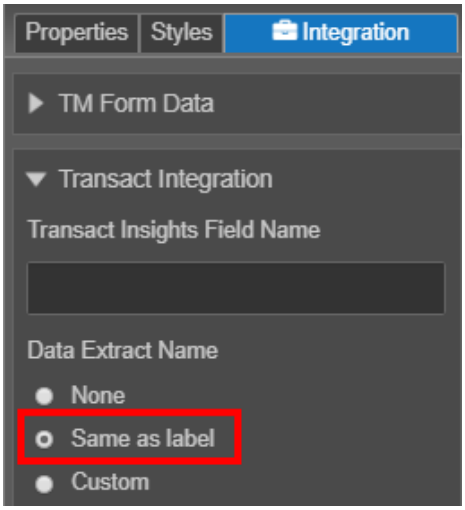
Throughout this page, we cover an example Maestro form that is intended to be used with the *Review and Approval – Multi Step Group Review* job controller service in TM.

Data Extracts

There are certain components from your form that are required to be saved as data extracts for use in the collaboration jobs process. For instance, in the form we are demonstrating here, an applicant enters a loan amount. If the reviewer approves the loan, the *Loan Amount* on the *Request* tab must then be checked to see if it exceeds the threshold of $20,000. If so, the loan must be co-approved by a manager. For this to be achieved, we need to pass the entered loan amount to the job controller service in TM in the form of data extracts.

To set a component to be included in data extracts:

1. Select the component you wish to extract by selecting it on the *Structure* panel or directly from the form editor
2. Switch to the *Integration* panel on the right-hand side of the window
3. Expand the *Transact Integration* section
4. Click on the *Same as label* radio button
   Alternatively, if you wish the data extract name to be different from the form component name, click on the *Custom* radio button and enter a name in the entry box provided. This depends on the name of the variable used within TM such as in the job controller service. For more information on data extracts see the Submission Data Extract section of the Maestro documentation.

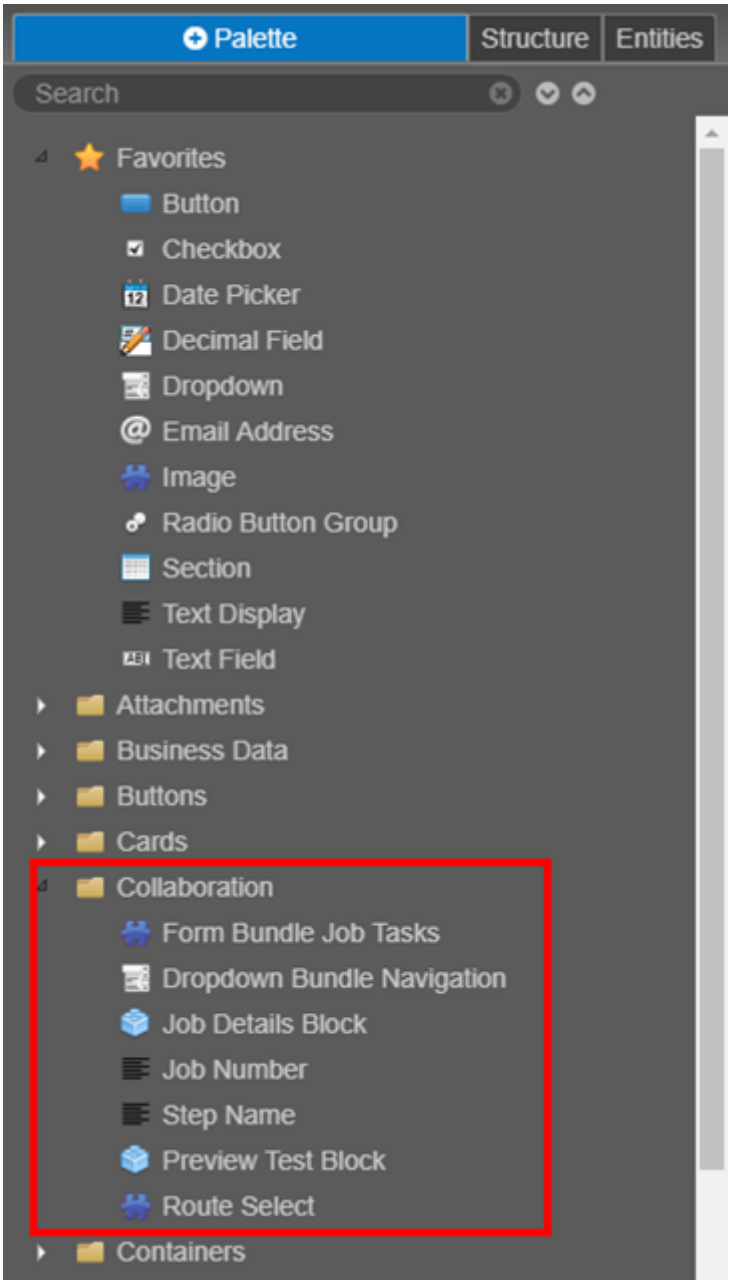


Collaboration Job Components

Collaboration job components have been created to assist the form builder in combining the collaboration jobs features into their forms.

To use a collaboration job component, go to the *Palette* panel on the left-hand side:

1. Expand the *Collaboration* folder.
2. Drag and drop any of the components on to your form as needed.



The following explains the collaboration job components:

Form Bundle Job Tasks

This component is used for form bundles as a way of navigating and keeping track of forms that are still to be completed. It appears as a set of links for navigating to the applications for products previously selected. It has extra features such as allowing a user to click on only the applications that have not been completed and allowing to download a copy of any completed application.

Dropdown Bundle Navigation

It provides a similar feature as the *Form Bundle Job Tasks* but is presented as a simple dropdown.

Job Details Block

This component is for displaying the job details on the form. You can also display the job details by selecting the *Include Collaboration Job Details* checkbox in the *Form Options* (expand *User Experience*) window. Using the Job Details Block allows you to customize the location of the job details within you form and /or template.

Job Number

This component displays the job number.

Step Number

This component displays the step name of the step that the collaboration job is currently on.

Preview Test Block

This provides a section in your form that can be used to test the form without having to render the form in Transact Manager. The block is used for testing purposes and can be viewed from the preview mode in the Maestro editor. This allows the form builder to preview the form and make sure that the rules are set up correctly without having to leave Maestro. That is, without the form being first built to Transact Manager and associated with a job controller. It is important to note that the Preview Test Block will never display in a built/rendered form as it only appears when using the Maestro *Preview* mode.

The following components may be entered:

- *Job Reference* will be shown in the relevant job details and/or job number fields. That is, it will appear in the Job Number and Job Details components where they appear on your form, or in the job details in the heading if you have ticked the Include Collaboration Job Details in the Form Options.
- *Step Name* is quite useful for testing the rules set up for presenting the correct form depending on which step name the form is up to. As this is a dropdown you must first set up the options to mimic the step name values of the job controller, which is also used in the visibility and editability rules set up for this form. To do this:
    - Switch to the *Properties* panel on the right-hand side
    - Expand the *Dropdown* section
    - Click on the *Options* button
    - Enter each of the correct step name values for each row under the *Label* column. Make sure that the first one is left blank which is testing for when the applicant first submits the application.
    - Click the *Save* button.
- *Available Routes* is used for the Route Select dropdown component. It will plug in the correct values in the Route Select dropdown, which is the same as when passed from the job controller. The values must be entered as a pipe delimited string. You will also need to select the suitable step name from the Step Name checkbox above.

Route Select

This component is a dropdown that will present to a reviewer or manager reviewing the application a list of options. A review or manager will choose an option to indicate their decision. This component will dynamically plug-in the values for the options from the job controller associated with the form. This may be useful but there has been a preference towards using radio button group component instead. This is also covered below in the Using a Radio Button Group for the *Reviewer and Manager Decision* section.

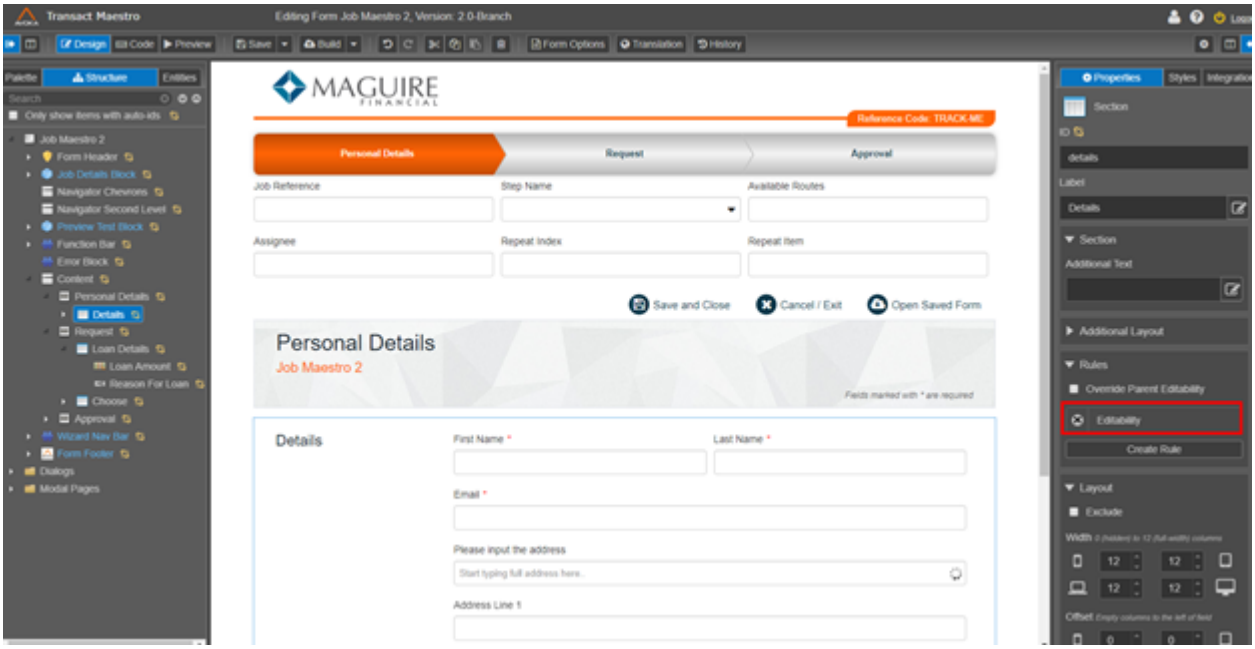
Form Rules

For a collaboration job workflow, it often requires to be 'shared' amongst the relevant parties, which may be the applicant, the reviewer and/or the manager. For the form to be presented suitably to each user the form rules need to be specified. For instance, the form rules need to be set up to manage such things as:

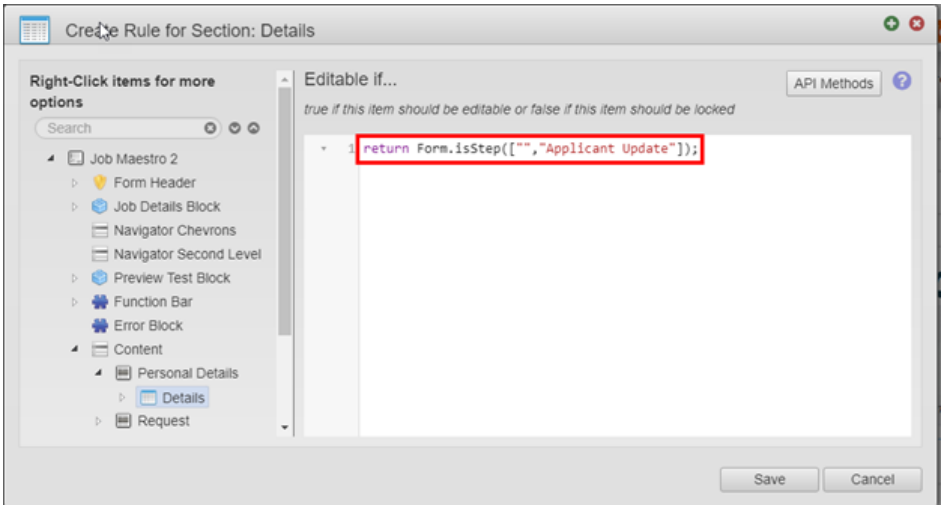
- An applicant can enter their personal and request details but can't approve their own application. Where a reviewer or manager had declined an applicant's request the applicant may update their initial request.
- A reviewer can indicate their decision. However, a reviewer can view but not update any of the applicant's personal or request details.
- A manager can indicate their decision which is an additional decision to the reviewer decision. However, a manager can view but not update any of the applicant's personal or request details. Also, a manager is like a reviewer except they can also view but not update the reviewer's decision.

To allow only the applicant to enter or update their personal or request details we must create an editability rule.

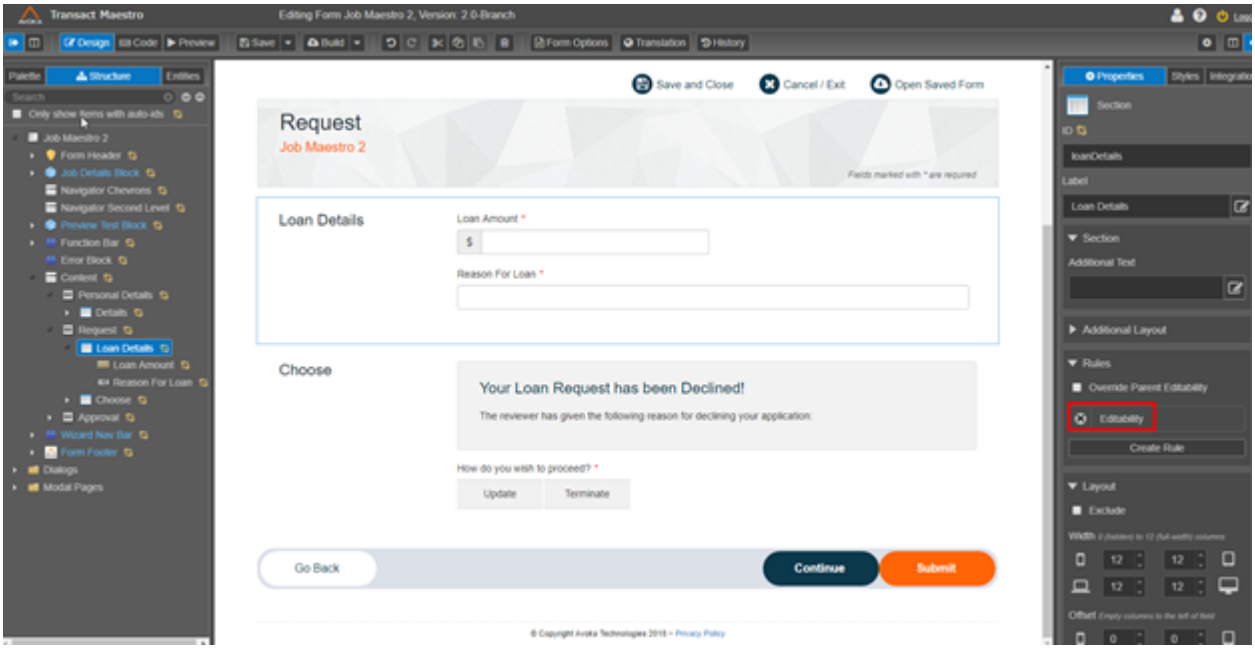
For the applicant's personal details, create an editability rule as shown.



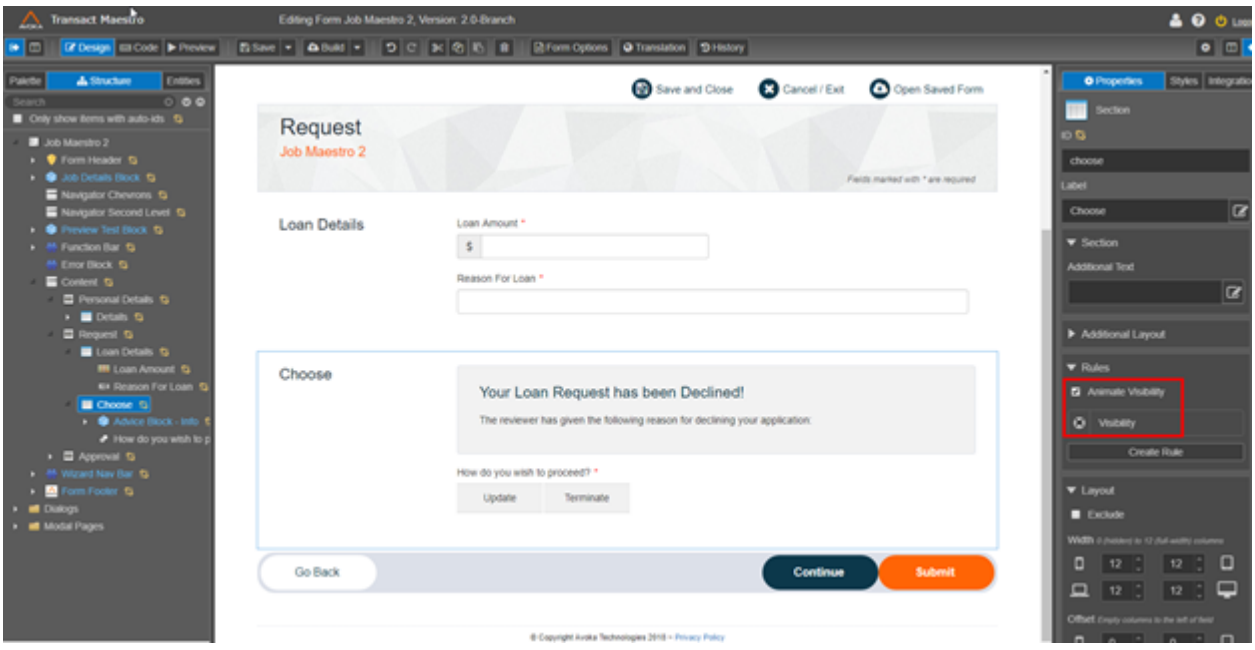
Create the following editability rule, but make sure that the second parameter matches the step name in your intended job controller service in TM. In this example, the step name requesting the applicant to update their details is the *Applicant Update* step.



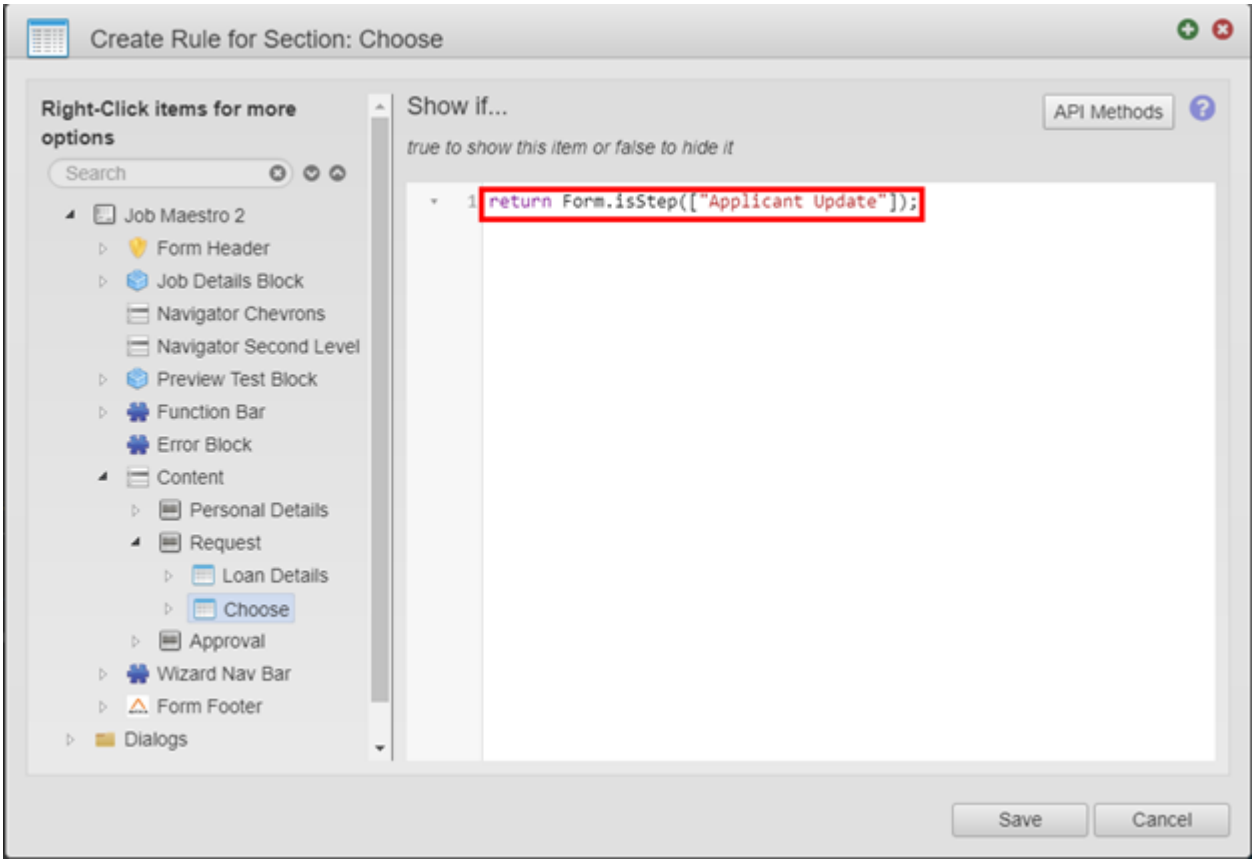
You will also need to add this same editability rule to the applicant's request.



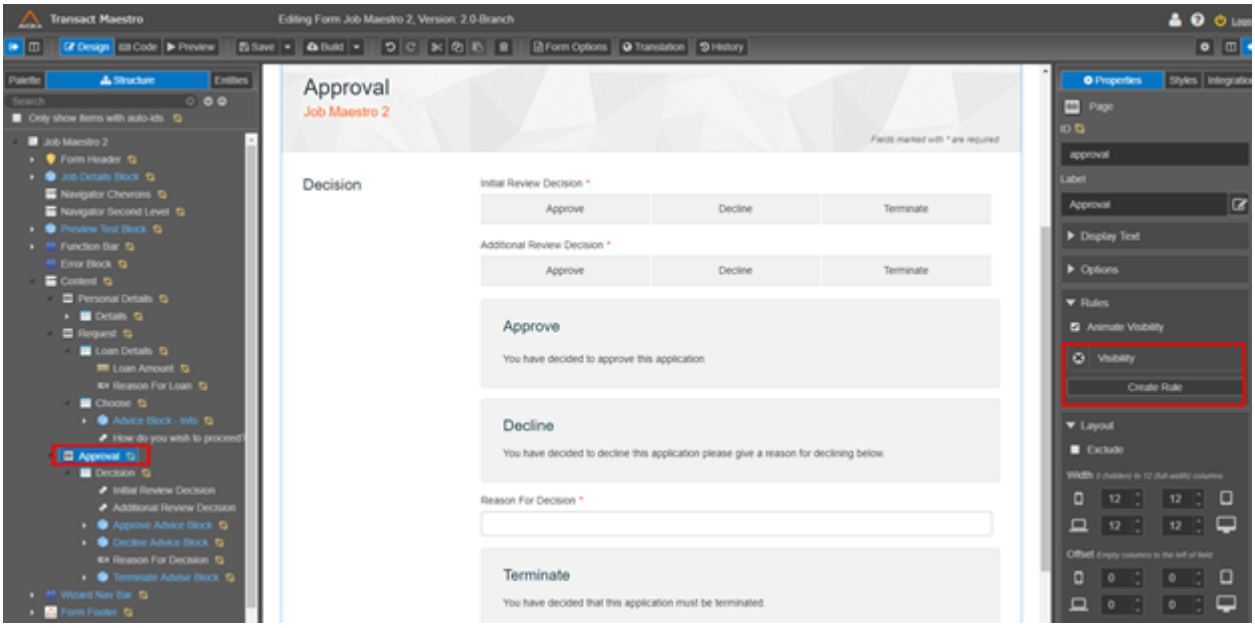
Also, when an applicant's application had been declined they must see the relevant information and choose either to update or terminate their loan application. For this, the following visibility rule must be created.

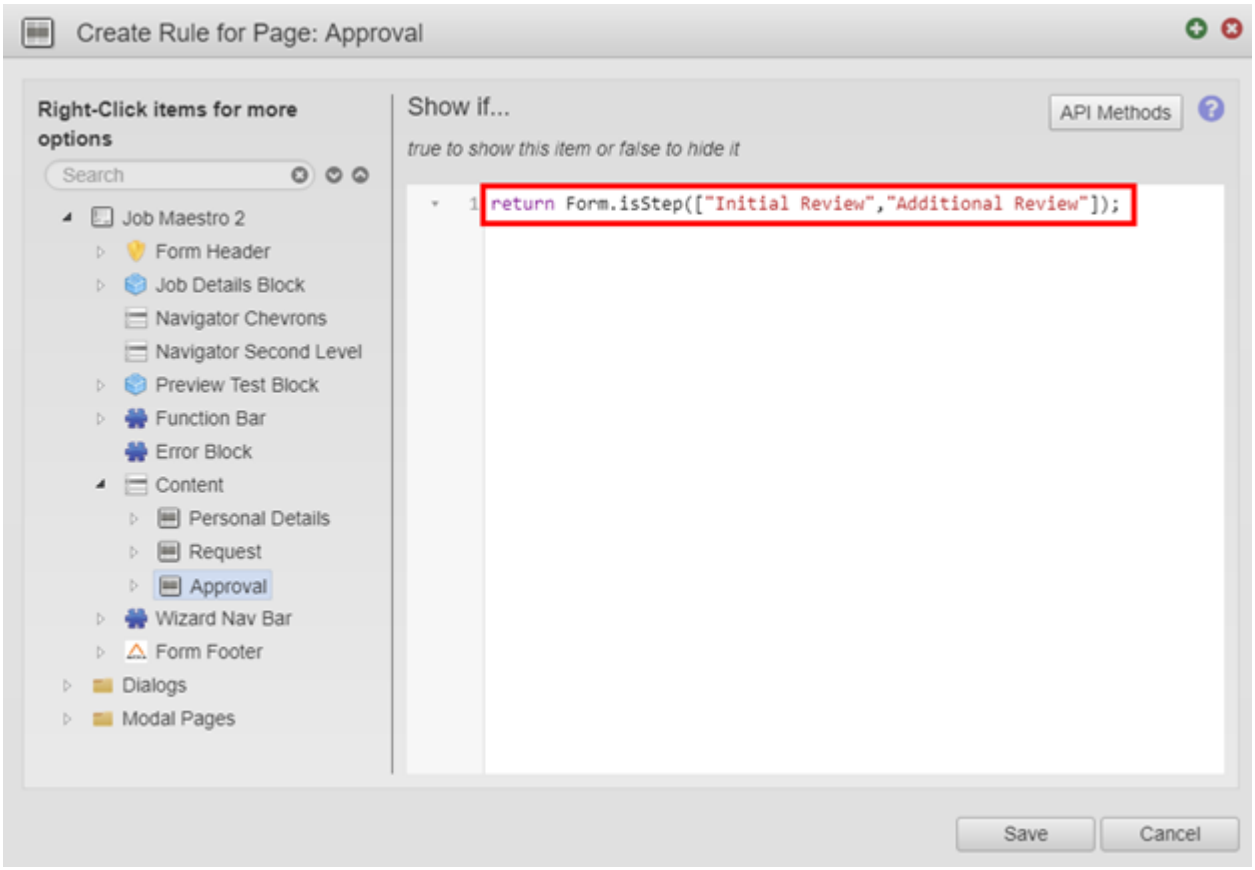


It is only relevant for this section to appear when the application had been previously declined.



A visibility rule is created to only show this tab to either reviewer or manager.





More form rules are explained in *Using a Radio Button Group for the Reviewer and Manager Decision* section.

Using a Radio Button Group for the Reviewer and Manager Decision

Both the reviewer and the manager need to be able to indicate their decision on the form. Previously, we covered the *Route Select* component which dynamically loads each of the possible decisions for a reviewer and a manager, from the job controller service in TM, into this dropdown. However, the consensus is that, for this feature, a radio button group is much better because:

- It looks nicer than a dropdown
- Feedback can be clearer as to their decision
- The form receipt indicates the decision that was made
- It accommodates for additional instructions and/or feedback to the reviewer and manager

The only negative is that, at this stage, the radio button group doesn't allow for the dynamic load of the radio button options based on the job definition service in TM. You must hard code the possible decisions as radio button options on the radio buttons group.

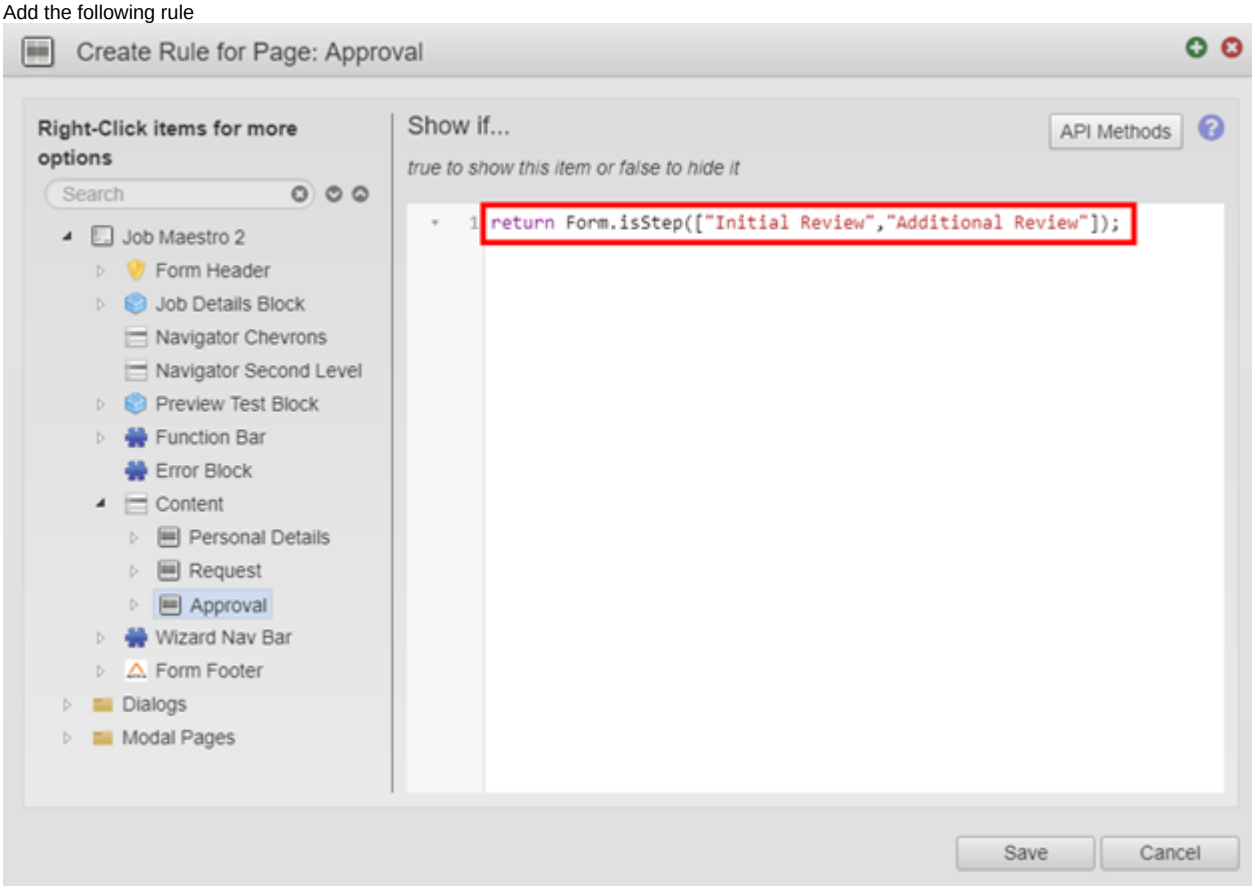
For this example, the reviewer and the manager are both involved in the collaboration job, so we will use the *Job Controller Service*, *Review and Approval – Multi Step Group Review.*

As this decision feature is not available for the applicant, it makes sense to add a separate tab to your form which can be shared amongst the reviewer and manager. Then make the approval or decision tab only visible to the reviewer and the manager. We have named it the *Approval* tab.

To do this select the *Approval* tab:

1. From the *Properties* panel, click the *Create Rule* button

2. Add the following rule



3. Click the *Save* button

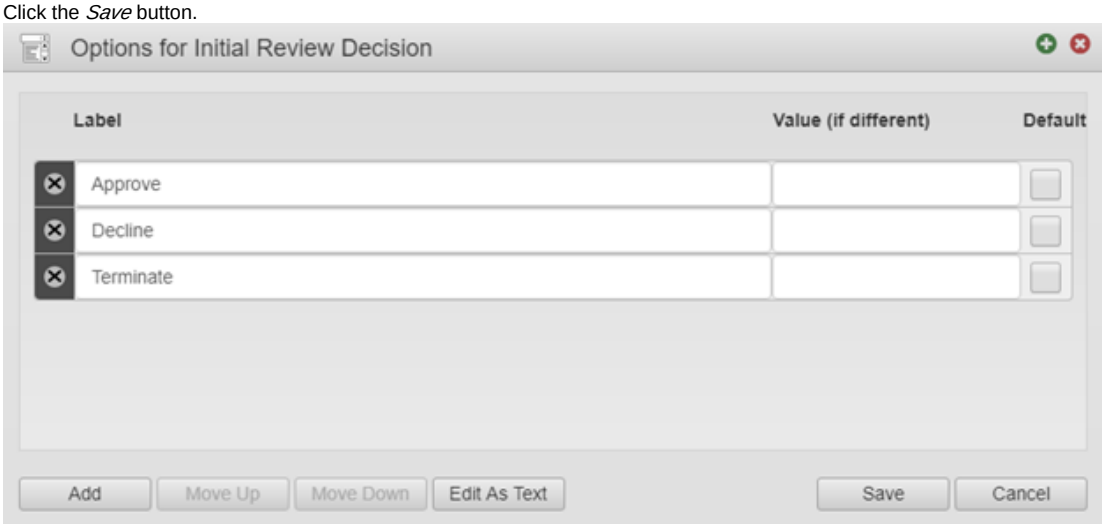
Being a shared tab, we need to manage the form interaction correctly for each of the parties involved. For instance, the following rules apply:

- A reviewer can indicate their decision but cannot see the manager's decision.
- A manager can indicate their decision which is an additional decision to the reviewer decision. A manager is like a reviewer except they can also view but not update the reviewer's decision.

To keep it simple, we will cover some of the rules here, but you will need to determine all the rules that apply for your form.

To create a radio button group for the reviewer decision, go to your form in Maestro:

1. Add a new tab for the reviewer and manager decision. We have named it *Approval*.
2. Drag and drop the *Radio Button Group* component onto your form. We have named it *Initial ReviewDecision*.
3. From the *Properties* panel on the right-hand side, expand the *Button Group* section.
4. Click the *Options* button.
5. Enter each of the decision options under the *Label* column. In our example, we have 3 radio button options, *Approve*, *Decline* and *Terminate*.
6. Click the *Save* button.



Next you will need to make the decision mandatory to minimize ambiguity.

7. Expand the *Rule* section
8. Select the *Mandatory* checkbox

To create a radio button group for the manager decision, repeat the above steps 2 to 8 and give it a suitable name. We named our radio button group *Additional Review Decision*.

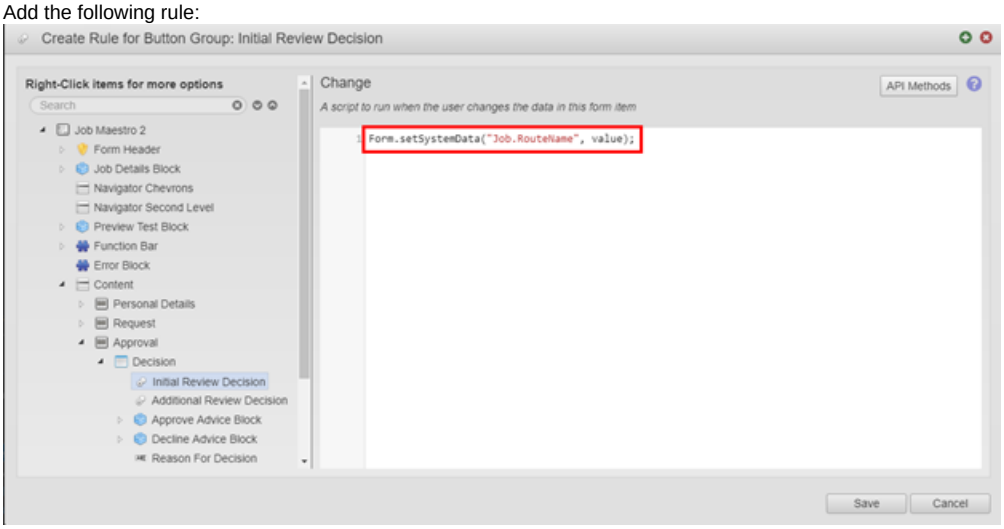
As explained above, the two radio button groups require different rules.

For the reviewer, the *Initial Review Decision* radio button group, we need to create the following two rules:

Change Rule

An added complexity exists with sharing this form regarding which decision must be used for the job controller to decide what to do next. For this we need to create a change rule which plugs the reviewer decision to the system profile route name, *Job.RouteName*, only when the reviewer has changed this. This means that, on change, the reviewer decision will be used by the job controller. To do this:

1. From the *Properties* panel, click the *Create Rule* button
2. Add the following rule:

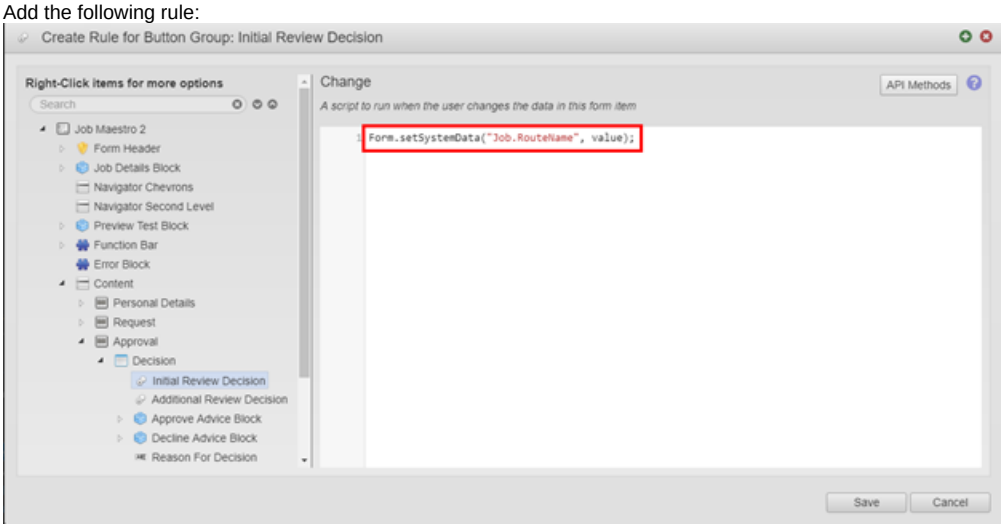


3. Click the *Save* button

Editability Rule

As the reviewer can also be seen by the manager we need to make sure that only the reviewer can update this. To do this:

1. From the *Properties* panel, click the *Create Rule* button
2. Add the following rule:



3. Click the *Save* button

For the manager, the *Additional Review Decision* radio button group, we need to create the following two rules:

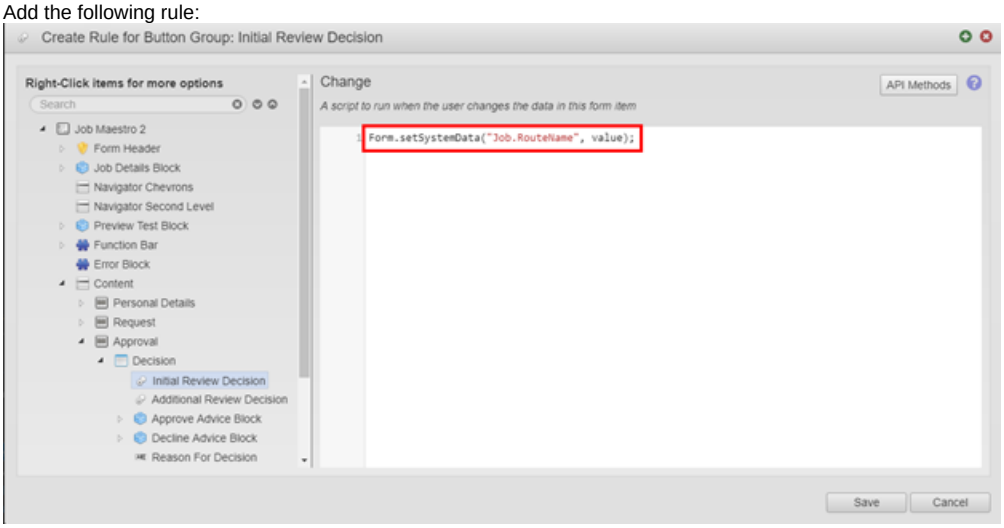
Change Rule

The same change rule as for the reviewer needs to be added here to allow the manager's decision to be used for directing what happens next when this has been changed.

Visibility Rule

This radio button group must only be visible to a manager. To do this:

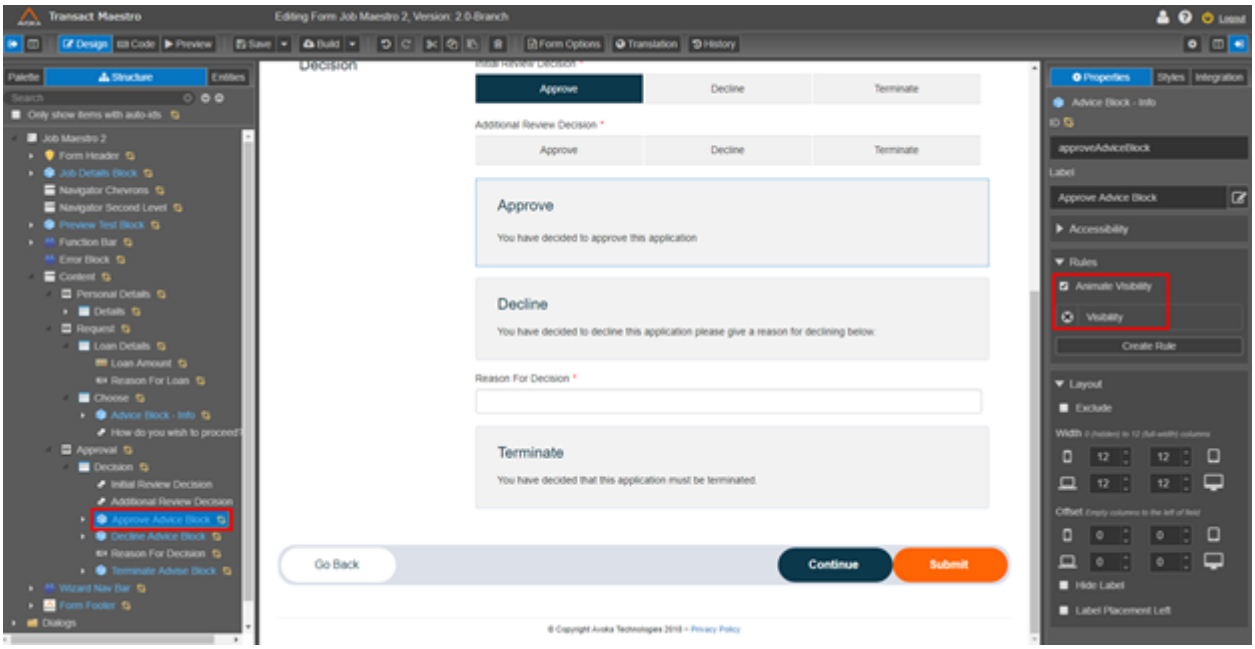
1. From the *Properties* panel, click the *Create Rule* button
2. Add the following rule:



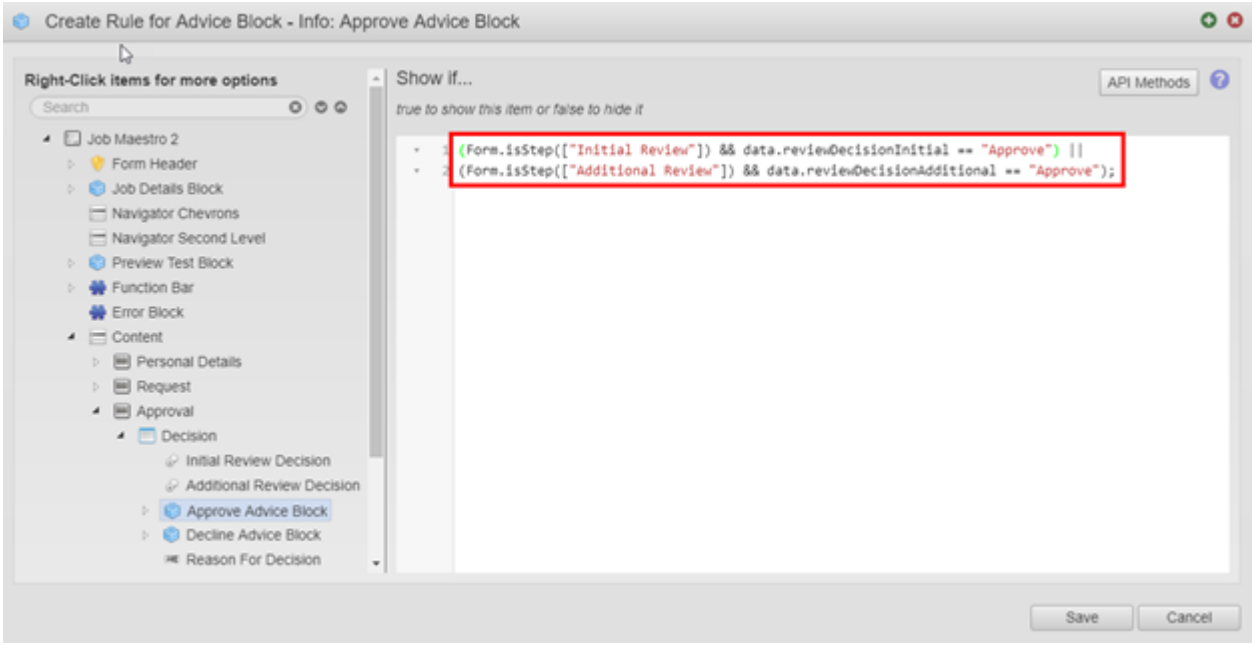
3. Click the *Save* button

Some other rules that may be relevant are:

In this case an *Approve Advice Block* pops up when the reviewer or manager clicks the *Approve* radio button option, to highlight the decision they are about to submit.

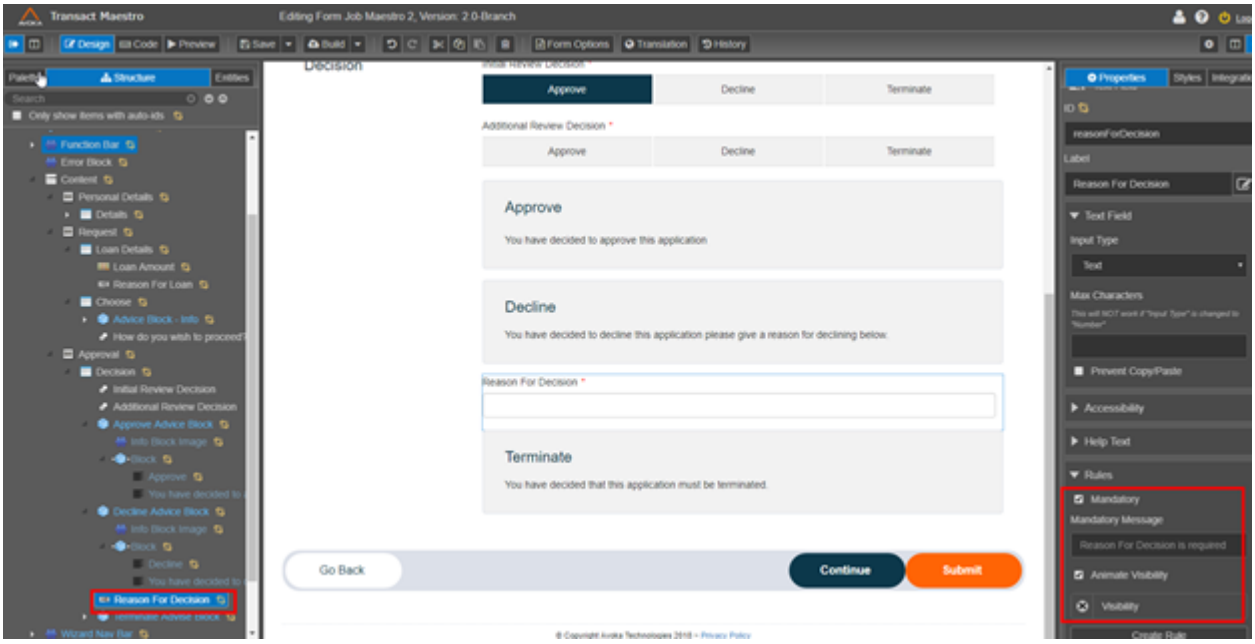


For this the visibility rule is:

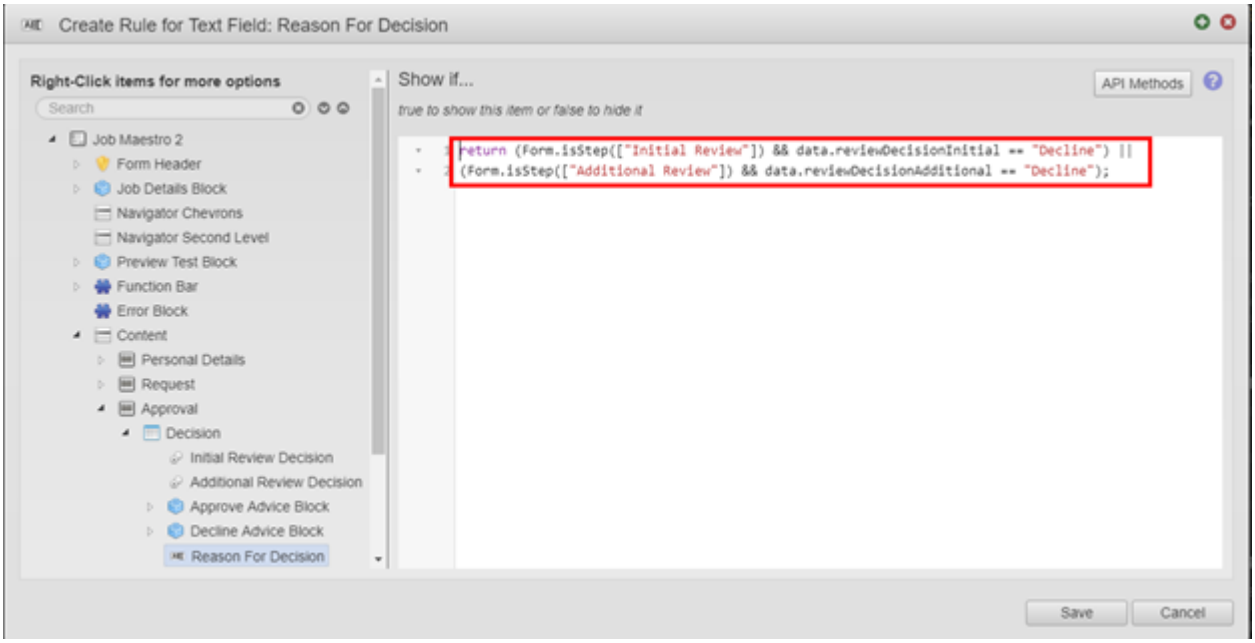


When either the reviewer or the manager declines the application, they must enter a reason for their decision.

The following visibility rule is required for the *Reason for Decision* field which is ticked as *Mandatory*.



And the visibility rule is:


Create Rule for Text Field: Reason For Decision
Right-Click items for more options
Search
Job Maestro 2
Form Header
Job Details Block
Navigator Chevrons
Navigator Second Level
Preview Test Block
Function Bar
Error Block
Content
Personal Details
Request
Approval
Decision
Initial Review Decision
Additional Review Decision
Approve Advice Block
Decline Advice Block
Reason For Decision
Show if...
API Methods
true to show this item or false to hide it
return (Form.isStep(["Initial Review"]) && data.reviewDecisionInitial == "Decline") ||
(Form.isStep(["Additional Review"]) && data.reviewDecisionAdditional == "Decline");
Save
Cancel

Transact Manager Form Configuration

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

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

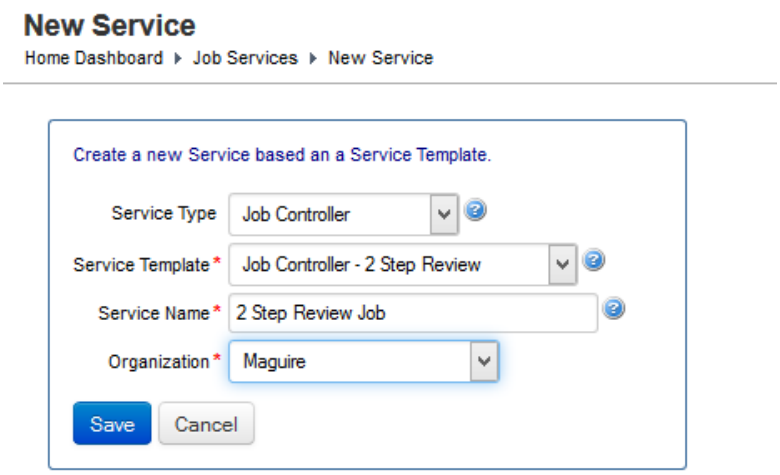
- Form Properties: can be reference in the job definition to provide form specific setting.
- Form Groups: control access to the form
- Form Delivery Method: this has to be setup for the job controller to complete.

Create Job Controller

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

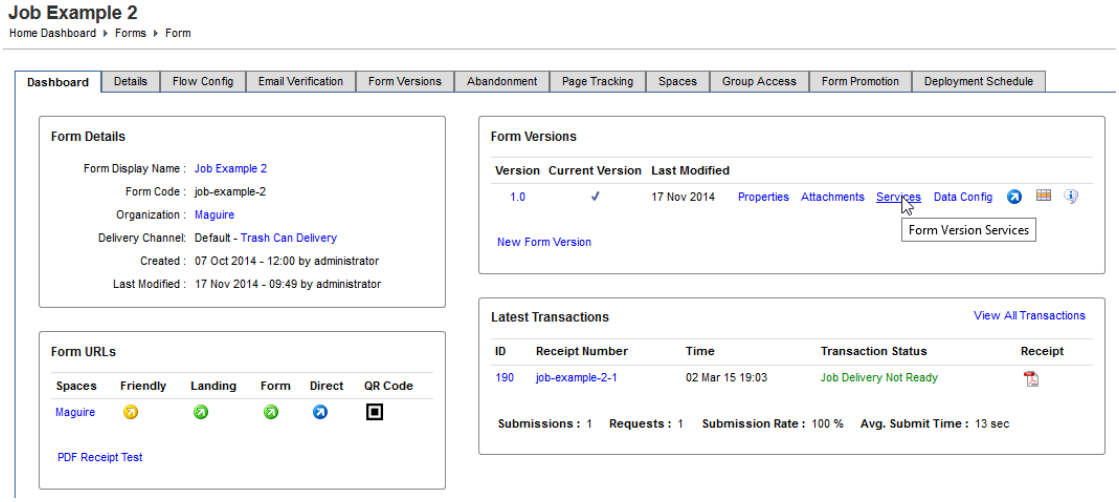
Before we configure the Form, we first have to create a new **Job Controller** service instance from the service template **Job Controller - 2 Step Review**. Please refer to the 2 Step Review example here.

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



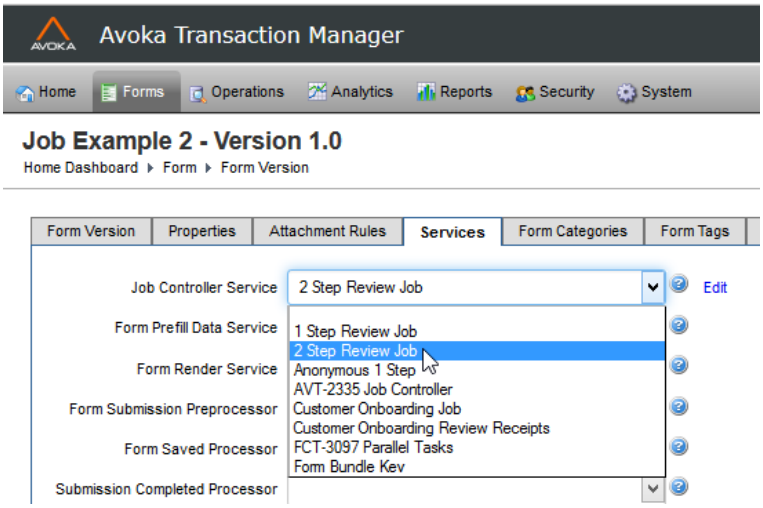
Form Imported into Transact Manager

In the Form Dashboard



Assign the Job Controller service

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

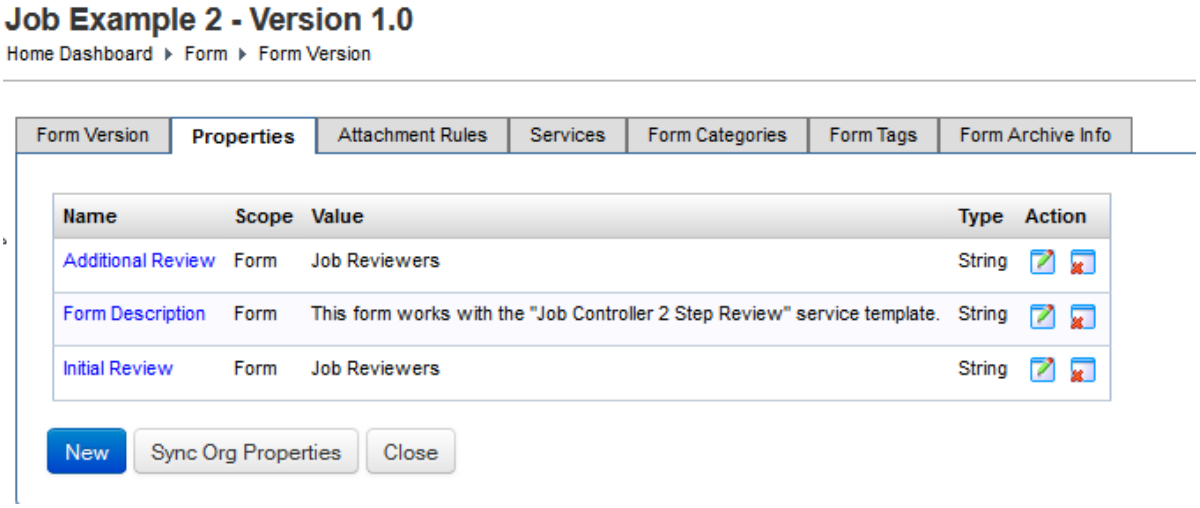


Form Properties

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.

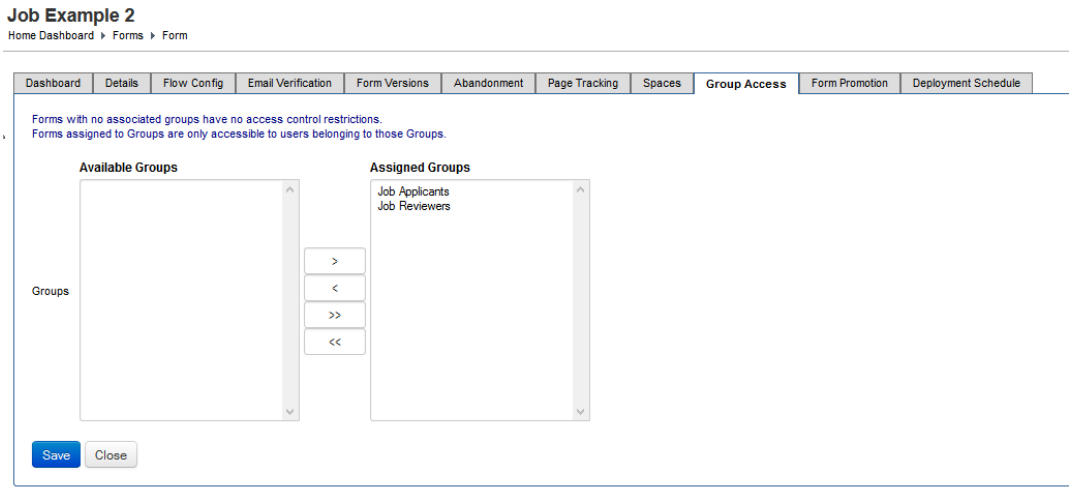
NOTE: By default, both will be mapped to the same Job Reviewers form group.



Form Groups

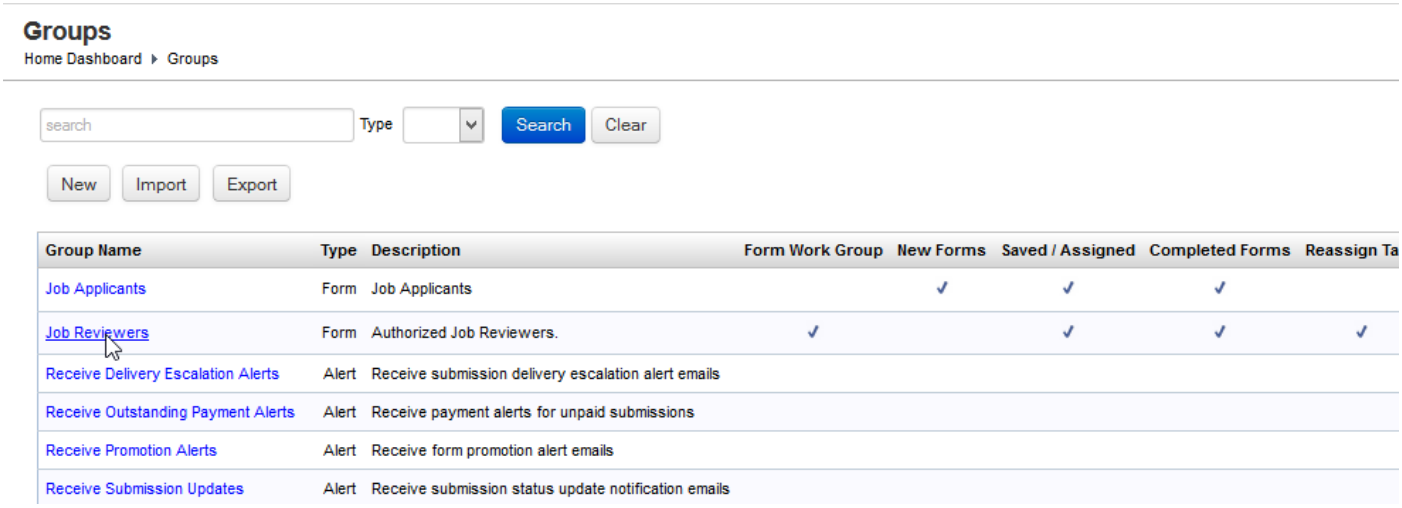
Form groups are used to control access the following access to a form. By default if a form group is not assigned to a form it will be available to any user in the organisation to start.

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.

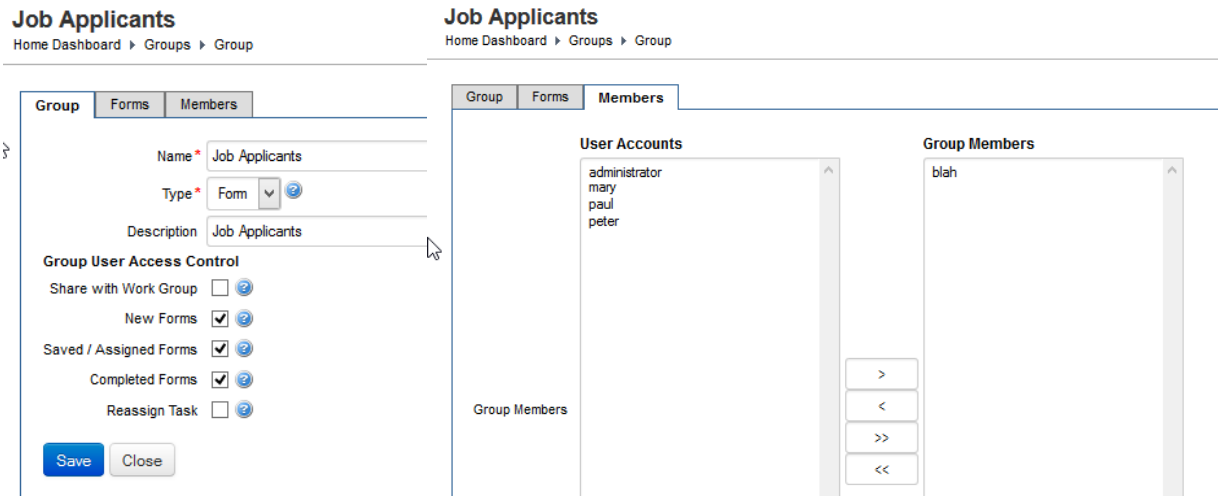


The purpose of the **Job Applicants** form groups is to restrict who can start the form. The **Job Reviewers** cannot initiate the form but can participate in reviewing the submitted forms.

Select the **Job Applicants** group.

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

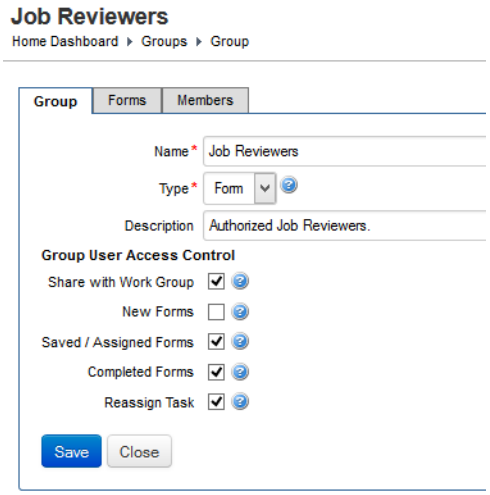
Select the **Members** tab, add your user to this tab.



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.



Job Groups

Job groups are used for the *Reviews* tab in the form space to:

- Control a job coordinator's access to jobs.
- Facilitate a job coordinator's search of jobs by filtering on job group.

It is important to note that there are two ways that job groups can be created. A job group:

- Will be automatically created (recommended method) when a job is run for the first time and the job group is:
  - Specified in the job definition.
  - Not already created.
  - Can be manually created from the *Security -> Groups* menu item and creating a group of type *Job*. For more information, refer to the *Groups* section.

These job groups should match the job groups as defined in the job definition of the applicable job(s).

In either case, whether job groups are created automatically or manually, the job groups will need to be assigned to the applicable job coordinators' user accounts.

This is done from the *Security > Groups* menu item. Select each job group and assign the applicable user accounts from the *Members* tab. Then the job group will appear as an option in the job coordinator's *Job Groups* dropdown from the *Reviews* tab

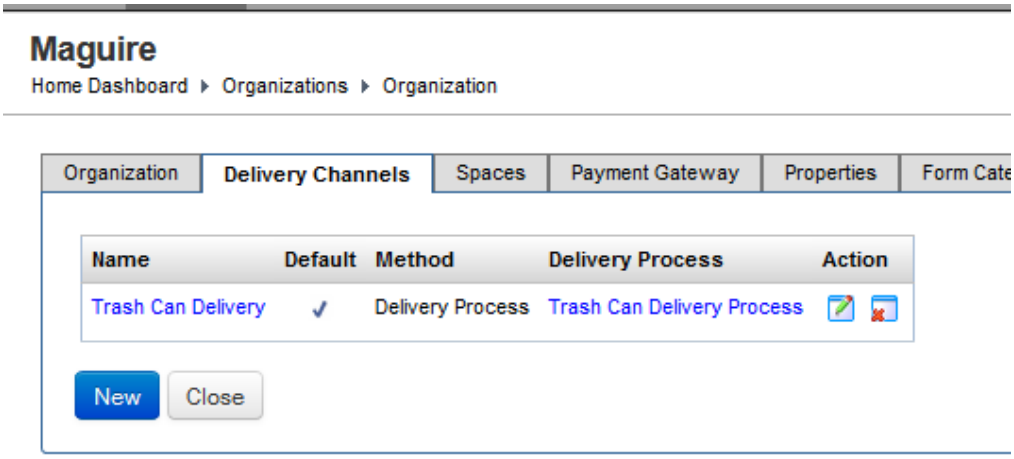
Delivery Method

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.

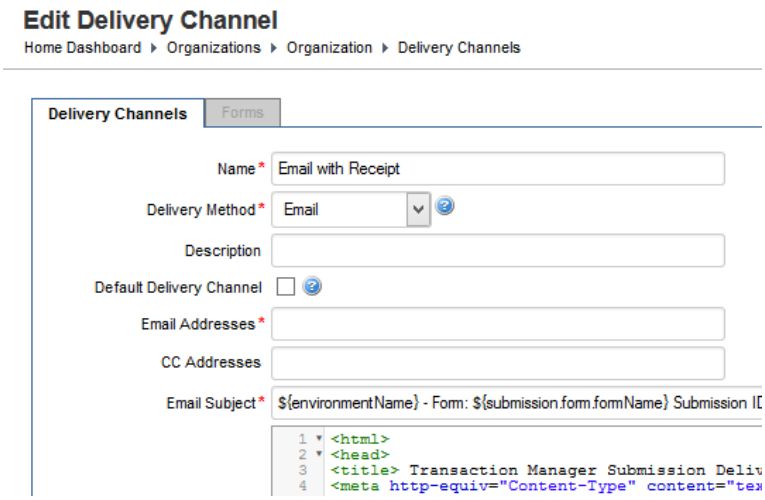
Forms -> **Organizations**, select **Maguire** to edit.



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.


Edit Delivery Channel
Home Dashboard ▸ Organizations ▸ Organization ▸ Delivery Channels
Delivery Channels
Forms
Name *
Email with Receipt
Delivery Method *
Email
Description
Default Delivery Channel
Email Addresses *
CC Addresses
Email Subject *
${environmentName} - Form: ${submission.form.formName} Submission ID
<html>
<head>
<title> Transaction Manager Submission Deliv
<meta http-equiv="Content-Type" content="tex

Task Assign Repeat Context (Advanced Topic)

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>Assignee</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>AssignRepeatIndex</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>AssignRepeatItem</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"

Admin Operations

Developers and Administrator

Developers

Developers will need access to the Organisations, Forms, 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

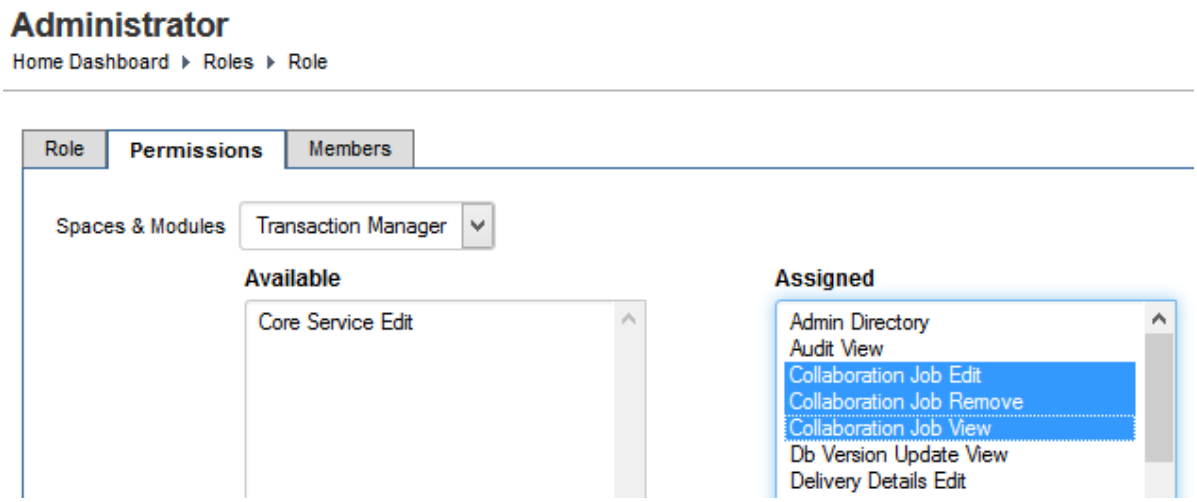
System Administrators will need access to the developers section above above roles to move forms and jobs between environments, to maintain and assign users to the correct groups and roles.

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



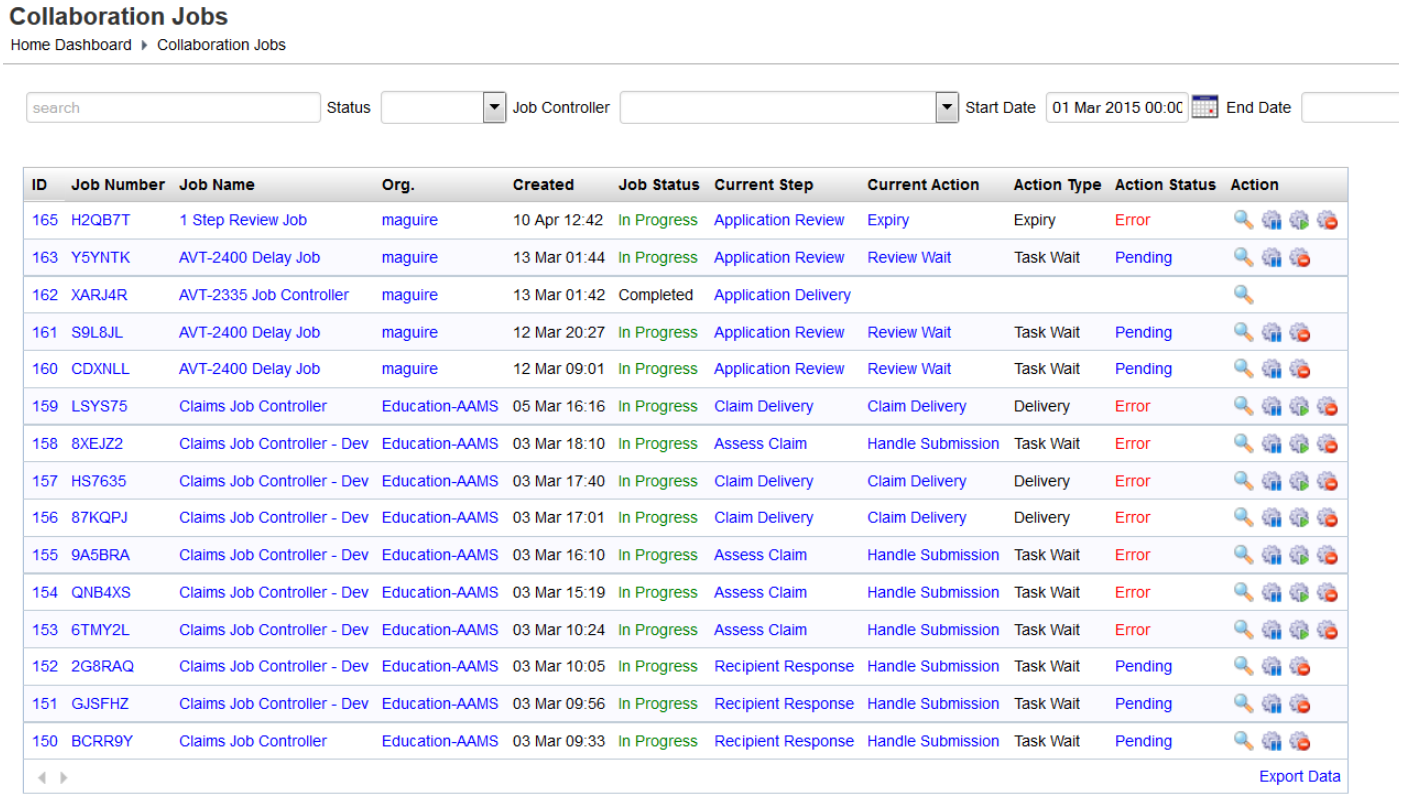
Collaboration Job View allows one to review the entries.

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 Jobs Management

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



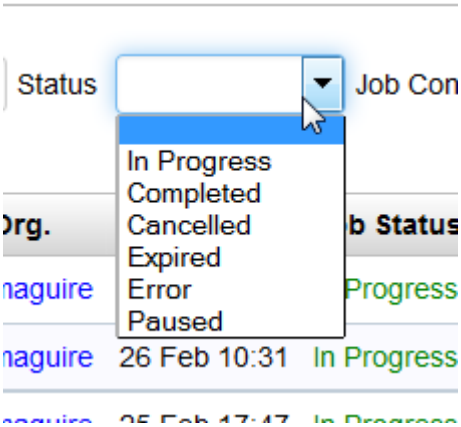
Job Search

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:

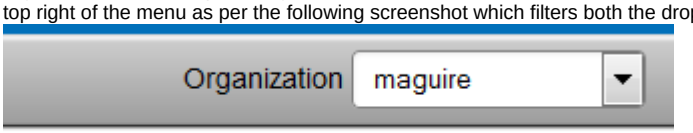
1. A **full word** search over a number of Job fields such as Job Number and Job Name.
2. The current job status can be selected from a drop down.



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

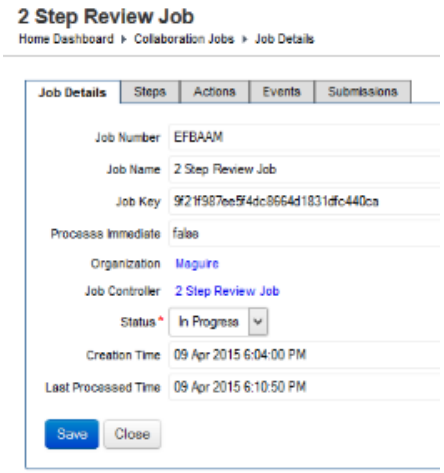
- (Job) **ID**, **Job Number**, **Job Name** selecting any of these will take you to the Job Detail page for the job instance (as described below)
- **Org.** (Organization) clicking on this link is a shortcut to the Organization Edit page.
- **Created Date:** The date the job was created
- **Job Status**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

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

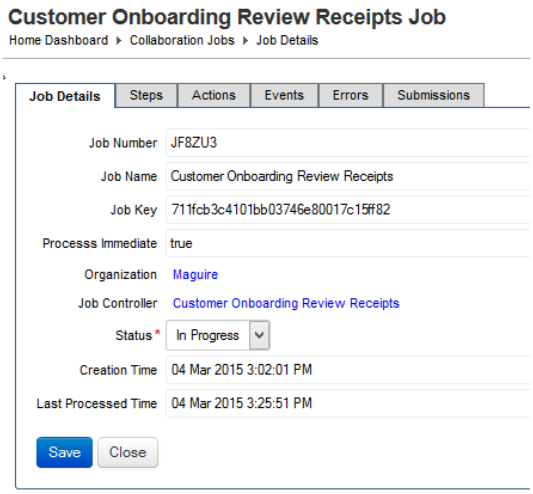
Icon	Details
	This icon takes you to the Job Detail page for the selected job instance
	This icon pauses the job instance.
	This executes the job instance immediately. This is useful for a jobs developer trouble shooting an action service. They can make a programming or configuration change then press 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#Job Execution.

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

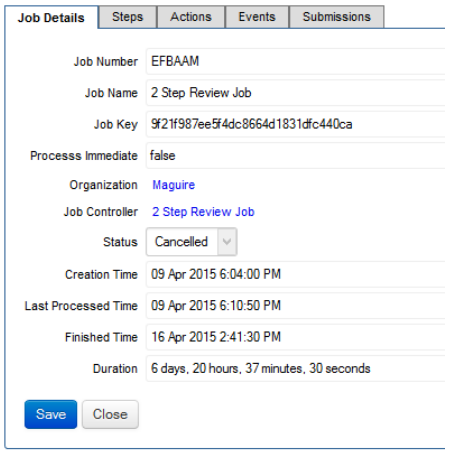
Job Controller: The name of the Job Controller which has a link to the Job Controller's Editor opening in the Job Definition page.

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.

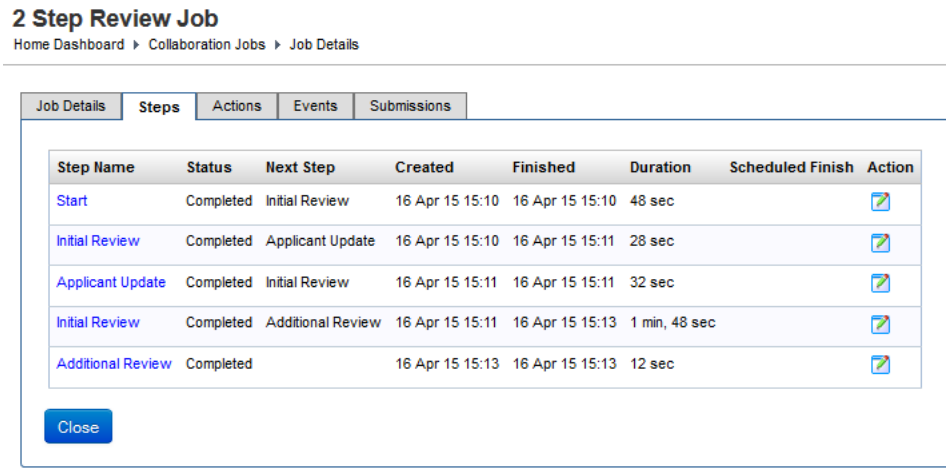
For **In Progress** and **Paused** Jobs it shows the Job Creation Time and Last Processed Time.

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



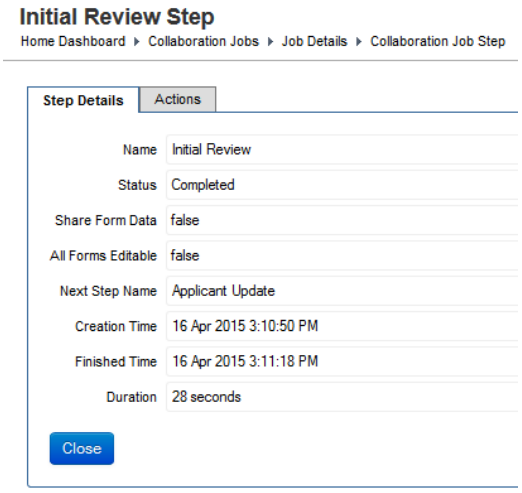
Steps

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

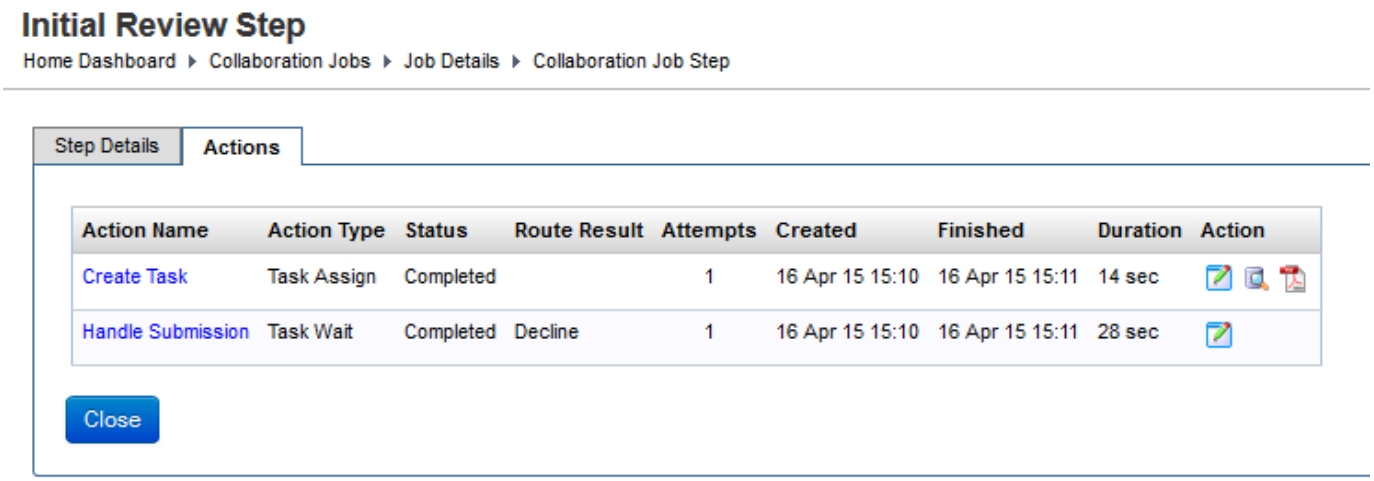


Clicking on the Action icon will take you to the steps details

Step Details



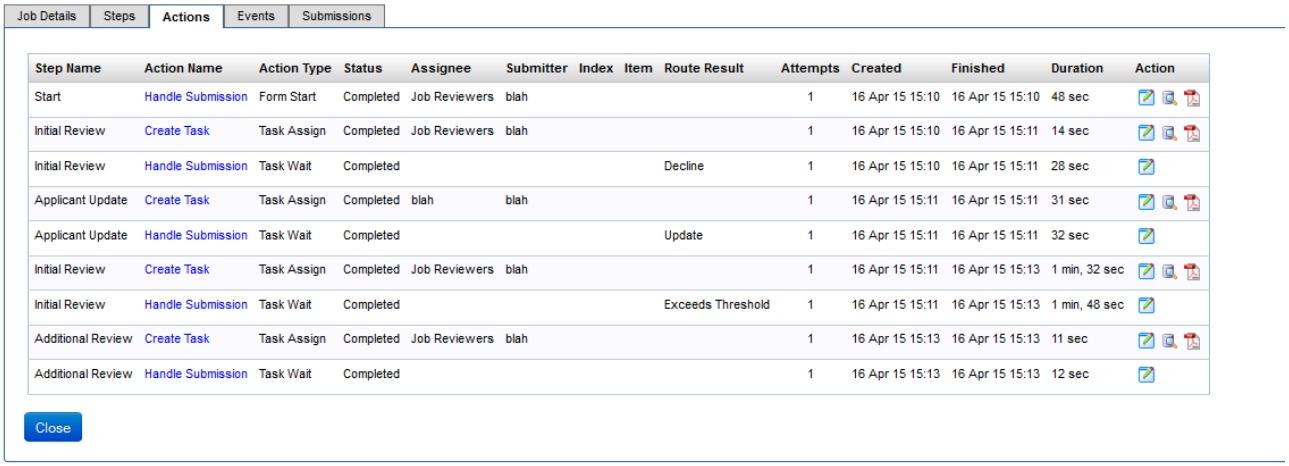
Clicking on the Actions Tab shows only the Actions that are run for that step. this screen is discussed in the Actions section.



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.

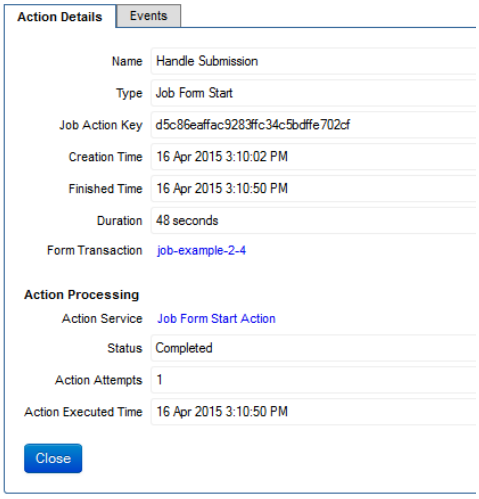
The Action Name is a link that can be clicked on to get to the Action Detail page. Clicking on the  icon will take you to the Action Detail page.



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

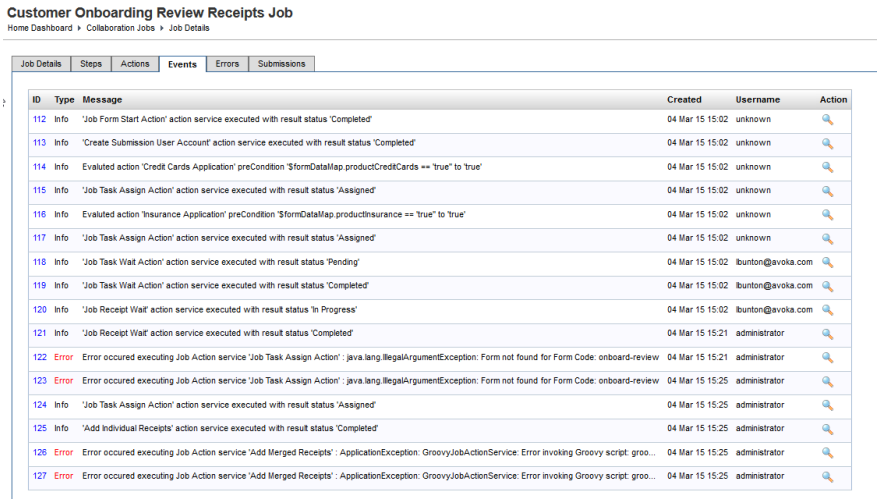


The Action Details page gives even more detail about the action instance.

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

```
27  final ACTION_PROPERTY_WAIT_TIME_MINUTES = "Wait Time Minutes"
28
29  JobEventLogService jobEventLogService = new JobEventLogService()
30
31  boolean isLogEvents = serviceParameters["Log Events"]
32
33  def actionResult = new ActionResult(Status.IN_PROGRESS)
34  def jobAction = actionContext.getJobAction()
35
36  // logEvent closure
37  def logEvent = { msg ->
38      if ( isLogEvents ) {
39          // Logs to Job Action Event Tab
40          jobEventLogService.logInfoEventWithAction((String) msg, jobAction)
41      }
42  }
43
44  String waitTimeString = actionStepProperties.getProperty(ACTION_PROPERTY_WAIT_TIME_MINUTES)
45  Validate.notEmpty(waitTimeString, "The Action Property - " + ACTION_PROPERTY_WAIT_TIME_MINUTES + " is empty")
46
47  def waitTimeMinutes = waitTimeString.isInteger() ? waitTimeString.toInteger() : 0;
48
49  Validate.isTrue(waitTimeMinutes > 0, "The Action Property - " + ACTION_PROPERTY_WAIT_TIME_MINUTES + " is less than 1 or i
50
51  def dateStart = jobAction.getTimeCreated()
52  def dateLater = DateUtils.addMinutes(dateStart, waitTimeMinutes)
53
54  logEvent( "dateStart: "+ dateStart + ", waitTimeMinutes: " + waitTimeMinutes + " , dateLater: " + dateLater )
55
```

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.

Error Details

Home Dashboard ▸ Job Services ▸ Job Event ▸ Job Details ▸ Error Details

Error ID 41
Time Wed Mar 04 15:25:47 AEDT 2015
Name GroovyJobActionService
Message GroovyJobActionService
User Message Error invoking Groovy script: groovy.lang.MissingMethodException: No signature of method: com.avoka.fc.core.service.AttachmentService.addInFormAttachment() is applicable for argument types: (com.avoka.fc.core.entity.Submission, java.lang.String, java.lang.String, [B) values: [[<Objectid:Submission, submission_oid=195>, committed, [ta
Context

```
groovyScript=/* Groovy Job Action service which is used to execute job step 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 groovy.json.JsonSlurper

import org.apache.commons.lang.Validate

import com.avoka.fc.core.dao.DaoFactory
import com.avoka.fc.core.dao.SubmissionDataDao
import com.avoka.fc.core.entity.Job
import com.avoka.fc.core.entity.JobAction
import com.avoka.fc.core.entity.Submission
import com.avoka.fc.core.service.AttachmentService
import com.avoka.fc.core.service
```

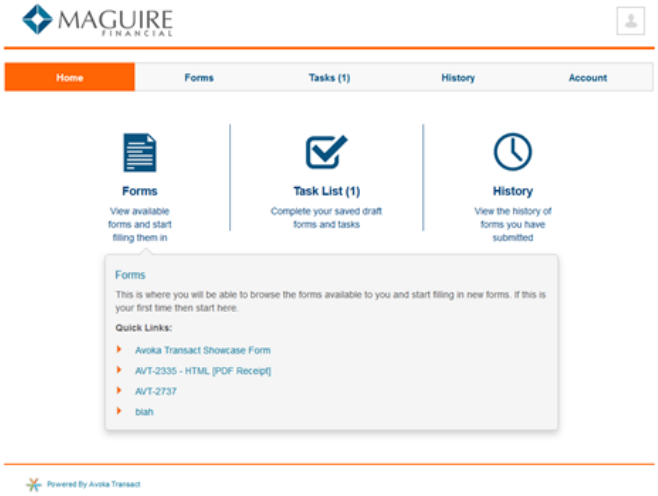
Space / Module Transaction Manager
Request GET http://localhost:9080/manager/admin/job/job-search.htm?actionLink=process.job&value=358_wid=ee5d
IP Address 127.0.0.1
User Agent Mozilla/5.0 (Windows NT 6.3; WOW64; rv:35.0) Gecko/20100101 Firefox/35.0

Error Stack Trace

```
groovy.lang.MissingMethodException: No signature of method: com.avoka.fc.core.service.AttachmentService.addInFormAttachment() is applicable for argument types: (com.avoka.fc.core.entity.Submission, java.lang.String, java.lang.String, [B) values: [(
Possible solutions: addInFormAttachment(com.avoka.fc.core.entity.Submission, org.apache.commons.fileupload.FileItem, com.avoka.fc.core.bean.RequiredAttachmentMetadata, java.lang.String), addInFormAttachment(com.avoka.fc.core.entity.Submission, [B,
        at org.codehaus.groovy.runtime.ScriptBytecodeAdapter.unwrap(ScriptBytecodeAdapter.java:55)
        at org.codehaus.groovy.runtime.callsite.PojoMetaClassSite.call(PojoMetaClassSite.java:46)
        at org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCall(CallSiteArray.java:45)
        at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call(AbstractCallSite.java:108)
        at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call(AbstractCallSite.java:128)
        at Script1.run(Script1.groovy:72)
        at groovy.lang.GroovyShell.evaluate(GroovyShell.java:518)
        at groovy.lang.GroovyShell.evaluate(GroovyShell.java:556)
        at groovy.lang.GroovyShell.evaluate(GroovyShell.java:527)
        at com.avoka.core.groovy.GroovyScriptRuntime.executeScript(GroovyScriptRuntime.java:176)
        at com.avoka.fc.core.service.job.impl.GroovyJobActionService.execute(GroovyJobActionService.java:87)
        at com.avoka.fc.core.service.job.impl.JobControllerService$8.execute(JobControllerService.java:1139)
        at com.avoka.fc.core.util.TxnExecutor.execute(TxnExecutor.java:54)
        at com.avoka.fc.core.service.job.impl.JobControllerService.performJobAction(JobControllerService.java:1137)
```

Form Space

Transact uses form spaces a means to host forms, list tasks assigned to users and show 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 several 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 from your company website which opens an unsecured form page in the portal. The available list of forms is manually maintained by your company 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 advance features of the secured portal such as Form, To-do and History pages as described below.

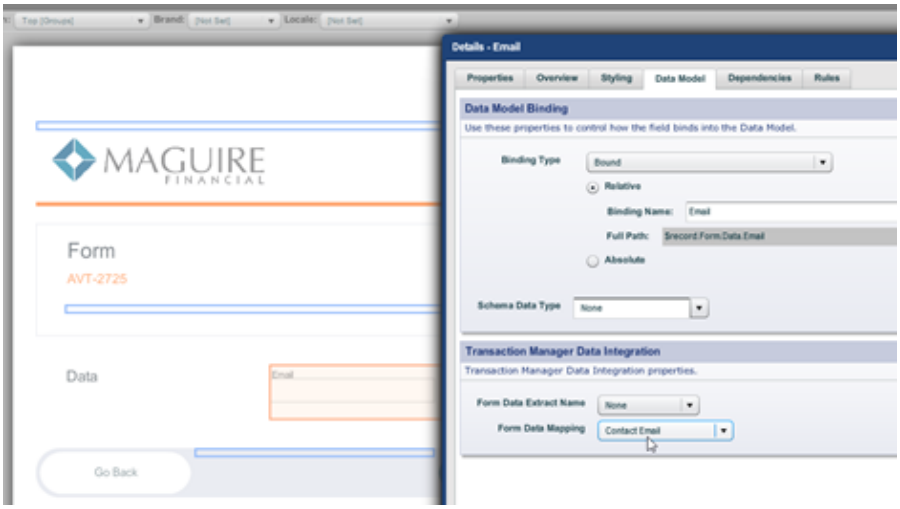
An anonymous user can still participate:

- 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 user as the job progresses.

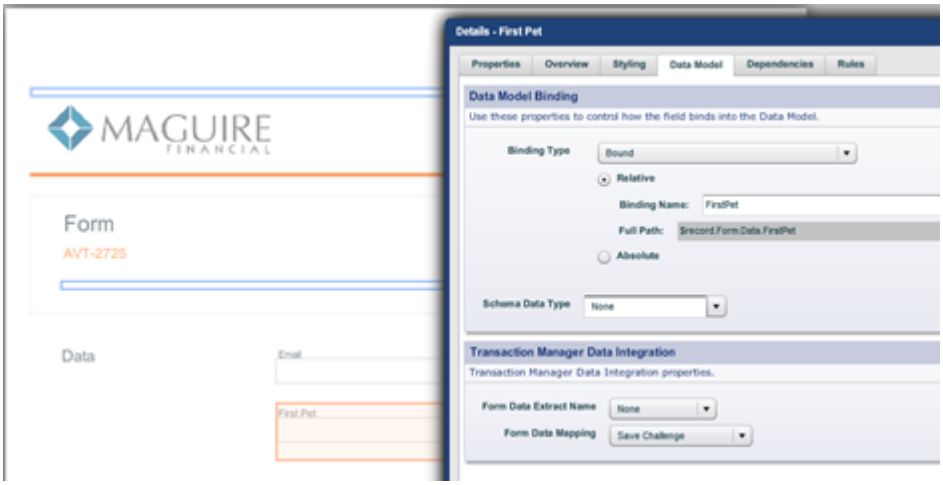
Anonymous users must provide their email address. Optionally a security question can be set to open saved forms and tasks.

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.

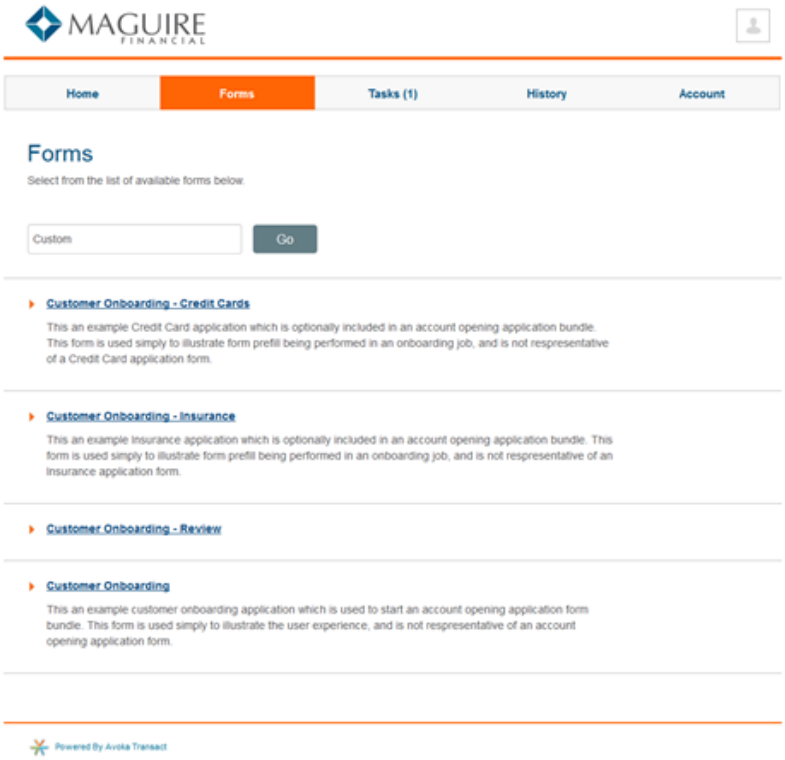
Common Portal Features

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

Forms Tab

This *Forms* tab list all the forms available for a user to start. The user can filter the results:

- Enter a string in the search box.
- Click on the *Go* button.



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 and 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 Tab

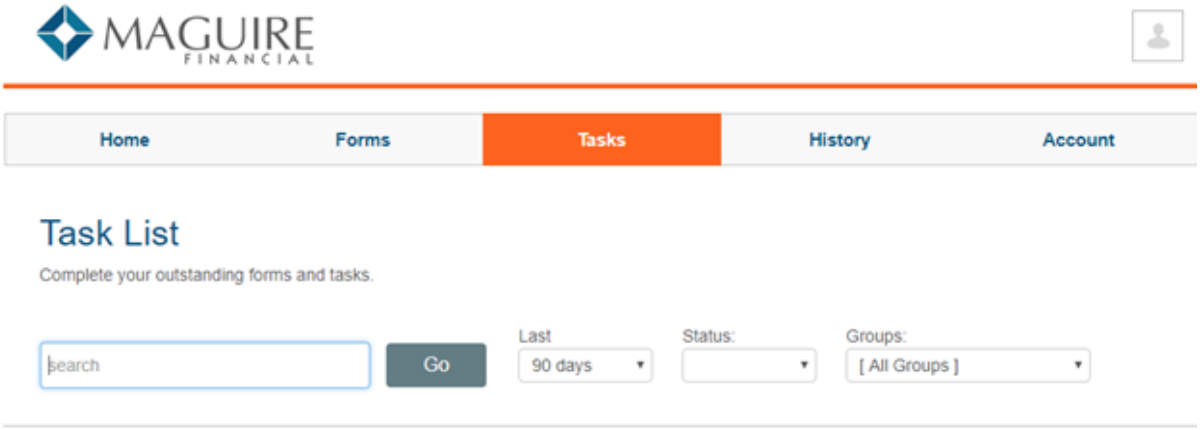
This page lists Saved Forms, Assigned Tasks and Submitted forms. Note submitted forms are ones that require additional attachments or payments and are not using most. Most Collaboration Jobs will use the inform transaction management features of Maguire which means the submitted forms will be empty.

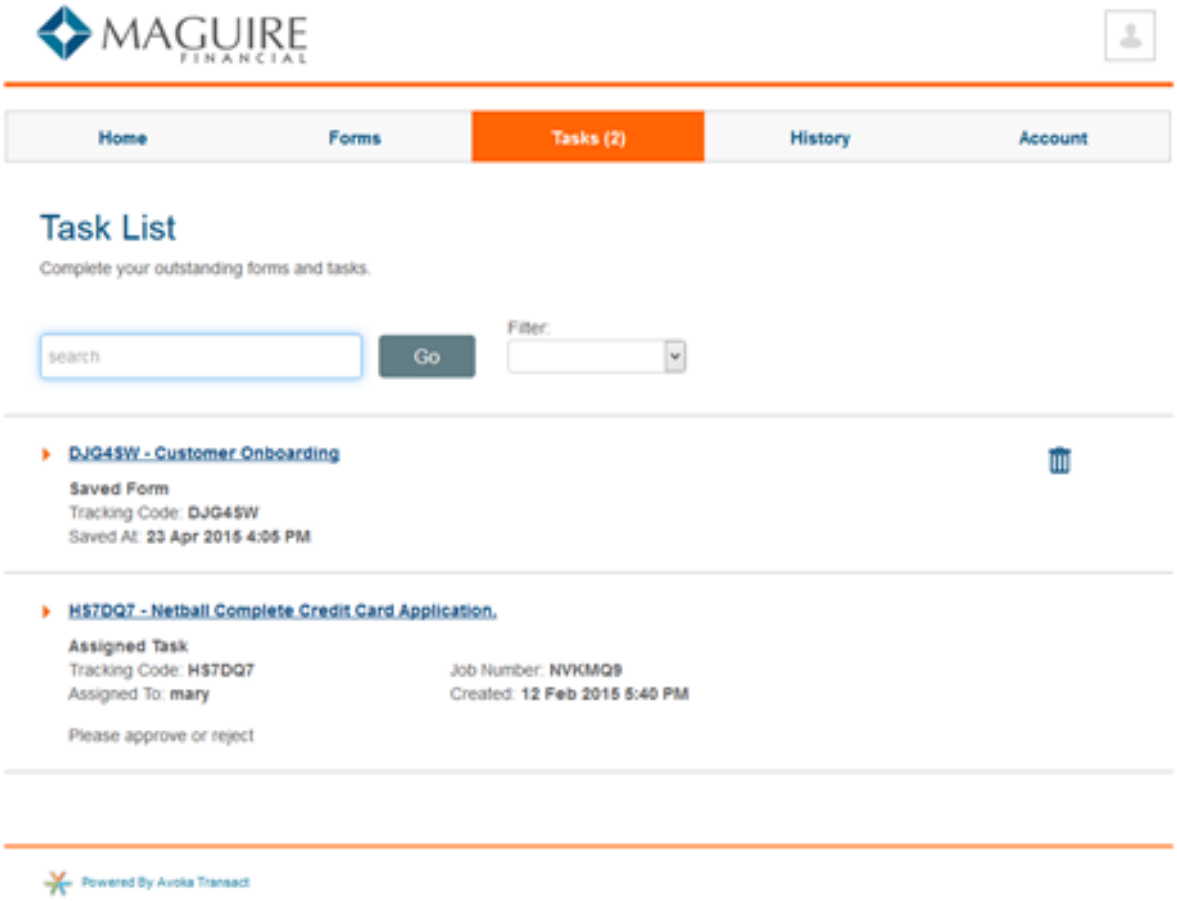
The user can filter their tasks on:

- A keyword (the job name).
- Jobs started the last number of days up to and including a year by selecting from the *Last* dropdown. This dropdown defaults to the last 90 days.
- Jobs that are currently *Assigned*, *Saved* or *Submitted* by selecting from the *Status* dropdown.
- When the form space property *Search Filter Group Select* checkbox is ticked:
    - Form groups by selecting from the *Groups* dropdown:
        - Blank for any tasks that are assigned to the user account that you are logged in to.
        - *[All Groups]* for all form groups.
        - A chosen form group.

otherwise

- Select the *Group Items* checkbox to only show group items.





Assigned Task Types

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

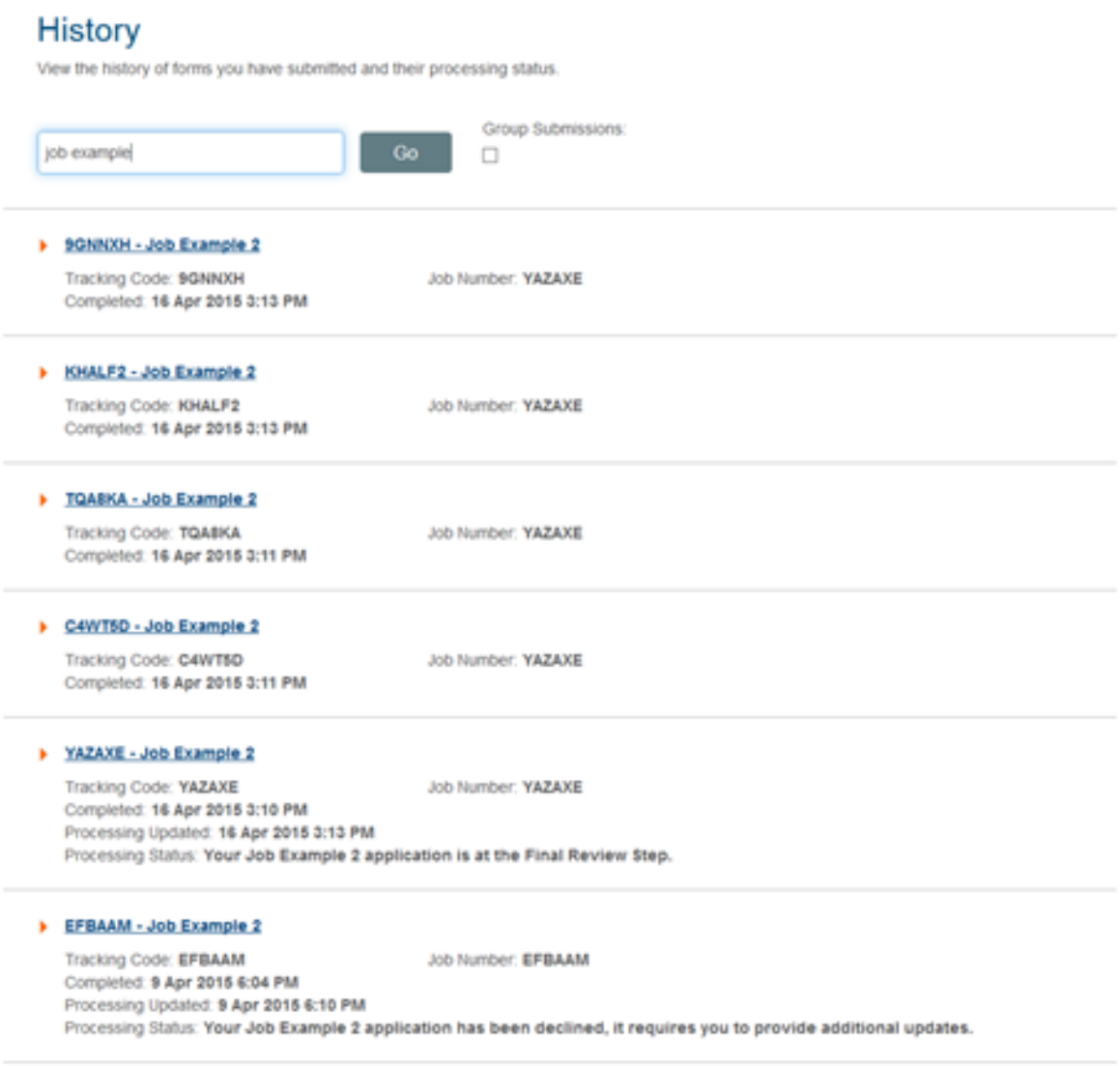
There are 3 task types:

- **Form Task**: The form xml can be pre-filled. Assigned to a User or a Group and displayed to secure users in the To-do 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 Tab

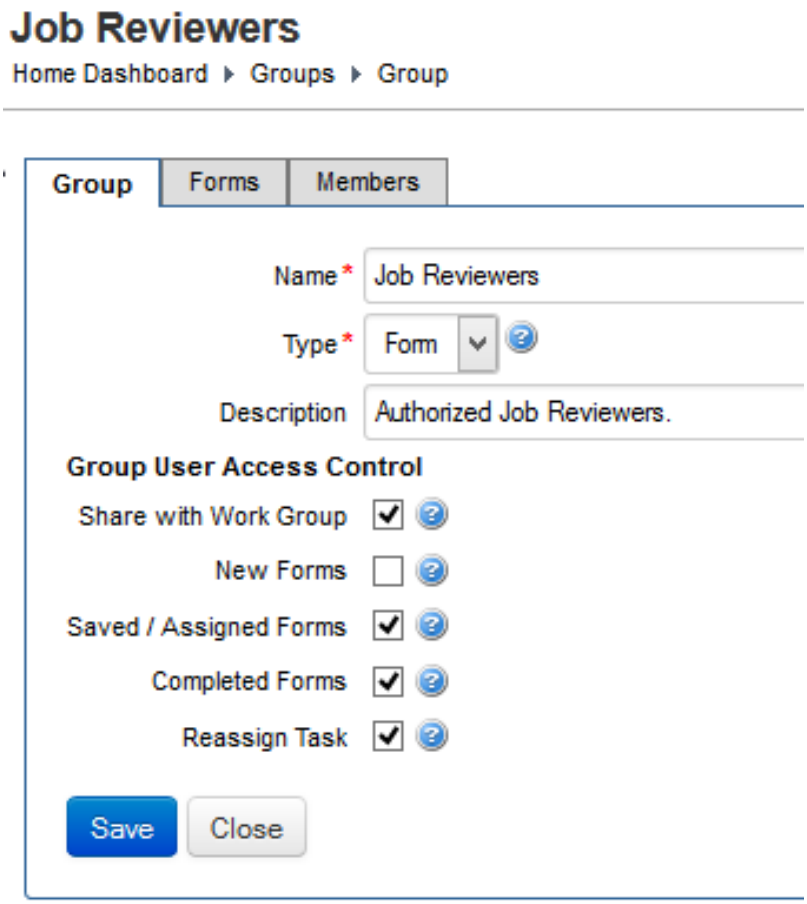
The history page shows a list of all submissions that a user had. A submission that starts a job can contain a processing status that is updated as the job progresses through its steps.

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.

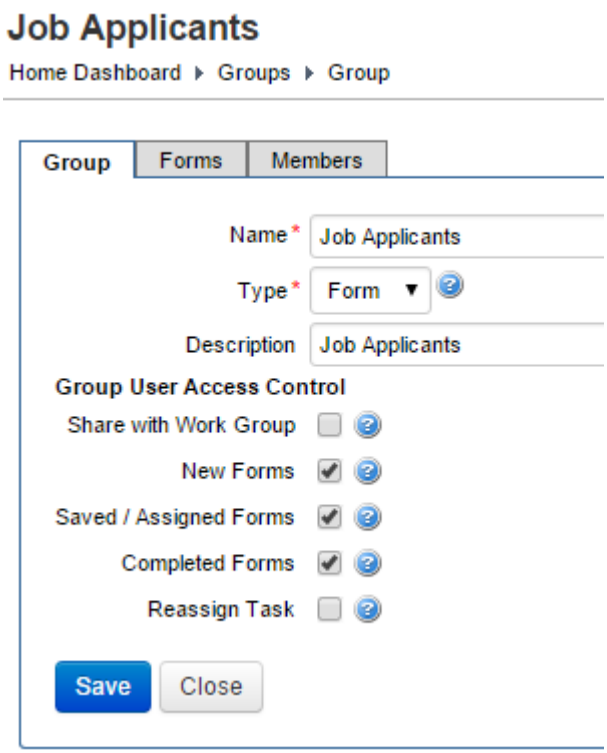
Completed Forms: Access control allowing users to view completed form group submissions in the history page.

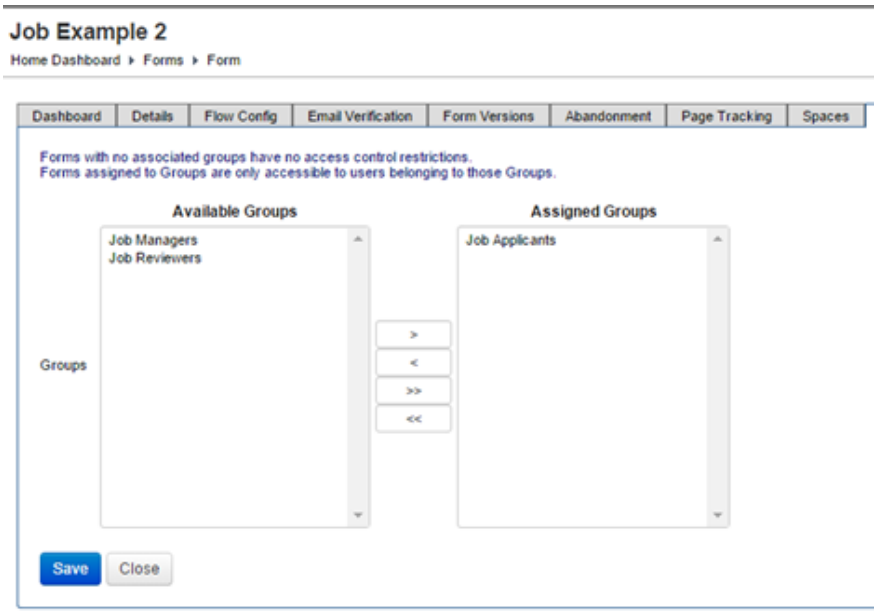
Reassign Task: Access control allowing users to perform Claim, Reassign tasks to other users.

Authenticated User Roles

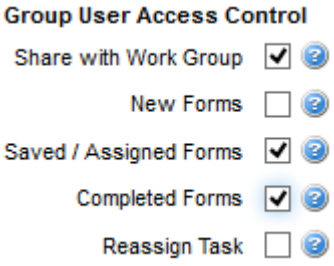
There are several authenticated user roles

- **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.

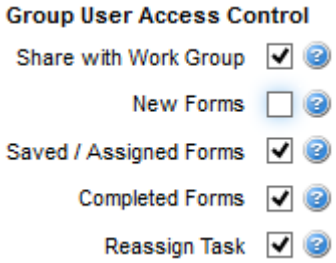




- **Reviewers**: These will be reviewing jobs as part of their job on a daily basis. They might be part of a group that looks after a particular step.



- **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 user's submission is ignored.

2. **Task Enable Claiming**: (Transact Version 4.1+)
   Task claiming can be setup for Group tasks. Claiming allows a task 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

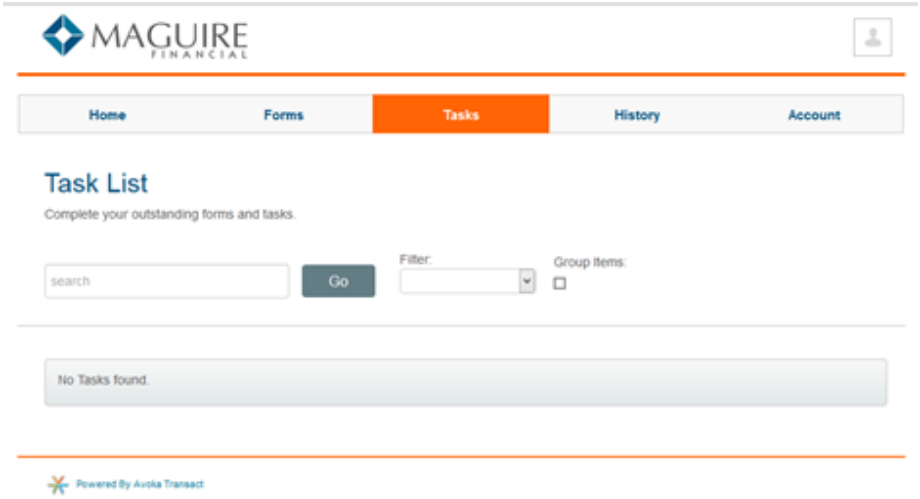
Group Task Claiming needs to be configure when the task is created. This is done the Job Task Assign - Action by the **Task Enable Claiming** action property as seen the extract from Job Example 2 below.

Job Task Assign - Action Definition Expand source

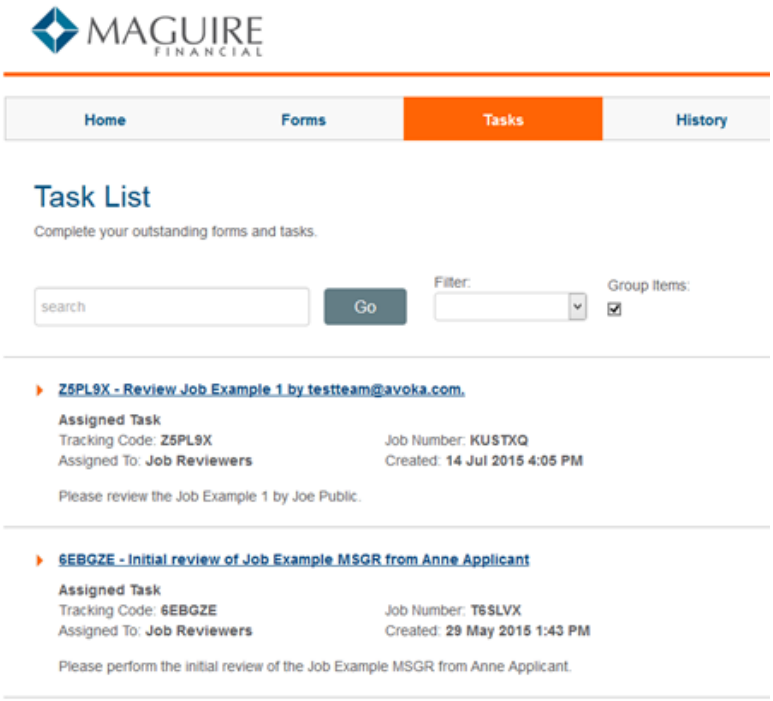
```
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "Your ${submission.formName} application is at the Final
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 Group.

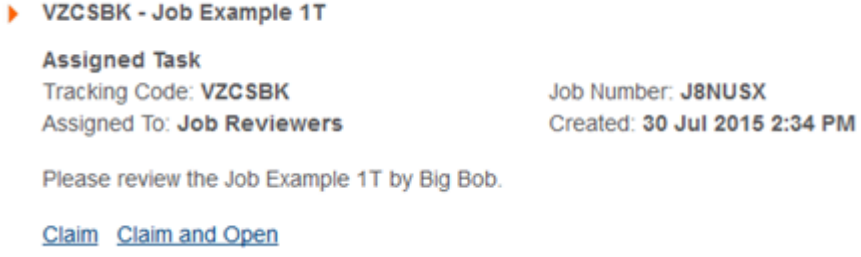


Roger can open the Job Example 1 Task by clicking on the link. This opens the form.

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.

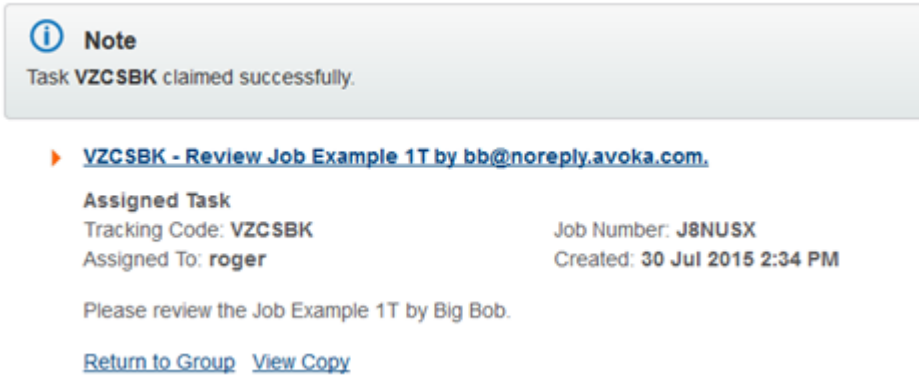
Example - Task Claiming Enabled

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

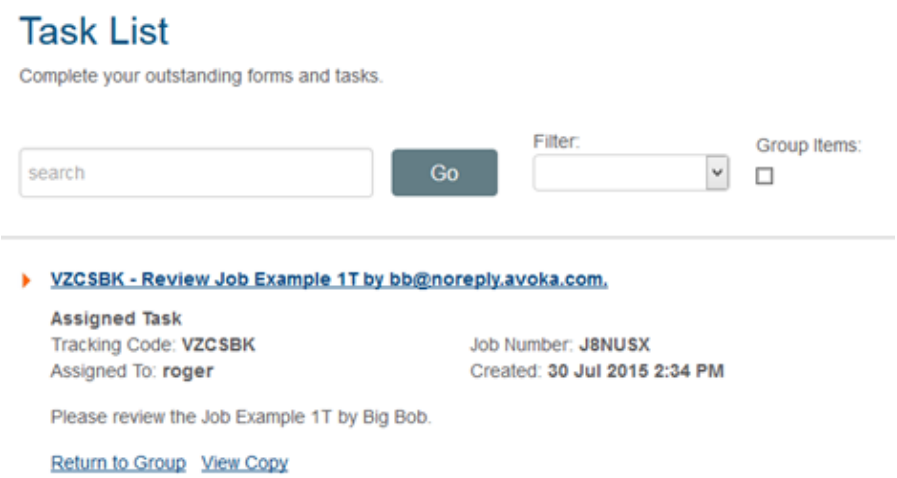


Clicking **Claim and Open** will open the form. Other users will be restricted from editing the form.

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



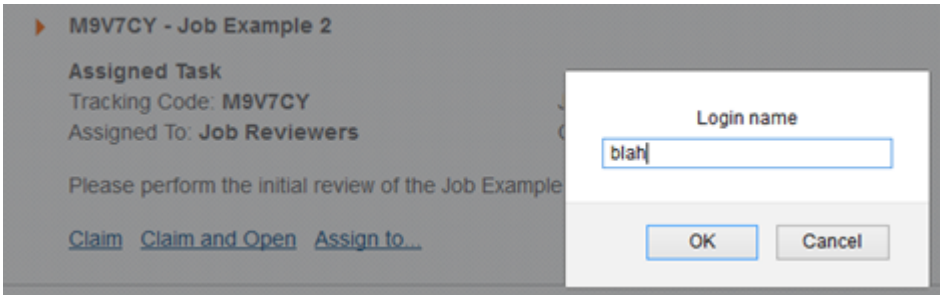
Actions:

- **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.

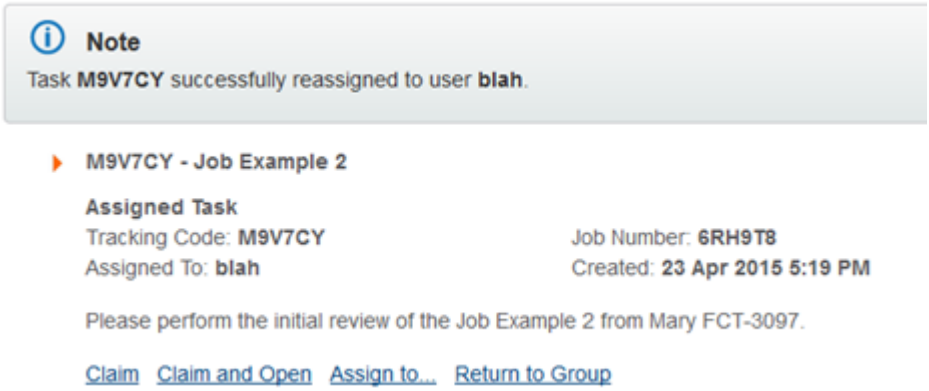
Opening the form as a Job Manager

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

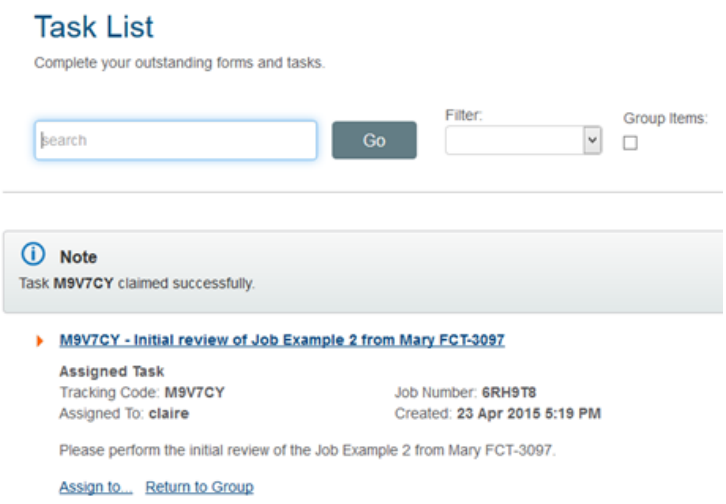
Assign To allows the form to be allocated to someone else in the group (Brian Lah). We need to enter their login name (blah) as follows.



We can see the task is now assigned to blah



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



Job Coordinator

A job coordinator or business system coordinator monitors and manages collaboration jobs. As coordinators, they do not necessarily get involved with actioning tasks. Instead, they manage the overall jobs processing and only intervene when required to do so.

With large scale operations, such as those having several regions and job coordinators managing their group(s) of jobs, the implementation of job groups facilitates the jobs to be filtered on one or all job groups.

Reviews Tab

Job coordinators can switch to the *Reviews* tab in the form space and access all review and approval jobs. They may wish to search for a job and check:

- The job details.
- The step the job is up to.
- Information about the job's current task and its status.
- History of the job's previous submissions including submission receipts.

The *Reviews* tab allows the coordinator to check for jobs by using the filtering options to obtain a suitable search result list (refer to screenshot below).

1. You may wish to filter the list by one or more of the following:

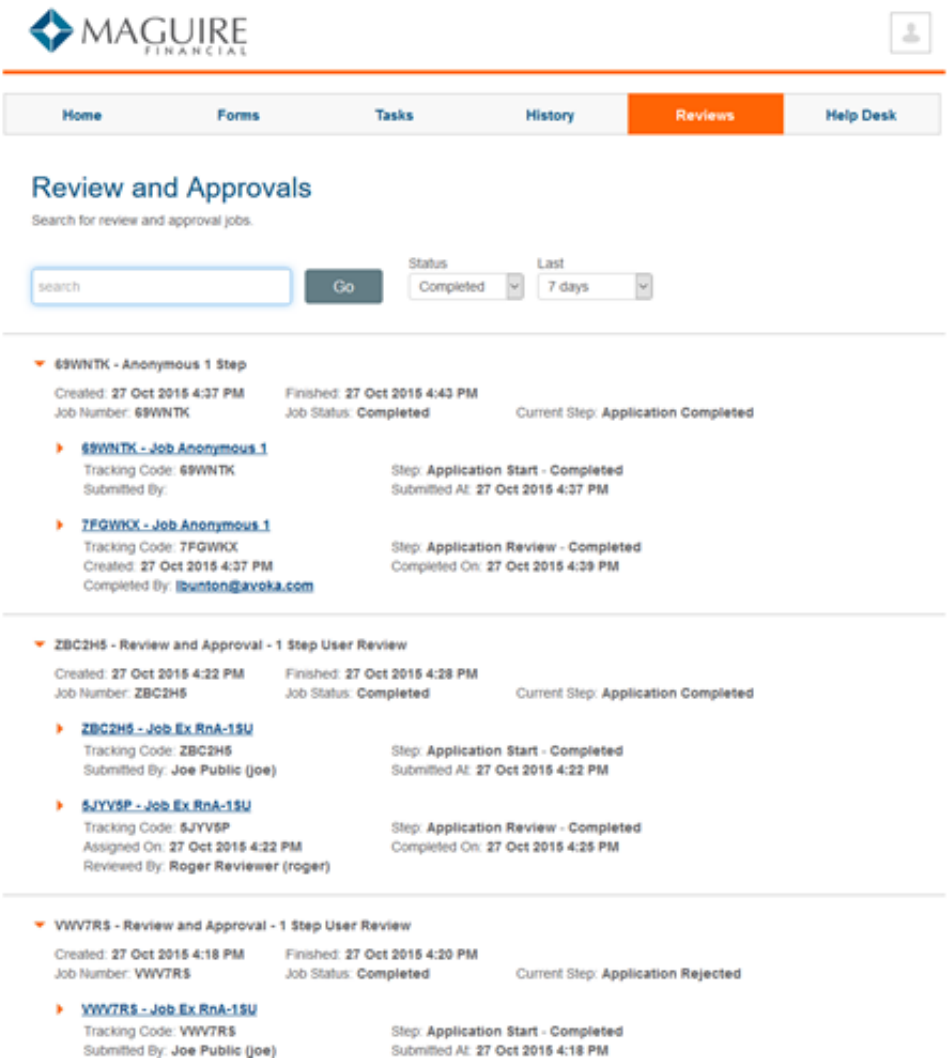
- In the search box, enter one of:
    - Job number
    - Tracking code
    - Contact email
    - By default, only the review and approval jobs for the last 90 days will be listed. To see the last jobs going back to a different period, select one of the periods from the *Last* dropdown.
    - By default, only the review and approval jobs that are in progress will be listed. That is, the *Status* dropdown will initially have the ***In Progress*** option selected. To see jobs of:
        - All statuses, select blank from the *Status* dropdown.
        - A specific status, select the status from the *Status* dropdown.
        - By default, all jobs from all the job groups, ***[All Groups]***, that you are responsible for will be listed. Select from the *Job Groups* dropdown:
            - ***[No Groups]*** to see jobs that had not been allocated to any job groups.

This option is required because jobs not allocated to any job group do not appear when ***[All Groups]*** is selected.

In the case where your company allocate all jobs to job groups, this option will allow for the changeover period where you can still see jobs before the job groups were introduced.

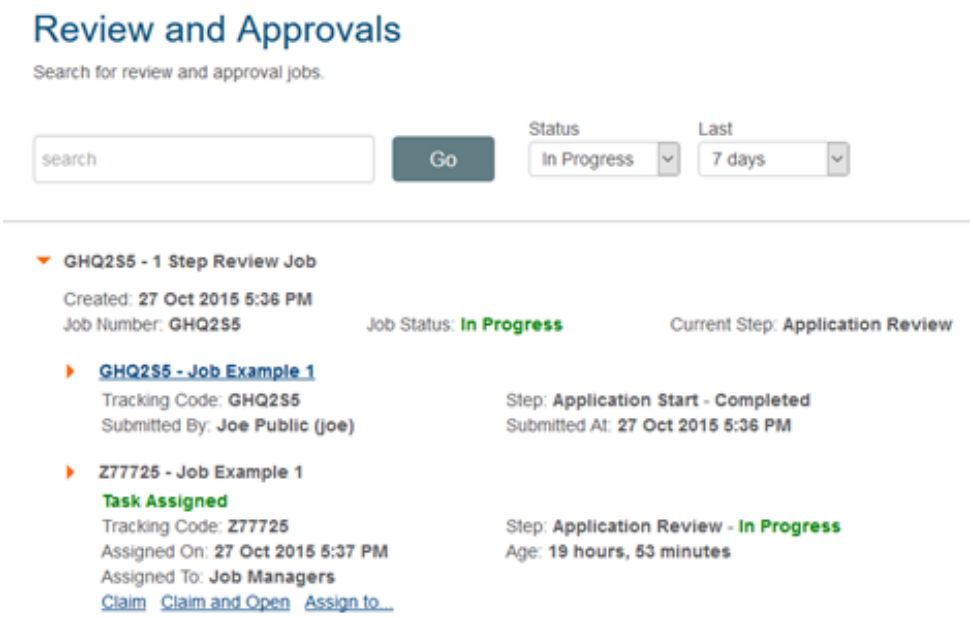
- A specific job group to see jobs associated to that job group. Only the job groups that you are responsible for will be listed in the *Job Groups* dropdown.

1. Click the *Go* button.



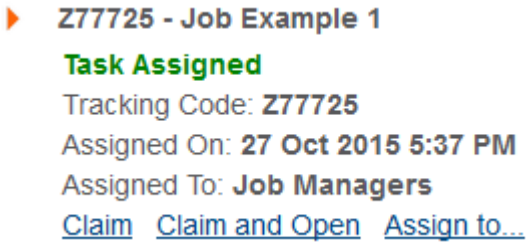
In Progress Jobs

The coordinator can select the jobs that are 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 *Reviews* tab, a job coordinator must be given the following permissions:

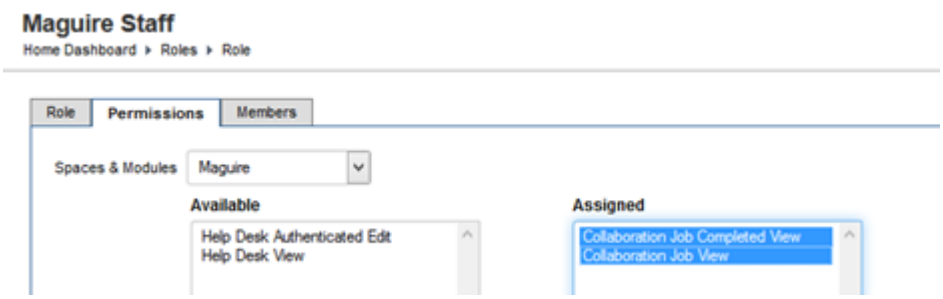
- Collaboration Job View
- Collaboration Job Completed View

Every Transact form space has a role created with the name, <form space name> *Staff,* which contains these permissions as per the screenshot below.

For instance, a *Maguire Staff* role exists for the *Maguire* form space, having the correct permissions to allow job coordinators access to the *Reviews* tab from within the *Maguire* space.

To view the permissions assigned to the *Maguire Staff* role:

1. Navigate to *Security* from the menu bar, and then click *Roles* from the dropdown menu.
    a. Find and click on either the *Maguire Staff* link or the *Edit* icon () in the *Action* column from the same row.
    b. Switch to the *Permissions* tab.
    c. Select *Maguire* from the *Spaces & Modules* dropdown.



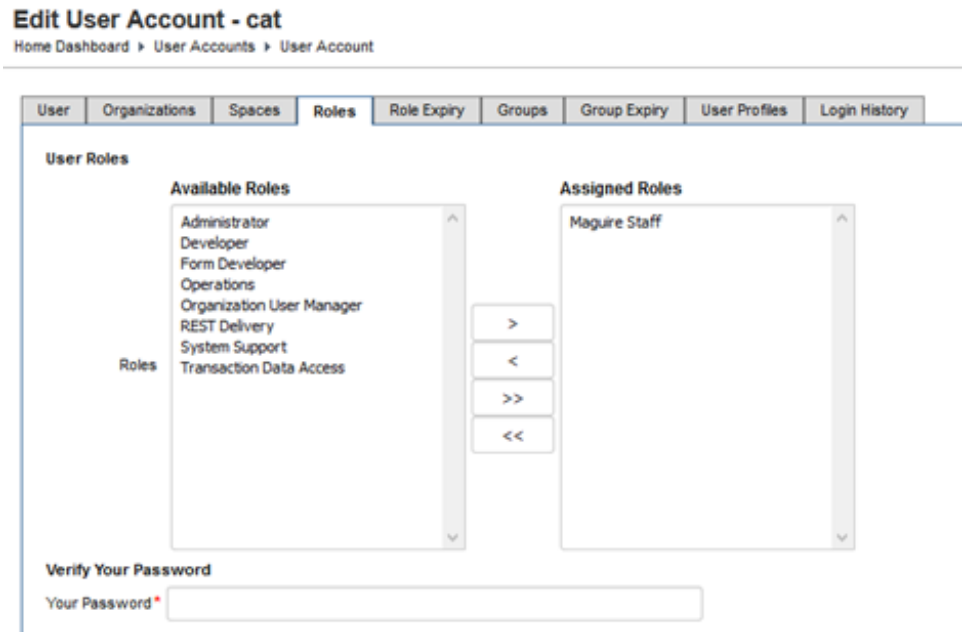
It is not recommended to alter these standard roles.

To function as job coordinators, their user accounts need to be assigned to the applicable:

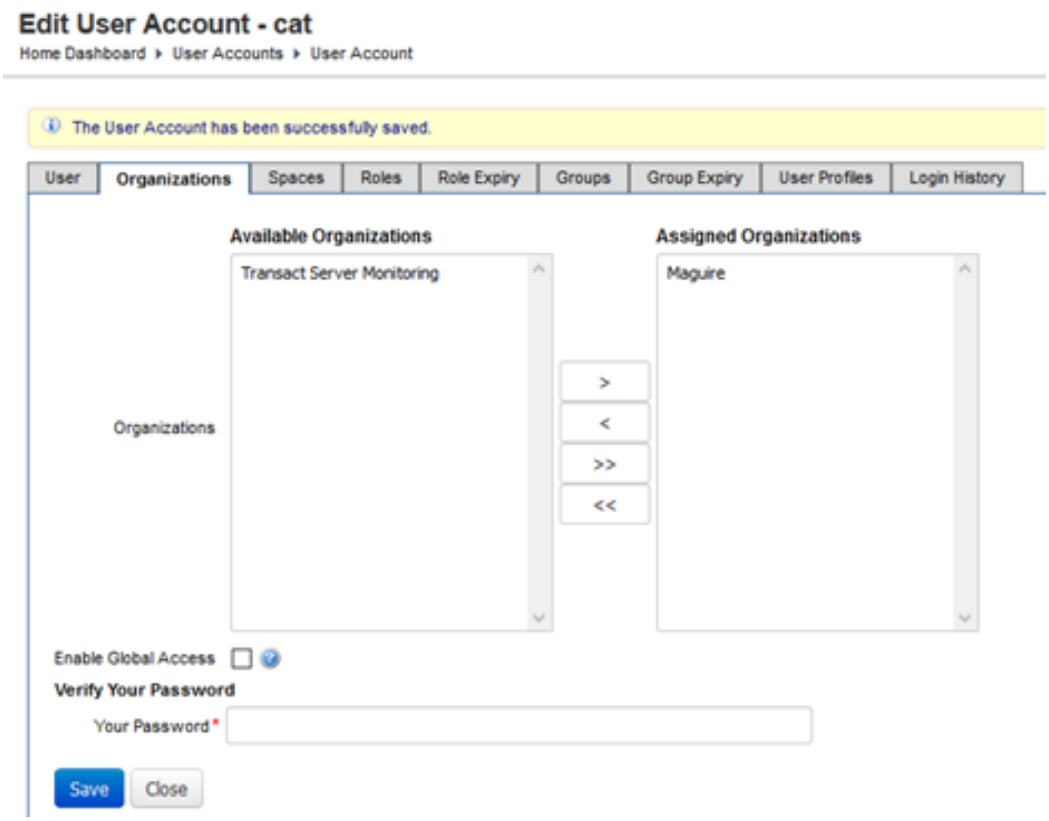
- Form space staff role(s).
- Job groups.
- Organization(s).

User Account

As an example, the job coordinator responsible for the *Maguire* form space needs to be assigned to the *Maguire Staff* role.



To view all the jobs including their receipts, the job coordinator user account must also belong to the same organization(s) that the related forms belong to. For instance, the user account *cat* is a job coordinator for all jobs linked to forms for the *Maguire* organization by being assigned the *Maguire* organization.



Step By Step Examples

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

- 1 Step Review Job
- Multi Step Group Review Job
- Customer Onboarding Job - Form Bundle
- Maestro Customer Onboarding Job - Form Bundle
- Maestro Form and Job Example Assets

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"
> ```

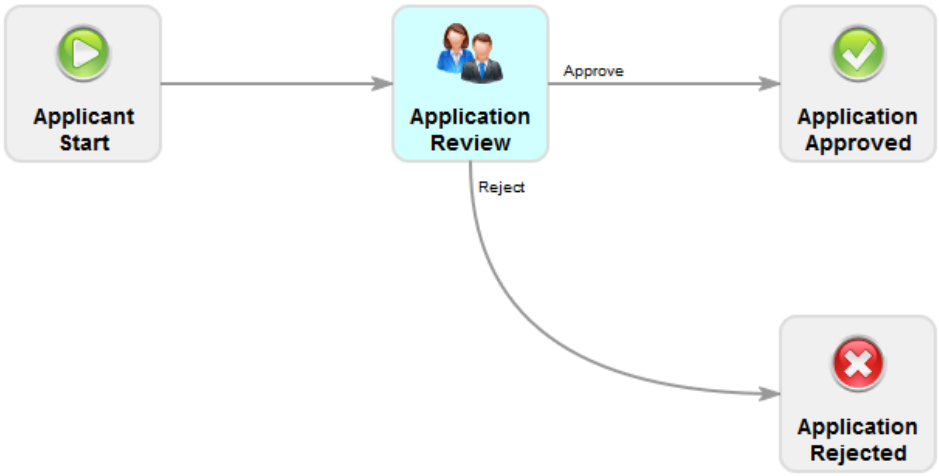
Create Test Users

```
/**
 * 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"
    }
}
```

1 Step Review Job

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:

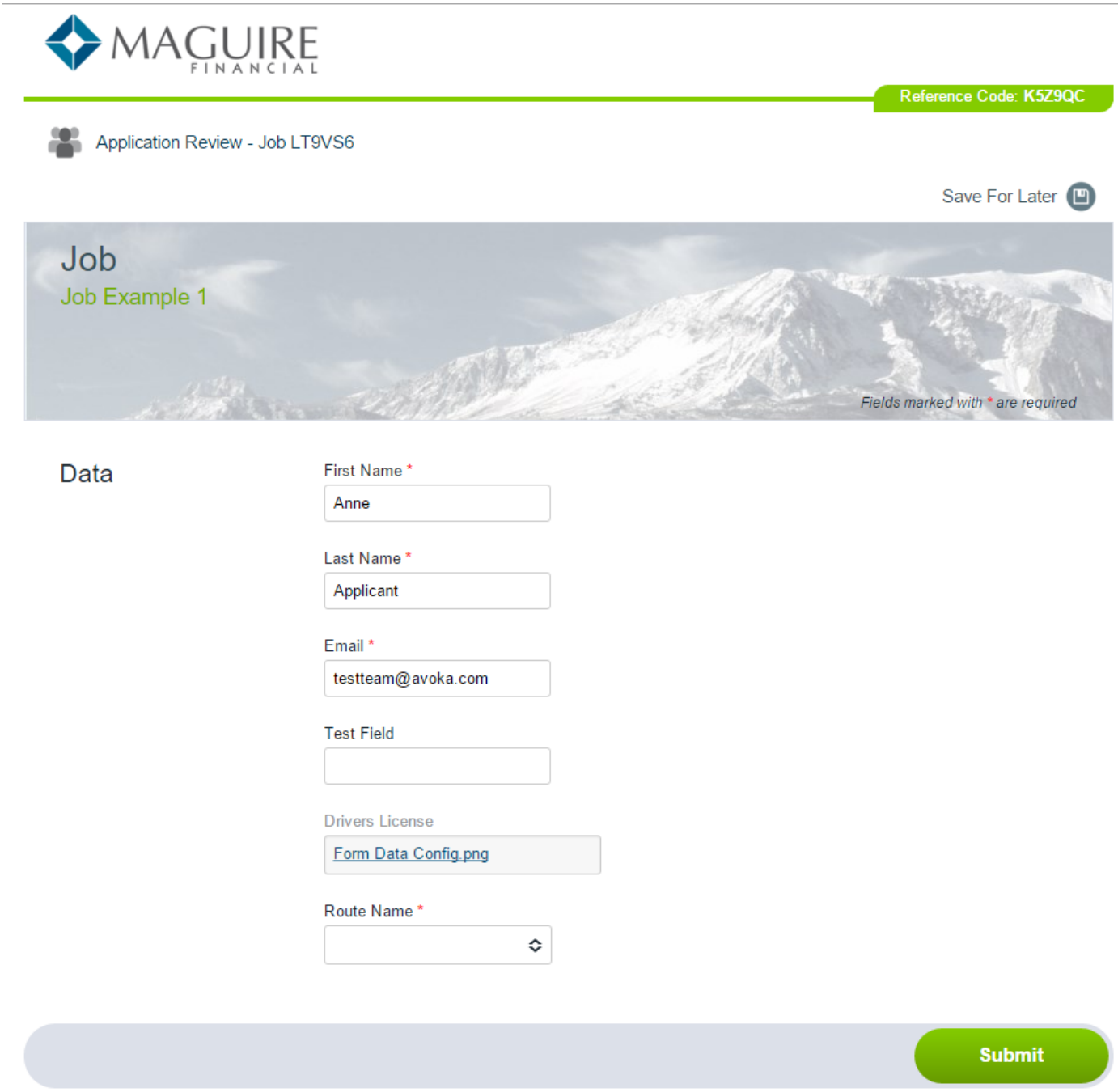
- An application form is filled in and submitted by a user.
- The form next goes to the **Application Review** step where it is reviewed by a member of the **Job Reviewer** group.
- The reviewer may **Accept** or **Reject** the application.
    - If it is rejected the job moves to the **Application Rejected** 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. and for Composer.
>
> Other forms can use the same Job Controller as long as they have these same Data Extract field.

Other:

Test Field (Text Field)

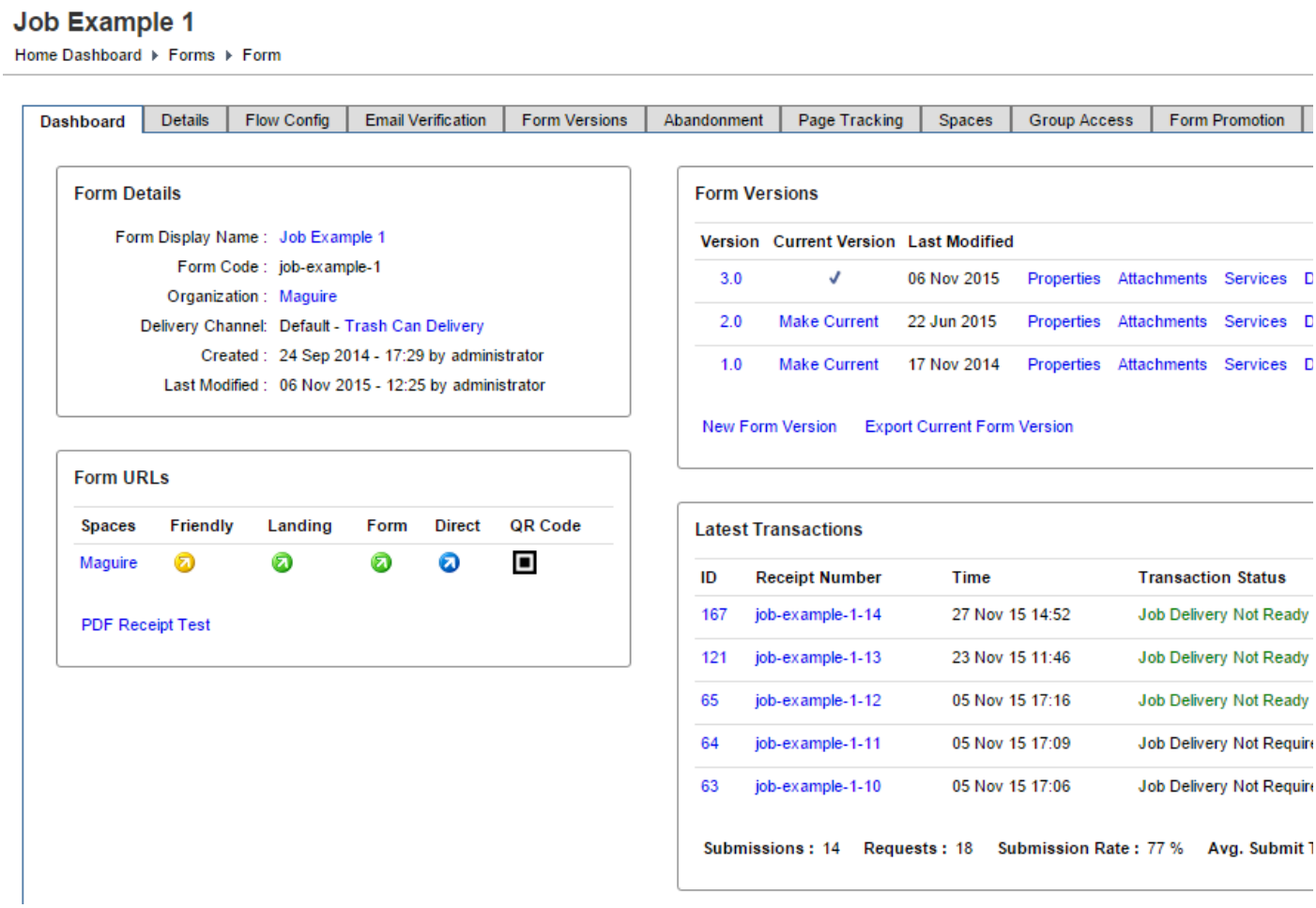
Drivers License (Inform Attachment Field). This is setup so it's read only at the Application Review step.

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 > 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.

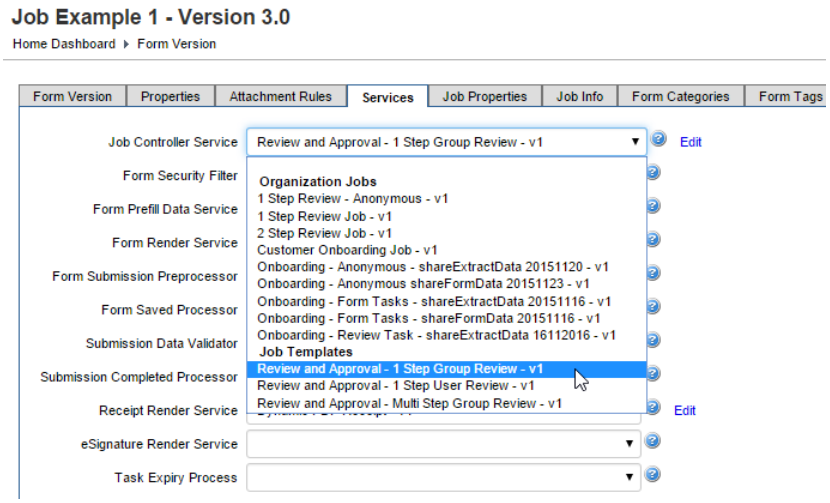
Set the Job Controller Service

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 > 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.

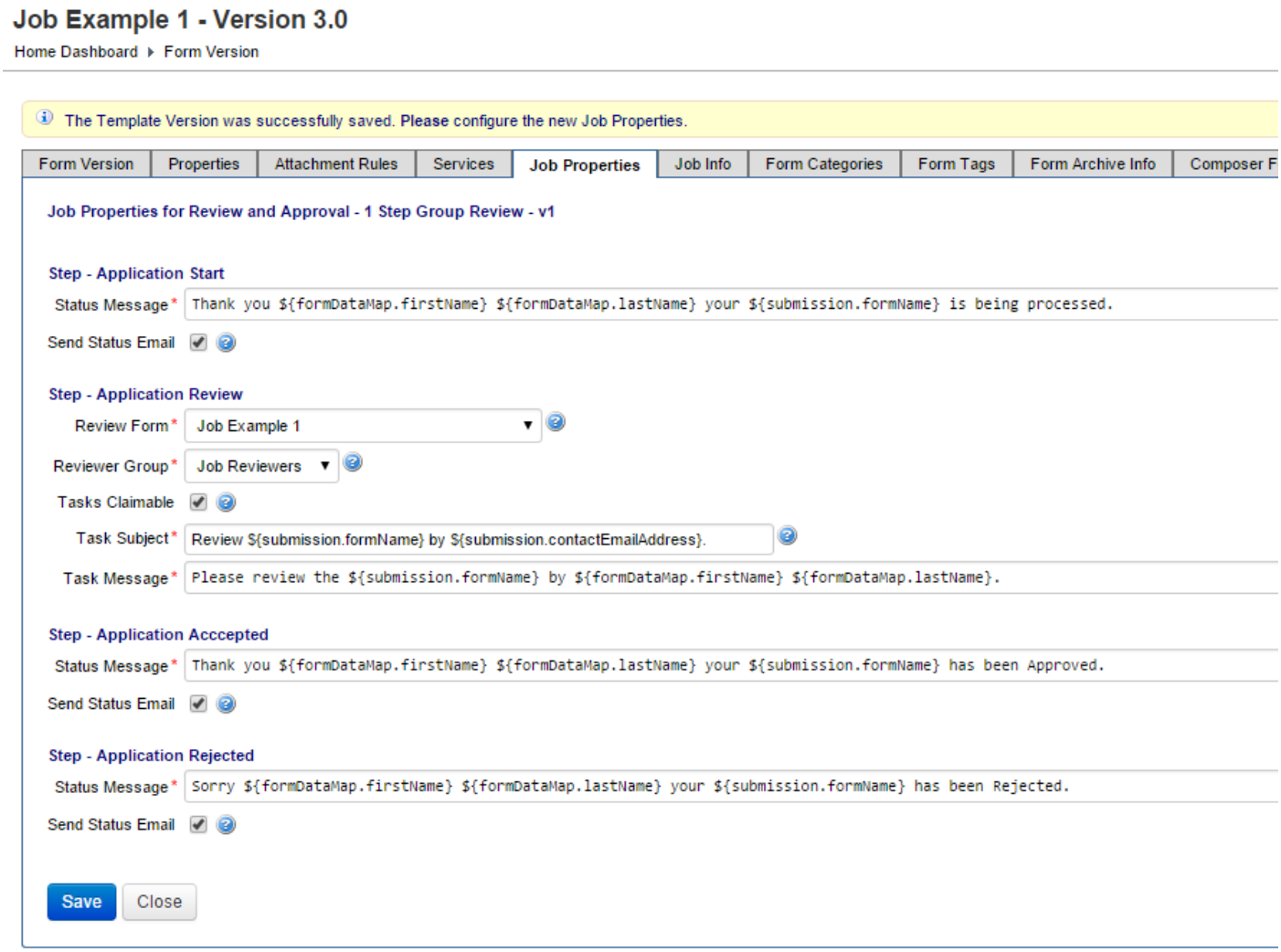
Under the Step - Application Review

> 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 > 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..

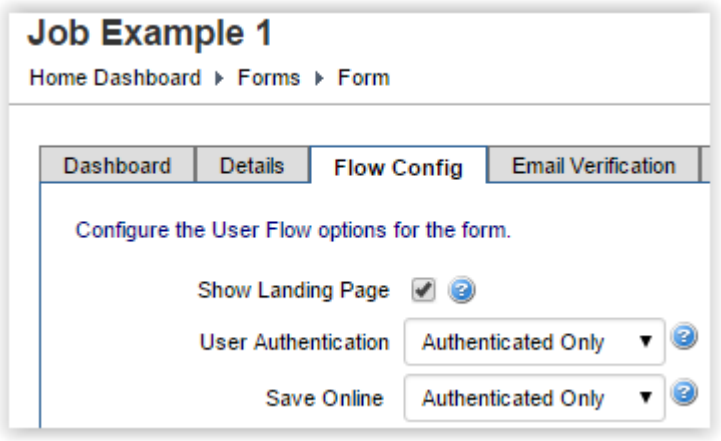
1. Anne Applicant who does the Application Submission
2. Roger Reviewer who does the Application Review
   We will add Roger to the **Job Reviewers** group.

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 > 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**.



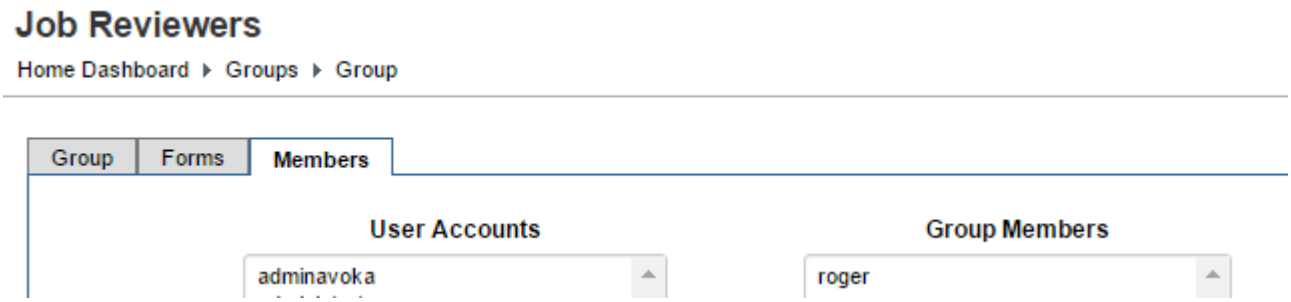
Set Job Reviews Group

> 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

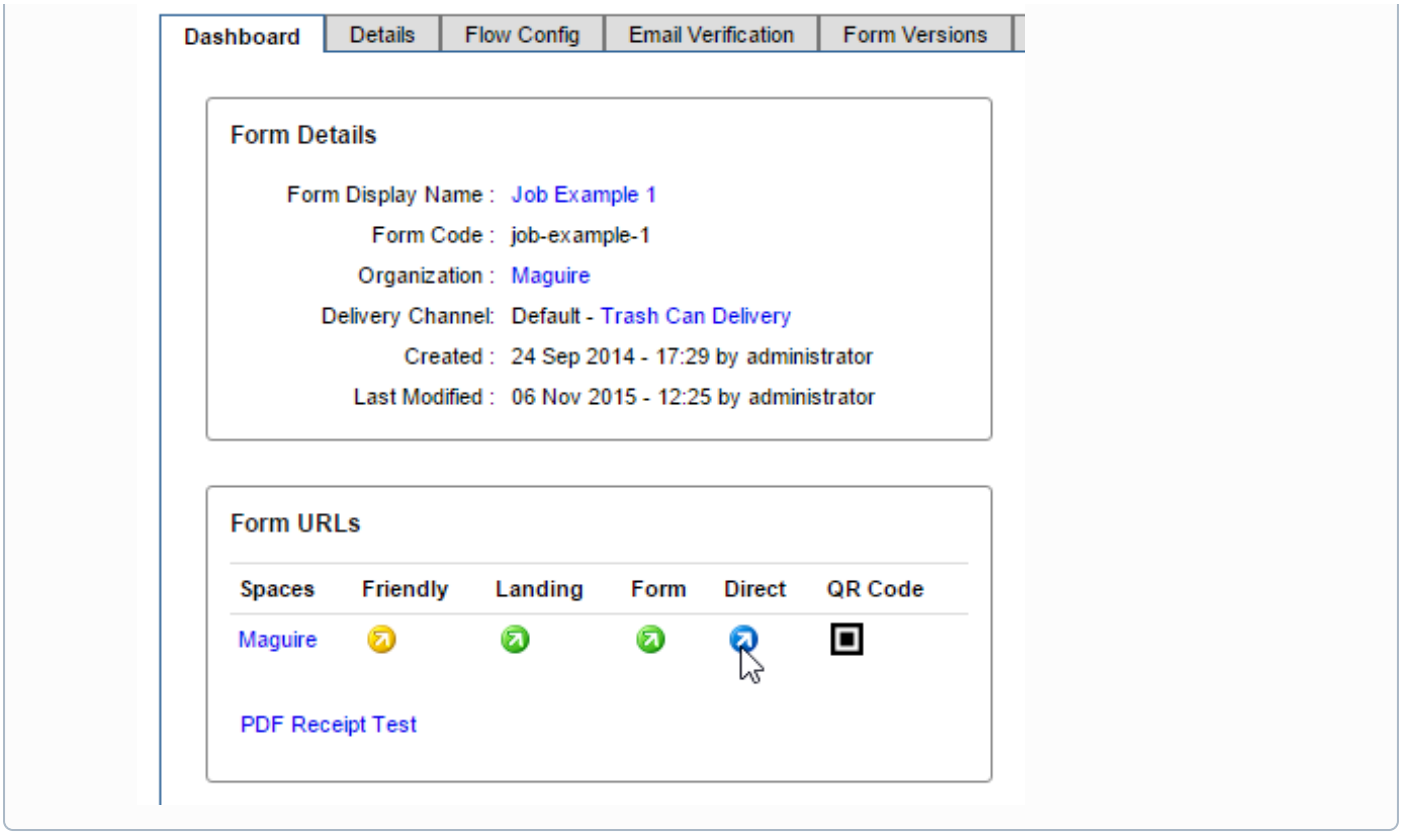
Note

To Run the example you should be familiar with concepts covered in Form Space > Secured Form Space

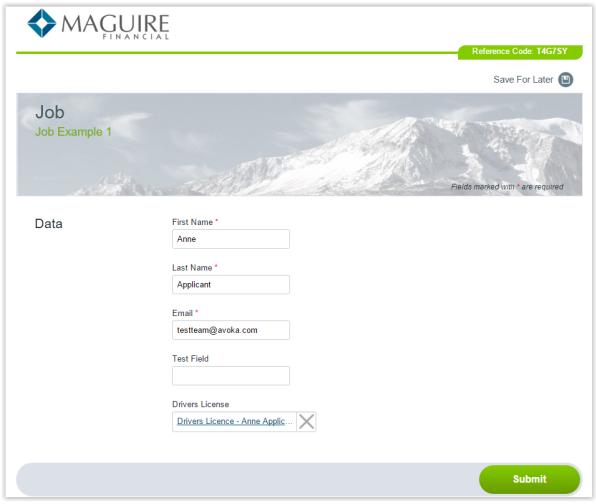
** **Tip for Development and Testing**

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.

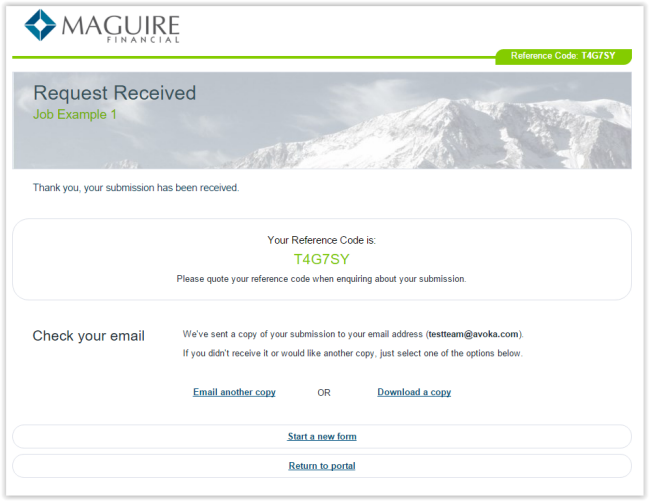


> Upload an image file to the Drivers Licence Field.

> 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.

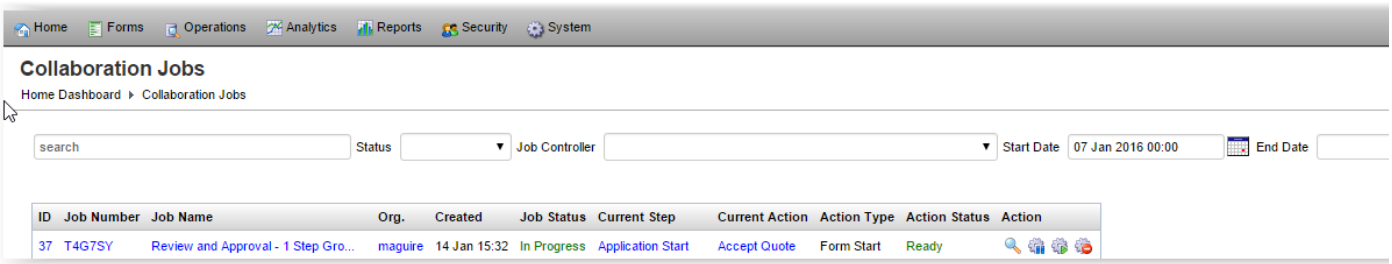
> Menu Operations > Collaboration Jobs

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 > Job Execution)

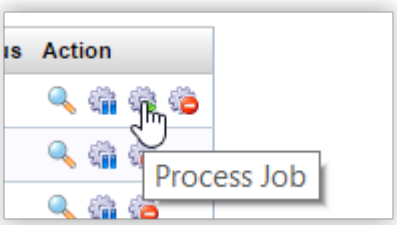


Note

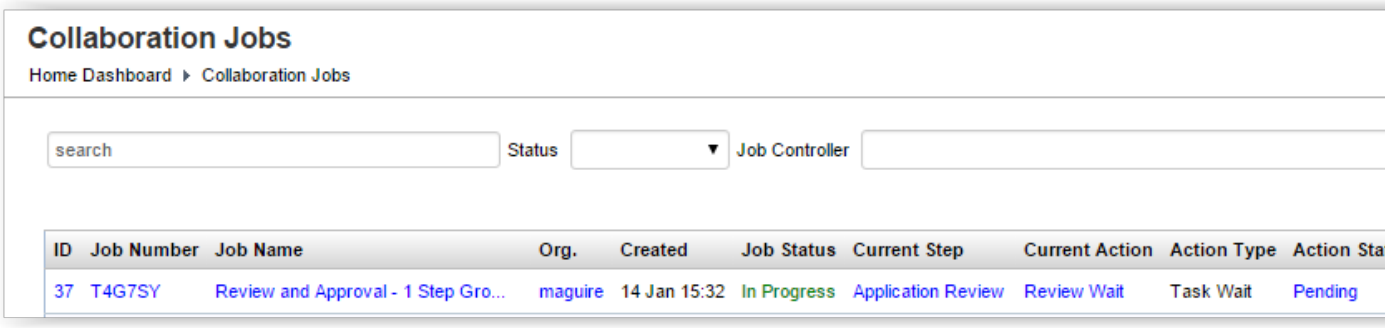
Be familiar with Admin Operations#Collaboration Job Management

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.

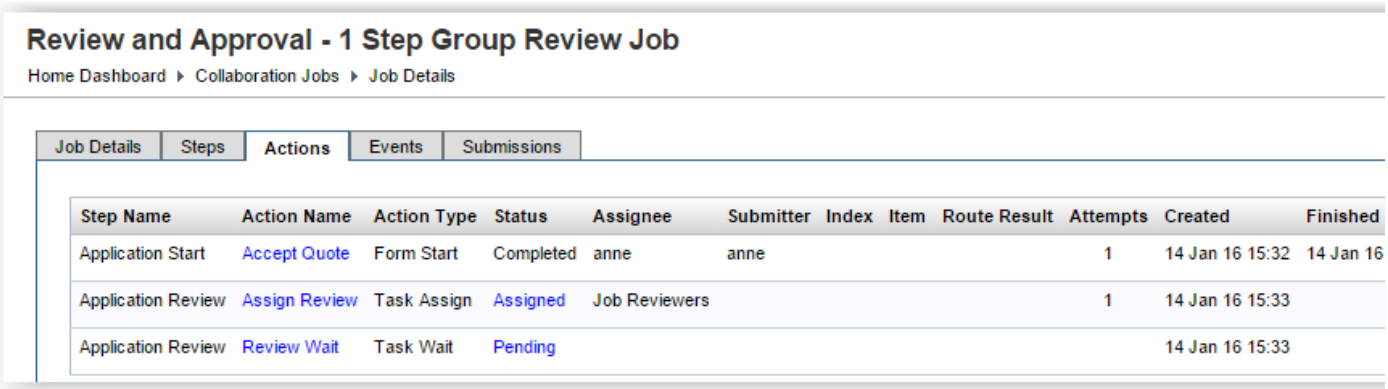


After the job has been processed it moves to the **Application Review** step



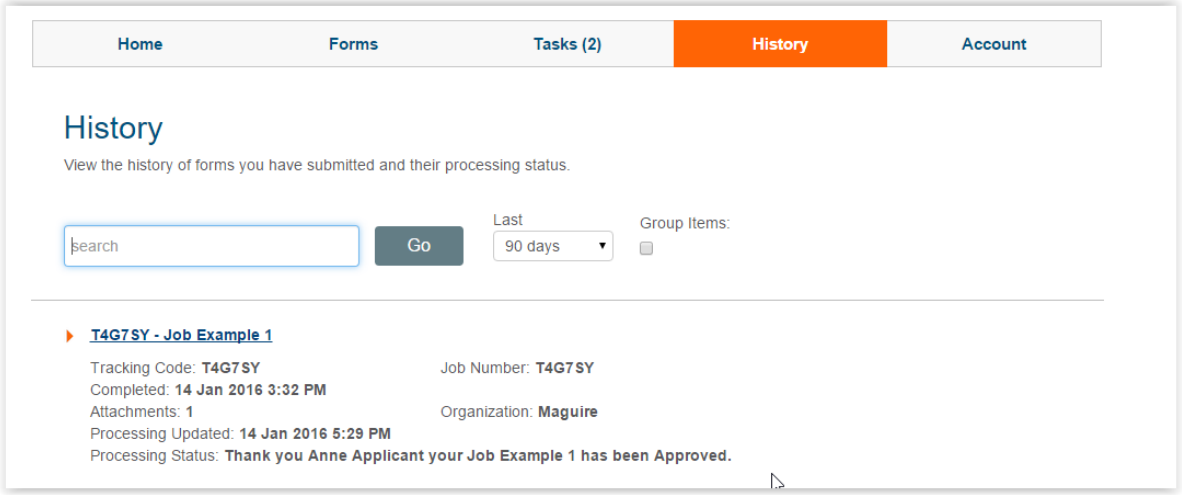
> Drill down into the Actions tab for this Job instance.

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



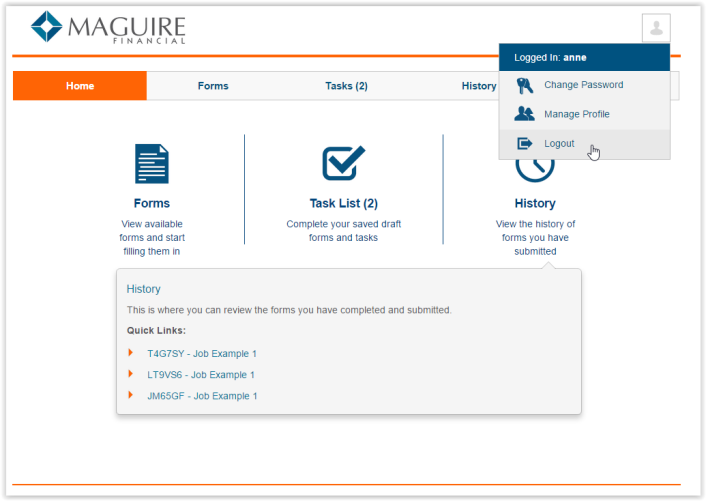
Maguire Form Space - Anne

> Click on History Tab



We see the Processing Status

> Logout as Anne



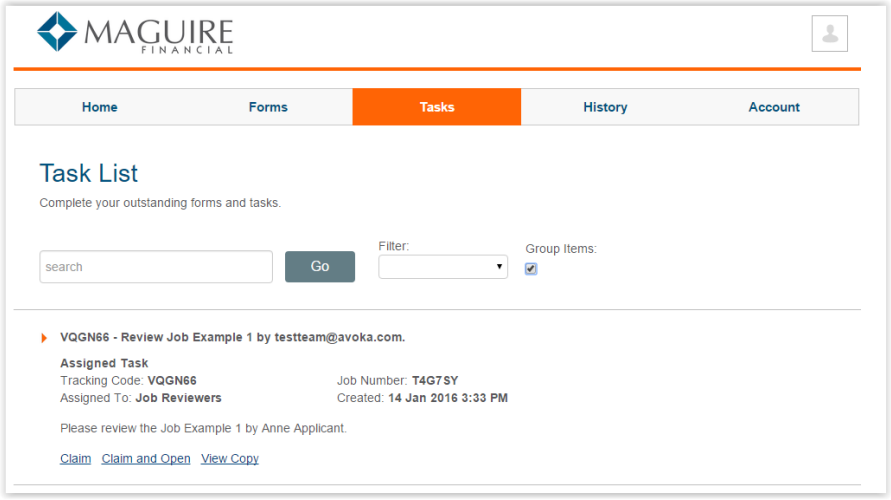
Maguire Form Space - Roger

> 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 > Rules

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

> Select Route Name: Approve

>Select Submit

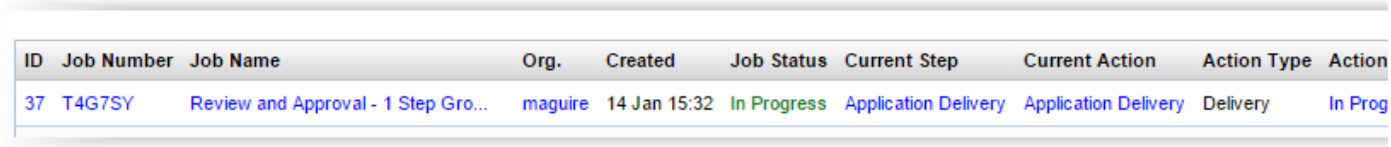
The Confirmation page will be displayed

> 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

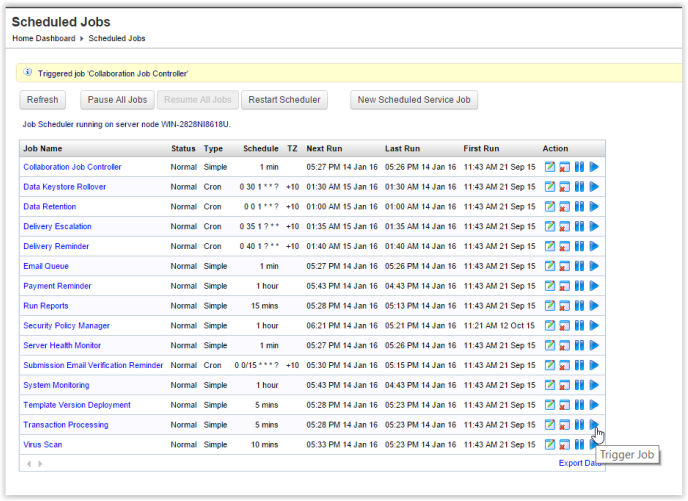
Note

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.

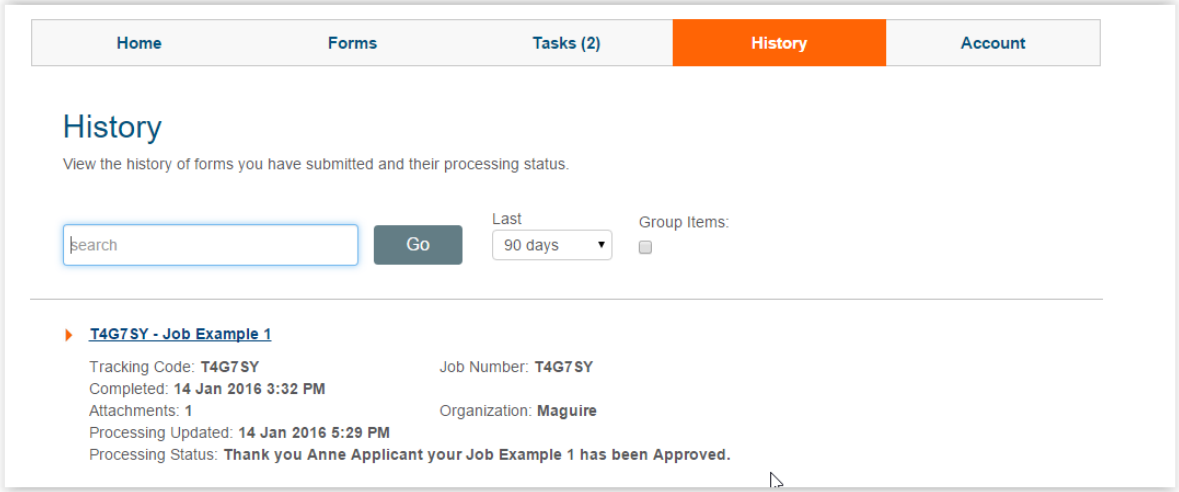


Maguire Form Space - Anne

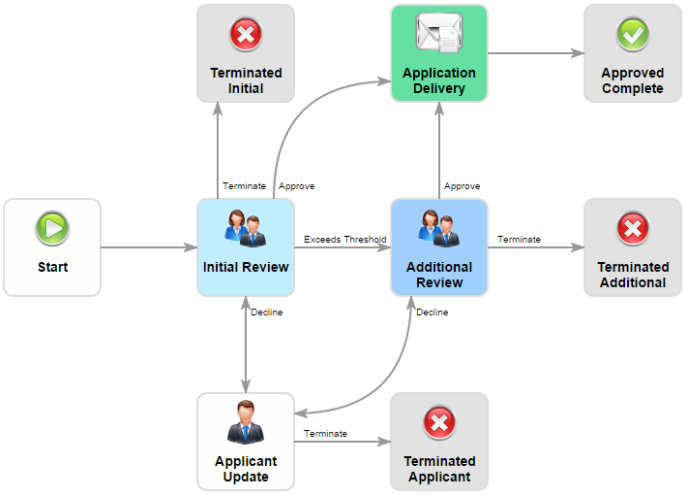
> Login as Anne

> 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.

Multi Step Group Review Job



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).

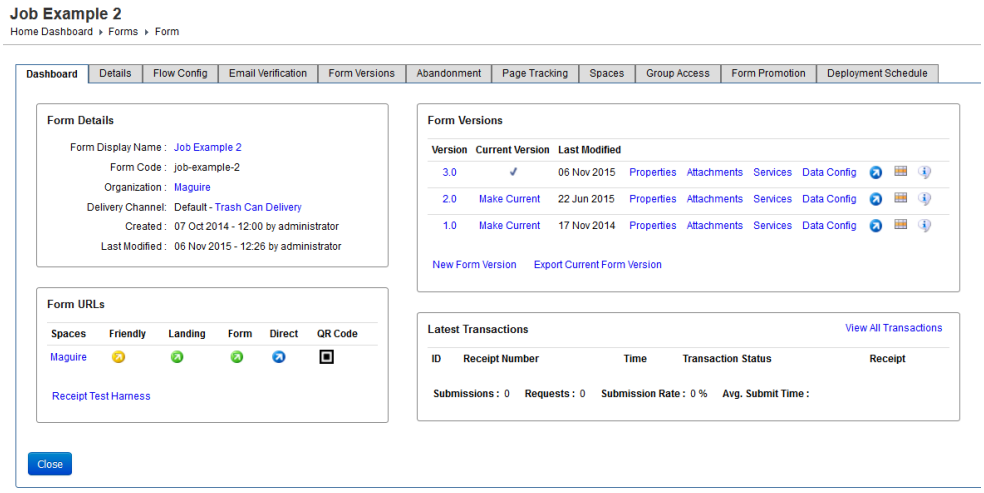
The Setup of the Form, Job Controller and Actors are discussed below.

> **Note**
>
> This example assumes you are familiar with concepts introduced in 1 Step Review Job 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:

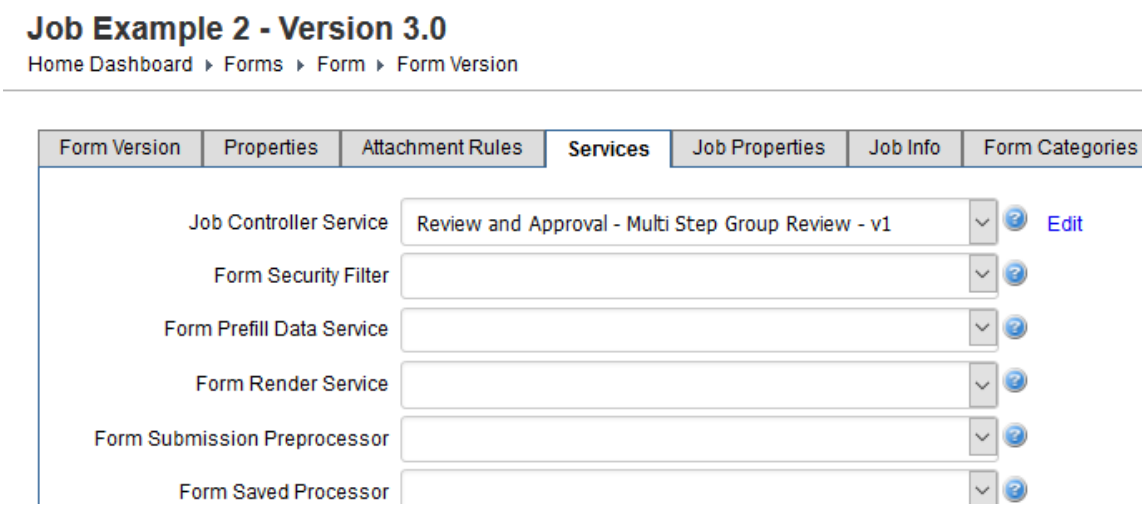
Form versions 1 and 2 were part of previous releases. They are both minimal forms like Job Example 1.

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

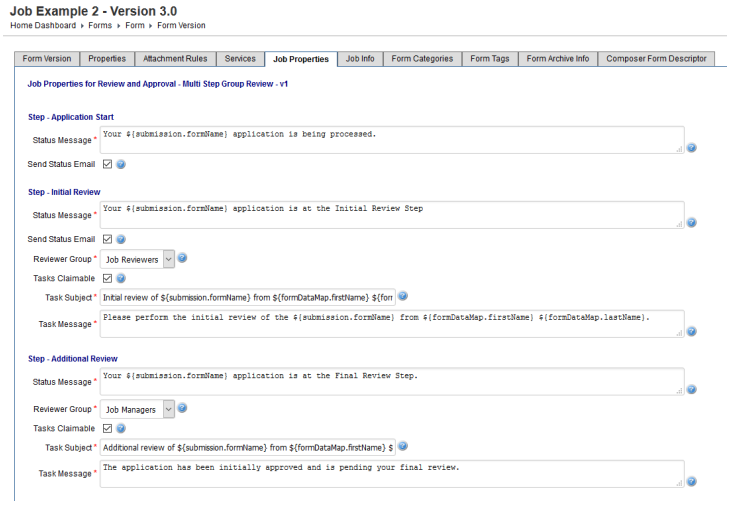
visibility and editability rules based upon the Job Step.

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**

Job Definition

```
{
  "jobDetails": {
    "name": "Review and Approval - Multi Step Group Review",
    "version": "4.2.0"
  },
  "properties": [
    { "name": "Task Form Code",  "value": "$func.startFormCode()" }
  ],
  "steps": [
    {
      "name": "Start",
      "type": "start",
      "actions": [
        {
          "name": "Handle Submission",
          "type": "Job Form Start",
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty('Start Send Email')" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Start Process Message')" }
          ]
        }
      ],
      "routes": [
```

```
        { "name": "Default",  "nextStep": "Initial Review" }
      ]
    },
    {
      "name": "Initial Review",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty('Initial Review Send Email')"
},
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Initial Review Process Message')" },
            { "name": "Task Assign Group",  "value": "$func.formProperty('Initial Review Group')" },
            { "name": "Task Enable Claiming",  "value": "$func.formProperty('Initial Review Enable Claiming')" },
            { "name": "Task Message",  "value": "$func.formProperty('Initial Review Task Message')" },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "$func.formProperty('Initial Review Task Subject')" },
            { "name": "Task Type",  "value": "Review" }
          ]
        },
        {
          "name": "Handle Submission",
          "type": "Job Task Wait",
          "properties": [
            { "name": "Conditional Route Name",  "value": "#if ( $formDataMap.routeName == 'Approve' &&
$formDataMap.loanAmount >= 20000 ) Exceeds Threshold #end" }
          ]
        }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Exceeds Threshold",  "nextStep": "Additional Review",  "display": "false" },
        { "name": "Terminate",  "nextStep": "Terminated Initial" }
      ]
    },
    {
      "name": "Additional Review",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Additional Review Process
Message')" },
            { "name": "Task Assign Group",  "value": "$func.formProperty('Additional Review Group')" },
            { "name": "Task Enable Claiming",  "value": "$func.formProperty('Additional Review Enable
Claiming')" },
            { "name": "Task Message",  "value": "$func.formProperty('Additional Review Task Message')" },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "$func.formProperty('Additional Review Task Subject')" },
            { "name": "Task Type",  "value": "Review" }
          ]
        },
        {
          "name": "Handle Submission",
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Decline",  "nextStep": "Applicant Update" },
        { "name": "Terminate",  "nextStep": "Terminated Additional" }
      ]
    },
    {
      "name": "Applicant Update",
      "type": "",
      "actions": [
        {
          "name": "Create Task",
          "type": "Job Task Assign",
          "properties": [
```

```
            { "name": "Process Message Submission",  "value": "$func.startSubmission()" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Applicant Update Process Message')"
},
            { "name": "Task Assign User",  "value": "$func.startUser()" },
            { "name": "Task Message",  "value": "$func.formProperty('Applicant Update Task Message')" },
            { "name": "Task Review Previous Step",  "value": "true" },
            { "name": "Task Subject",  "value": "$func.formProperty('Applicant Update Task Subject')" },
            { "name": "Task Type",  "value": "Review" }
          ]
        },
        {
          "name": "Handle Submission",
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Update",  "nextStep": "##PREVIOUS_STEP" },
        { "name": "Terminate",  "nextStep": "Terminated Applicant" }
      ]
    },
    {
      "name": "Application Delivery",
      "type": "",
      "actions": [
        {
          "name": "Application Delivery",
          "type": "Job Delivery"
        },
        {
          "name": "Application Delivery Wait",
          "type": "Job Delivery Wait"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Approved Complete" }
      ]
    },
    {
      "name": "Approved Complete",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty('Approved Complete Send
Email')" },
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Approved Complete Process
Message')" }
          ]
        }
      ]
    },
    {
      "name": "Terminated Initial",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty('Terminated Initial Send
Email')" },
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Terminated Initial Process
Message')" }
          ]
        }
      ]
    },
    {
      "name": "Terminated Additional",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
```

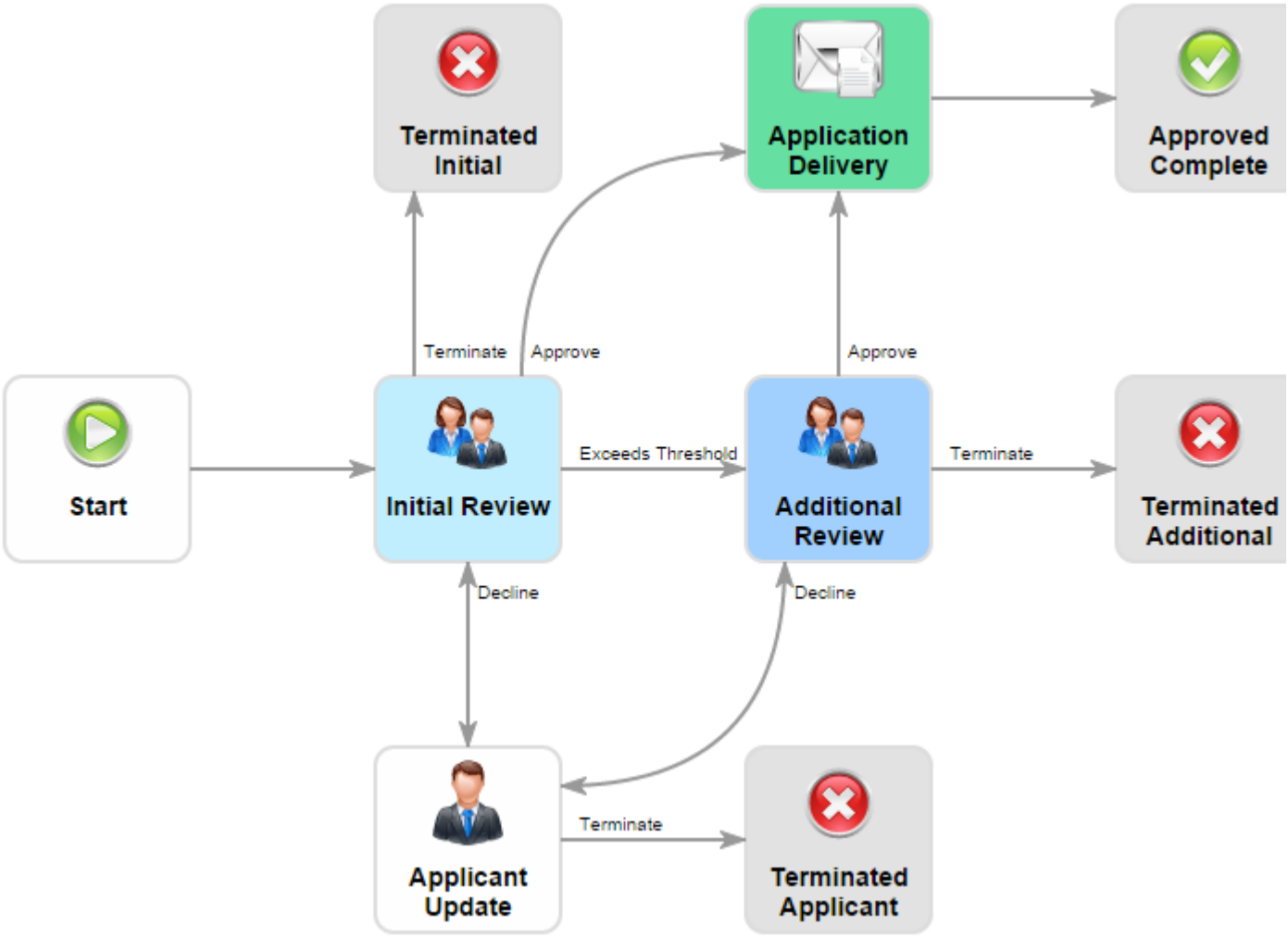
```
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty('Terminated Additional Send
Email')" },
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Terminated Additional Process
Message')" }
          ]
        }
      ]
    },
    {
      "name": "Terminated Applicant",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "$func.formProperty('Terminated Applicant Send
Email')" },
            { "name": "Process Message Submission Step",  "value": "Start" },
            { "name": "Process Message Text",  "value": "$func.formProperty('Terminated Applicant Process
Message')" }
          ]
        }
      ]
    }
  ]
}
```

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



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#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#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

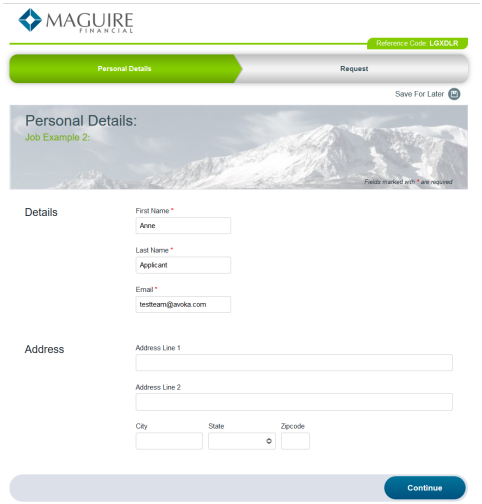
The routes in the form below use radio buttons.

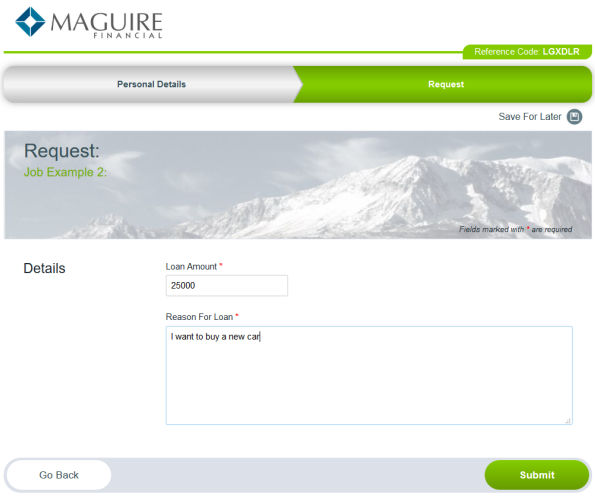
Please see Composer#Routeing Using the Standard Radio Button Group

Example Execution

Anne- Start

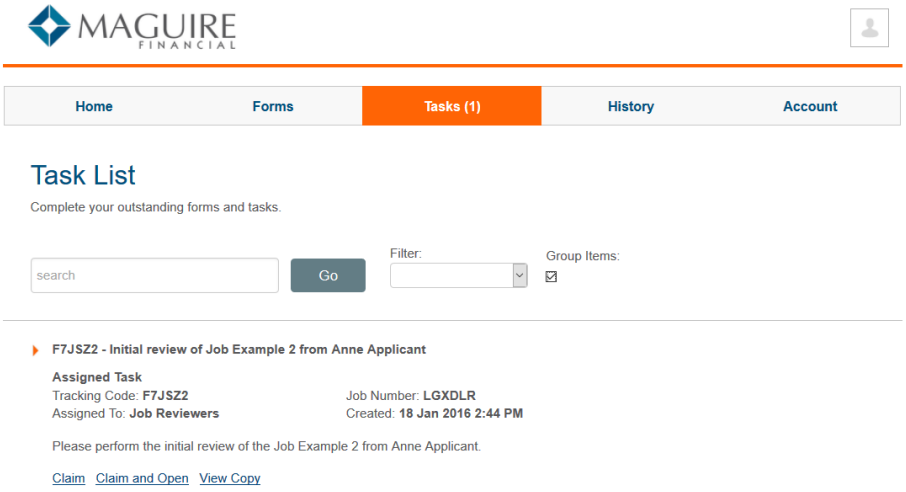
Fills in the 2 sections and submits.



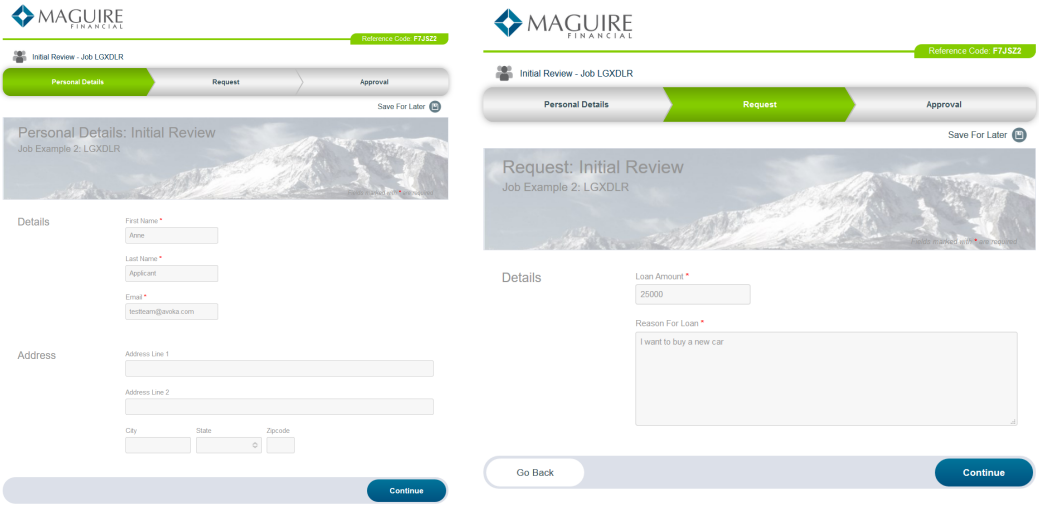


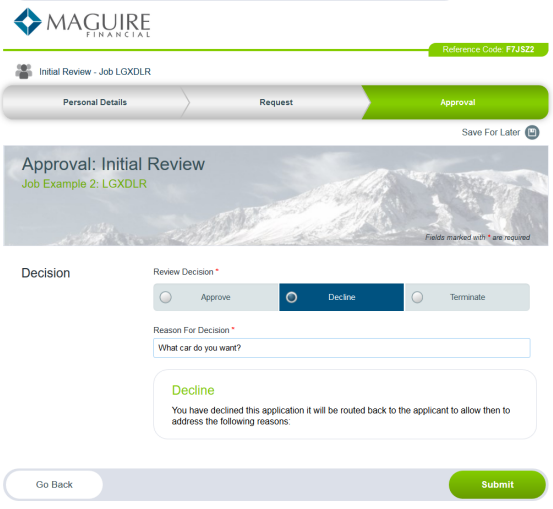
Roger - Initial Review

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?"

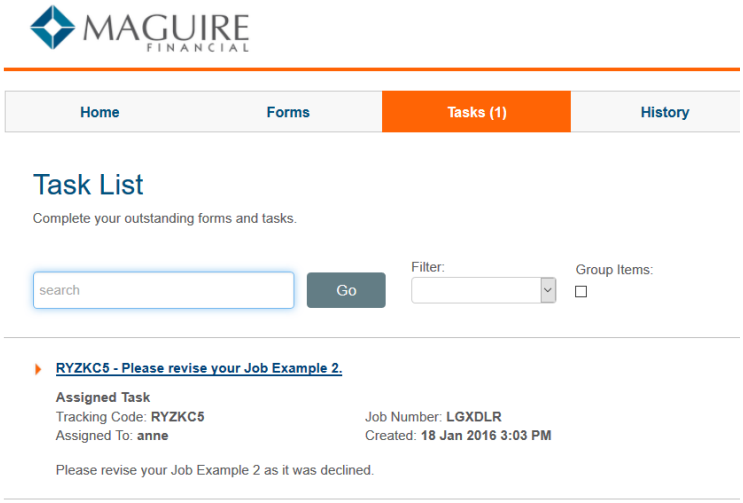




> **Note**
>
> See Composer#Setting Route with Business Rule when using more that 1 radio button groups to control the Route Name. *

Anne - Applicant Update

Sees the tasks assigned to her

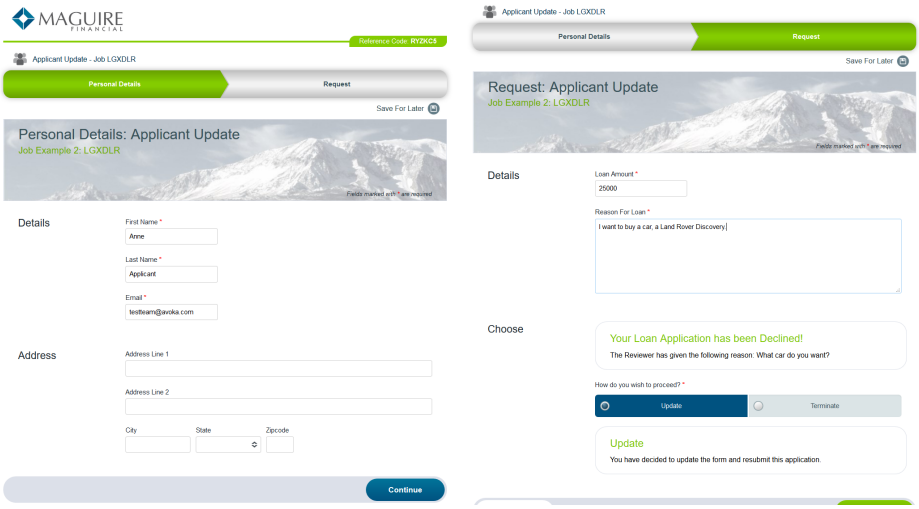


Open Form, Note the Request now has an extra section Choose.

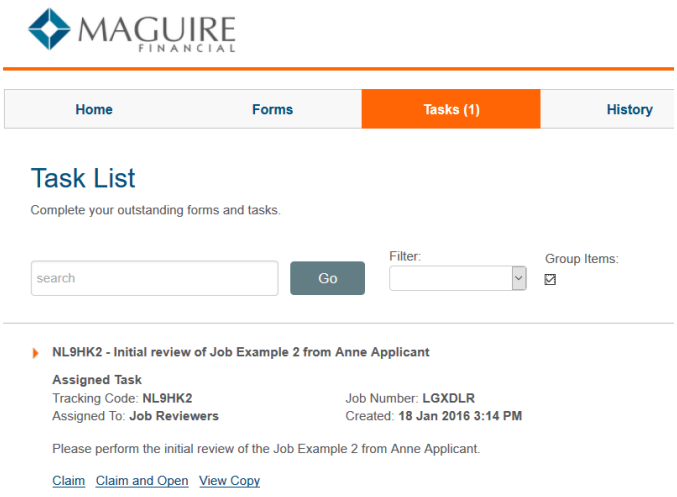
>Select the Update radio button*

>Type the new car make into the details.

>Click Submit



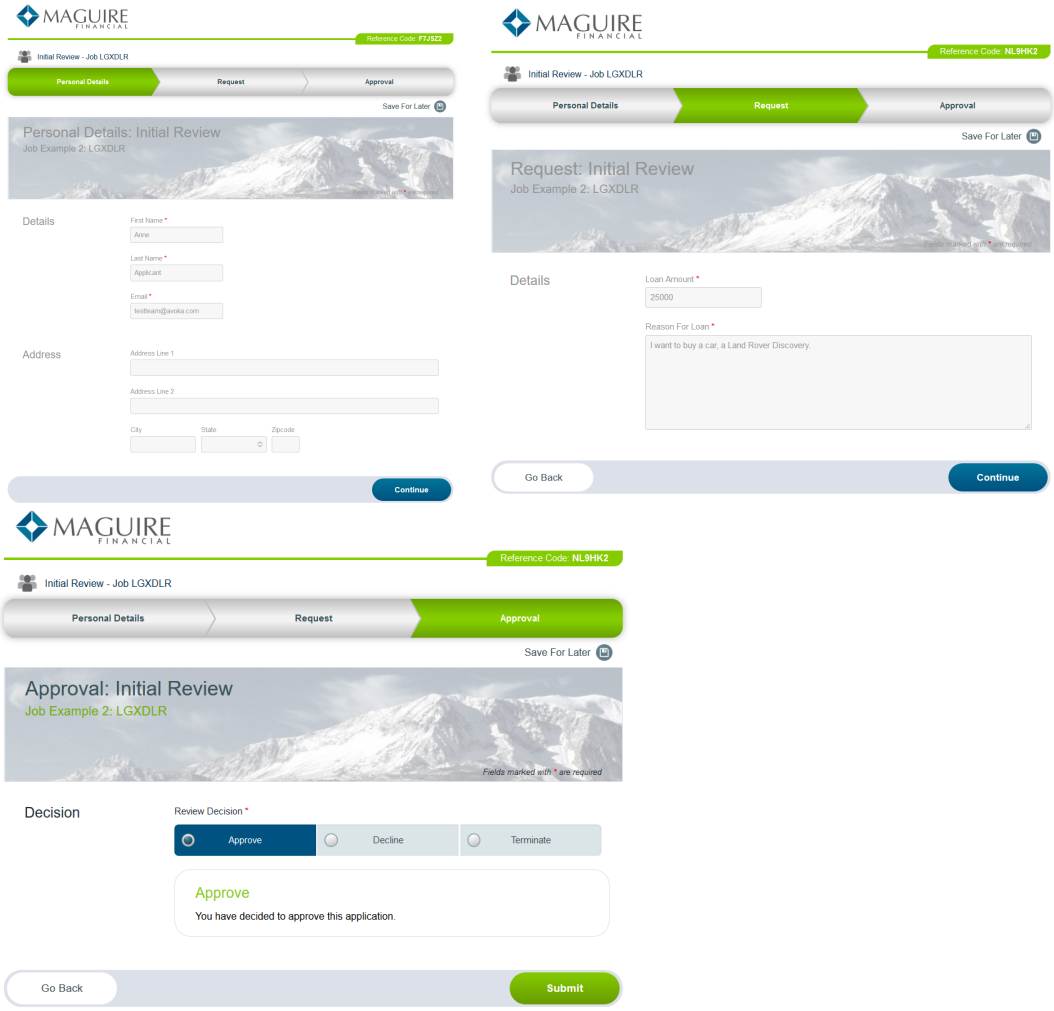
Roger - Initial Review



> Select Claim and Open

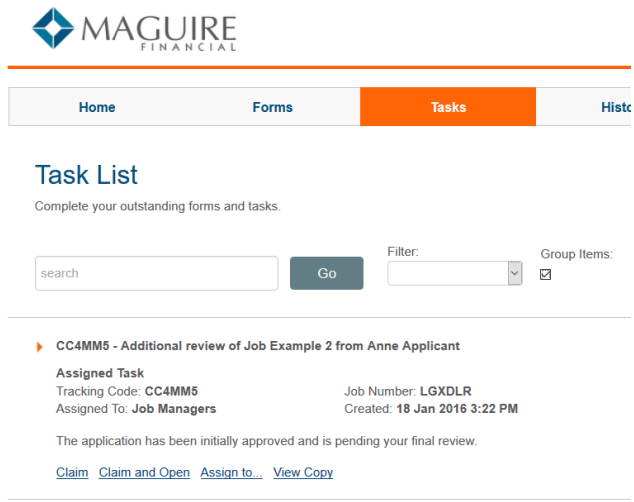
> Review Request - Reason For Loan

> In Approval section select the Approve Radio Button.



Mike - Additional Review

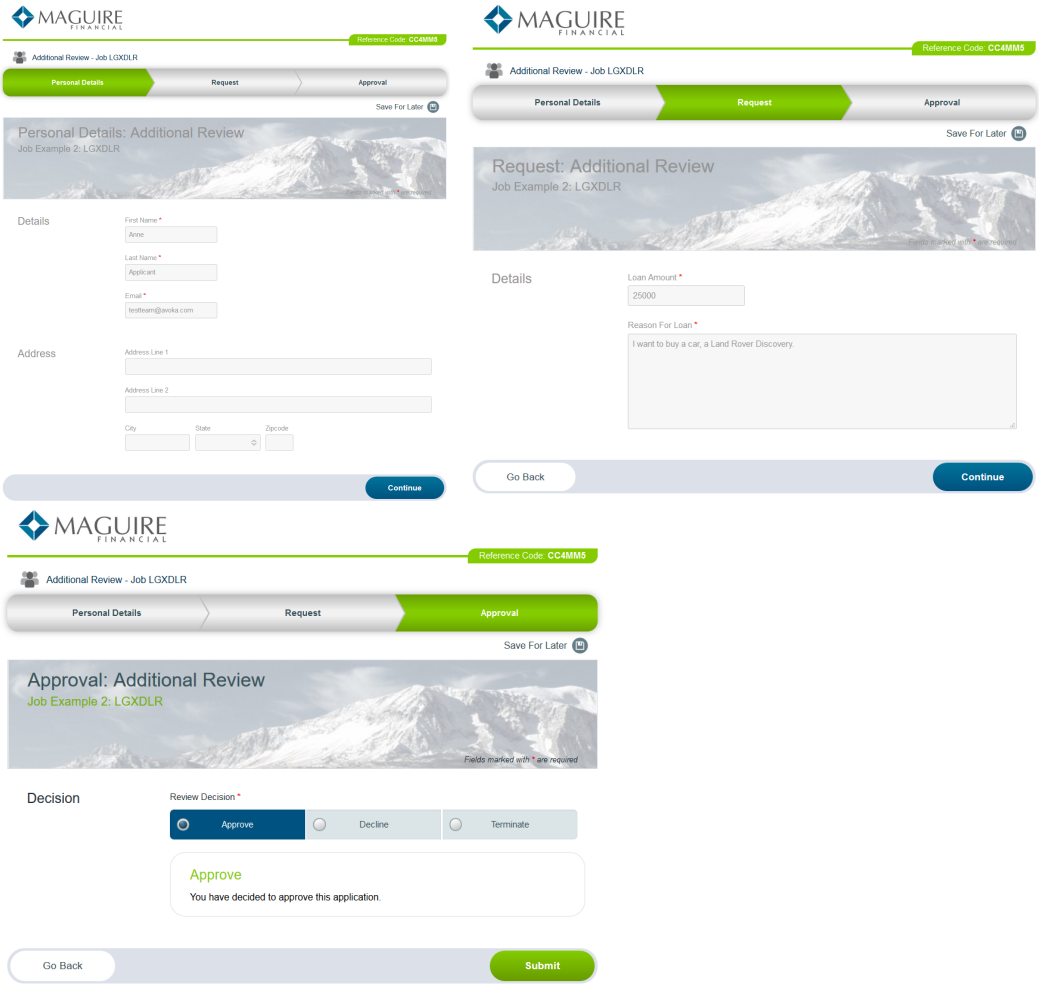
As the loan amount was greater that $20,000 the job is routed to Mike



> 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#Visibility and Editability Rules** section.

Customer Onboarding Job - Form Bundle

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

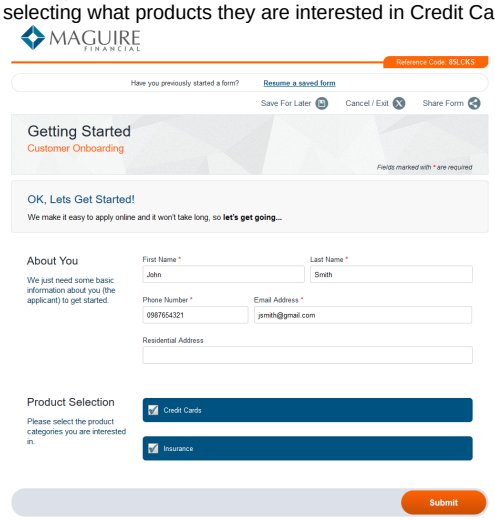
A separate advanced example demonstrates

- 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.

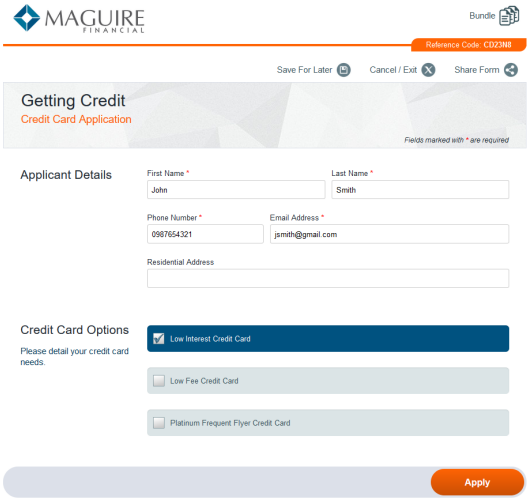


Click Submit

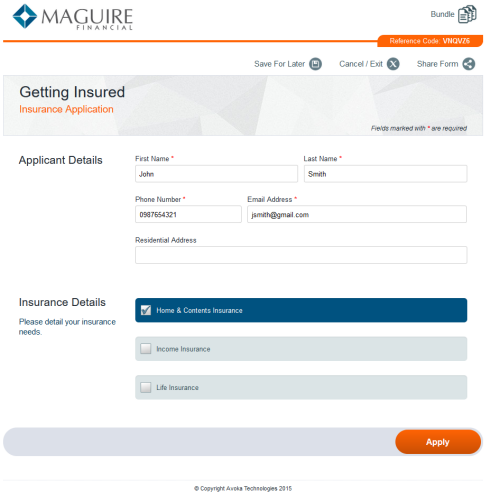
A collaboration job is started and the processing immediately progresses to the Additional Products step (ref Job Definition below).

Here the following Tasks are created.

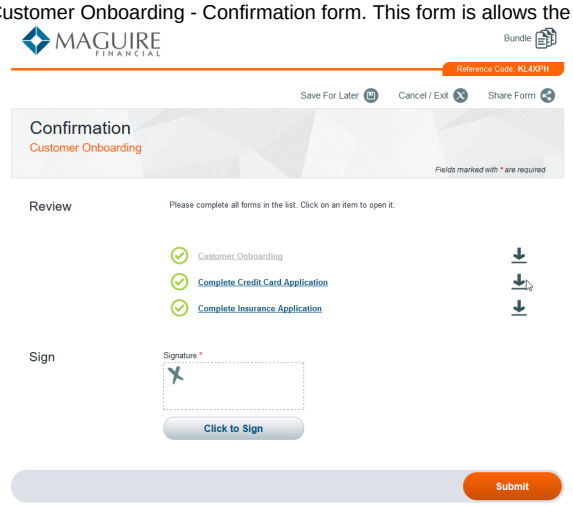
1. Customer Onboarding - Credit Card form (created if Credit Card was checked: "preCondition": "$formDataMap.productCreditCards == 'true'",).



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

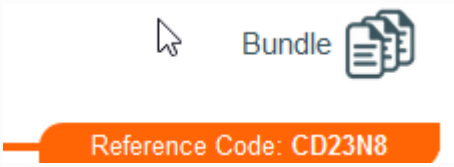


The Redirect Next functionality takes the user to the next form / task that has not be completed.

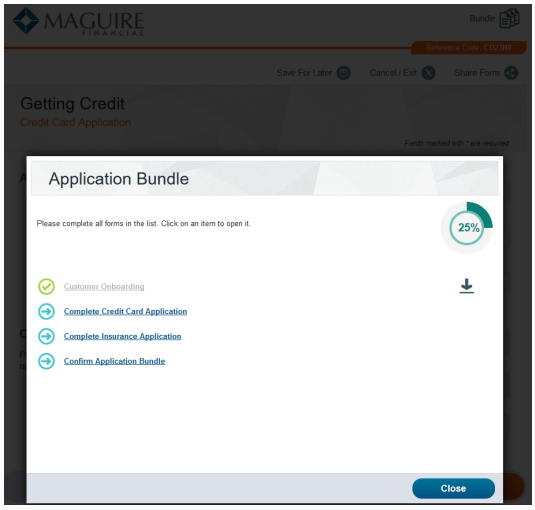
A happy day scenario would open the credit card. Submitting would take them to the insurance form then the confirmation page.

Note

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



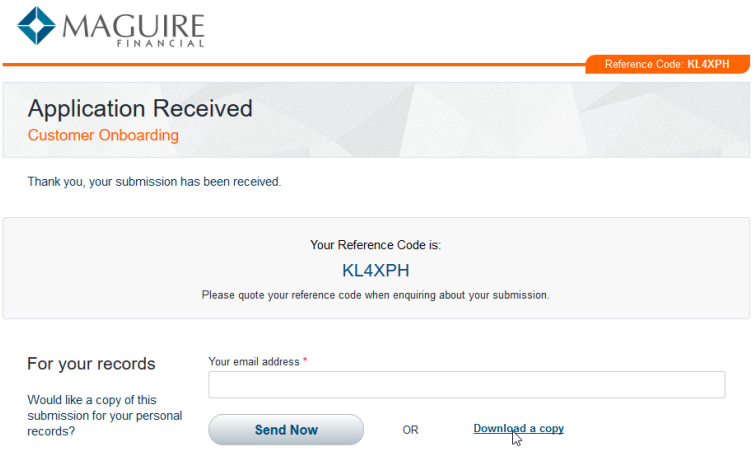
Clicking on this brings up the Application Bundle navigation screen where its possible to jump to other forms in the bundle in a different order to the Redirect Next.



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.



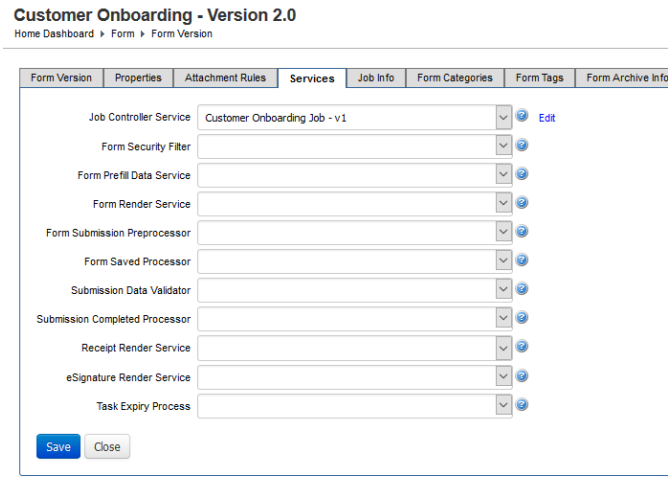
Setup

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"
    }
  ]
}
```

Maestro Customer Onboarding Job - Form Bundle

This is the Maestro version of the Composer Customer Onboarding Job - Form Bundle example.

Assets

The maestro forms and collaboration job can be found in this application package.

System Requirements

Form Creation must be completed in Maestro 5.1 +

Forms and Collaboration Job should run on TM 5.0 +

Execution

This follows the same lines as the composer example. Please run the **Maestro Onboarding - Triage** form.

Maestro Form and Job Example Assets

The follow application package contain the Maestro Form with the Job for Transact 5.1 release.

Form	Job	Relates to Composer / Job
Anonymous Maestro	Anonymous 1 Step Job	Anonymous Tasks Example 1
Job Maestro 1	use existing Job Review and Approval - 1 Step Group Review	1 Step Review Job
Job Maestro 2	use existing Job Review and Approval - Multi Step Group Review	Multi Step Group Review Job
Maestro Onboarding - Triage (Start Form) Maestro Onboarding - Credit Cards Maestro Onboarding - Insurance Maestro Onboarding - Confirmation	Maestro Onboarding	Customer Onboarding Job - Form Bundle

Advanced Examples

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

- Task Cancellation Processor
- Customer Onboarding Job - Dynamic Form Bundles
- Step Expiry
- Anonymous Tasks Example 1
- Asynchronous Step Execution
- Task Types - Anonymous, Form and Review
- Cancelling an Anonymous Task (Maestro Form)
- Maestro Form Bundle - Test Cases

Task Cancellation Processor

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.

Customer Onboarding Job - Dynamic Form Bundles

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

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.

Step Expiry

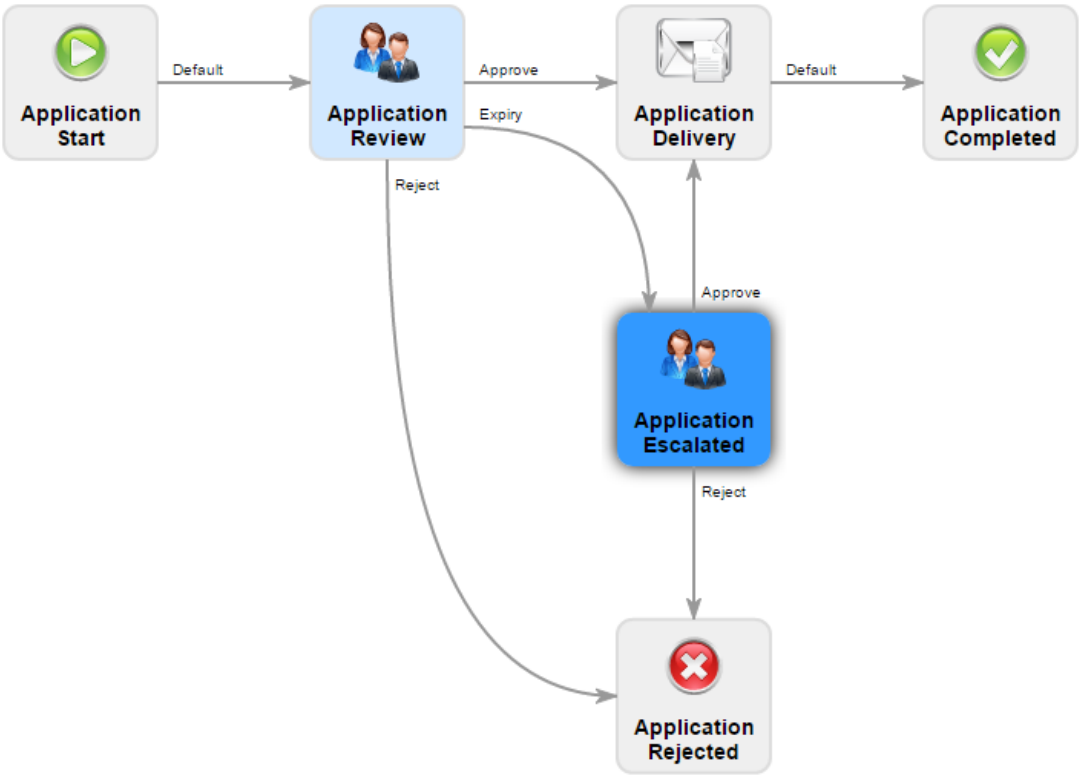
This example demonstrates how to setup expiry for a Step.

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.

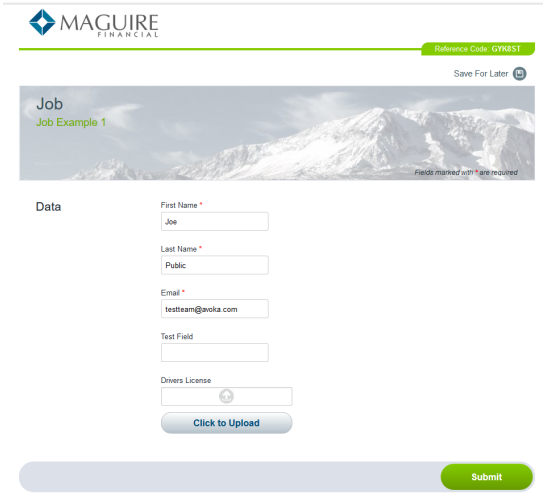


Note: This was modified to test this scenario detailed here: Expiry step loses Job Form Data Map values.

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.

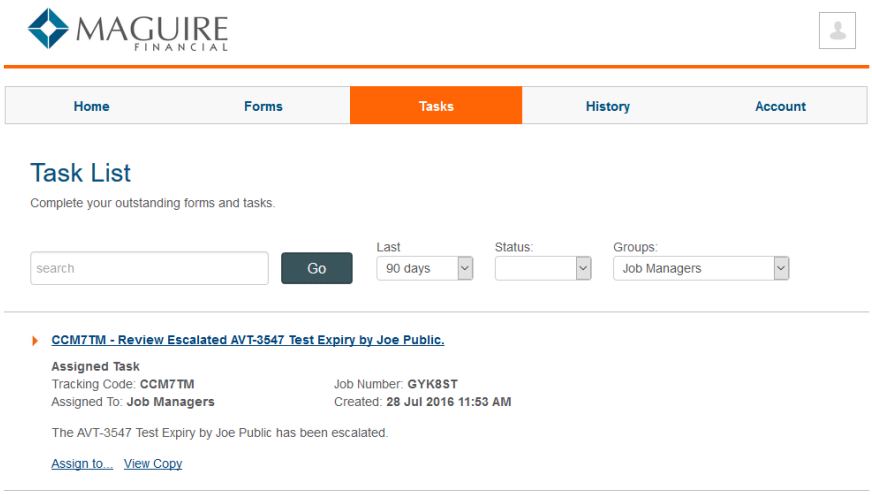
When the step expires these 2 actions are expired. The Application Review - Expiry action (Custom Groovy Service) runs to clean up the submissions for this step. For more details see the Job Expiry Service below.

3. Application Escalated

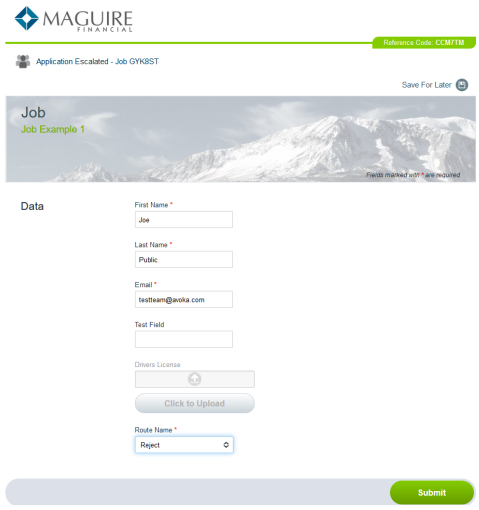
Log in as Mike Manager (Job Manager)

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

Anonymous Tasks Example 1

Summary

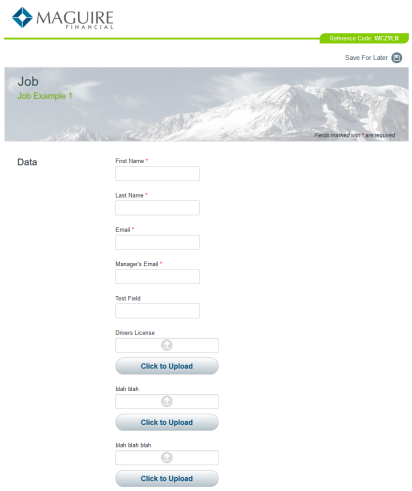
The following contains a flow similar to the 1 Step Review Job but with an anonymous tasks.

Application Package

Components

Form

Job Example 1 - Anonymous

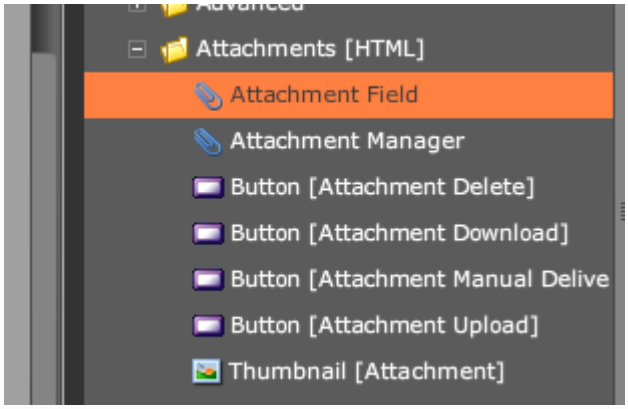


The extra fields are Managers email. This is who the Job Step Application Review is assigned to.

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

1 Step Review Anonymous


```
{
  "jobDetails": {
    "name": "Anonymous 1 Step",
    "processSubmitImmediate": "false",
    "version": "SNAPSHOT"
  },
  "steps": [
    {
      "name": "Application Start",
      "type": "start",
      "actions": [
        {
          "name": "Accept Quote",
          "type": "Job Form Start",
          "properties": [
            { "name": "Process Message Send Email",  "value": "true" },
            { "name": "Process Message Text",  "value": "Thank you ${formDataMap.firstName} ${formDataMap.
lastName} your ${submission.formName} is being processed." }
          ]
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Review" }
      ]
    },
    {
      "name": "Application Review",
      "type": "",
      "actions": [
        {
          "name": "Assign Review",
          "type": "Job Task Assign",
          "properties": [
            { "name": "Task Assign Email",  "value": "${formDataMap.managersEmail}" },
            { "name": "Task Form Code",  "value": "$func.startFormCode()" },
            { "name": "Task Message",  "value": "Please review the ${submission.formName} by ${formDataMap.
firstName} ${formDataMap.lastName}." },
            { "name": "Task Subject",  "value": "Review ${submission.formName} by ${submission.
contactEmailAddress}." },
            { "name": "Task Assign Portal",  "value": "Maguire" },
            { "name": "Task Form XML Data",  "value": "$func.startSubmissionXml()" },
            { "name": "Task Attachments Submission Step",  "value": "Application Start" },
            { "name": "Task Send Email",  "value": "true" },
            { "name": "Task Email Message",  "value": "$func.formProperty('Anonymous Task Email Message')" },
            { "name": "Task Email Subject",  "value": "$func.formProperty('Anonymous Task Email Subject')" },
            { "name": "Task Type",  "value": "Anonymous" }
          ]
        },
        {
          "name": "Review Wait",
          "type": "Job Task Wait"
        }
```

```
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Reject",  "nextStep": "Application Rejected" }
      ]
    },
    {
      "name": "Application Delivery",
      "type": "",
      "actions": [
        {
          "name": "Application Delivery",
          "type": "Job Delivery"
        },
        {
          "name": "Application Delivery Wait",
          "type": "Job Delivery Wait"
        }
      ],
      "routes": [
        { "name": "Default",  "nextStep": "Application Completed" }
      ]
    },
    {
      "name": "Application Completed",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "true" },
            { "name": "Process Message Submission Step",  "value": "Application Start" },
            { "name": "Process Message Text",  "value": "Thank you ${formDataMap.firstName} ${formDataMap.
lastName} your ${submission.formName} has been Approved." }
          ]
        }
      ]
    },
    {
      "name": "Application Rejected",
      "type": "endpoint",
      "actions": [
        {
          "name": "Process Message",
          "type": "Job Process Message",
          "properties": [
            { "name": "Process Message Send Email",  "value": "true" },
            { "name": "Process Message Submission Step",  "value": "Application Start" },
            { "name": "Process Message Text",  "value": "Sorry ${formDataMap.firstName} ${formDataMap.lastName}
your ${submission.formName} has been declined." }
          ]
        }
      ]
    }
  ]
}
```

Asynchronous Step Execution

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)`

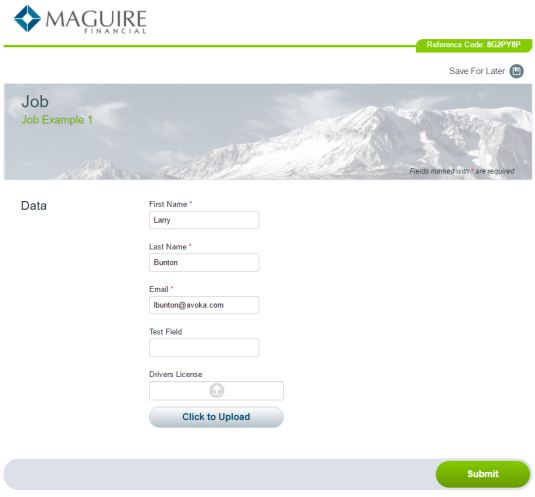
The following code can be run in the Groovy console. I have included it in the example **AVT-4097 Fake Callback.**

- 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

AVT-4097 Invoke Wait Job

Home Dashboard ▸ Collaboration Jobs ▸ Job Details

Job Details | Steps | Actions | Submissions | Events

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

Groovy Console

Home Dashboard ▸ Collaboration Jobs

Groovy Script successfully executed in 46 ms

Groovy Script | Service Parameters

Groovy Script

```
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 = "8G2PY8P"
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
```

Secure Fluent API Only | Execution Timeout (sec) 60

Execute Script | Syntax Check | Clear Output | Clear Script | Save Script

Transact Services Guide | Transact Fluent API Javadoc

Execution Output

```
result = true
```

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		

AVT-4097 Invoke Wait Job

Home Dashboard ▸ Collaboration Jobs ▸ Job Details

Job Details | Steps | Actions | Submissions | Events

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	Completed						1	28 Sep 16 14:29	
Async Invoke Wait	Wait For Callback	Action Wait	Completed						1	28 Sep 16 14:29	28 Sep 16 14

Close

Application Package


application-packa...le-2016-09-28.zip

Task Types - Anonymous, Form and Review

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

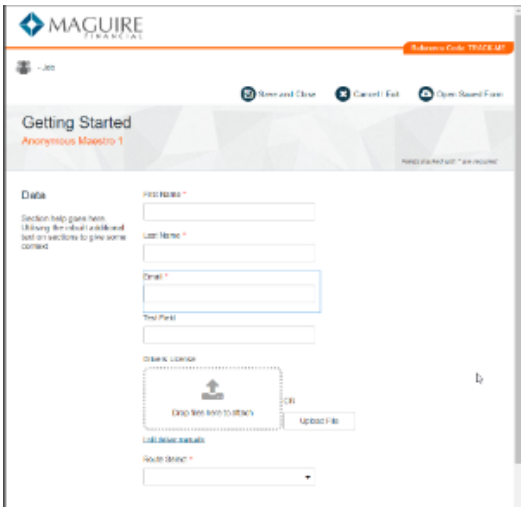
```
{
  "jobDetails": {
    "name": "1 Step Review - Form Task with Attach",
    "version": "4.2.0"
  },
  "steps": [
    {
      "name": "Application Start",
      "type": "start",
      "actions": [
        {
          "name": "Accept Quote",
          "type": "Job Form Start"
        }
      ],
      "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"
    }
  ]
}
```

Cancelling an Anonymous Task (Maestro Form)

Summary

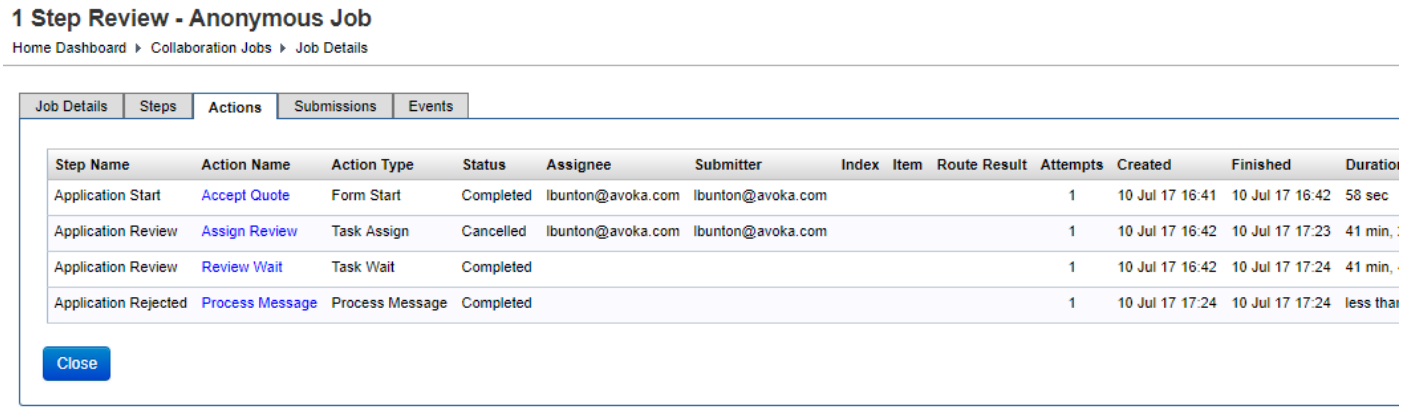
The following example we have a Maestro form that is put through a 1 step Anonymous review workflow. Cancelling an authenticated task works the same way.



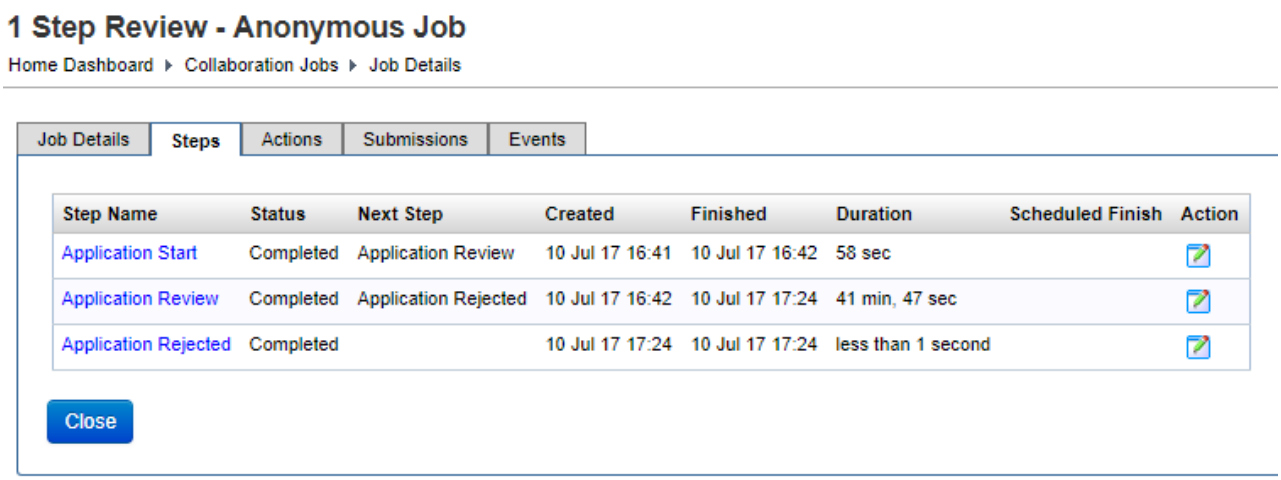
When we initially submit the form we enter our details. The task that is created will get assigned back to the address in the email field. You will get notified of a saved email with a link to open the form.

You have opened the form associated with the Application Review - Assign Review action. Then click on the Cancel Exit button. Cancelling does not return any form.xml and hence the route is empty. The empty route is picked up by the new Default route which send to the next step Application rejected.

Here we can see the actions that have been executed. We can see the status of the Application Review has been changed to cancelled and the route result is blank.



Here we can see that when the default route is routing to the next step Application Rejected.



Application Package

Collaboration Job

We needed to add the Default route to the application Review Step. This is because the cancel does not submit any form data resulting in the routeName being empty.

Job Definition - Application Review Step

```
    {
      "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 ${submission.
contactEmailAddress}." },
            { "name": "Task Assign Portal",  "value": "Web Plug-in" },
            { "name": "Task Form XML Data",  "value": "$func.startSubmissionXml()" },
            { "name": "Task Attachments Submission Step",  "value": "Application Start" },
            { "name": "Task Send Email",  "value": "true" },
            { "name": "Task Email Message",  "value": "$func.formProperty('Anonymous Task Email Message')" },
            { "name": "Task Email Subject",  "value": "$func.formProperty('Anonymous Task Email Subject')" },
            { "name": "Task Type",  "value": "Anonymous" }
          ]
        },
        {
          "name": "Review Wait",
          "type": "Job Task Wait"
        }
      ],
      "routes": [
        { "name": "Approve",  "nextStep": "Application Delivery" },
        { "name": "Reject",  "nextStep": "Application Rejected" },
        { "name": "Default",  "nextStep": "Application Rejected" }
      ]
    },
```

Maestro Form Bundle - Test Cases

End to End Example

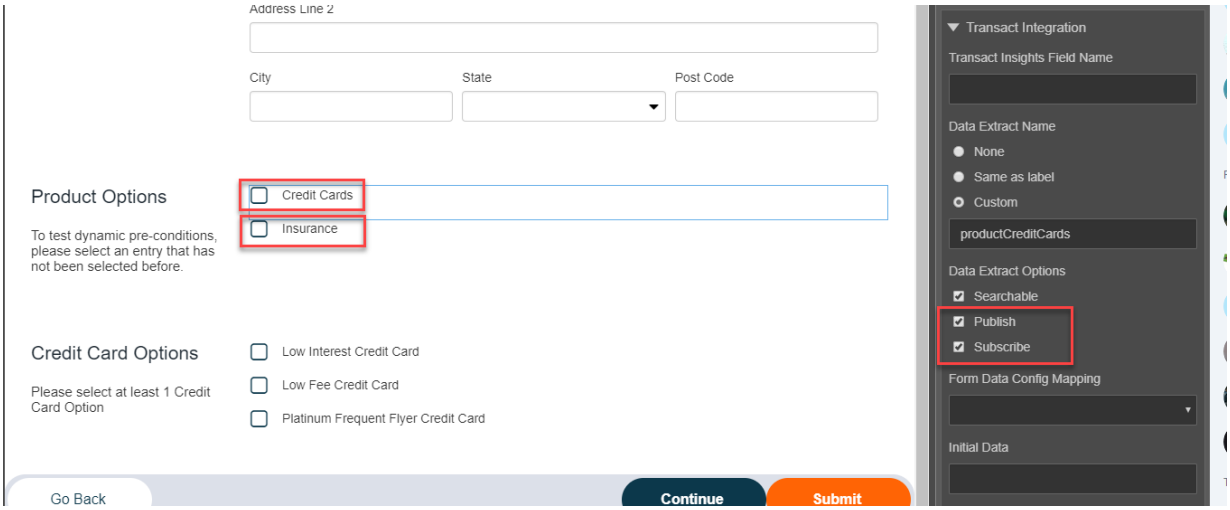
Unknown macro: 'multiexcerpt'

Overview

The test cases extend the Maestro Onboarding example.

The Credit Card Application and Insurance Application forms have been replaced with forms that include the Product Options which enables us to test the dynamicPreCondition attribute.

The Product options - Credit Card and insurance check boxes are set to both Publish and Subscribe. This allows for testing of the shareExtractData step attribute.



Application Package

Forms

These forms are part of the standard Maestro Onboarding example.

Maestro Onboarding - Confirmation
Maestro Onboarding - Credit Cards
Maestro Onboarding - Insurance
Maestro Onboarding - Triage

These are the 2 forms that substitute the equivalent standard form
MTest Onboarding - Credit Cards
MTest Onboarding - Insurance

These are used instead of the standard triage form. They are used to start authenticated bundle tests of type Form and Review.
MTest Onboarding - Triage Form
MTest Onboarding - Triage Review

Services

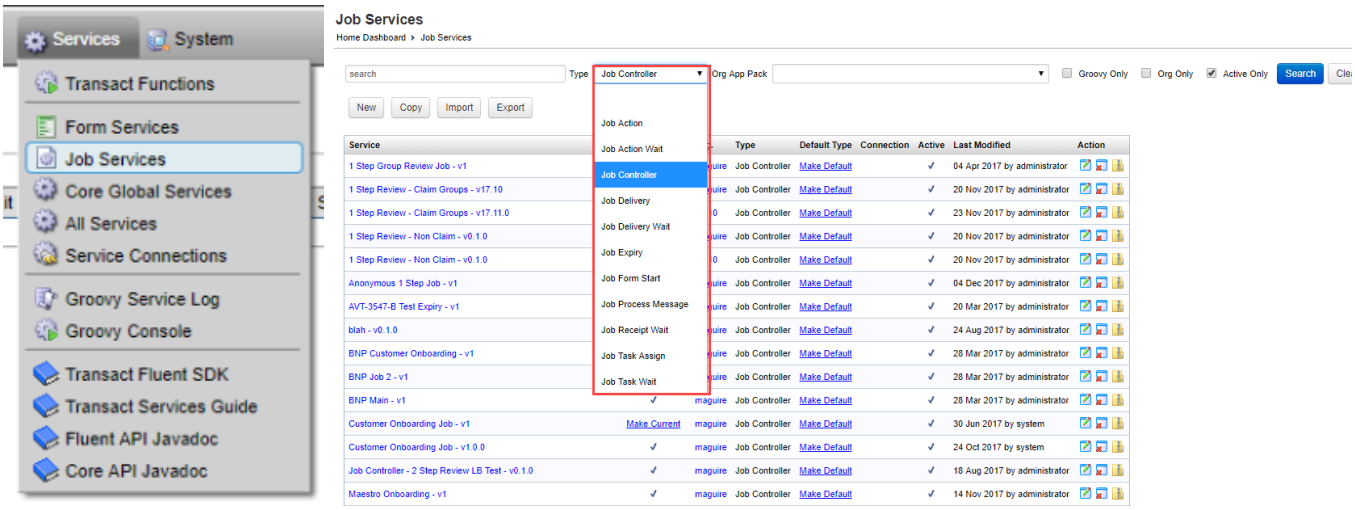
Job Controller	Description
Maestro Onboarding - v17.10.0	The standard job controller, Anonymous with shareFormData, Not Dynamic
Maestro Onboard-Anony-SharedExtractData - v17.10.0	Anonymous with shareExtractData, Not Dynamic
mOnboarding - Anonymous - Dynamic - shareExtractData - v17.10	Anonymous with shareExtractData, Dynamic
mOnboarding - Anonymous - Dynamic - shareFormData - v17.10.0	Anonymous with shareFormData, Not Dynamic
mOnboarding - Anonymous - No Dynamic - v17.10.0	Anonymous No sharing, Not Dynamic
mOnboarding - Form Tasks - shareExtractData - v17.10.0	Authenticated Form Task, shareExtractData, Dynamic
mOnboarding - Form Tasks - shareFormData - v17.10.0	Authenticated Form Task, shareFormData, Dynamic
mOnboarding - Review Task - shareExtractData - v17.10.0	Authenticated Review Task, shareExtractData, Dynamic

Job Services - Standard and Custom Services

Job Services

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

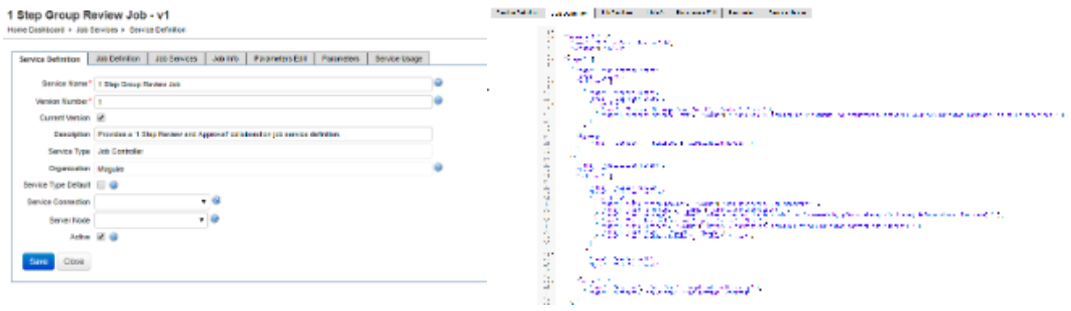
This bring up the Job Services Search Screen, by default the it will list services of type Job Controllers. We can see the other types of services in the type drop down.

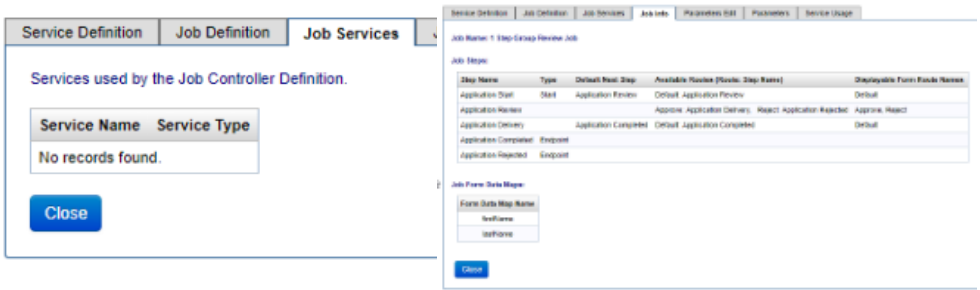


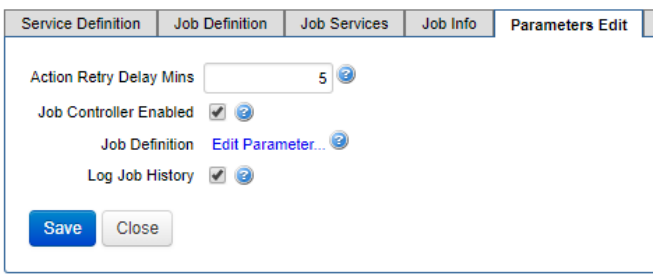
Service Types

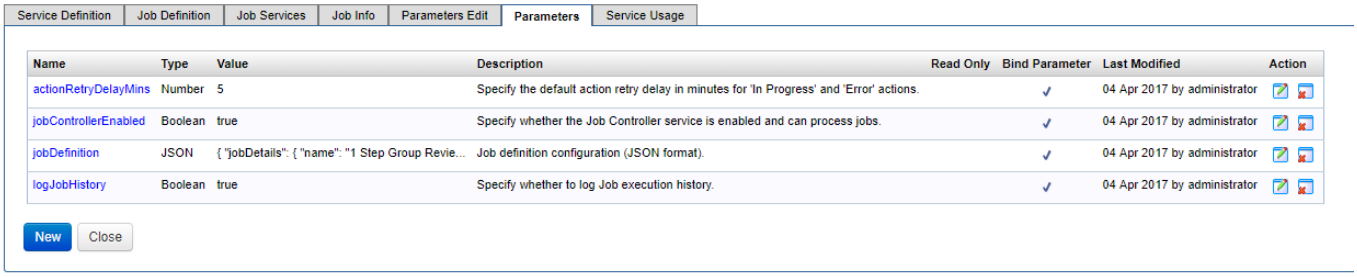
Job Controller

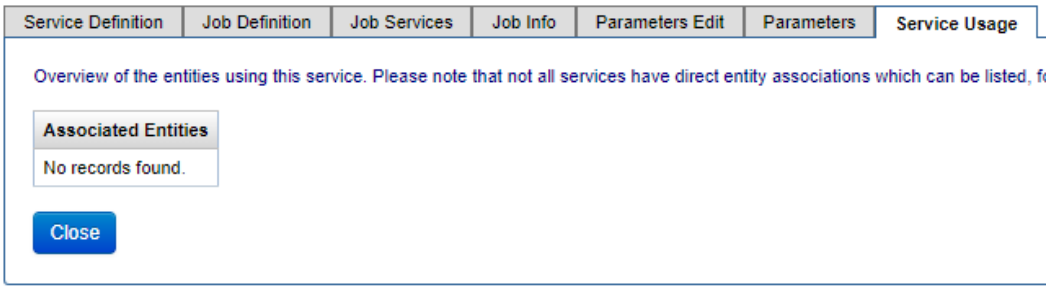
The Job Controller is responsible for holding the processing definition of the collaboration job. It has a Job Definition (JSON) that describes when and how it calls the Action Services and Job Function.











The screenshots above show the tabs of a Job Controller Service:

- the first tab displays the standard service definition. Here you can edit the service name.
- the Job Definition tab allows the Job Definition JSON to be edited.
- the Job Services tab show the other services that are being used by this Job Controller, that is referenced by the JSON
- the Job Info tab summarizes the Job Steps as well as listing the referenced data extracts.
- the Parameters tab is where you can perform CRUD operations on parameters.
- the Parameters Edit tab give a quicker way of editing the service parameters.
- the Service Usage shows the forms that are linked to

Job Parameters

The Standard Job Controller Service Parameters are:

Name	Type	Default Value	Description	Read Only	Bind Parameter
actionRetryDelayMins	Number	5	Specify the default action retry delay in minutes for 'In Progress' and 'Error' actions.		
jobControllerEnabled	Boolean	true	Specify whether the Job Controller service is enabled and can process jobs. This is useful for debugging test systems and selectively enabling certain jobs in production.		
jobDefinition	JSON	{ "jobDetails": { "name": "1 Step Group Revie...	Job definition configuration (JSON format). 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		
logJobHistory	Boolean	true	Specify whether to log Job execution history.		

Additional Optional Job Controller Service Parameters:

Name	Type	Value	Description	Read Only	Bind Parameter
jobDiagram	JSON	{ "jobDiagra": { "name": "1 Step Group Revie..	Holds the job diagram format created by the external NextStep tool. Note this is not reference by		
Form Property Editor Definition	JSON	[{ "name": "Step - Application Start", ...	Templated job form properties editor definition (JSON format). See Job Templates below.		

Job Template

Since TM 4.2.0.

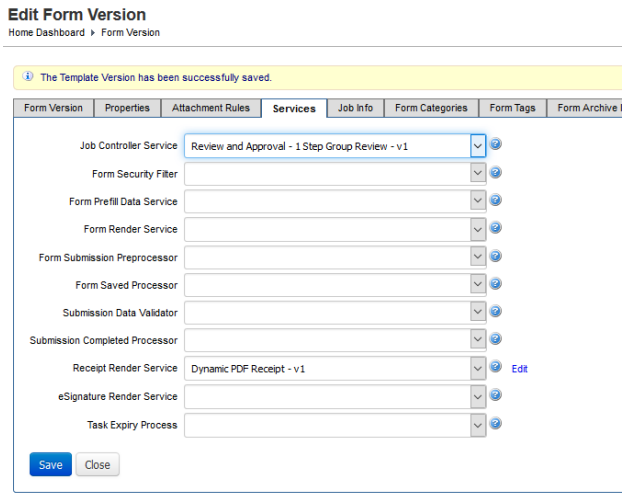
Job Templates are an enhanced form of Job Controller Service that allows a user select and customise a Collaboration Job without having to modify the JSON definition. This allows for:

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

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

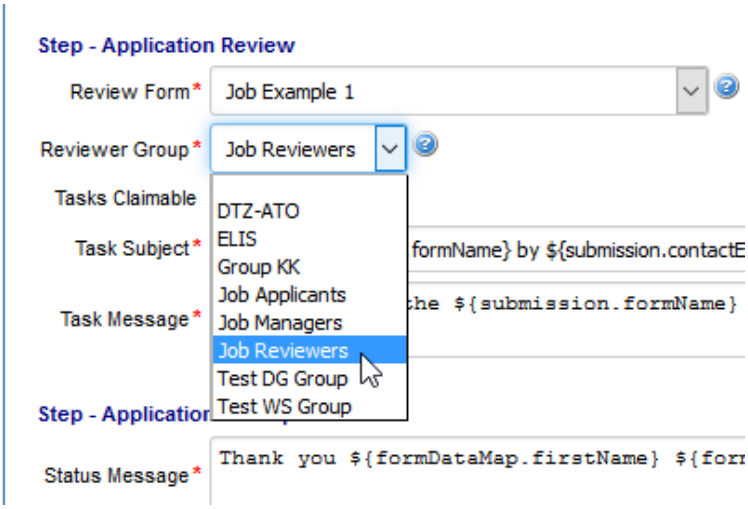
This is predominantly used for review and approvals jobs. Transact comes with 3 example templates:

Review and Approval - 1 Step Group Review - v1	✓	Job Controller	Make Default	✓	14 Mar
Review and Approval - 1 Step User Review - v1	✓	Job Controller	Make Default	✓	14 Mar
Review and Approval - Multi Step Group Review - v1	✓	Job Controller	Make Default	✓	14 Mar



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 a vaule from 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.

A new custom template can be created by adding the "Form Property Editor Definition" above to a standard Job Controller.

Action Services

Standard Types and Java Implementation

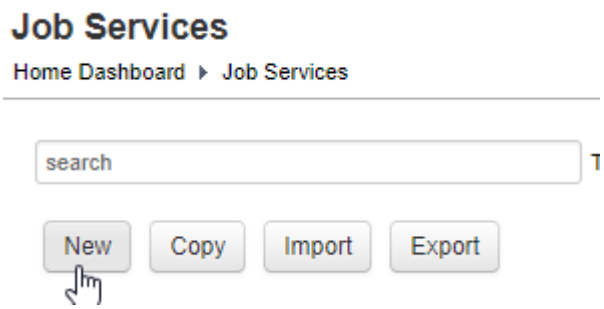
The table below makes reference to standard java class names which can be found in the API documentation here com.avoka.fc.core.service.job.impl.

Type	Job Action Class Name com.avoka.fc.core.service.job.impl	Description
Job Action		This type can be implemented using the GrooveyJobActionService or the new FluentJobActionService
Job Action Wait	**JobActionWaitService**	This action service will wait until all the steps previous "Invoked" actions have been completed.
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 message to the initial Submission. These appear in the user's submission history. It can also send a status update via email.
Job Expiry		This type can be implemented using the GroovyJobActionService. Create a GroovyJobActionServie or FluentJobActionService (See Creating Action Service Types other than Job Action)
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 Transact 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

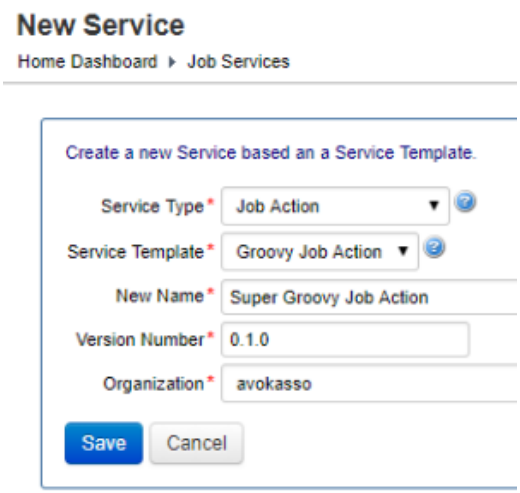
		• Task Assignment: These properties are used in the creation and assignment of user and group tasks. Tasks can be categorised as:
		  • Standard Tasks: to an authenticated user or group.
		  • Review Tasks: these are the similar to the standard tasks but form xml and attachments are copied from a previous 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.

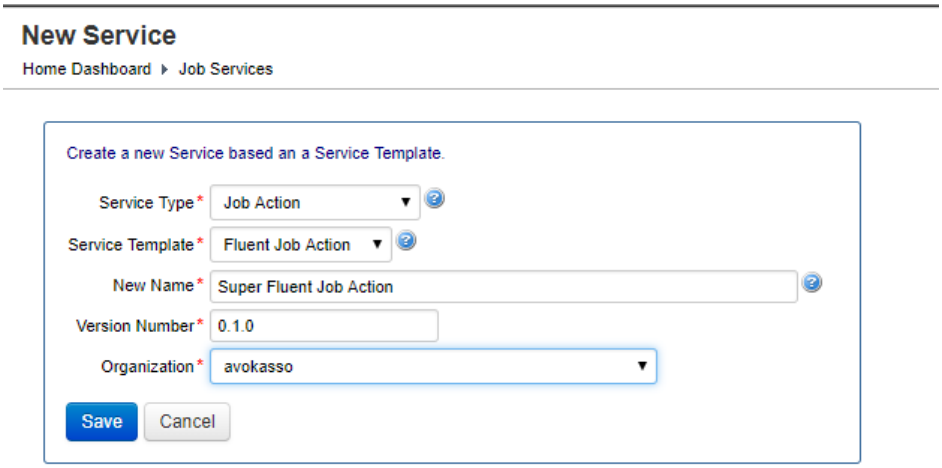
Creating Custom Job Actions



Groovy Job Action (4.0 +)



Fluent (5.0+)



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

Custom Action Properties.

The standard Action Property classes each define a fixed set of Action Properties. What each action property does is defined in the API documentation here com.avoka.fc.core.service.job.impl. The JobTaskAssignService defines 39 separate properties.

When creating a new Job Action it is possible to add action properties to your code. Doing so makes your Job action more reuseable accross your job and application.

Job Definition

```
"properties": [
  { "name": "ABC Widget Name",  "value": "Aplle Blue Catapillar" },
  ..
]
```

Groovy Job Action

```
def abcName = actionStepProperties.getProperty("ABC Widget Name")
```

Gotcha's

1. By default there is no submission associated with the Job Actions
   $formDataMap.name does not resolve as the Job Action doesn not have any concept of a submission.
   $func.previousSubmission().formDataMap.name works as we have used the submission from the previous step.
   $job.submission(0).name works as we have used the first submission linked to the job.

2. Not Supported in Fluent Job Action Classes (17.10.1)

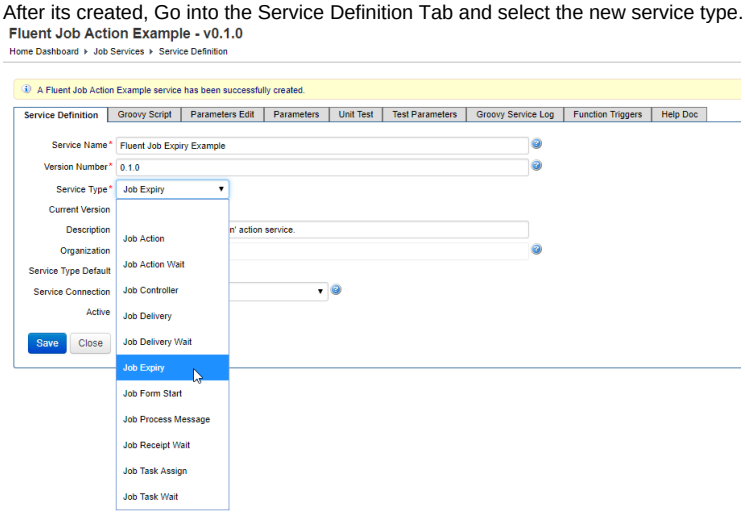
Making a Custom Job Action a different type (Advanced)

The following technique is used to create a Fluent or Groovy Job Action but assign a new type.

- Job Expiry Service
- Replacing Job Task Wait

1. Create a Job Action as per Above

2. After its created, Go into the Service Definition Tab and select the new service type.



Asynchronous Wait Pairs

The following type pairs into a asynchronous wait mechanism as per below.

- **Job Task Assign** and **Job Task Wait**
- **Job Delivery** and **Job Delivery Wait**

Unknown macro: 'multiexcerpt-include'

Custom Wait Pairs

In addition its possible to implement a custom Asynchronous Wait Mechanism with the following types.

Job Action and **Job Action Wait**.

Job Functions

Provided the job functions which are available for use in Job Definition - Action property values, and in Step preConditions and Action preConditions.

These functions are described in the JobFunctions API the invoke method extracted below and creating a new Groovy serving below again.:

invoke

The `invoke` function invoke the specified Groovy Service name, with the given lists of args and return the result.

The example below invokes the Groovy Service named 'Manager Lookup' passing it an empty parameter map.

```
 "properties": [
   { "name": "Task Assign User",  "value": "$func.invoke('Manager Lookup')" },
   ..
 ]
```

The example below invokes the Groovy Service named 'Reviewer Lookup' passing it a list of arguments.

```
 "properties": [
   { "name": "Task Assign User",  "value": "$func.invoke('Reviewer Lookup', $formDataMap.state, $formDataMap.email)"
},
   ..
 ]
```

In the example above the invoke function parameters are passed to the Groovy Service as an `args` list object. For example:

```
 // Groovy Script args parameter.
 def state = args[0]
 def email = args[1]
```

The Groovy Service will also be passed the `job` (`Job`) and `jobAction` (`JobAction`) parameters, of the currently execution job and job action. This enables you to access all the properties of the currently executing job.

```
 // Job and JobAction parameters.
 def jobName = job.getName()
 def stepName = jobAction.getJobStep().getName()
 def actionName = jobAction.getName()

 println jobName + ' : ' + stepName + ' : ' + actionName
```

Groovy Services

From the menu select Services Form Service

You will be presented with the form services

Fluent Service (Not Yet Implemented)

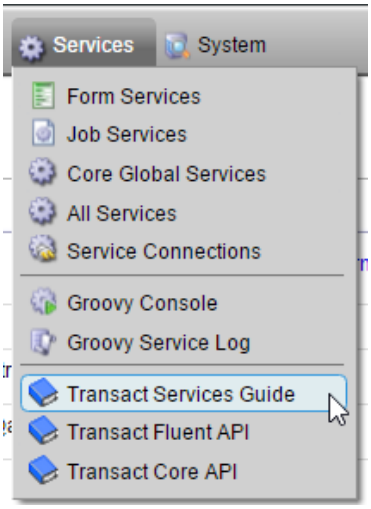
There are a number of Groovy services in the Advanced Example page

Documentation

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

Transact Services Guide

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



Collaboration Jobs API

Click here to see the Transact Collaboration Jobs API Reference

The following is a summary of the classes and their use.

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. 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.

JobTask WaitService	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 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

Job Reuse

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 Job Actions.
- 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')" },

Chaining and Form Bundles

Prerequisites

Form Design page covers form design for collaboration jobs

Maestro form design page covers the Maestro widgets for collaboration jobs.

Composer form design covers looks at the composer widgets used by collaboration jobs.

Overview

What is a Form Bundle?

Often with customer on-boarding scenarios the customer may be after several products. A form bundle contains 2 or more Forms grouped together in a one step.

Form bundles can be used instead of building a single monolithic form. Breaking a large form in to more manageable chunks makes it easier to develop and maintain. Form bundles enables concurrent development of form "parts".

A typical form bundles may get a user to enter their basic details in a triage form and what options they are interested in.

When the user hits submit on the triage form:

1. The next section of collaboration job, the bundle step is executed.
2. Task are associated with Job Action, if the preconditions are met the task is created and assigned to the user that submitted the triage form.
3. The browser is redirected to load the next available Form Task. The experience for the user is like they have gone to the next page of a bigger form.

End to End Form Bundle Flow Example

Unknown macro: 'multiexcerpt-include'

What is Avoka Client Services implementing for our Clients

Picture the flow above but a form bundle made to look like its one form. This can be achieved with some minor form changes.

- Changing the submit button label to Next on all but the final Application Confirmation form.
- Adding a back button to go to the previous form.

In addition client services has Custom Navigator Maestro Widgets that allow a user to easily jump to any part of the form.

Job Definition Elements

This page details by feature the Step Attributes, Action Attributes, Step Properties and Action Attributes relating to form bundles

Step Attributes Flags related to bundles only, 2+ tasks. Note the following are ignored if there is only 1 task in the step:

- **dynamicPreConditions:** if true action preconditions can be re-evaluated after forms with the step, if false they are only evaluated on the triage form.
- **shareFormData:** method for copying data between form tasks within a bundle step. This copies the entire form xml.
- **shareExtractData:** method for copying data between form tasks within a bundle step. This copies the data extracts using a publish subscribe method.
- **allFormsEditable:** if true the form tasks withing the bundle are mutable until the step has completed.

Step Attributes that can be used with 1+ Tasks:

- **showPreviousForms:** this will show the previous forms.
- **redirectNext:** This allows form chaining along with the Action Attribute redirect next.

Step Properties:

This is used to specify the triage step and is only required if triage form is not the start form.

```
"properties": [
  {"name": "Pre Condition Submission Step", "value" : "Triage Step Name"}
],
```

Action Attributes related to bundles:

- **preCondition:** A velocity template that
- **redirectNext:** This allows form chaining along with the Step Attribute redirect next.

Features

Chaining and Bundling

The type of UI flow mentioned above is often referred to as *Form Chaining*. Its possible to configure a collaboration job to have 1 Task created per Step and still implement chaining between the steps..

This can be done with the use of **redirectNext** step and action attributes see below.

For form chaining to work the collaboration job must run in **Process Immediate Mode**.

For any chaining or form bundle job to work as intended they must use a different job processing mode that Review and Approval

Process Immediate Mode

Unknown macro: 'multiexcerpt-include'

Form Bundle - Process Immediate Mode

Unknown macro: 'multiexcerpt-include'

processSubmitImmeadiate set true as per the Job Definition below (see attribute highlighted in red).

Form Bundle - Job Definition JSON

```
{
  "jobDetails": {
    "name": "2 Step Review Job",
    "processSubmitImmediate": "true",

    "version": "17.10.0"
  },
  "steps":[
    ...
  ]

}
```

Additional Features Form Bundles

Sharing Data between Tasks.

There are 3 valid options that can be taken to share data between forms in a bundle.

1. **No Sharing**: Set the step attribute **shareFormData is false,** and **shareExtractData is false**.
   In this case no data is copied between the forms.

2. **Shared XML Data Model**: Set the step attribute **shareFormData is true**, and **shareExtractData is false**.
   When a Form Task is submitted the data is automatically written to the other forms. To use this All forms need to use the same data model.
   This is the most common for of sharing data between forms..

3. **Share Extract Data**: Set the step attribute **shareFormData is false**, and **shareExtractData is true**.
   This can be used where the forms do not have a common data model. This type of Sharing is also called Publish - Subscribe sharing of Data Extracts
   Form A sets a field as a data extract Demo Type . Form B subscribes to the same data extract **Demo Type**.
   Form A is opened, the Demo Type data extract value is changed *wert* to *blah* and then form A is submitted.
   The Demo Type data extract is is published on Save, as Form B has subscribed to the value it picks its data extract and form xml is updated with the *blah* value.
   The page Maestro Form Bundle - Test Cases - Overview has a diagram showing how publish subscribe is configured.

Show Previous Forms

The Form Bundle Context is populated in the task when the step attribute **showPreviousForms is true**.
Click on the link above to see the content of the data that is returned to the form. Can you see how it can be used by form and form widget that display the previous forms in a bundle.

Forms in the Same Step Editable

Forms in a bundle step are editable when the step attribute **allFormsEditable** is **true** and when the step has still not completed. This happens even if the form tasks in the step have previously been submitted.

There are Collaboration Job Bundle Navigation widgets which list form links to previous forms. Clicking on a link will open the selected form by redirecting to it. It may also be possible to click on a link to download a receipt for a previous submitted form.

Form Chaining - redirectNext

redirectNext is both a Step and an Action attribute

For form chaining to be used withing a steps set the step attributeset **redirectNext is true**

For form chainging from 1 action to another the action set the action attribute redirectNext is true.

Lets analyse the Customer Onboarding Job - Form Bundle. In particular lets look at the Additional Products Step see JSON below.

We can see

- the Steps has **"redirectNext": true,**
- The first 2 Task Actions,Credit Card Application and Insurance Application Forms, have redirect next **"redirectNext": true,**
- The last Action Application Confirmation does not have a redirectNext attribute or **"redirectNext": false,**

The Triage form is filled in, depending upon the user selection in the triage form and the either or both Credit Card and Insurance **preCondition** will evaluate to true.

Lets say both products selected. The user submits the triage form it chains to the Credit Card. The order for the next available form is determined by the order of the Action in the Step that is

Triage Form Credit Card Insurance Confirmation

The form chaning contiues to the confirmation page. However at the confirmation page it breaks the chaining flow and goes to the Application Confirmation form's - Submission Confirmation page. At this page the user can download a combined PDF receipt.

Additional Products Step

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

Change of User

Form Chaining can happen only when the current user has access newly created task. That is form changing happens:

- Anonymous Forms the email address needs to stay the same.
- Authenticated task either:
    - the current user is assigned to the new task
    - the task is assigned to a group that the current user belongs to
    - group tasks are claimable are also required to be claimed by the current user.

When there is a change of assignment that does not meet the conditions above, additional tasks cannont be openned by the current user. The current form stops chaining and goes to the forms submission confirmation page.

Legacy Mode

There is an legacy UI flow that was in earlier revisions of form bundles 4.2 did not load the next available form, that is did not perform form chaining.

When a user submitts the submission the user is then presented with a list of forms on the forms submission confirmation page, one for each form that they were to completed.

We recommend that form chaining be used as it is a better user experience.

PreConditions

All the Job Task Assign Actions in a bundle steps can have optional PreConditions set. Here is an example from

```
    "preCondition": "$formDataMap.productCreditCards == 'true'",
```

Pre conditions trigger if the velocity template on the right evaluates to true. When triggered the action instance and associated task will be created.

By default the preconditions are evaluated against the **start submission**. If the previous submission isn't the start submission, its possible that the data extract fields are populated in the previous submission but are missing from the start submission. If this is the case the precondition may not fire.

To fix this we will need to specify the **Pre Condition Submission Step** to be used for evaluating pre conditions. Please see the step properties array in bold below. Note the "*Triage Step Name*" should be replaced with the actual step name that contains triage form submission.

```
{
  "name": "A Form Bundle",
  "type": "",
  "dynamicPreConditions": true,
  "shareFormData": true,
  "showPreviousForms": true,
  "redirectNext": true,
  "actions": [ ... ],
  "properties": [
    {"name": "Pre Condition Submission Step", "value" : "Triage Step Name"}
  ],
  "routes": [ ... ]
}
```

Dynamic PreConditions

Normal or static preconditions are evaluated once when the step is first run using the data extracts for the precondition submission.

Dynamic preconditions fire after the submission of any task in the bundle. The data extracts will be taken from the bundle form is submitted and evaluated against the actions whose preconditions have not already fired.

Note: actions that have already been created because previously the preconditions was true, are not affected if their preconditions now evaluate to false.

To configure dynamicPreconditions in a bundle step add the following step attribute:

```
    "dynamicPreConditions": true,
```

Dynamic Precondition and Action Order

The order that the forms are made available in a form bundle is taken from the order they are listed in the job definition. With dynamic preconditions the next form loaded is the highest one on the list that hasn't been submitted.
I will use the **Additional Products** step of a collaboration job as an example. Please see page with summary of the Job Definition below

Lets say we pick the insurance product option only in the triage form. The result is the insurance application and the application confirmation are created.

The next available form is the Insurance Application and this is opened next. In this form we now select the Credit Card product. The dynamic precondition is triggered and the Credit Card Application form is created.

The next available form is the Credit Card Application as it is at the top of the list. We submit this form.

As the Insurance Application was already submitted the chaining opens next available is the Application Confirmation.

The Application Confirmation lists all the forms in order that they appear in the job and the step.

```
{
  "name": "Additional Products",
  "type": "",
  "dynamicPreConditions": true,
  "shareFormData": true,
  "showPreviousForms": true,
  "redirectNext": true,
  "actions": [
    {
      "name": "Credit Cards Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productCreditCards == 'true'",
          ...
    },
    {
      "name": "Insurance Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productInsurance == 'true'",
      ...
    },
    {
      "name": "Appliction Confirmation",
      "type": "Job Task Assign",
      ...
        },
    ...
  ],
  "routes": [ ... ]
}
```

User Authentication

Form Bundles support both Anonymous and Authenticated (Based on Form or Review Task) Flows. Please see Task Types - Anonymous, Form and Review for a detailed view of what action properties are available to the task.

Note: Mixing of task types (Anonymous, Form and Review) in the same form bundle step is Not supported.

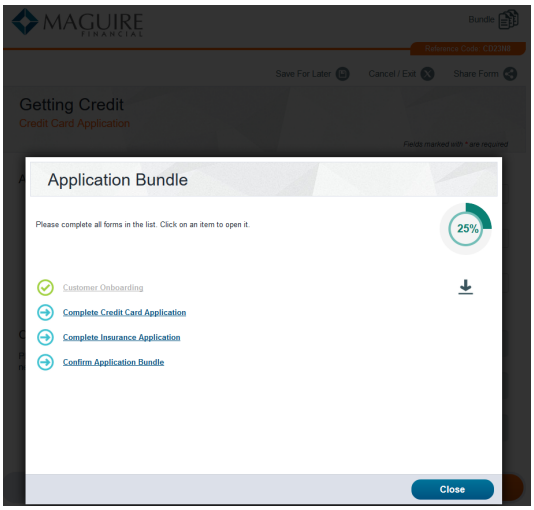
The following links to the form bundles setup as well as giving more advance examples of dynamic form bundles.

- Anonymous Form Bundles
- Authenticated Form Bundles

Form Bundle Context

Collaboration Jobs provides additional context information to Form Bundles created using Maestro or the Composer / Maguire Template.

This information is automatically used by forms and widgets to provide a list of all the other forms in the bundle, there completion status and a way to navigate to the the chosen form as well as print a receipt. The image below show a composer widget.



The partial form xml shown below is from the Customer Onboarding Job - Form Bundle. The data is created in the **//SystemProfile/Job/StepTasks** xml node. It is a list of the previous and current step tasks for the form bundle.

Form XML

```
<SystemProfile>
    <DisplayMode>Entry</DisplayMode>
    <HostContext>Page</HostContext>
    <ReceiptNumber></ReceiptNumber>
    <SubmitDateString></SubmitDateString>
    <SubmissionMessage></SubmissionMessage>
    <FormDataServiceURL>https://tm.maestro.avoka.com/maguire/servlet/FormDynamicDataServlet</FormDataServiceURL>
    <RequestLogKey>ab45894a127f2e26a3e516ba4291080a</RequestLogKey>
    <TrackingCode>PT3F6P</TrackingCode>
    <Job>
        <ReferenceNumber>QF3FJK</ReferenceNumber>
        <StepName>Additional Products</StepName>
        <AvailableRoutes>Default</AvailableRoutes>
        <RouteName></RouteName>
        <Assignee>jsmith@gmail.com</Assignee>
        <AssignRepeatIndex></AssignRepeatIndex>
        <AssignRepeatItem></AssignRepeatItem>
        <Description></Description>
        <StepTasks>
            <Task>
                <StepName>Customer Onboarding</StepName>
                <MenuLabel>Customer Onboarding</MenuLabel>
                <MenuCategory>Customer Onboarding</MenuCategory>
                <TaskSubject>Customer Onboarding</TaskSubject>
                <FormCompleted>true</FormCompleted>
                <FormEditable>false</FormEditable>
                <FormName>Customer Onboarding</FormName>
                <ReceiptURL>https://tm.maestro.avoka.com/maguire/servlet/FormReceipt.pdf?
submitKey=dfe15aab5345ee23505021f889ac0ff1&amp;renderMode=file</ReceiptURL>
                <CurrentForm>false</CurrentForm>
                <TaskMessage></TaskMessage>
                <ScheduledTime></ScheduledTime>
                <ExpiryTime></ExpiryTime>
                <FormURL></FormURL>
            </Task>
            <Task>
                <StepName>Additional Products</StepName>
                <MenuLabel>Complete Credit Card Application</MenuLabel>
                <MenuCategory>Additional Products</MenuCategory>
                <TaskSubject>Complete Credit Card Application</TaskSubject>
                <TaskMessage>Please complete the Credit Card Application form.</TaskMessage>
                <FormCompleted>false</FormCompleted>
                <FormEditable>true</FormEditable>
```

```
                <FormName>Customer Onboarding - Credit Cards</FormName>
                <FormURL>https://tm.maestro.avoka.com/maguire/form.htm?
submitKey=2ef6da250d9d1dbee74749172f64e3e4</FormURL>
                <CurrentForm>true</CurrentForm>
                <ScheduledTime></ScheduledTime>
                <ExpiryTime></ExpiryTime>
                <ReceiptURL></ReceiptURL>
            </Task>
            <Task>
                <StepName>Additional Products</StepName>
                <MenuLabel>Complete Insurance Application</MenuLabel>
                <MenuCategory>Additional Products</MenuCategory>
                <TaskSubject>Complete Insurance Application</TaskSubject>
                <TaskMessage>Please complete the Insurance Application form.</TaskMessage>
                <FormCompleted>false</FormCompleted>
                <FormEditable>true</FormEditable>
                <FormName>Customer Onboarding - Insurance</FormName>
                <FormURL>https://tm.maestro.avoka.com/maguire/form.htm?
submitKey=ff13980c03dccd2963ba8c6be469e248</FormURL>
                <CurrentForm>false</CurrentForm>
                <ScheduledTime></ScheduledTime>
                <ExpiryTime></ExpiryTime>
                <ReceiptURL></ReceiptURL>
            </Task>
            <Task>
                <StepName>Additional Products</StepName>
                <MenuLabel>Confirm Application Bundle</MenuLabel>
                <MenuCategory>Additional Products</MenuCategory>
                <TaskSubject>Confirm Application Bundle</TaskSubject>
                <TaskMessage>Please sign to complete Application bundle.</TaskMessage>
                <FormCompleted>false</FormCompleted>
                <FormEditable>true</FormEditable>
                <FormName>Customer Onboarding - Confirmation</FormName>
                <FormURL>https://tm.maestro.avoka.com/maguire/form.htm?
submitKey=8e6b6352eb284b03b10ea36fa0b454f2</FormURL>
                <CurrentForm>false</CurrentForm>
                <ScheduledTime></ScheduledTime>
                <ExpiryTime></ExpiryTime>
                <ReceiptURL></ReceiptURL>
            </Task>
        </StepTasks>
    </Job>
```

Anonymous Form Bundles

Anonymous Bundles require the use of an email address. This was a product design decision, it allows the end user to partially fill out the bundle forms and then complete the form at a later time. Typically a triage form is filled out and submitted. If the user is asked to enter their email on the triage form then the user can be sent an email with a link which can get them back into editing the form bundle.

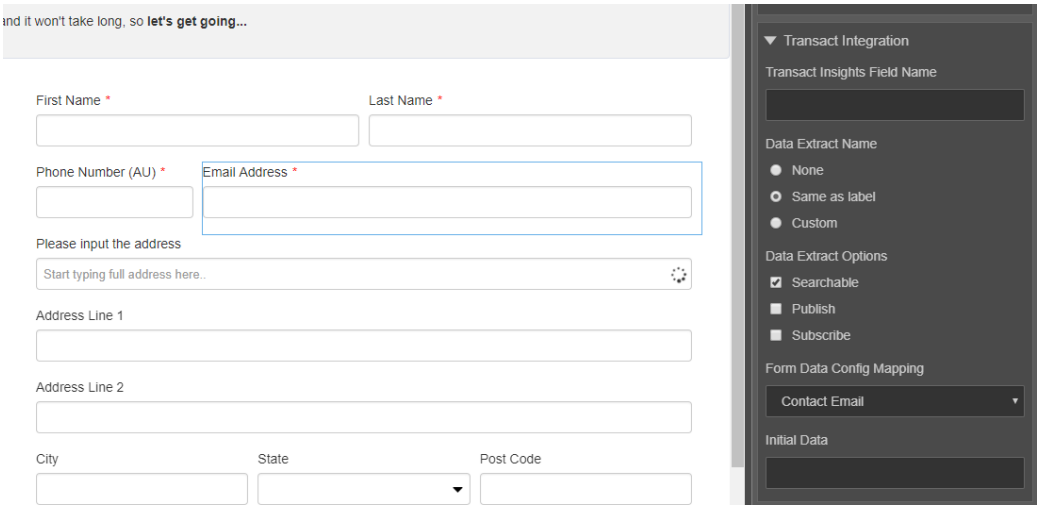
It is possible to use a hard coded contact email as per the Form Configuration section below. Consideration should be given to how much information will be have to be re-keyed by the end user is unable to complete the bundle.

Anonymous Bundle forms should be protected by save challenge.

This example uses the **Maestro Onboarding Job** and there are four Maestro forms.

The Form Configuration details the setup of the email address in the triage form. The Job Definition - Additional Products section below shows the configuration bundle step.

Form Configuration



The screenshot above is taken from Maestro Onboarding - Triage Form which starts the Anonymous Job.

The data extract is setup here **Same as label** this can be referenced further on in the collaboration job below as **$formDataMap.emailAddress**

The **Form Data Config Mapping** has to be set to **Contact Email Address.**

Important! This email address cannot be blank. Where the email address is to be entered by the end user it should be mandatory.

Optionally it is possible in the form to hard code an email address such as onboarding@company.com by:

- setting the **initial data** value to **onboarding@company.com**
- we may want to creating a Maestro form rule to hide the email address or to make it read only.
- A hard coded address is not recommended as the user may need to .

Job Definition - Additional Products

```
{
  "name": "Additional Products",
  "type": "",
  "dynamicPreConditions": true,
  "shareExtractData": false,
  "shareFormData": true,
  "allFormsEditable": true,
  "showPreviousForms": true,
  "redirectNext": true,
  "actions": [
    {
      "name": "Credit Cards Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productCreditCards == 'true'",
      "redirectNext": true,
      "properties": [
        { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
        { "name": "Task Form Code",  "value": "maestroOnBoardingCre" },
        { "name": "Task Message",  "value": "Please complete the Credit Card Application form." },
        { "name": "Task Subject",  "value": "Complete Credit Card Application" },
        { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
        { "name": "Task Type",  "value": "Anonymous" },
        { "name": "Task Save Challenge",  "value": "${formDataMap.phoneNumber}" }
      ]
    },
    {
      "name": "Insurance Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productInsurance == 'true'",
      "redirectNext": true,
      "properties": [
        { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
        { "name": "Task Form Code",  "value": "maestroOnBoardingIns" },
        { "name": "Task Message",  "value": "Please complete the Insurance Application form." },
        { "name": "Task Subject",  "value": "Complete Insurance Application" },
        { "name": "Task Input XML Prefill",  "value": "$func.startSubmissionXml()" },
        { "name": "Task Type",  "value": "Anonymous" },
        { "name": "Task Save Challenge",  "value": "${formDataMap.phoneNumber}" }
      ]
    },
    {
      "name": "Application Confirmation",
```

Highlighted above are the action properties that are typically used for an Anonymous Bundle.

1. { "name": "Task Type", "value": "Anonymous" },

2. { "name": "Task Assign Email", "value": "$formDataMap.emailAddress" },
   Anonymous Form Bundles require an email to be set. the Job Task Assign service will produce and error if the value is blank.
   The email address can be set to a value or read from a data extract from the triage form as per note below.
3. { "name": "Task Input XML Prefill", "value": "$func.startSubmissionXml()" },
   This is how the Anonymous form task gets the XML Data from the previous submission.
   Note: $func.stepOrStartSubmissionXml() can be used instead, and is required if the step is changed to use dynamicPreConditions. (ref to API Link Job Functions)
4. { "name": "Task Save Challenge", "value": "${formDataMap.phoneNumber}" }
   This is how the Save Challenge data is set for the form.

It is possible for anonymous tasks to copy attachments from a previous task. See the API **JobTaskAssignService** Action Properties (**Task Attachments Previous Step** and **Task Attachments Previous Step**)

However sharing of attachments is NOT supported in Form Bundles.

Examples

Example 1

The Job Controller below has been extracted from the **mOnboarding - Anonymous - shareExtractData** job definition JSON.

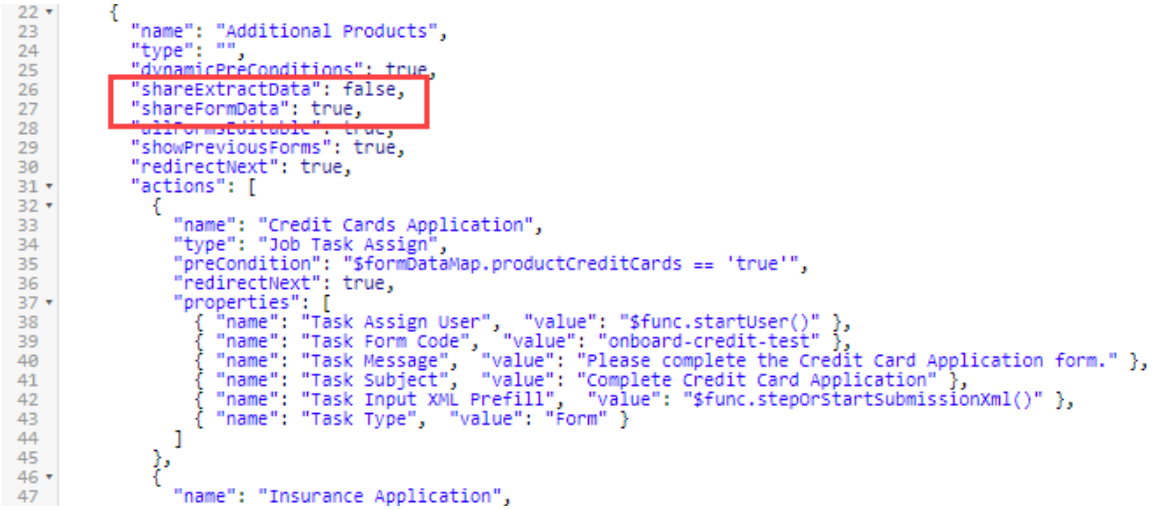
Its features setup in the with step level attribute flags **dynamicPreConditions, ShareExtractData**. As well as the standard **allFormsEditable, showPreviousForms and redirectNext**.

```
{
  "name": "Additional Products",
  "type": "",
  "dynamicPreConditions": true,
  "shareExtractData": true,
  "shareFormData": false,
  "allFormsEditable": true,
  "showPreviousForms": true,
  "redirectNext": true,
  "actions" : [
    {
      "name": "Credit Cards Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productCreditCards == 'true'",
      "redirectNext": true,
      "properties": [
        { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
        { "name": "Task Form Code",  "value": "mtestonboardingcred2" },
        { "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": "$func.stepOrStartSubmissionXml()" },
        { "name": "Task Type",  "value": "Anonymous" },
        { "name": "Task Save Challenge",  "value": "${formDataMap.phoneNumber}" }
      ]
    },
    {
      "name": "Insurance Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productInsurance == 'true'",
      "redirectNext": true,
      "properties": [
        { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
        { "name": "Task Form Code",  "value": "mtestonboardinginsu2" },
        { "name": "Task Message",  "value": "Please complete the Insurance Application form." },
        { "name": "Task Subject",  "value": "Complete Insurance Application" },
        { "name": "Task Input XML Prefill",  "value": "$func.stepOrStartSubmissionXml()" },
        { "name": "Task Type",  "value": "Anonymous" },
        { "name": "Task Save Challenge",  "value": "${formDataMap.phoneNumber}" }
      ]
    },
    {
      "name": "Application Confirmation",
      "type": "Job Task Assign",
```

Example 2

The Job Controller below has been extracted from the **mOnboarding - Anonymous - shareFormData** job definition JSON.

Its features setup in the with step level attribute flags **dynamicPreConditions, ShareFormData**. As well as the standard **allFormsEditable, showPreviousForms and redirectNext**.



```
{
  "name": "Additional Products",
  "type": "",
  "dynamicPreConditions": true,
  "shareExtractData": false,
  "shareFormData": true,
  "allFormsEditable": true,
  "showPreviousForms": true,
  "redirectNext": true,
  "actions": [
    {
      "name": "Credit Cards Application",
      "type": "Job Task Assign",
      "preCondition": "$formDataMap.productCreditCards == 'true'",
      "redirectNext": true,
      "properties": [
        { "name": "Task Assign User",  "value": "$func.startUser()" },
        { "name": "Task Form Code",  "value": "onboard-credit-test" },
        { "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.stepOrStartSubmissionXml()" },
        { "name": "Task Type",  "value": "Form" }
      ]
    },
    {
      "name": "Insurance Application",
```

Authenticated Form Bundles

Prerequisites

To understand this example it is recommended that you get familiar with the following pages:

Form Space

Task Types - Anonymous, Form and Review

Chaining and Form Bundles

Anonymous Form Bundles

The example collaboration jobs can be found here

.Maestro Form Bundle - Test Cases

Overview

User and Group Action Properties For Form Bundle

Authenticated tasks in a form bundle can be assigned to a group

- **User Task**: the current user is assigned to the new task.

  { "name": "Task Assign User", "value": "$func.startUser()" },

  or alternatively

  { "name": "Task Assign User", "value": "$func.previousUser()" },

- **Non Claimable Tasks:** the task is assigned to a group that the current user belongs to.
  Lets assume the user is in the **accounts** group.

  { "name": "Task Assign Group", "value": "accounts" },

  or if there is a **supervisor** group for the task

  { "name": "Task Assign Groups", "value": "accounts,supervisor" },

- **Claimable Group Tasks**: are also required to be claimed by the current user. The claiming is required for the redirect Next to work.

  { "name": "Task Assign User", "value": "$func.startUser()" },
  { "name": "Task Assign Group", "value": "accounts" },

  or if there is a **supervisor** group for the task

  { "name": "Task Assign User", "value": "$func.previousUser()" },
  { "name": "Task Assign Groups", "value": "accounts,supervisor" },

Task Types

Authenticated Form bundles can be build from are based upon Form and Review Tasks Types. Whilst Review Type tasks can be made to work in a bundle we would recommend sticking to Form Task Types.

Form Tasks

Example 1

The Job Controller below has been extracted from the **mOnboarding - Form Tasks - shareExtractData** job definition JSON.

Its features setup in the with step level attribute flags **dynamicPreConditions, ShareExtractData**. As well as the standard **allFormsEditable, showPreviousForms and redirectNext**.

All form tasks in the bundle require action properties of

- { "name": "Task Type", "value": "Form" }
- { "name": "Task Input XML Prefill", "value": "$func.stepOrStartSubmissionXml()" },
  alternatively the following
  { "name": "Task Input XML Prefill", "value": "$func.stepOrPreviousSubmissionXml()" },

$func.stepOrStartSubmission or $func.stepOrPreviousSubmissionXml() are required for dynamic preconditions to work. They also work just as well for static bundles.

.

Example 2

Form Task Bundle configured to shareFormData. The only difference are the highlighted flags.

Review Task

Review tasks are primaraly used for review and approval processes that use the same form to review as was submitted by the user. They are configured by specifying a previous step or submission. The review task will copy the XML data, attachments from the triage form. You need to specifying a form code so the review task does not use the triage form.

Review tasks should only setup with shareExtractData option.

Example 3

This example has been extracted from **mOnboarding - Review Task - shareExtractData**.

The flags are the same as Example 1,

The Task Type should be setup as Review

{ "name": "Task Type", "value": "Review" }

The following gets the review data from a previous step.

{ "name": "Task Review Previous Step", "value": "true" },

Transact Collaboration Jobs API Reference

com.avoka.fc.core.service.job

- ActionContext
- ActionResult
- ActionResult.Status
- IJobActionService
- IJobController
- JobEventLogService

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

- ActionDef
- IDefJsonSerializer
- JobDef
- JsonSerializerUtils
- StepDef

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

- AbstractJobActionService
- ActionStepProperties
- GroovyJobActionService
- JobActionUtils
- JobActionWaitService
- JobControllerService
- JobDeliveryService
- JobDeliveryWaitService
- JobFormStartService
- JobFunctions
- JobProcessMessageService
- JobReceiptWaitService
- JobTaskAssignActionBuilder
- JobTaskAssignService
- JobTaskWaitService

Constants

com.avoka.fc.core.service.job

Interfaces

- IJobActionService
- IJobController

Classes in Package com.avoka.fc.core.service.job

- ActionContext
- ActionResult
- JobEventLogService

Enums

- ActionResult.Status

ActionContext

Package:

- com.avoka.fc.core.service.job

Class ActionContext

- java.lang.Object
- com.avoka.fc.core.service.job.ActionContext

```
public class ActionContext
extends java.lang.Object
```

Provide a Job Action execution context.

Since:

- 4.0.0

Constructor Summary
Constructors

Constructor and Description
ActionContext-com.avoka.fc.core.entity.JobAction-com.avoka.fc.core.service.job.config.JobDef-java.util.Map-java.util.Map-(com.avoka.fc.core.entity.JobAction jobAction, JobDef jobDef, java.util.Map<java.lang.String,java.lang.String> jobControllerParameters, java.util.Map<java.lang.String,java.lang.String> jobProperties) Create a Job ActionContext with the given Job Action, Job Def and Job Properties.
ActionContext-com.avoka.fc.core.entity.JobAction-com.avoka.fc.core.service.job.config.JobDef-java.util.Map-java.util.Map-com.avoka.fc.core.service.job.config.ActionDef-(com.avoka.fc.core.entity.JobAction jobAction, JobDef jobDef, java.util.Map<java.lang.String,java.lang.String> jobControllerParameters, java.util.Map<java.lang.String,java.lang.String> jobProperties, ActionDef actionDef) Create a Job ActionContext with the given Job Action, Job Def and Job Properties.

Method Summary
All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
ActionDef	getActionDef--()
com.avoka.fc.core.entity.JobAction	getJobAction--()
java.util.Map<java.lang.String,java.lang.String>	getJobControllerParameters--() Returns the map of job controller service definition service parameter values
JobDef	getJobDef--()
java.util.Map<java.lang.String,java.lang.String>	getJobProperties--()
StepDef	getStepDef--()
java.lang.String	toString--()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ActionContext

```
public ActionContext(com.avoka.fc.core.entity.JobAction jobAction,

     JobDef jobDef,
                  java.util.Map<java.lang.String,java.lang.String> jobControllerParameters,
                  java.util.Map<java.lang.String,java.lang.String> jobProperties)
```

Create a Job ActionContext with the given Job Action, Job Def and Job Properties.

Parameters:

- jobAction - the Job Action (required)

- `jobDef` - the Job Def (required)
- `jobControllerParameters` - the map of job controller service definition service parameter values (required)
- `jobProperties` - the Job Properties (required)

ActionContext

```
public ActionContext(com.avoka.fc.core.entity.JobAction jobAction,

     JobDef jobDef,
                    java.util.Map<java.lang.String,java.lang.String> jobControllerParameters,
                    java.util.Map<java.lang.String,java.lang.String> jobProperties,

     ActionDef actionDef)
```

Create a Job ActionContext with the given Job Action, Job Def and Job Properties.

Parameters:

- `jobAction` - the Job Action (required)
- `jobDef` - the Job Def (required)
- `jobControllerParameters` - the map of job controller service definition service parameter values (required)
- `jobProperties` - the Job Properties (required)
- `actionDef` - the ActionDef action definition (required)

Since:

- 4.3.4

Method Detail

---

getJobDef

```
public
     JobDef getJobDef()
```

Returns:

- the jobDef

---

getStepDef

```
public
     StepDef getStepDef()
```

Returns:

- the stepDef

---

getActionDef

```
public
     ActionDef getActionDef()
```

Returns:

- the actionDef

---

getJobAction

```
public com.avoka.fc.core.entity.JobAction getJobAction()
```

Returns:

- the jobAction

---

getJobProperties

```
public java.util.Map<java.lang.String,java.lang.String> getJobProperties()
```

Returns:

- the jobProperties

---

getJobControllerParameters

```
public java.util.Map<java.lang.String,java.lang.String> getJobControllerParameters()
```

Returns the map of job controller service definition service parameter values

Returns:

- the map of job controller service definition service parameter values

---

toString

`public java.lang.String toString()`

Overrides:

- `toString` in class `java.lang.Object`

Returns:

- the string representation of this object

ActionResult

Package:

- com.avoka.fc.core.service.job

Class ActionResult

- java.lang.Object
- com.avoka.fc.core.service.job.ActionResult

```
public class ActionResult
extends java.lang.Object
```

Provides the Job Action result class.

Since:

- 4.0.0

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
static class	ActionResult.Status The delivery status values.

Constructor Summary

Constructors

Constructor and Description
ActionResult-com.avoka.fc.core.service.job.ActionResult.Status-(ActionResult.Status status) Create a new ActionResult object with the given status.

Method Summary

All Methods Static Methods Instance Methods Concrete Methods Deprecated Methods

Modifier and Type	Method and Description
java.util.Date	getExpiryTime--()
java.lang.Integer	getMaxRetryAttempts--()
java.lang.Integer	getMaxRetryAttepts--() Deprecated. use getMaxRetryAttempts() instead
java.lang.String	getMessage--()
java.util.Date	getNextRetryAttempt--()
java.lang.String	getRoute--()
java.lang.String	getStatus--()
boolean	isStatusError--()
void	setExpiryTime-java.util.Date-(java.util.Date expiryTime)
void	setMaxRetryAttempts-java.lang.Integer-(java.lang.Integer maxRetryAttempts)
void	setMaxRetryAttepts-java.lang.Integer-(java.lang.Integer maxRetryAttepts) Deprecated. use setMaxRetryAttempts(Integer) instead
void	setMessage-java.lang.String-(java.lang.String message)
void	setNextRetryAttempt-java.util.Date-(java.util.Date nextRetryAttempt)
void	setRoute-java.lang.String-(java.lang.String route)
static ActionResult.Status	toActionStatus-java.lang.String-(java.lang.String status)

	Return the corresponding Status enumeration for the text status.
java.lang.String	toString--()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ActionResult

```
public ActionResult(
     ActionResult.Status status)
```

Create a new ActionResult object with the given status.

Parameters:

- status - the result status

Method Detail

getRoute

```
public java.lang.String getRoute()
```

Returns:

- the route

---

setRoute

```
public void setRoute(java.lang.String route)
```

Parameters:

- route - the route to set

---

getStatus

```
public java.lang.String getStatus()
```

Returns:

- the status

---

isStatusError

```
public boolean isStatusError()
```

Returns:

- true if the action result status was 'Error'

---

getNextRetryAttempt

```
public java.util.Date getNextRetryAttempt()
```

Returns:

- the nextRetryAttempt

---

setNextRetryAttempt

```
public void setNextRetryAttempt(java.util.Date nextRetryAttempt)
```

Parameters:

- nextRetryAttempt - the nextRetryAttempt to set

---

getMaxRetryAttempts

```
public java.lang.Integer getMaxRetryAttempts()
```

Returns:

- the maxRetryAttempts

---

setMaxRetryAttempts

```
public void setMaxRetryAttempts(java.lang.Integer maxRetryAttempts)
```

Parameters:

- `maxRetryAttempts` - the maxRetryAttempts to set

---

getMaxRetryAttepts

```
@Deprecated
public java.lang.Integer getMaxRetryAttepts()
```

Deprecated. use `getMaxRetryAttempts()` instead

Returns:

- the maxRetryAttempts

---

setMaxRetryAttepts

```
@Deprecated
public void setMaxRetryAttepts(java.lang.Integer maxRetryAttepts)
```

Deprecated. use `setMaxRetryAttempts(Integer)` instead

Parameters:

- `maxRetryAttepts` - the maxRetryAttempts to set

---

getExpiryTime

```
public java.util.Date getExpiryTime()
```

Returns:

- the expiryTime

---

setExpiryTime

```
public void setExpiryTime(java.util.Date expiryTime)
```

Parameters:

- `expiryTime` - the expiryTime to set

---

getMessage

```
public java.lang.String getMessage()
```

Returns:

- the action result message

---

setMessage

```
public void setMessage(java.lang.String message)
```

Parameters:

- `message` - the action result message

---

toActionStatus

```
public static
     ActionResult.Status toActionStatus(java.lang.String status)
```

Return the corresponding Status enumeration for the text status.

Parameters:

- `status` - the text status value (required)

Returns:

- the corresponding Status enumeration for the text status

Since:

- 5.0

---

toString

`public java.lang.String toString()`

Overrides:

- `toString` in class `java.lang.Object`

Returns:

- the string representation of this object.

ActionResult.Status

Package:

- com.avoka.fc.core.service.job

Enum ActionResult.Status

- java.lang.Object

java.lang.Enum< ActionResult.Status>

- com.avoka.fc.core.service.job.ActionResult.Status

All Implemented Interfaces:

- java.io.Serializable, java.lang.Comparable< ActionResult.Status>

Enclosing class:

- ActionResult

```
public static enum ActionResult.Status
extends java.lang.Enum<
     ActionResult.Status>
```

The delivery status values.

Enum Constant Summary

Enum Constants

Enum Constant and Description
ASSIGNED
COMPLETED
ERROR
IN_PROGRESS
INVOKED
PENDING

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
static ActionResult.Status	valueOf-java.lang.String-(java.lang.String name) Returns the enum constant of this type with the specified name.
static ActionResult.Status[]	values--() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum

compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

ASSIGNED

```
public static final
     ActionResult.Status ASSIGNED
```

COMPLETED

```
public static final
     ActionResult.Status COMPLETED
```

ERROR

```
public static final
     ActionResult.Status ERROR
```

PENDING

```
public static final
     ActionResult.Status PENDING
```

IN_PROGRESS

```
public static final
     ActionResult.Status IN_PROGRESS
```

INVOKED

```
public static final
     ActionResult.Status INVOKED
```

Method Detail

values

```
public static
     ActionResult.Status[] values()
```

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (ActionResult.Status c : ActionResult.Status.values())
    System.out.println(c);
```

Returns:

- an array containing the constants of this enum type, in the order they are declared

valueOf

```
public static
     ActionResult.Status valueOf(java.lang.String name)
```

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

- `name` - the name of the enum constant to be returned.

Returns:

- the enum constant with the specified name

Throws:

- `java.lang.IllegalArgumentException` - if this enum type has no constant with the specified name
- `java.lang.NullPointerException` - if the argument is null

IJobActionService

Package:

- com.avoka.fc.core.service.job

Interface IJobActionService

All Known Implementing Classes:

- AbstractJobActionService, GroovyJobActionService, JobActionWaitService, JobDeliveryService, JobDeliveryWaitService, JobFormStartService, JobProcessMessageService, JobReceiptWaitService, JobTaskAssignService, JobTaskWaitService

---

```
public interface IJobActionService
```

Provides a Job Action and Expiry service interface.

Since:

- 4.0.0

Method Summary

All Methods Instance Methods Abstract Methods

Modifier and Type	Method and Description
ActionResult	execute-com.avoka.fc.core.service.job.ActionContext-(ActionContext actionContext) Execute the action and return the result.
java.lang.Integer	getMaxErrorRetryAttempts--() Return the maximum number of error retry attempts should be performed.
java.lang.Integer	getRetryDelayMins--() Return the minimum action execute retry delay in minutes.
java.lang.String	validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-(ActionStepProperties actionStepProperties, com.avoka.fc.core.entity.Client client) Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Method Detail

execute

```
ActionResult execute(
ActionContext actionContext)
```

Execute the action and return the result.

Parameters:

- `actionContext` - the job action execution context (required)

Returns:

- the action result

---

validateProperties

```
java.lang.String validateProperties(
     ActionStepProperties actionStepProperties,
                                 com.avoka.fc.core.entity.Client client)
```

Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Parameters:

- `actionStepProperties` - the action step properties (required)
- `client` - the Job Controller client (required)

Returns:

- null if the properties are valid or null otherwise

---

getMaxErrorRetryAttempts

```
java.lang.Integer getMaxErrorRetryAttempts()
```

Return the maximum number of error retry attempts should be performed.

Returns:

- the maximum number of error retry attempts should be performed

---

getRetryDelayMins

```
java.lang.Integer getRetryDelayMins()
```

Return the minimum action execute retry delay in minutes.

Returns:

- the minimum action execute retry delay in minutes

IJobController

Package:

- com.avoka.fc.core.service.job

Interface IJobController

All Known Implementing Classes:

- JobControllerService

```
public interface IJobController
```

Provides the Job Controller service interface which orchestrates the job workflow.

Since:

- 4.0.0

Method Summary

All Methods Instance Methods Abstract Methods

Modifier and Type	Method and Description
boolean	cancelJob-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job) Cancel the job, abandon any Assigned or Saved Tasks and make submission data ready for purging.
boolean	cancelledTaskSubmission-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Provides a callback to tell the Job controller that the given task submission has been cancelled.
boolean	completedJobAsyncAction-java.lang.String-(java.lang.String jobActionKey) Provides a callback to tell the Job controller that the specified asynchronous action ('Job Async Action') has been completed.
boolean	completedSubmissionDelivery-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Provides a callback to tell the Job controller that the specified submission delivery has been completed.
boolean	completedTaskSubmission-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Provides a callback to tell the Job controller that the given task submission has been completed.
com.avoka.fc.core.entity.Job	createJobWithClient-com.avoka.fc.core.entity.Client-(com.avoka.fc.core.entity.Client client) Create a job with the client, initializing the first step and action and put in 'Ready' state.
com.avoka.fc.core.entity.Job	createJobWithSubmission-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Create a job with the submission, initializing the first step and action, and using the given submission and put step action in 'Ready' state.
boolean	isJobControllerEnabled--() Return true if the job controller service is enabled and should be processed by the scheduled job.
boolean	pausedJob-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job) Pause the job from performing any further processing.
void	processJob-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job) Process the job when a job action is in a 'Ready' state.
boolean	resumeJob-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job) Resume the job to perform further processing.

Method Detail

createJobWithClient

```
com.avoka.fc.core.entity.Job createJobWithClient(com.avoka.fc.core.entity.Client client)
```

Create a job with the client, initializing the first step and action and put in 'Ready' state.
If the controller is configured to process the job immediately after creation, this method will also process the job before it is returned.

Parameters:

- `client` - the client organization which owns the job (required)

Returns:

- a new job instance

---

createJobWithSubmission

`com.avoka.fc.core.entity.Job createJobWithSubmission(com.avoka.fc.core.entity.Submission submission)`

Create a job with the submission, initializing the first step and action, and using the given submission and put step action in 'Ready' state.
If the controller is configured to process the job immediately after creation, this method will also process the job before it is returned. This can be useful to create an Task assigned to the user who created the job with the initial submission.

Parameters:

- `submission` - the initial job submission (required)

Returns:

- a new job instance

---

processJob

`void processJob(com.avoka.fc.core.entity.Job job)`

Process the job when a job action is in a 'Ready' state.

Parameters:

- `job` - the Job instance to process

---

completedTaskSubmission

`boolean completedTaskSubmission(com.avoka.fc.core.entity.Submission submission)`

Provides a callback to tell the Job controller that the given task submission has been completed. The job controller should progress the associated task assign action status to 'Completed' and the next task wait action status to 'Ready'.

Parameters:

- `submission` - the completed task submission (required)

Returns:

- true if the submission task was found and associated "Task Assigned" was set to "Completed" or false otherwise

---

cancelledTaskSubmission

`boolean cancelledTaskSubmission(com.avoka.fc.core.entity.Submission submission)`

Provides a callback to tell the Job controller that the given task submission has been cancelled. The job controller should progress the associated task assign action status to 'Cancelled' and the next task wait action status to 'Ready'.

Parameters:

- `submission` - the cancelled task submission (required)

Returns:

- true if the submission task was found and associated "Task Assigned" was set to "Cancelled" or false otherwise

Since:

- 4.3.0

---

completedJobAsyncAction

`boolean completedJobAsyncAction(java.lang.String jobActionKey)`

Provides a callback to tell the Job controller that the specified asynchronous action ('Job Async Action') has been completed. The job controller should progress the associated action status to 'Completed' and the next wait action status to 'Ready'.

Parameters:

- `jobActionKey` - the unique job action key identifier (required)

Returns:

- true if the job action was found and set to "Completed" or false otherwise

---

completedSubmissionDelivery

```
boolean completedSubmissionDelivery(com.avoka.fc.core.entity.Submission submission)
```

Provides a callback to tell the Job controller that the specified submission delivery has been completed. The job controller should progress the any associated delivery wait action status to 'Ready'.

Parameters:

- `submission` - the associated delivered submission (required)

Returns:

- true if the submission task was found and associated "Delivery Wait" was set to "Completed" or false otherwise

---

cancelJob

```
boolean cancelJob(com.avoka.fc.core.entity.Job job)
```

Cancel the job, abandon any Assigned or Saved Tasks and make submission data ready for purging.

Parameters:

- `job` - the job instance to cancel (required)

Returns:

- true if the job was cancelled, or false if the job was not "In Progress"

---

pausedJob

```
boolean pausedJob(com.avoka.fc.core.entity.Job job)
```

Pause the job from performing any further processing.

Parameters:

- `job` - the job instance to pause (required)

Returns:

- true if the job was paused

---

resumeJob

```
boolean resumeJob(com.avoka.fc.core.entity.Job job)
```

Resume the job to perform further processing.

Parameters:

- `job` - the job instance to pause (required)

Returns:

- true if the job was resumed

---

isJobControllerEnabled

```
boolean isJobControllerEnabled()
```

Return true if the job controller service is enabled and should be processed by the scheduled job. This property can be used to prevent a controllers jobs from being processed while some other maintenance action is being performed.

Returns:

- true if the controller service is enabled

JobEventLogService

Package:

- com.avoka.fc.core.service.job

Class JobEventLogService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- com.avoka.fc.core.service.job.JobEventLogService

```
public class JobEventLogService
extends com.avoka.fc.core.service.CayenneService
```

Provides a Job Event Logging service.

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
`JobEventLogService--()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`void`	`logErrorEventWithAction-java.lang.String-com.avoka.fc.core.entity.JobAction-(java.lang.String message, com.avoka.fc.core.entity.JobAction jobAction)` Log a job error event with the given message and job action
`void`	`logErrorEventWithJob-java.lang.String-com.avoka.fc.core.entity.Job-(java.lang.String message, com.avoka.fc.core.entity.Job job)` Log a job error event with the given message and job action
`void`	`logInfoEventWithAction-java.lang.String-com.avoka.fc.core.entity.JobAction-(java.lang.String message, com.avoka.fc.core.entity.JobAction jobAction)` Log a job info event with the given message and job action
`void`	`logInfoEventWithJob-java.lang.String-com.avoka.fc.core.entity.Job-(java.lang.String message, com.avoka.fc.core.entity.Job job)` Log a job info event with the given message and job action
`void`	`logWarnEventWithAction-java.lang.String-com.avoka.fc.core.entity.JobAction-(java.lang.String message, com.avoka.fc.core.entity.JobAction jobAction)` Log a job warn event with the given message and job action
`void`	`logWarnEventWithJob-java.lang.String-com.avoka.fc.core.entity.Job-(java.lang.String message, com.avoka.fc.core.entity.Job job)` Log a job warn event with the given message and job action

Methods inherited from class com.avoka.fc.core.service.CayenneService

`commitChanges, rollbackChanges`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

JobEventLogService

```
public JobEventLogService()
```

Method Detail

logErrorEventWithAction

```
public void logErrorEventWithAction(java.lang.String message,
                                     com.avoka.fc.core.entity.JobAction jobAction)
```

Log a job error event with the given message and job action

Parameters:

- `message` - error message (required)
- `jobAction` - the job action to be associated with the event log

---

logErrorEventWithJob

```
public void logErrorEventWithJob(java.lang.String message,
                                 com.avoka.fc.core.entity.Job job)
```

Log a job error event with the given message and job action

Parameters:

- `message` - error message (required)
- `job` - the job to be associated with the event log

---

logWarnEventWithAction

```
public void logWarnEventWithAction(java.lang.String message,
                                   com.avoka.fc.core.entity.JobAction jobAction)
```

Log a job warn event with the given message and job action

Parameters:

- `message` - error message (required)
- `jobAction` - the job action to be associated with the event log

---

logWarnEventWithJob

```
public void logWarnEventWithJob(java.lang.String message,
                                com.avoka.fc.core.entity.Job job)
```

Log a job warn event with the given message and job action

Parameters:

- `message` - error message (required)
- `job` - the job to be associated with the event log

---

logInfoEventWithAction

```
public void logInfoEventWithAction(java.lang.String message,
                                   com.avoka.fc.core.entity.JobAction jobAction)
```

Log a job info event with the given message and job action

Parameters:

- `message` - error message (required)
- `jobAction` - the job action to be associated with the event log

---

logInfoEventWithJob

```
public void logInfoEventWithJob(java.lang.String message,
                                com.avoka.fc.core.entity.Job job)
```

Log a job info event with the given message and job action

Parameters:

- `message` - error message (required)
- `job` - the job to be associated with the event log

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

Interfaces

- IDefJsonSerializer

Classes in Package com.avoka.fc.core.service.job.config

- ActionDef
- JobDef
- JsonSerializerUtils
- StepDef

ActionDef

Package:

- com.avoka.fc.core.service.job.config

Class ActionDef

- java.lang.Object
- com.avoka.fc.core.service.job.config.ActionDef

All Implemented Interfaces:

- IDefJsonSerializer

```
public class ActionDef
extends java.lang.Object
implements
     IDefJsonSerializer
```

Provides an immutable Action configuration class.

Action JSON Configuration

```
{
   "name" : "Handle Submission",
   "type" : "Task Wait",
   "serviceName" : "Review And Approval Handle Submission",
   "serviceVersion" : 3,
   "preCondition" : "$formDatMap.type == 'Home Insurance'"
}
```

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
ActionDef-java.util.Map-java.lang.String-java.lang.String-int-(java.util.Map actionMap, java.lang.String stepName, java.lang.String jobJson, int sequence) Create an Action configuration object from the given action configuration map (JSON).
ActionDef-java.util.Map-java.lang.String-java.lang.String-int-boolean-(java.util.Map actionMap, java.lang.String stepName, java.lang.String jobJson, int sequence, boolean useJobJsonErrorChecking) Create an Action configuration object from the given action configuration map (JSON).

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
java.lang.String	getName--()
java.lang.String	getPreCondition--()
java.util.Map<java.lang.String,java.lang.String>	getPropertyMap--()
java.lang.String	getPropertyValue-java.lang.String-(java.lang.String propertyName)
int	getSequence--()
java.lang.String	getServiceName--()
java.lang.Integer	getServiceVersion--()
java.lang.String	getType--()
java.lang.Boolean	isPropertyTemplateEvaluation-java.lang.String-(java.lang.String propertyName) Return true if the property value should be evaluated with a Velocity template or false otherwise
boolean	isRedirectNext--()
java.lang.String	toString--()
void	writeJson-java.lang.StringBuilder-(java.lang.StringBuilder builder)

	Appends the JSON text representation of this object to the string builder.

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

ActionDef

```
public ActionDef(java.util.Map actionMap,
                 java.lang.String stepName,
                 java.lang.String jobJson,
                 int sequence)
```

Create an Action configuration object from the given action configuration map (JSON). A validation exception is returned if the Action JSON (String): action must contain an non empty name action must contain an valid type @see JobAction.TYPE_VALUES action must contain an non empty serviceName action can contain an expiryServiceName if specified the expiryServiceName if specified cannot be empty.

Parameters:

- `actionMap` - the action configuration map JSON (required)
- `stepName` - the name of the step (required)
- `jobJson` - this job definition JSON source (required)
- `sequence` - the job action sequence in the parent job step

ActionDef

```
public ActionDef(java.util.Map actionMap,
                 java.lang.String stepName,
                 java.lang.String jobJson,
                 int sequence,
                 boolean useJobJsonErrorChecking)
```

Create an Action configuration object from the given action configuration map (JSON). A validation exception is returned if the Action JSON (String): action must contain an non empty name action must contain an valid type @see JobAction.TYPE_VALUES action must contain an non empty serviceName action can contain an expiryServiceName if specified the expiryServiceName if specified cannot be empty.

Parameters:

- `actionMap` - the action configuration map JSON (required)
- `stepName` - the name of the step (required)
- `jobJson` - this job definition JSON source (required)
- `sequence` - the job action sequence in the parent job step
- `useJobJsonErrorChecking` - boolean if set to true will use the JobJson Error Checking.

Since:

- 4.3.4

Method Detail

getName

```
public java.lang.String getName()
```

Returns:

- the action name

---

getType

```
public java.lang.String getType()
```

Returns:

- the action type

---

getSequence

```
public int getSequence()
```

Returns:

- the action sequence in the parent step

Since:

- 4.3.0

---

getServiceName

`public java.lang.String getServiceName()`

Returns:

- the service name

---

getServiceVersion

`public java.lang.Integer getServiceVersion()`

Returns:

- the service version

Since:

- 5.1.0

---

getPreCondition

`public java.lang.String getPreCondition()`

Returns:

- the action creation pre-condition

---

isRedirectNext

`public boolean isRedirectNext()`

Returns:

- true if the action should redirect the user to the next form

Since:

- 4.3.0

---

getPropertyValue

`public java.lang.String getPropertyValue(java.lang.String propertyName)`

Parameters:

- `propertyName` - the name of the property

Returns:

- the value(String) for the corresponding property name

---

isPropertyTemplateEvaluation

`public java.lang.Boolean isPropertyTemplateEvaluation(java.lang.String propertyName)`

Return true if the property value should be evaluated with a Velocity template or false otherwise

Parameters:

- `propertyName` - the name of the property

Returns:

- true if the property value should be evaluated with a Velocity template or false otherwise

Since:

- 4.1.0

---

getPropertyMap

`public java.util.Map<java.lang.String,java.lang.String> getPropertyMap()`

Returns:

- a map of step property values

---

toString

`public java.lang.String toString()`

Overrides:

- `toString` in class `java.lang.Object`

Returns:

- the string representation of this object

---

writeJson

`public void writeJson(java.lang.StringBuilder builder)`

Description copied from interface: `IDefJsonSerializer#writeJson-java.lang.StringBuilder-`

Appends the JSON text representation of this object to the string builder.

Specified by:

- `IDefJsonSerializer#writeJson-java.lang.StringBuilder-` in interface `IDefJsonSerializer`

Parameters:

- `builder` - JSON string builder to write to

See Also:

- `IDefJsonSerializer.writeJson(StringBuilder)`

IDefJsonSerializer

Package:

- com.avoka.fc.core.service.job.config

Interface IDefJsonSerializer

All Known Implementing Classes:

- ActionDef, JobDef, StepDef

```
public interface IDefJsonSerializer
```

Provides an interface serializing Job, Step and Action Definition objects as a JSON text.

Since:

- 4.0.0

Method Summary

All Methods Instance Methods Abstract Methods

Modifier and Type	Method and Description
`void`	`writeJson-java.lang.StringBuilder-(java.lang.StringBuilder builder)` Appends the JSON text representation of this object to the string builder.

Method Detail

---

writeJson

```
void writeJson(java.lang.StringBuilder builder)
```

Appends the JSON text representation of this object to the string builder.

Parameters:

- `builder` - JSON string builder to write to

JobDef

Package:

- com.avoka.fc.core.service.job.config

Class JobDef

- java.lang.Object
- com.avoka.fc.core.service.job.config.JobDef

All Implemented Interfaces:

- IDefJsonSerializer

```
public class JobDef
extends java.lang.Object
implements
     IDefJsonSerializer
```

Provides an immutable Job definition configuration class. The Job configuration contains an object graph of StepConfig objects.

Job JSON Configuration

```
{
   "jobDetails" : {
      "name" : "Review and Approval Job"
   }
   "jobGroups" : [
     {
       "name" : "HL Melbourne",
       "description" : "Home Loan Customer Care Group for Vic"
     },
     {
       "name" : "HL Sydney",
       "description" : "Home Loan Customer Care Group for NSW"
     }
   ]
   "steps" : [
      { ... },
      { ... },
      { ... }
   ]
}
```

Since:

- 4.0.0

Field Summary

Fields

Modifier and Type	Field and Description
`static java.lang.String`	`DESCRIPTION` The JSON job details description attribute.
`static java.lang.String`	`NAME` The JSON job details name attribute.
`static java.lang.String`	`PRE_CONDITION_SUBMISSION_STEP` The pre-condition submission step name.

Constructor Summary

Constructors

Constructor and Description
`JobDef-java.lang.String-(java.lang.String jobJson)` Creates the Job configuration object from the given JSON map.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
java.lang.String	getDefaultNextStepName-java.lang.String-(java.lang.String currentStepName) Return the default next route next step for the given step name.
java.lang.String	getDescription--()
java.lang.String	getDisplayName--()
java.lang.String	getJobDetailsPropertyValue-java.lang.String-(java.lang.String name) Return the job details property value.
java.util.List<java.util.Map>	getJobGroups--() Get the Job Groups as a List<Map>.
java.lang.String	getName--() Return the name of this job configuration.
java.lang.String	getNextStepName-java.lang.String-java.lang.String-(java.lang.String currentStepName, java.lang.String routeName) Return the next route step name for the given step and and route name.
java.lang.String	getPreviousStepName-java.lang.String-(java.lang.String currentStepName) Return the name of the previous step form the current step name.
java.util.Map<java.lang.String,java.lang.String>	getPropertyMap--()
java.lang.String	getPropertyValue-java.lang.String-(java.lang.String propertyName)
java.lang.String	getReferenceNumberLabel--()
StepDef	getStartStepDef--() Return the start step definition.
StepDef	getStepDef-java.lang.String-(java.lang.String stepName) Return the step definition for the given step name.
java.util.List< StepDef>	getStepDefList--() Return the step definition list.
java.util.Map<java.lang.String, StepDef>	getStepDefMap--()
boolean	isProcessSubmitImmediate--() Return true if user form submit operations should be processed immediately in the user thread.
void	setName-java.lang.String-(java.lang.String value) Set the name of this job configuration.
void	setVersion-java.lang.String-(java.lang.String version) Set the version of this job configuration.
java.lang.String	toJSONString--()
java.lang.String	toString--() Return the string representation of this object.
void	writeJson-java.lang.StringBuilder-(java.lang.StringBuilder builder) Appends the JSON text representation of this object to the string builder.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

PRE_CONDITION_SUBMISSION_STEP

public static final java.lang.String PRE_CONDITION_SUBMISSION_STEP

The pre-condition submission step name.

See Also:

- Constants#com.avoka.fc.core.service.job.config.JobDef.PRE_CONDITION_SUBMISSION_STEP

NAME

`public static final java.lang.String NAME`

The JSON job details name attribute.

See Also:

- Constants#com.avoka.fc.core.service.job.config.JobDef.NAME

DESCRIPTION

`public static final java.lang.String DESCRIPTION`

The JSON job details description attribute.

See Also:

- Constants#com.avoka.fc.core.service.job.config.JobDef.DESCRIPTION

Constructor Detail

JobDef

`public JobDef(java.lang.String jobJson)`

Creates the Job configuration object from the given JSON map. A validation exception is returned if the Job configuration JSON (String): does not contain a jobDetails or a jobDetails.name does not contain a **steps** or the **steps** Array is empty. a step does not have a valid name. validation in the StepDefinition fails (StepDefinition.java)

Parameters:

- `jobJson` - this job definition JSON source (required)

Method Detail

getName

`public java.lang.String getName()`

Return the name of this job configuration.

Returns:

- the job name

---

setName

`public void setName(java.lang.String value)`

Set the name of this job configuration.

Parameters:

- `value` - the job name

---

getDisplayName

`public java.lang.String getDisplayName()`

Returns:

- the job details user display name.

---

getDescription

`public java.lang.String getDescription()`

Returns:

- the job details user description.

---

getReferenceNumberLabel

`public java.lang.String getReferenceNumberLabel()`

Returns:

- the job details reference number display label.

---

isProcessSubmitImmediate

```
public boolean isProcessSubmitImmediate()
```

Return true if user form submit operations should be processed immediately in the user thread.

Returns:

- true if user form submit operations should be processed immediately in the user thread

---

setVersion

```
public void setVersion(java.lang.String version)
```

Set the version of this job configuration.

Parameters:

- `version` - the job name

---

getJobDetailsPropertyValue

```
public java.lang.String getJobDetailsPropertyValue(java.lang.String name)
```

Return the job details property value.

Parameters:

- `name` - the job detail property name

Returns:

- the Job details property value

---

getStepDef

```
public
     StepDef getStepDef(java.lang.String stepName)
```

Return the step definition for the given step name.

Parameters:

- `stepName` - the step name

Returns:

- the step definition for the given step name.

---

getStepDefMap

```
public java.util.Map<java.lang.String,
     StepDef> getStepDefMap()
```

Returns:

- the map of step definitions keyed on step name.

---

getStepDefList

```
public java.util.List<
     StepDef> getStepDefList()
```

Return the step definition list.

Returns:

- the step definition list

---

getStartStepDef

```
public
     StepDef getStartStepDef()
```

Return the start step definition.

Returns:

- the start step definition.

---

getNextStepName

```
public java.lang.String getNextStepName(java.lang.String currentStepName,
                                        java.lang.String routeName)
```

Return the next route step name for the given step and and route name.

Parameters:

- `currentStepName` - the current step name
- `routeName` - the route name

Returns:

- the name of the next step

Throws:

- `java.lang.IllegalArgumentException` - if the current step was not found

---

getDefaultNextStepName

```
public java.lang.String getDefaultNextStepName(java.lang.String currentStepName)
```

Return the default next route next step for the given step name.

Parameters:

- `currentStepName` - the current steps name (required)

Returns:

- the default next route next step for the given step name

Throws:

- `java.lang.IllegalArgumentException` - if the current step was not found

---

getPreviousStepName

```
public java.lang.String getPreviousStepName(java.lang.String currentStepName)
```

Return the name of the previous step form the current step name.

Parameters:

- `currentStepName` - the current step name (required)

Returns:

- the name of the previous step form the current step name

Throws:

- `java.lang.IllegalArgumentException` - if the current step was not found or is a start step

---

getPropertyValue

```
public java.lang.String getPropertyValue(java.lang.String propertyName)
```

Parameters:

- `propertyName` - the name of the property

Returns:

- the value(String) for the corresponding property name

---

getPropertyMap

```
public java.util.Map<java.lang.String,java.lang.String> getPropertyMap()
```

Returns:

- a map of step property values

---

toString

`public java.lang.String toString()`

Return the string representation of this object.

Overrides:

- `toString` in class `java.lang.Object`

Returns:

- the string representation of this object

---

toJSONString

`public java.lang.String toJSONString()`

Returns:

- the JSON formatted string representation of this object.

---

writeJson

`public void writeJson(java.lang.StringBuilder builder)`

Description copied from interface: `IDefJsonSerializer#writeJson-java.lang.StringBuilder-`

Appends the JSON text representation of this object to the string builder.

Specified by:

- `IDefJsonSerializer#writeJson-java.lang.StringBuilder-` in interface `IDefJsonSerializer`

Parameters:

- `builder` - JSON string builder to write to

See Also:

- `IDefJsonSerializer.writeJson(StringBuilder)`

---

getJobGroups

`public java.util.List<java.util.Map> getJobGroups()`

Get the Job Groups as a List<Map>. The Map has the keys name, description

Returns:

- the job groups as a List<Map>

Since:

- 17.10.0

JsonSerializerUtils

Package:

- com.avoka.fc.core.service.job.config

Class JsonSerializerUtils

- java.lang.Object
- com.avoka.fc.core.service.job.config.JsonSerializerUtils

```
public class JsonSerializerUtils
extends java.lang.Object
```

Provides a utility class for rendering JobDef, StepDef and ActionDef to build JSON output.

Since:

- 4.0.0

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
`static java.util.List<java.util.Map<java.lang.String, java.lang.String>>`	`convertMap2JSONList-java.util.Map-java.lang.String-java.lang.String-(java.util.Map<java.lang.String, java.lang.String> map2Convert, java.lang.String keyName, java.lang.String valueName)` Converts a map that holds pair value such as the routeMap to a List<map<String,String>> which can be used to generate a JSON Array.
`static void`	`renderAttribute-java.lang.StringBuilder-java.lang.String-java.lang.Object-int-boolean-boolean-java.lang.String:A-(java.lang.StringBuilder builder, java.lang.String attName, java.lang.Object jsonObj, int padSpaces, boolean isFirst, boolean layoutValueSameLine, java.lang.String[] subFieldOrder)` Renders a string representation of a JSON Attribute name and value
`static void`	`renderJsonArray-java.lang.StringBuilder-java.util.List-int-boolean-java.lang.String:A-(java.lang.StringBuilder builder, java.util.List listJsonArray, int padSpaces, boolean itemSameLine, java.lang.String[] itemAttributeOrderArrray)` Adds a string representation of a JSON Array to a string builder.
`static void`	`renderJsonObject-java.lang.StringBuilder-java.util.Map-int-boolean-java.lang.String:A-(java.lang.StringBuilder builder, java.util.Map jsonObjectMap, int padSpaces, boolean layoutSameLine, java.lang.String[] attributeOrderArrray)` Adds a string representation of a JSON object to a string builder.

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Method Detail

---

renderJsonObject

```
public static void renderJsonObject(java.lang.StringBuilder builder,
                                    java.util.Map jsonObjectMap,
                                    int padSpaces,
                                    boolean layoutSameLine,
                                    java.lang.String[] attributeOrderArrray)
```

Adds a string representation of a JSON object to a string builder. This only supports JSON object which have attribute values that are strings.

Parameters:

- `builder` - the String Builder that has the string value appended to.
- `jsonObjectMap` - a Map that holds the JSON object.
- `padSpaces` - an integer representing the left padding associated with used if not on the same line
- `layoutSameLine` - boolean if true the JSON string will appear on the same line.
- `attributeOrderArrray` - String[] contains a array of Attribute names in the order to appear in the JSON

---

renderJsonArray

```
public static void renderJsonArray(java.lang.StringBuilder builder,
                                   java.util.List listJsonArray,
                                   int padSpaces,
                                   boolean itemSameLine,
                                   java.lang.String[] itemAttributeOrderArrray)
```

Adds a string representation of a JSON Array to a string builder. This assumes Each array item contains an JSON Object.

Parameters:

- `builder` - the String Builder that has the string value appended to.
- `listJsonArray` - List containing the data to be converted to a JSON Array.
- `padSpaces` - an integer representing the left padding associated with used if not on the same line.
- `itemSameLine` - boolean if set to true will render each item on the same line
- `itemAttributeOrderArrray` - String[] contains a list of String Attribute names in the order to appear in the JSON Object Items

---

renderAttribute

```
public static void renderAttribute(java.lang.StringBuilder builder,
                                   java.lang.String attName,
                                   java.lang.Object jsonObj,
                                   int padSpaces,
                                   boolean isFirst,
                                   boolean layoutValueSameLine,
                                   java.lang.String[] subFieldOrder)
```

Renders a string representation of a JSON Attribute name and value

Parameters:

- `builder` - the String Builder that has the string value appended to.
- `attName` - this the JSON attribute name
- `jsonObj` - this is the attribute value
- `padSpaces` - this is the attribute spacing
- `isFirst` - boolean if set treats this a the first element. It constructs the first "{" or "["
- `layoutValueSameLine` - boolean this is passed to either the renderJsonArray or renderJsonObject.
- `subFieldOrder` - String[] JSON objects attributes will be sorted in this name order, if null then attributes will be ordered by name alphabetically.

---

convertMap2JSONList

```
public static java.util.List<java.util.Map<java.lang.String,java.lang.String>> convertMap2JSONList(java.util.
Map<java.lang.String,java.lang.String> map2Convert,
                                                                                                    java.lang.String
keyName,
                                                                                                    java.lang.String
valueName)
```

Converts a map that holds pair value such as the routeMap to a List<map<String,String>> which can be used to generate a JSON Array.

Parameters:

- `map2Convert` - Map<String,Stringg> that holds the pair value.
- `keyName` - the JSON attribute String name for the key
- `valueName` - The JSON attribute String name for the value

Returns:

- List<map<String,String>> the converted List that can be passed to renderJsonArray

StepDef

Package:

- com.avoka.fc.core.service.job.config

Class StepDef

- java.lang.Object
- com.avoka.fc.core.service.job.config.StepDef

All Implemented Interfaces:

- IDefJsonSerializer

```
public class StepDef
extends java.lang.Object
implements
     IDefJsonSerializer
```

Provides an immutable StepConfig configuration class.

Step JSON Configuration

```
{
   "name" : "Initial Review",
   "type" : "",
   "dynamicPreConditions": true,
   "shareFormData" : true,
   "shareExtractData": false,
   "allFormsEditable" : true,
   "showPreviousForms" : true,
   "redirectNext": true,
   "expiryRule" : "+5d",
   "expiryServiceName" : "Review Expiry Service",
   "expiryServiceVersion" : "12",
   "actions": [
      { "name" : "Assign Review",  "type" : "Job Task Assign" },
      { "name" : "Review Wait",  "type" : "Job Task Wait" }
   ],
   "routes": [
      { "name" : "Default",  "nextStep" : "Management Review" },
      { "name" : "Expiry",  "nextStep" : "Review Expiry" }
   ],
   "properties": [
       { "name": "Task Review Submission Step",  "value": "Start" },
       { "name": "Task Group",  "value": "Application Reviewers" }
   ]
}
```

Since:

- 4.0.0

Field Summary

Fields

Modifier and Type	Field and Description
`static java.lang.String`	`DEFAULT_ROUTE_NAME` The "Default" route name.
`static java.lang.String`	`EXPIRY_ROUTE_NAME` The "Expiry" route name.
`static java.lang.String`	`NEXT_STEP_PREVOUS_STEP` nextStep special field value: previous step
`static java.lang.String`	`TYPE_ENDPOINT` the step type endpoint attribute value
`static java.lang.String`	`TYPE_START` the step type start attribute value
`static java.lang.String[]`	`VALID_TYPES` Array Containing the valid step types

Constructor Summary

Constructors

Constructor and Description
StepDef-java.util.Map-java.util.List-java.lang.String-(java.util.Map<java.lang.String,java.lang.Object> stepMap, java.util.List<java.lang.String> allStepNames, java.lang.String jobJson) Create a StepConfig using the given JSON Job Step configuration map.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
java.util.List< ActionDef>	getActionDefList--()
java.lang.String	getDefaultRouteStep--()
java.util.List<java.lang.String>	getDisplayRoutes--() Return the List of the available route names to display on the form
java.lang.String	getExpiryRouteStep--()
java.lang.String	getExpiryRule--()
java.lang.String	getExpiryServiceName--()
java.lang.Integer	getExpiryServiceVersion--()
java.lang.String	getName--()
java.lang.String	getNextStep-java.lang.String-(java.lang.String routeName)
java.lang.String	getPreCondition--()
java.util.Map<java.lang.String,java.lang.String>	getPropertyMap--()
java.lang.String	getPropertyValue-java.lang.String-(java.lang.String propertyName)
java.util.Map<java.lang.String,java.lang.String>	getRouteNextStepMap--() Return the route result to step name map.
java.lang.String	getType--()
boolean	isAllFormsEditable--()
boolean	isDynamicPreConditions--()
boolean	isRedirectNext--()
boolean	isShareExtractData--()
boolean	isShareFormData--()
boolean	isShowPreviousForms--()
boolean	isTypeEndpoint--()
boolean	isTypeStart--()
java.lang.String	toString--() Return the string representation of this object.
void	writeJson-java.lang.StringBuilder-(java.lang.StringBuilder builder) Appends the JSON text representation of this object to the string builder.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

DEFAULT_ROUTE_NAME

public static final java.lang.String DEFAULT_ROUTE_NAME

The "Default" route name.

See Also:

- Constants#com.avoka.fc.core.service.job.config.StepDef.DEFAULT_ROUTE_NAME

EXPIRY_ROUTE_NAME

```
public static final java.lang.String EXPIRY_ROUTE_NAME
```

The "Expiry" route name.

See Also:

- Constants#com.avoka.fc.core.service.job.config.StepDef.EXPIRY_ROUTE_NAME

NEXT_STEP_PREVOUS_STEP

```
public static final java.lang.String NEXT_STEP_PREVOUS_STEP
```

nextStep special field value: previous step

See Also:

- Constants#com.avoka.fc.core.service.job.config.StepDef.NEXT_STEP_PREVOUS_STEP

TYPE_START

```
public static final java.lang.String TYPE_START
```

the step type start attribute value

See Also:

- Constants#com.avoka.fc.core.service.job.config.StepDef.TYPE_START

TYPE_ENDPOINT

```
public static final java.lang.String TYPE_ENDPOINT
```

the step type endpoint attribute value

See Also:

- Constants#com.avoka.fc.core.service.job.config.StepDef.TYPE_ENDPOINT

VALID_TYPES

```
public static final java.lang.String[] VALID_TYPES
```

Array Containing the valid step types

Constructor Detail

StepDef

```
public StepDef(java.util.Map<java.lang.String,java.lang.Object> stepMap,
               java.util.List<java.lang.String> allStepNames,
               java.lang.String jobJson)
```

Create a StepConfig using the given JSON Job Step configuration map. A validation exception is returned if the step for the Job definition JSON (String): step does not contain a name has an invalid step type If the type is not an endpoint then the JSON: must have an actions attribute that is an array and it must contain at least 1 action validation in the ActionDefinition fails (ActionDefinition.java) must have a routes attribute that is an Array, and it must contain at least 1 route each route must have an name and nextStep attributes which are not empty the nextStep in the route must be either ##PREVIOUS_STEP() (excluding step of type start), or a valid step name contained in the job. must have an customProperties attribute that is an Array. Note this can be an empty array custom properties must have a name, the value (String) is optional and can be empty

Parameters:

- `stepMap` - the step configuration map JSON (required)
- `allStepNames` - the list of job configuration step names used to validate this step configuration (required)
- `jobJson` - this job definition JSON source (required)

Method Detail

getName

```
public java.lang.String getName()
```

Returns:

- the step name

---

getType

```
public java.lang.String getType()
```

Returns:

- the step type

---

getExpiryRule

```
public java.lang.String getExpiryRule()
```

Returns:

- the expiry rule

---

getExpiryServiceName

```
public java.lang.String getExpiryServiceName()
```

Returns:

- the expiry service name

---

getExpiryServiceVersion

```
public java.lang.Integer getExpiryServiceVersion()
```

Returns:

- the expiry service version number

Since:

- 5.1.0

---

getPreCondition

```
public java.lang.String getPreCondition()
```

Returns:

- the action creation pre-condition

---

getActionDefList

```
public java.util.List<
     ActionDef> getActionDefList()
```

Returns:

- a list of Actions Definition

---

getNextStep

```
public java.lang.String getNextStep(java.lang.String routeName)
```

Parameters:

- `routeName` - the name of the route

Returns:

- the stepName (String) for the Next Step

---

getRouteNextStepMap

```
public java.util.Map<java.lang.String,java.lang.String> getRouteNextStepMap()
```

Return the route result to step name map.

Returns:

- the route result to step name map.

---

getDisplayRoutes

```
public java.util.List<java.lang.String> getDisplayRoutes()
```

Return the List of the available route names to display on the form

Returns:

- the List of the available route names to display on the form

---

getDefaultRouteStep

```
public java.lang.String getDefaultRouteStep()
```

Returns:

- the default route step name.

---

getExpiryRouteStep

```
public java.lang.String getExpiryRouteStep()
```

Returns:

- the expiry route step name.

---

getPropertyValue

```
public java.lang.String getPropertyValue(java.lang.String propertyName)
```

Parameters:

- `propertyName` - the name of the property

Returns:

- the value(String) for the corresponding property name

---

getPropertyMap

```
public java.util.Map<java.lang.String,java.lang.String> getPropertyMap()
```

Returns:

- a map of step property values

---

isDynamicPreConditions

```
public boolean isDynamicPreConditions()
```

Returns:

- true if the step configured to use dynamic preCondition evaluation for step actions

Since:

- 4.3.0

---

isShareExtractData

```
public boolean isShareExtractData()
```

Returns:

- true if the step submissions should share extract data

Since:

- 4.3.0

---

isShareFormData

```
public boolean isShareFormData()
```

Returns:

- true if the step submissions should share form data

Since:

- 4.1.8

---

isAllFormsEditable

```
public boolean isAllFormsEditable()
```

Returns:

- true if the step submissions should share form data

Since:

- 4.2.0

---

isShowPreviousForms

```
public boolean isShowPreviousForms()
```

Returns:

- true if the step forms show all the previous step forms

Since:

- 4.3.0

---

isRedirectNext

```
public boolean isRedirectNext()
```

Returns:

- true if the step should redirect the user to the next form

Since:

- 4.3.0

---

isTypeEndpoint

```
public boolean isTypeEndpoint()
```

Returns:

- true if an "endpoint" type step.

---

isTypeStart

```
public boolean isTypeStart()
```

Returns:

- true if an "start" type step.

---

toString

```
public java.lang.String toString()
```

Return the string representation of this object.

Overrides:

- `toString` in class `java.lang.Object`

Returns:

- the string representation of this object

---

writeJson

```
public void writeJson(java.lang.StringBuilder builder)
```

Description copied from interface: `IDefJsonSerializer#writeJson-java.lang.StringBuilder-`

Appends the JSON text representation of this object to the string builder.

Specified by:

- `IDefJsonSerializer#writeJson-java.lang.StringBuilder-` in interface `IDefJsonSerializer`

Parameters:

- `builder` - JSON string builder to write to

See Also:

- `IDefJsonSerializer.writeJson(StringBuilder)`

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

Classes in Package com.avoka.fc.core.service.job.impl

- AbstractJobActionService
- ActionStepProperties
- GroovyJobActionService
- JobActionUtils
- JobActionWaitService
- JobControllerService
- JobDeliveryService
- JobDeliveryWaitService
- JobFormStartService
- JobFunctions
- JobProcessMessageService
- JobReceiptWaitService
- JobTaskAssignActionBuilder
- JobTaskAssignService
- JobTaskWaitService

AbstractJobActionService

Package:

- com.avoka.fc.core.service.job.impl

Class AbstractJobActionService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- com.avoka.fc.core.service.job.impl.AbstractJobActionService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobActionService

Direct Known Subclasses:

- GroovyJobActionService, JobActionWaitService, JobDeliveryService, JobDeliveryWaitService, JobFormStartService, JobProcessMessageService, JobReceiptWaitService, JobTaskAssignService, JobTaskWaitService

---

```
public abstract class AbstractJobActionService
extends com.avoka.fc.core.service.CayenneService
implements
     IJobActionService, com.avoka.fc.core.service.IServiceDefinitionAware
```

Provides an abstract Job Action Service.

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
AbstractJobActionService--()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
java.lang.Integer	getMaxErrorRetryAttempts--() Return the maximum number of error retry attempts should be performed.
java.lang.Integer	getRetryDelayMins--() Return the minimum action execute retry delay in minutes.
com.avoka.fc.core.entity.ServiceDefinition	getServiceDefinition--() Return the service definition.
void	setMaxErrorRetryAttempts-java.lang.Integer-(java.lang.Integer attempts) Set the maximum number of action attempts.
void	setRetryDelayMins-java.lang.Integer-(java.lang.Integer delayMins) Set the minimum action execute retry delay in minutes.
void	setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-(com.avoka.fc.core.entity.ServiceDefinition serviceDefinition) Set the service definition.
java.lang.String	validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-(ActionStepProperties actionStepProperties, com.avoka.fc.core.entity.Client client) Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Methods inherited from class com.avoka.fc.core.service.CayenneService

commitChanges, rollbackChanges

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface com.avoka.fc.core.service.job. IJobActionService
`IJobActionService#execute-com.avoka.fc.core.service.job.ActionContext-`

Constructor Detail

AbstractJobActionService

```
public AbstractJobActionService()
```

Method Detail

validateProperties

```
public java.lang.String validateProperties(
     ActionStepProperties actionStepProperties,
                                          com.avoka.fc.core.entity.Client client)
```

Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Specified by:

- `IJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-` in interface `IJobActionService`

Parameters:

- `actionStepProperties` - the action step properties (required)
- `client` - the Job Controller client (optional)

Returns:

- null if the properties are valid or null otherwise

See Also:

- `IJobActionService.validateProperties(ActionStepProperties, Client)`

---

getMaxErrorRetryAttempts

```
public java.lang.Integer getMaxErrorRetryAttempts()
```

Description copied from interface: `IJobActionService#getMaxErrorRetryAttempts--`

Return the maximum number of error retry attempts should be performed.

Specified by:

- `IJobActionService#getMaxErrorRetryAttempts--` in interface `IJobActionService`

Returns:

- the maximum number of error retry attempts should be performed

See Also:

- `IJobActionService.getMaxErrorRetryAttempts()`

---

setMaxErrorRetryAttempts

```
public void setMaxErrorRetryAttempts(java.lang.Integer attempts)
```

Set the maximum number of action attempts.

Parameters:

- `attempts` - the maximum number of error retry attempts should be performed

---

getRetryDelayMins

```
public java.lang.Integer getRetryDelayMins()
```

Description copied from interface: `IJobActionService#getRetryDelayMins--`

Return the minimum action execute retry delay in minutes.

Specified by:

- `IJobActionService#getRetryDelayMins--` in interface `IJobActionService`

Returns:

- the minimum action execute retry delay in minutes.

See Also:

- `IJobActionService.getRetryDelayMins()`

---

setRetryDelayMins

`public void setRetryDelayMins(java.lang.Integer delayMins)`

Set the minimum action execute retry delay in minutes.

Parameters:

- `delayMins` - the retry delay in minutes

---

getServiceDefinition

`public com.avoka.fc.core.entity.ServiceDefinition getServiceDefinition()`

Return the service definition.

Specified by:

- `getServiceDefinition` in interface `com.avoka.fc.core.service.IServiceDefinitionAware`

Returns:

- the service definition

See Also:

- `IServiceDefinitionAware.getServiceDefinition()`

---

setServiceDefinition

`public void setServiceDefinition(com.avoka.fc.core.entity.ServiceDefinition serviceDefinition)`

Set the service definition.

Specified by:

- `setServiceDefinition` in interface `com.avoka.fc.core.service.IServiceDefinitionAware`

Parameters:

- `serviceDefinition` - the service definition

See Also:

- `IServiceDefinitionAware.setServiceDefinition(ServiceDefinition)`

ActionStepProperties

Package:

- com.avoka.fc.core.service.job.impl

Class ActionStepProperties

- java.lang.Object
- com.avoka.fc.core.service.job.impl.ActionStepProperties

```
public class ActionStepProperties
extends java.lang.Object
```

Provides a class to resolve action step properties.

Resolution Order

Action step properties are resolved in the following order:

lookup in the action properties of the Job Definition lookup in the action service definition service parameters lookup in the step properties of the current step in the lookup in the job controller service definition parameters lookup in the job properties of the Job Definition

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
`ActionStepProperties-com.avoka.fc.core.service.job.ActionContext-( ActionContext actionContext)` Create a ActionStepProperties object with the given action context.
`ActionStepProperties-com.avoka.fc.core.service.job.ActionContext-com.avoka.fc.core.service.job.IJobActionService-( ActionContext actionContext, IJobActionService jobActionService)` Create a ActionStepProperties object with the given action context and action service.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`java.lang. Boolean`	`getBooleanProperty-java.lang.String-(java.lang.String name)` Return the date property value for the given name.
`java.lang. Boolean`	`getBooleanProperty-java.lang.String-java.util.Map-(java.lang.String name, java.util.Map<java.lang.String, java.lang.String> formDataMap)` Return the date property value for the given name.
`java.sql.Date`	`getDateProperty-java.lang.String-(java.lang.String name)` Return the date property value for the given name, date format (yyyy-[m]m-[d]d).
`java.sql.Date`	`getDateProperty-java.lang.String-java.util.Map-(java.lang.String name, java.util.Map<java.lang.String,java. lang.String> formDataMap)` Return the date property value for the given name, date format (yyyy-[m]m-[d]d).
`java.lang.Double`	`getDoubleProperty-java.lang.String-(java.lang.String name)` Return the double property value for the given name.
`java.lang.Double`	`getDoubleProperty-java.lang.String-java.util.Map-(java.lang.String name, java.util.Map<java.lang.String,java. lang.String> formDataMap)` Return the double property value for the given name.
`java.lang. Integer`	`getIntegerProperty-java.lang.String-(java.lang.String name)` Return the integer property value for the given name.
`java.lang. Integer`	`getIntegerProperty-java.lang.String-java.util.Map-(java.lang.String name, java.util.Map<java.lang.String, java.lang.String> formDataMap)` Return the integer property value for the given name.
`JobDef`	`getJobDef--()`

	Return the Job Definition associated with these action step properties.
java.util.Set<java.lang.String>	getKeySet--() Return the set resolved of available property keys (names).
java.lang.String	getProperty-java.lang.String-(java.lang.String name) Return the property value for the given name.
java.lang.String	getProperty-java.lang.String-java.util.Map-(java.lang.String name, java.util.Map<java.lang.String,java.lang.String> formDataMap) Return the property value for the given name.
java.lang.Object	getPropertyObject-java.lang.String-java.util.Map-(java.lang.String name, java.util.Map<java.lang.String,java.lang.String> formDataMap) Return the property value for the given name.
java.lang.Object	getPropertyObject-java.lang.String-java.util.Map-java.util.Map-(java.lang.String name, java.util.Map<java.lang.String,java.lang.String> formDataMap, java.util.Map<java.lang.String,java.lang.Object> model) Return the property value for the given name.
java.lang.String	getRawPropertyValue-java.lang.String-(java.lang.String name) Return the raw property value before property function or StringTemplate conversion is run.
boolean	isValidRoute-java.lang.String-(java.lang.String routeName) return true if the routeName is valid
void	setUsePropertyFunctions-boolean-(boolean usePropertyFunctions) Specify whether to use property functions, can be switched off to enable property validation.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ActionStepProperties

```
public ActionStepProperties(
     ActionContext actionContext,

     IJobActionService jobActionService)
```

Create a ActionStepProperties object with the given action context and action service.

Parameters:

- actionContext - the job action context (required)
- jobActionService - the job action service (required)

ActionStepProperties

```
public ActionStepProperties(
     ActionContext actionContext)
```

Create a ActionStepProperties object with the given action context. Note by using this constructor ActionService Properties cannot be resolved for the Job Action.

Parameters:

- actionContext - the job action context (required)

Since:

- 4.3.0

Method Detail

getBooleanProperty

```
public java.lang.Boolean getBooleanProperty(java.lang.String name)
```

Return the date property value for the given name.

Parameters:

- name - the name of the action step custom property or the service definition parameter name

Returns:

- the resolved boolean property value

---

getBooleanProperty

```
public java.lang.Boolean getBooleanProperty(java.lang.String name,
                                            java.util.Map<java.lang.String,java.lang.String> formDataMap)
```

Return the date property value for the given name.

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name
- `formDataMap` - the form data map (optional)

Returns:

- the resolved boolean property value

---

getDateProperty

```
public java.sql.Date getDateProperty(java.lang.String name)
```

Return the date property value for the given name, date format (yyyy-[m]m-[d]d).

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name

Returns:

- the resolved date property value

---

getDateProperty

```
public java.sql.Date getDateProperty(java.lang.String name,
                                     java.util.Map<java.lang.String,java.lang.String> formDataMap)
```

Return the date property value for the given name, date format (yyyy-[m]m-[d]d).

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name
- `formDataMap` - the form data map (optional)

Returns:

- the resolved date property value

---

getDoubleProperty

```
public java.lang.Double getDoubleProperty(java.lang.String name)
```

Return the double property value for the given name.

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name

Returns:

- the resolved double property value

---

getDoubleProperty

```
public java.lang.Double getDoubleProperty(java.lang.String name,
                                          java.util.Map<java.lang.String,java.lang.String> formDataMap)
```

Return the double property value for the given name.

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name
- `formDataMap` - the form data map (optional)

Returns:

- the resolved double property value

---

getIntegerProperty

```
public java.lang.Integer getIntegerProperty(java.lang.String name)
```

Return the integer property value for the given name.

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name

Returns:

- the resolved integer property value

---

getIntegerProperty

```
public java.lang.Integer getIntegerProperty(java.lang.String name,
                                            java.util.Map<java.lang.String,java.lang.String> formDataMap)
```

Return the integer property value for the given name.

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name
- `formDataMap` - the form data map (optional)

Returns:

- the resolved integer property value

---

getProperty

```
public java.lang.String getProperty(java.lang.String name)
```

Return the property value for the given name.

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name

Returns:

- the resolved property value

---

getProperty

```
public java.lang.String getProperty(java.lang.String name,
                                    java.util.Map<java.lang.String,java.lang.String> formDataMap)
```

Return the property value for the given name.

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name
- `formDataMap` - the form data map (optional)

Returns:

- the resolved property value

---

getPropertyObject

```
public java.lang.Object getPropertyObject(java.lang.String name,
                                          java.util.Map<java.lang.String,java.lang.String> formDataMap)
```

Return the property value for the given name.

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name
- `formDataMap` - the form data map (optional)

Returns:

- the resolved property value

---

getPropertyObject

```
public java.lang.Object getPropertyObject(java.lang.String name,
                                          java.util.Map<java.lang.String,java.lang.String> formDataMap,
                                          java.util.Map<java.lang.String,java.lang.Object> model)
```

Return the property value for the given name.

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name
- `formDataMap` - the form data map (optional)
- `model` - template model Map (required)

Returns:

- the resolved property value

---

getRawPropertyValue

```
public java.lang.String getRawPropertyValue(java.lang.String name)
```

Return the raw property value before property function or StringTemplate conversion is run.

Parameters:

- `name` - the name of the action step custom property or the service definition parameter name

Returns:

- the raw property value String before property function or StringTemplate conversion

---

getKeySet

```
public java.util.Set<java.lang.String> getKeySet()
```

Return the set resolved of available property keys (names).

Returns:

- the set resolved of available property keys (names)

---

getJobDef

```
public
     JobDef getJobDef()
```

Return the Job Definition associated with these action step properties.

Returns:

- the Job Definition associated with these action step properties

---

setUsePropertyFunctions

```
public void setUsePropertyFunctions(boolean usePropertyFunctions)
```

Specify whether to use property functions, can be switched off to enable property validation.

Parameters:

- `usePropertyFunctions` - specify whether to execute property functions

---

isValidRoute

```
public boolean isValidRoute(java.lang.String routeName)
```

return true if the routeName is valid

Parameters:

- `routeName` - the selected route name String from the form submission

Returns:

- true if the routeName is contained in the stepDef

GroovyJobActionService

Package:

- com.avoka.fc.core.service.job.impl

Class GroovyJobActionService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- AbstractJobActionService
- com.avoka.fc.core.service.job.impl.GroovyJobActionService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobActionService

---

```
public class GroovyJobActionService
extends
     AbstractJobActionService
```

Provides a Groovy Job Action Service.

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
GroovyJobActionService--()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
ActionResult	execute-com.avoka.fc.core.service.job.ActionContext-(ActionContext actionContext) Execute the action and return the result.
java.lang.Integer	getExecutionTimeout--() Return the GroovyScript execution timeout in milliseconds.
java.lang.String	getGroovyScript--()
java.lang.Boolean	isGroovyDebugLogging--() Return true if GroovyLogger DEBUG level recording is enabled.
boolean	isGroovyLoggingEnabled--() Return true if the Groovy Service logging is enabled.
boolean	isGroovyTypeChecked--() Return true if the Groovy runtime should perform static type checking.
void	setExecutionTimeout-java.lang.Integer-(java.lang.Integer timeout) Return the GroovyScript execution timeout in milliseconds.
void	setGroovyDebugLogging-java.lang.Boolean-(java.lang.Boolean debugEnabled) Set whether GroovyLogger DEBUG level recording is enabled.
void	setGroovyLoggingEnabled-java.lang.Boolean-(java.lang.Boolean loggingEnabled) Specify whether the Groovy Service logging is enabled.
void	setGroovyScript-java.lang.String-(java.lang.String script)
void	setGroovyTypeChecked-boolean-(boolean typeChecked) Specify whether the Groovy runtime should perform static type checking.

Methods inherited from class com.avoka.fc.core.service.job.impl. AbstractJobActionService

AbstractJobActionService#getMaxErrorRetryAttempts--, AbstractJobActionService#getRetryDelayMins--, AbstractJobActionService#getServiceDefinition--, AbstractJobActionService#setMaxErrorRetryAttempts-java.lang.Integer-, AbstractJobActionService#setRetryDelayMins-java.lang.Integer-, AbstractJobActionService#setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-, AbstractJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-

Methods inherited from class com.avoka.fc.core.service.CayenneService

commitChanges, rollbackChanges

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

GroovyJobActionService

```
public GroovyJobActionService()
```

Method Detail

execute

```
public
     ActionResult execute(
     ActionContext actionContext)
```

Description copied from interface: IJobActionService#execute-com.avoka.fc.core.service.job.ActionContext-

Execute the action and return the result.

Parameters:

- actionContext - the job action execution context (required)

Returns:

- the action result

See Also:

- IJobActionService.execute(ActionContext)

---

getExecutionTimeout

```
public java.lang.Integer getExecutionTimeout()
```

Return the GroovyScript execution timeout in milliseconds.

Returns:

- the GroovyScript execution timeout in milliseconds

---

setExecutionTimeout

```
public void setExecutionTimeout(java.lang.Integer timeout)
```

Return the GroovyScript execution timeout in milliseconds.

Parameters:

- timeout - the GroovyScript execution timeout in milliseconds

---

isGroovyDebugLogging

```
public java.lang.Boolean isGroovyDebugLogging()
```

Return true if GroovyLogger DEBUG level recording is enabled.

Returns:

- the groovyDebugLoggingEnabled

Since:

- 4.3.4

setGroovyDebugLogging

`public void setGroovyDebugLogging(java.lang.Boolean debugEnabled)`

Set whether GroovyLogger DEBUG level recording is enabled.

Parameters:

- `debugEnabled` - the groovyDebugLoggingEnabled to set

Since:

- 4.3.4

---

isGroovyLoggingEnabled

`public boolean isGroovyLoggingEnabled()`

Return true if the Groovy Service logging is enabled.

Returns:

- true if the Groovy Service logging is enabled.

Since:

- 4.2.0

---

setGroovyLoggingEnabled

`public void setGroovyLoggingEnabled(java.lang.Boolean loggingEnabled)`

Specify whether the Groovy Service logging is enabled.

Parameters:

- `loggingEnabled` - specify whether the Groovy Service logging is enabled

Since:

- 4.2.0

---

getGroovyScript

`public java.lang.String getGroovyScript()`

Returns:

- the Groovy Script content to execute

---

setGroovyScript

`public void setGroovyScript(java.lang.String script)`

Parameters:

- `script` - the Groovy Script content to execute

---

isGroovyTypeChecked

`public boolean isGroovyTypeChecked()`

Return true if the Groovy runtime should perform static type checking.

Returns:

- true if the Groovy runtime should perform static type checking

Since:

- 4.0.0

---

setGroovyTypeChecked

`public void setGroovyTypeChecked(boolean typeChecked)`

Specify whether the Groovy runtime should perform static type checking.

Parameters:

- `typeChecked` - specify whether the Groovy runtime should perform static type checking

Since:

- 4.0.0

JobActionUtils

Package:

- com.avoka.fc.core.service.job.impl

Class JobActionUtils

- java.lang.Object
- com.avoka.fc.core.service.job.impl.JobActionUtils

```
public class JobActionUtils
extends java.lang.Object
```

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

Since:

- 4.0.0

Field Summary
Fields

Modifier and Type	Field and Description
`static java.lang.String`	`PROPERTY_PROCESS_MESSAGE_EMAIL_MESSAGE_TEMPLATE` The processing message email message template name.
`static java.lang.String`	`PROPERTY_PROCESS_MESSAGE_EMAIL_SUBJECT_TEMPLATE` The processing message email message template name.
`static java.lang.String`	`PROPERTY_PROCESS_MESSAGE_EMAIL_TO` The processing message email to.
`static java.lang.String`	`PROPERTY_PROCESS_MESSAGE_SEND_EMAIL` The processing message send email.
`static java.lang.String`	`PROPERTY_PROCESS_MESSAGE_SUBMISSION` The processing message submission.
`static java.lang.String`	`PROPERTY_PROCESS_MESSAGE_SUBMISSION_STEP` The processing message submission step.
`static java.lang.String`	`PROPERTY_PROCESS_MESSAGE_TEXT` The processing message text.

Constructor Summary
Constructors

Constructor and Description
`JobActionUtils--()`

Method Summary
All Methods Static Methods Concrete Methods Deprecated Methods

Modifier and Type	Method and Description
`static void`	`addSubmissionProcessingStatus-com.avoka.fc.core.service.job.ActionContext-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Submission-( ActionContext actionContext, ActionStepProperties actionStepProperties, com.avoka.fc.core.entity.Submission submission)` Add processing status message to the submission.
`static java.util.Map<java.lang.String,java.lang.Object>`	`convertNumericPropertyValues-java.util.Map-(java.util.Map<java.lang.String,java.lang.String> inputMap)` Return a Map with the numeric string property values converted to Integer, Long or Double.
`static ActionResult`	`expireStepTasksAndActions-com.avoka.fc.core.entity.JobAction-java.lang.String-(com.avoka.fc.core.entity.JobAction expiryJobAction, java.lang.String expiryRouteName)` Called from an Job Expiry service this will expire the tasks and actions for a job.
`static java.util.`	`getCompletedSubmissionsForStep-com.avoka.fc.core.entity.JobStep-(com.avoka.fc.core.entity.JobStep step)`

List<com.avoka.fc.core.entity.Submission>	Returns a List of completed submissions for the step, in id order, for the given job step or and empty List
static com.avoka.fc.core.entity.JobStep	getFirstStep-com.avoka.fc.core.entity.Job-java.lang.String-(com.avoka.fc.core.entity.Job job, java.lang.String stepName) Gets the earliest step instance for the specified step name working forwards from earliest to latest steps.
static java.util.Map<java.lang.String,java.lang.String>	getFormDataMap-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Returns the formDataMap for the current submission
static java.util.Map<java.lang.String,java.lang.String>	getFormDataMapPreviousStep-com.avoka.fc.core.entity.JobAction-(com.avoka.fc.core.entity.JobAction currentJobAction) Returns the formDataMap for the previous JobStep submission
static java.util.Set<java.lang.String>	getFormDataMapRefs-java.lang.String-(java.lang.String jsonString) Return the set of formDataMap references in the JSON string.
static com.avoka.fc.core.entity.Submission	getFormStartSubmission-com.avoka.fc.core.entity.JobAction-(com.avoka.fc.core.entity.JobAction currentJobAction) Gets the original submission that started the Job's form start action.
static com.avoka.fc.core.entity.UserAccount	getFormStartUser-com.avoka.fc.core.entity.JobAction-(com.avoka.fc.core.entity.JobAction currentJobAction) Returns the user that made the original submission.
static com.avoka.fc.core.entity.JobStep	getLastStep-com.avoka.fc.core.entity.Job-java.lang.String-(com.avoka.fc.core.entity.Job job, java.lang.String stepName) Gets the latest step instance for the specified step name working backwards from latest to earliest steps.
static com.avoka.fc.core.entity.Submission	getLastSubmission-com.avoka.fc.core.service.job.ActionContext-java.lang.String-(ActionContext actionContext, java.lang.String stepName) Gets the submission for the specified step name working backwards from latest to earliest steps.
static com.avoka.fc.core.entity.JobStep	getPreviousStep-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job) Return the previous step relative to the job's current step, or null if current step is first step.
static com.avoka.fc.core.entity.Submission	getPreviousSubmission-com.avoka.fc.core.entity.JobAction-(com.avoka.fc.core.entity.JobAction currentJobAction) Gets the submission from the previous step action.
static com.avoka.fc.core.entity.Submission	getRedirectNextTask-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Get the next task to redirect the user to for an Application Bundle workflow.
static java.lang.String	getRouteName-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Returns the route name from the given submission.
static com.avoka.fc.core.entity.Submission	getStepSubmission-com.avoka.fc.core.entity.Job-java.lang.String-(com.avoka.fc.core.entity.Job job, java.lang.String stepName) Return the first submission for the given step name or null if not found.
static java.util.List<com.avoka.fc.core.entity.Submission>	getSubmissionsForLastStep-com.avoka.fc.core.entity.Job-java.lang.String-(com.avoka.fc.core.entity.Job job, java.lang.String stepName) Returns a List of submissions, in action order, for the latest job step and given step name or and empty List
static java.util.List<com.avoka.fc.core.entity.Submission>	getSubmissionsForStep-com.avoka.fc.core.entity.JobStep-(com.avoka.fc.core.entity.JobStep step) Returns a List of submissions, in action order, for the given job step or and empty List
static java.util.List<com.avoka.fc.core.entity.Submission>	getSubmissionsForStep-com.avoka.fc.core.entity.Job-java.lang.String-(com.avoka.fc.core.entity.Job job, java.lang.String stepName) Returns a List of submissions, in id order, for the job and given step name or and empty List
static java.lang.Integer	getThreadAssignRepeatIndex--() Return the thread local Task Assign Repeat Index value [1..n].
static java.lang.String	getThreadAssignRepeatItem--() Return the thread local Task Assign Repeat Item.
static boolean	isDynamicProperty-java.lang.String-(java.lang.String propertyValue) Return true if property value is dynamically resolved or false if a static literal property value.
static void	setThreadAssignRepeatIndex-java.lang.Integer-(java.lang.Integer index) Set the thread local Task Assign Repeat Index value [1..n].
static void	setThreadAssignRepeatItem-java.lang.String-(java.lang.String item) Set the thread local Task Assign Repeat Item.
static java.lang.	toJobHistoryJsonString-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job)

String	Return the job JSON debug representation of this object for debugging purposes.
static java.lang.String	validatePropertyIsBoolean-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String propertyName) Validate the given actionStepProperties ensuring the given propertyName value is a boolean value if defined
static java.lang.String	validatePropertyIsDate-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String propertyName) Validate the given actionStepProperties ensuring the given propertyName value is a date value if defined
static java.lang.String	validatePropertyIsDouble-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String propertyName) Deprecated.
static java.lang.String	validatePropertyIsEmail-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String propertyName) Validate the given actionStepProperties ensuring the given propertyName value is a valid email if defined
static java.lang.String	validatePropertyIsForm-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-com.avoka.fc.core.entity.Client-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String formCodePropertyName, com.avoka.fc.core.entity.Client client, java.lang.String versionPropertyName) Validate the given actionStepProperties ensuring the given propertyName value is a valid form code if defined
static java.lang.String	validatePropertyIsFormCode-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-com.avoka.fc.core.entity.Client-(ActionStepProperties actionStepProperties, java.lang.String propertyName, com.avoka.fc.core.entity.Client client) Deprecated. use validatePropertyIsForm.
static java.lang.String	validatePropertyIsFormGroupNames-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String propertyName) Validate the given actionStepProperties ensuring the given propertyName value is a set of valid form group names
static java.lang.String	validatePropertyIsGroup-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String propertyName) Validate the given actionStepProperties ensuring the given propertyName value is a Group if defined
static java.lang.String	validatePropertyIsInteger-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String propertyName) Validate the given actionStepProperties ensuring the given propertyName value is a Integer value if defined
static java.lang.String	validatePropertyIsPortal-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-com.avoka.fc.core.entity.Client-(ActionStepProperties actionStepProperties, java.lang.String propertyName, com.avoka.fc.core.entity.Client client) Validate the given actionStepProperties ensuring the given propertyName value is a Portal if defined
static java.lang.String	validatePropertyIsStep-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String propertyName) Validate the given actionStepProperties ensuring the given propertyName value is a valid Step name if defined
static java.lang.String	validatePropertyIsUser-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String propertyName) Validate the given actionStepProperties ensuring the given propertyName value is a User if defined
static java.lang.String	validatePropertyNotBlank-com.avoka.fc.core.service.job.impl.ActionStepProperties-java.lang.String-(ActionStepProperties actionStepProperties, java.lang.String propertyName) Validate the given actionStepProperties ensuring the given propertyName value is not blank.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

PROPERTY_PROCESS_MESSAGE_TEXT

public static final java.lang.String PROPERTY_PROCESS_MESSAGE_TEXT

The processing message text.

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobActionUtils.PROPERTY_PROCESS_MESSAGE_TEXT

PROPERTY_PROCESS_MESSAGE_SUBMISSION

public static final java.lang.String PROPERTY_PROCESS_MESSAGE_SUBMISSION

The processing message submission.

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobActionUtils.PROPERTY_PROCESS_MESSAGE_SUBMISSION

PROPERTY_PROCESS_MESSAGE_SUBMISSION_STEP

```
public static final java.lang.String PROPERTY_PROCESS_MESSAGE_SUBMISSION_STEP
```

The processing message submission step.

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobActionUtils.PROPERTY_PROCESS_MESSAGE_SUBMISSION_STEP

PROPERTY_PROCESS_MESSAGE_SEND_EMAIL

```
public static final java.lang.String PROPERTY_PROCESS_MESSAGE_SEND_EMAIL
```

The processing message send email.

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobActionUtils.PROPERTY_PROCESS_MESSAGE_SEND_EMAIL

PROPERTY_PROCESS_MESSAGE_EMAIL_TO

```
public static final java.lang.String PROPERTY_PROCESS_MESSAGE_EMAIL_TO
```

The processing message email to.

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobActionUtils.PROPERTY_PROCESS_MESSAGE_EMAIL_TO

PROPERTY_PROCESS_MESSAGE_EMAIL_SUBJECT_TEMPLATE

```
public static final java.lang.String PROPERTY_PROCESS_MESSAGE_EMAIL_SUBJECT_TEMPLATE
```

The processing message email message template name.

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobActionUtils.PROPERTY_PROCESS_MESSAGE_EMAIL_SUBJECT_TEMPLATE

PROPERTY_PROCESS_MESSAGE_EMAIL_MESSAGE_TEMPLATE

```
public static final java.lang.String PROPERTY_PROCESS_MESSAGE_EMAIL_MESSAGE_TEMPLATE
```

The processing message email message template name.

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobActionUtils.PROPERTY_PROCESS_MESSAGE_EMAIL_MESSAGE_TEMPLATE

Constructor Detail

JobActionUtils

```
public JobActionUtils()
```

Method Detail

getThreadAssignRepeatIndex

```
public static java.lang.Integer getThreadAssignRepeatIndex()
```

Return the thread local Task Assign Repeat Index value [1..n].

Returns:

- the thread local Task Assign Repeat Index value [1..n]

Since:

- 4.1.8

---

setThreadAssignRepeatIndex

```
public static void setThreadAssignRepeatIndex(java.lang.Integer index)
```

Set the thread local Task Assign Repeat Index value [1..n].

Parameters:

- `index` - set the thread local Task Assign Repeat Index value [1..n]

Since:

- 4.1.8

---

getThreadAssignRepeatItem

```
public static java.lang.String getThreadAssignRepeatItem()
```

Return the thread local Task Assign Repeat Item.

Returns:

- the thread local Task Assign Repeat Item

Since:

- 4.1.8

---

setThreadAssignRepeatItem

```
public static void setThreadAssignRepeatItem(java.lang.String item)
```

Set the thread local Task Assign Repeat Item.

Parameters:

- `item` - the thread local Task Assign Repeat Item

Since:

- 4.1.8

---

getFormStartSubmission

```
public static com.avoka.fc.core.entity.Submission getFormStartSubmission(com.avoka.fc.core.entity.JobAction
currentJobAction)
```

Gets the original submission that started the Job's form start action.

Parameters:

- `currentJobAction` - the JobAction associated with the Job Action Service.

Returns:

- if found returns the original submission associated with the FormStart Action

---

getFormDataMapRefs

```
public static java.util.Set<java.lang.String> getFormDataMapRefs(java.lang.String jsonString)
```

Return the set of formDataMap references in the JSON string.

Parameters:

- `jsonString` - the JSON string to parse (required)

Returns:

- the set of formDataMap references in the JSON string

Since:

- 4.2.0

---

getPreviousSubmission

```
public static com.avoka.fc.core.entity.Submission getPreviousSubmission(com.avoka.fc.core.entity.JobAction
currentJobAction)
```

Gets the submission from the previous step action.

Parameters:

- `currentJobAction` - the JobAction associated with the Job Action Service.

Returns:

- if found returns the previous step submission

---

getLastSubmission

```
public static com.avoka.fc.core.entity.Submission getLastSubmission(
     ActionContext actionContext,
                                                                    java.lang.String stepName)
```

Gets the submission for the specified step name working backwards from latest to earliest steps.

Parameters:

- `actionContext` - the ActionContext associated with the Job Action Service.
- `stepName` - the step name that you want to retrieve the submission from.

Returns:

- if found returns the submission for the specified step name working backwards from latest to earliest steps

---

getRedirectNextTask

```
public static com.avoka.fc.core.entity.Submission getRedirectNextTask(com.avoka.fc.core.entity.Submission
submission)
```

Get the next task to redirect the user to for an Application Bundle workflow.

Parameters:

- `submission` - the current submission being processed (required)

Returns:

- the next task to redirect the user to, or null if not defined

Since:

- 4.3.0

---

getLastStep

```
public static com.avoka.fc.core.entity.JobStep getLastStep(com.avoka.fc.core.entity.Job job,
                                                           java.lang.String stepName)
```

Gets the latest step instance for the specified step name working backwards from latest to earliest steps.

Parameters:

- `job` - the job containing the step
- `stepName` - the step name that you want to retrieve the step for.

Returns:

- if found returns the latest step instance with that stepName

Since:

- 4.2.0

---

getFirstStep

```
public static com.avoka.fc.core.entity.JobStep getFirstStep(com.avoka.fc.core.entity.Job job,
                                                            java.lang.String stepName)
```

Gets the earliest step instance for the specified step name working forwards from earliest to latest steps.

Parameters:

- `job` - the job containing the step instance
- `stepName` - the step name that you want to retrieve the step instance

Returns:

- if found returns the first step instance with that stepName

Since:

- 4.2.0

---

getPreviousStep

```
public static com.avoka.fc.core.entity.JobStep getPreviousStep(com.avoka.fc.core.entity.Job job)
```

Return the previous step relative to the job's current step, or null if current step is first step.

Parameters:

- `job` - the job to get the previous step from

Returns:

- the previous step, or null if the current step is the first step in a job

Since:

- 4.3.0

---

getStepSubmission

```
public static com.avoka.fc.core.entity.Submission getStepSubmission(com.avoka.fc.core.entity.Job job,
                                                                    java.lang.String stepName)
```

Return the first submission for the given step name or null if not found.

Parameters:

- `job` - the job (required)
- `stepName` - the job step name (required)

Returns:

- the first submission for the given step name or null if not found

---

getSubmissionsForStep

```
public static java.util.List<com.avoka.fc.core.entity.Submission> getSubmissionsForStep(com.avoka.fc.core.entity.
Job job,
                                                                                        java.lang.String stepName)
```

Returns a List of submissions, in id order, for the job and given step name or and empty List

Parameters:

- `job` - the job (required)
- `stepName` - the job step name (required)

Returns:

- a List of submissions in id order for the given step name otherwise and empty List

Since:

- 4.1.0

---

getSubmissionsForLastStep

```
public static java.util.List<com.avoka.fc.core.entity.Submission> getSubmissionsForLastStep(com.avoka.fc.core.
entity.Job job,
                                                                                            java.lang.String
stepName)
```

Returns a List of submissions, in action order, for the latest job step and given step name or and empty List

Parameters:

- `job` - the job (required)
- `stepName` - the job step name (required)

Returns:

- a List of submissions in id order for the given step name otherwise and empty List

Since:

- 4.1.0

---

getSubmissionsForStep

```
public static java.util.List<com.avoka.fc.core.entity.Submission> getSubmissionsForStep(com.avoka.fc.core.entity.
JobStep step)
```

Returns a List of submissions, in action order, for the given job step or and empty List

Parameters:

- `step` - the JobStep that holds the submissions (required)

Returns:

- a List of submissions in id order for the given step name otherwise and empty List

Since:

- 4.1.10

---

getCompletedSubmissionsForStep

```
public static java.util.List<com.avoka.fc.core.entity.Submission> getCompletedSubmissionsForStep(com.avoka.fc.core.
entity.JobStep step)
```

Returns a List of completed submissions for the step, in id order, for the given job step or and empty List

Parameters:

- `step` - the JobStep that holds the submissions (required)

Returns:

- a List of completed submissions for the step, in id order, for the given job step or and empty List

Since:

- 4.3.0

---

getFormStartUser

```
public static com.avoka.fc.core.entity.UserAccount getFormStartUser(com.avoka.fc.core.entity.JobAction
currentJobAction)
```

Returns the user that made the original submission. This is associated with the Job's Form Start Action

Parameters:

- `currentJobAction` - the JobAction associated with the Job Action Service.

Returns:

- if found returns the user associated with the original submission otherwise null

---

addSubmissionProcessingStatus

```
public static void addSubmissionProcessingStatus(
     ActionContext actionContext,

     ActionStepProperties actionStepProperties,
                                                com.avoka.fc.core.entity.Submission submission)
```

Add processing status message to the submission.

Parameters:

- `actionContext` - the actionContext for the JobAction (required)
- `actionStepProperties` - the ActionStepProperties for a particular JobAction instance (required)
- `submission` - the submission to provide a processing status message to (optional)

---

getFormDataMapPreviousStep

```
public static java.util.Map<java.lang.String,java.lang.String> getFormDataMapPreviousStep(com.avoka.fc.core.entity.
JobAction currentJobAction)
```

Returns the formDataMap for the previous JobStep submission

Parameters:

- `currentJobAction` - the current JobAction

Returns:

- the formDataMap Map<String, String> for the submission on a previous step

---

getFormDataMap

```
public static java.util.Map<java.lang.String,java.lang.String> getFormDataMap(com.avoka.fc.core.entity.Submission
submission)
```

Returns the formDataMap for the current submission

Parameters:

- `submission` - the Submission that you want the formDataMap from

Returns:

- the formDataMap Map<String, String> associated to the the

---

validatePropertyNotBlank

```
public static java.lang.String validatePropertyNotBlank(
     ActionStepProperties actionStepProperties,
                                                          java.lang.String propertyName)
```

Validate the given actionStepProperties ensuring the given propertyName value is not blank.

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test

Returns:

- the validation error message if the specified property is blank, or null otherwise

---

validatePropertyIsDate

```
public static java.lang.String validatePropertyIsDate(
     ActionStepProperties actionStepProperties,
                                                        java.lang.String propertyName)
```

Validate the given actionStepProperties ensuring the given propertyName value is a date value if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test

Returns:

- the validation error message if the specified property is date value if defined, or null otherwise

---

validatePropertyIsDouble

```
public static java.lang.String validatePropertyIsDouble(
     ActionStepProperties actionStepProperties,
                                                          java.lang.String propertyName)
```

Deprecated.

Validate the given actionStepProperties ensuring the given propertyName value is a Integer value if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test

Returns:

- the validation error message if the specified property is a Integer value if defined, or null otherwise

---

validatePropertyIsInteger

```
public static java.lang.String validatePropertyIsInteger(
     ActionStepProperties actionStepProperties,
                                                           java.lang.String propertyName)
```

Validate the given actionStepProperties ensuring the given propertyName value is a Integer value if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test

Returns:

- the validation error message if the specified property is a Integer value if defined, or null otherwise

---

validatePropertyIsBoolean

```
public static java.lang.String validatePropertyIsBoolean(
     ActionStepProperties actionStepProperties,
                                                    java.lang.String propertyName)
```

Validate the given actionStepProperties ensuring the given propertyName value is a boolean value if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test

Returns:

- the validation error message if the specified property is a boolean value if defined, or null otherwise

---

validatePropertyIsStep

```
public static java.lang.String validatePropertyIsStep(
     ActionStepProperties actionStepProperties,
                                                  java.lang.String propertyName)
```

Validate the given actionStepProperties ensuring the given propertyName value is a valid Step name if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test

Returns:

- the validation error message if the specified property is a valid Step name if defined, or null otherwise

---

validatePropertyIsEmail

```
public static java.lang.String validatePropertyIsEmail(
     ActionStepProperties actionStepProperties,
                                                   java.lang.String propertyName)
```

Validate the given actionStepProperties ensuring the given propertyName value is a valid email if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test

Returns:

- the validation error message if the specified property is a valid email if defined, or null otherwise

---

validatePropertyIsFormCode

```
public static java.lang.String validatePropertyIsFormCode(
     ActionStepProperties actionStepProperties,
                                                    java.lang.String propertyName,
                                                    com.avoka.fc.core.entity.Client client)
```

Deprecated. use validatePropertyIsForm.

Validate the given actionStepProperties ensuring the given propertyName value is a valid form code if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name
- `client` - the client organization

Returns:

- the validation error message if the specified property is a valid Step name if defined, or null otherwise

validatePropertyIsForm

```
public static java.lang.String validatePropertyIsForm(
     ActionStepProperties actionStepProperties,
                                                    java.lang.String formCodePropertyName,
                                                    com.avoka.fc.core.entity.Client client,
                                                    java.lang.String versionPropertyName)
```

Validate the given actionStepProperties ensuring the given propertyName value is a valid form code if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `formCodePropertyName` - the property name for the form code to test
- `client` - the client organization
- `versionPropertyName` - the property name for the form version to test

Returns:

- the validation error message if the specified property is a valid Step name if defined, or null otherwise

Since:

- 5.1.0

---

validatePropertyIsPortal

```
public static java.lang.String validatePropertyIsPortal(
     ActionStepProperties actionStepProperties,
                                                     java.lang.String propertyName,
                                                     com.avoka.fc.core.entity.Client client)
```

Validate the given actionStepProperties ensuring the given propertyName value is a Portal if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test
- `client` - the client organization

Returns:

- the validation error message if the specified property is a Portal if defined, or null otherwise

---

validatePropertyIsGroup

```
public static java.lang.String validatePropertyIsGroup(
     ActionStepProperties actionStepProperties,
                                                    java.lang.String propertyName)
```

Validate the given actionStepProperties ensuring the given propertyName value is a Group if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test

Returns:

- the validation error message if the specified property is a Group if defined, or null otherwise

---

validatePropertyIsFormGroupNames

```
public static java.lang.String validatePropertyIsFormGroupNames(
     ActionStepProperties actionStepProperties,
                                                          java.lang.String propertyName)
```

Validate the given actionStepProperties ensuring the given propertyName value is a set of valid form group names

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test

Returns:

- the validation error message if the specified property is set of valid form group names if defined, or null otherwise

Since:

- 4.1.0

validatePropertyIsUser

```
public static java.lang.String validatePropertyIsUser(
     ActionStepProperties actionStepProperties,
                                                    java.lang.String propertyName)
```

Validate the given actionStepProperties ensuring the given propertyName value is a User if defined

Parameters:

- `actionStepProperties` - the action step properties to validate
- `propertyName` - the property name to test

Returns:

- the validation error message if the specified property is a User if defined, or null otherwise

---

convertNumericPropertyValues

```
public static java.util.Map<java.lang.String,java.lang.Object> convertNumericPropertyValues(java.util.Map<java.lang.
String,java.lang.String> inputMap)
```

Return a Map with the numeric string property values converted to Integer, Long or Double.

Parameters:

- `inputMap` - the property name value map

Returns:

- a Map<String, Object> with property values converted to Integer, Long or Double

---

isDynamicProperty

```
public static boolean isDynamicProperty(java.lang.String propertyValue)
```

Return true if property value is dynamically resolved or false if a static literal property value.

Parameters:

- `propertyValue` - the property value to test (required)

Returns:

- true if property value is dynamically resolved or false if a static literal property value

Since:

- 4.1.0

---

expireStepTasksAndActions

```
public static
     ActionResult expireStepTasksAndActions(com.avoka.fc.core.entity.JobAction expiryJobAction,
                                            java.lang.String expiryRouteName)
```

Called from an Job Expiry service this will expire the tasks and actions for a job.

Parameters:

- `expiryJobAction` - the expiry service JobAction
- `expiryRouteName` - the route name that is returned in the ActionResult

Returns:

- ActionResult with the route name set as the expiryRouteName

Since:

- 4.3.2

---

getRouteName

```
public static java.lang.String getRouteName(com.avoka.fc.core.entity.Submission submission)
```

Returns the route name from the given submission. If blank then attempt to lookup in //SystemProfile/Job/RouteName.

Parameters:

- `submission` - the submission associated with a Job / JobAction. The submission needs to be associated with a job and the submission. getFormStatus() cannot be null.

Returns:

- the route name from the given submission

Since:

- 4.3.4

---

toJobHistoryJsonString

```
public static java.lang.String toJobHistoryJsonString(com.avoka.fc.core.entity.Job job)
```

Return the job JSON debug representation of this object for debugging purposes.

Parameters:

- `job` - the job to print out

Returns:

- the job JSON debug representation of this object for debugging purposes

Since:

- 5.1.0

JobActionWaitService

Package:

- com.avoka.fc.core.service.job.impl

Class JobActionWaitService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- AbstractJobActionService
- com.avoka.fc.core.service.job.impl.JobActionWaitService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobActionService

---

```
public class JobActionWaitService
extends
     AbstractJobActionService
```

Provides a 'Job Action Wait' Action Service.
This action service will wait until all the steps previous "Invoked" actions have been completed.

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
`JobActionWaitService--()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`ActionResult`	`execute-com.avoka.fc.core.service.job.ActionContext-( ActionContext actionContext)` Execute the action and return the result.

Methods inherited from class com.avoka.fc.core.service.job.impl. AbstractJobActionService

`AbstractJobActionService#getMaxErrorRetryAttempts--`, `AbstractJobActionService#getRetryDelayMins--`, `AbstractJobActionService#getServiceDefinition--`, `AbstractJobActionService#setMaxErrorRetryAttempts-java.lang.Integer-`, `AbstractJobActionService#setRetryDelayMins-java.lang.Integer-`, `AbstractJobActionService#setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-`, `AbstractJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-`

Methods inherited from class com.avoka.fc.core.service.CayenneService

`commitChanges, rollbackChanges`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

JobActionWaitService

```
public JobActionWaitService()
```

Method Detail

---

execute

```
public
     ActionResult execute(
     ActionContext actionContext)
```

Description copied from interface: `IJobActionService#execute-com.avoka.fc.core.service.job.ActionContext-`

Execute the action and return the result.

Parameters:

- `actionContext` - the job action execution context (required)

Returns:

- the action result

See Also:

- `IJobActionService.execute(ActionContext)`

JobControllerService

Package:

- com.avoka.fc.core.service.job.impl

Class JobControllerService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- com.avoka.fc.core.service.job.impl.JobControllerService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobController

---

```
public class JobControllerService
extends com.avoka.fc.core.service.CayenneService
implements
     IJobController, com.avoka.fc.core.service.IServiceDefinitionAware
```

Provides a Collaboration Job Controller service.

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
JobControllerService--()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
boolean	cancelJob-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job) Cancel the job, abandon any Assigned or Saved Tasks and make submission data ready for purging.
boolean	cancelledTaskSubmission-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Provides a callback to tell the Job controller that the given task submission has been cancelled.
boolean	completedJobAsyncAction-java.lang.String-(java.lang.String jobActionKey) Provides a callback to tell the Job controller that the specified asynchronous action ('Job Async Action') has been completed.
boolean	completedSubmissionDelivery-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Provides a callback to tell the Job controller that the specified submission delivery has been completed.
boolean	completedTaskSubmission-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Provides a callback to tell the Job controller that the given task submission has been completed.
com.avoka.fc.core.entity.Job	createJobWithClient-com.avoka.fc.core.entity.Client-(com.avoka.fc.core.entity.Client client) Create a job with the client, initializing the first step and action and put in 'Ready' state.
com.avoka.fc.core.entity.Job	createJobWithSubmission-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Create a job with the submission, initializing the first step and action, and using the given submission and put step action in 'Ready' state.
java.lang.Integer	getActionRetryDelayMins--()
java.lang.String	getJobDefinition--() Return the Job definition configuration (JSON format).
com.avoka.fc.core.entity.ServiceDefinition	getServiceDefinition--()

	Return the service definition.
boolean	isJobControllerEnabled--() Return true if the job controller service is enabled and should be processed by the scheduled job.
boolean	isLogJobHistory--() Set the log job history.
boolean	pausedJob-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job) Pause the job from performing any further processing.
void	processJob-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job) Process the job when a job action is in a 'Ready' state.
void	processJob-com.avoka.fc.core.entity.Job-boolean-(com.avoka.fc.core.entity.Job job, boolean logHistory) Process the job when a job action is in a 'Ready' state.
boolean	resumeJob-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job) Resume the job to perform further processing.
void	setActionRetryDelayMins-java.lang.Integer-(java.lang.Integer delayMins) Set the default action service retry delay minutes.
void	setJobControllerEnabled-boolean-(boolean enabled) Specify whether the Job Controller service is enabled.
void	setJobDefinition-java.lang.String-(java.lang.String jobDefinition) Set the Job definition configuration (JSON format).
void	setLogJobHistory-boolean-(boolean logJobHistory) Return the log job history
void	setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-(com.avoka.fc.core.entity.ServiceDefinition serviceDefinition) Set the service definition.
java.lang.String	toString--() Return the string representation of this object.

Methods inherited from class com.avoka.fc.core.service.CayenneService

commitChanges, rollbackChanges

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

JobControllerService

```
public JobControllerService()
```

Method Detail

getJobDefinition

```
public java.lang.String getJobDefinition()
```

Return the Job definition configuration (JSON format).

Returns:

- Job definition configuration (JSON format)

---

setJobDefinition

```
public void setJobDefinition(java.lang.String jobDefinition)
```

Set the Job definition configuration (JSON format).

Parameters:

- `jobDefinition` - the Job definition configuration (JSON format).

---

setJobControllerEnabled

`public void setJobControllerEnabled(boolean enabled)`

Specify whether the Job Controller service is enabled.

Parameters:

- `enabled` - specify whether the Job Controller service is enabled

---

getServiceDefinition

`public com.avoka.fc.core.entity.ServiceDefinition getServiceDefinition()`

Return the service definition.

Specified by:

- `getServiceDefinition` in interface `com.avoka.fc.core.service.IServiceDefinitionAware`

Returns:

- the service definition

See Also:

- `IServiceDefinitionAware.getServiceDefinition()`

---

setServiceDefinition

`public void setServiceDefinition(com.avoka.fc.core.entity.ServiceDefinition serviceDefinition)`

Set the service definition.

Specified by:

- `setServiceDefinition` in interface `com.avoka.fc.core.service.IServiceDefinitionAware`

Parameters:

- `serviceDefinition` - the service definition

See Also:

- `IServiceDefinitionAware.setServiceDefinition(ServiceDefinition)`

---

setActionRetryDelayMins

`public void setActionRetryDelayMins(java.lang.Integer delayMins)`

Set the default action service retry delay minutes.

Parameters:

- `delayMins` - the default action service retry delay minutes

---

getActionRetryDelayMins

`public java.lang.Integer getActionRetryDelayMins()`

Returns:

- the Mins the default action service retry delay minutes

---

isLogJobHistory

`public boolean isLogJobHistory()`

Set the log job history.

Returns:

- the logJobHistory

Since:

- 5.1.0

---

setLogJobHistory

`public void setLogJobHistory(boolean logJobHistory)`

Return the log job history

Parameters:

- `logJobHistory` - the logJobHistory to set

Since:

- 5.1.0

---

createJobWithClient

`public com.avoka.fc.core.entity.Job createJobWithClient(com.avoka.fc.core.entity.Client client)`

Description copied from interface: `IJobController#createJobWithClient-com.avoka.fc.core.entity.Client-`

Create a job with the client, initializing the first step and action and put in 'Ready' state.
If the controller is configured to process the job immediately after creation, this method will also process the job before it is returned.

Specified by:

- `IJobController#createJobWithClient-com.avoka.fc.core.entity.Client-` in interface `IJobController`

Parameters:

- `client` - the client organization which owns the job (required)

Returns:

- a new job instance

See Also:

- `IJobController.createJobWithClient(Client)`

---

createJobWithSubmission

`public com.avoka.fc.core.entity.Job createJobWithSubmission(com.avoka.fc.core.entity.Submission submission)`

Description copied from interface: `IJobController#createJobWithSubmission-com.avoka.fc.core.entity.Submission-`

Create a job with the submission, initializing the first step and action, and using the given submission and put step action in 'Ready' state.
If the controller is configured to process the job immediately after creation, this method will also process the job before it is returned. This can be useful to create an Task assigned to the user who created the job with the initial submission.

Specified by:

- `IJobController#createJobWithSubmission-com.avoka.fc.core.entity.Submission-` in interface `IJobController`

Parameters:

- `submission` - the initial job submission (required)

Returns:

- a new job instance

See Also:

- `IJobController.createJobWithSubmission(Submission)`

---

processJob

`public void processJob(com.avoka.fc.core.entity.Job job)`

Description copied from interface: `IJobController#processJob-com.avoka.fc.core.entity.Job-`

Process the job when a job action is in a 'Ready' state.

Specified by:

- `IJobController#processJob-com.avoka.fc.core.entity.Job-` in interface `IJobController`

Parameters:

- `job` - the Job instance to process

See Also:

- IJobController.processJob(Job)

---

processJob

```
public void processJob(com.avoka.fc.core.entity.Job job,
                       boolean logHistory)
```

Process the job when a job action is in a 'Ready' state.

Parameters:

- job - the Job instance to process
- logHistory - log the job execution history

Since:

- 5.1.0

---

completedTaskSubmission

```
public boolean completedTaskSubmission(com.avoka.fc.core.entity.Submission submission)
```

Description copied from interface: IJobController#completedTaskSubmission-com.avoka.fc.core.entity.Submission-

Provides a callback to tell the Job controller that the given task submission has been completed. The job controller should progress the associated task assign action status to 'Completed' and the next task wait action status to 'Ready'.

Specified by:

- IJobController#completedTaskSubmission-com.avoka.fc.core.entity.Submission- in interface IJobController

Parameters:

- submission - the completed task submission (required)

Returns:

- true if the submission task was found and associated "Task Assigned" was set to "Completed" or false otherwise

See Also:

- IJobController.completedTaskSubmission(Submission)

---

cancelledTaskSubmission

```
public boolean cancelledTaskSubmission(com.avoka.fc.core.entity.Submission submission)
```

Description copied from interface: IJobController#cancelledTaskSubmission-com.avoka.fc.core.entity.Submission-

Provides a callback to tell the Job controller that the given task submission has been cancelled. The job controller should progress the associated task assign action status to 'Cancelled' and the next task wait action status to 'Ready'.

Specified by:

- IJobController#cancelledTaskSubmission-com.avoka.fc.core.entity.Submission- in interface IJobController

Parameters:

- submission - the cancelled task submission (required)

Returns:

- true if the submission task was found and associated "Task Assigned" was set to "Cancelled" or false otherwise

See Also:

- IJobController.cancelledTaskSubmission(Submission)

---

completedJobAsyncAction

```
public boolean completedJobAsyncAction(java.lang.String jobActionKey)
```

Description copied from interface: IJobController#completedJobAsyncAction-java.lang.String-

Provides a callback to tell the Job controller that the specified asynchronous action ('Job Async Action') has been completed. The job controller should progress the associated action status to 'Completed' and the next wait action status to 'Ready'.

Specified by:

- IJobController#completedJobAsyncAction-java.lang.String- in interface IJobController

Parameters:

- `jobActionKey` - the unique job key identifier (required)

Returns:

- true if the job async action was found and set to "Completed" or false otherwise

See Also:

- `IJobController.completedJobAsyncAction(String)`

---

completedSubmissionDelivery

`public boolean completedSubmissionDelivery(com.avoka.fc.core.entity.Submission submission)`

Description copied from interface: `IJobController#completedSubmissionDelivery-com.avoka.fc.core.entity.Submission-`

Provides a callback to tell the Job controller that the specified submission delivery has been completed. The job controller should progress the any associated delivery wait action status to 'Ready'.

Specified by:

- `IJobController#completedSubmissionDelivery-com.avoka.fc.core.entity.Submission-` in interface `IJobController`

Parameters:

- `submission` - the associated delivered submission (required)

Returns:

- true if the submission task was found and associated "Delivery Wait" was set to "Completed" or false otherwise

See Also:

- `IJobController.completedSubmissionDelivery(Submission)`

---

cancelJob

`public boolean cancelJob(com.avoka.fc.core.entity.Job job)`

Description copied from interface: `IJobController#cancelJob-com.avoka.fc.core.entity.Job-`

Cancel the job, abandon any Assigned or Saved Tasks and make submission data ready for purging.

Specified by:

- `IJobController#cancelJob-com.avoka.fc.core.entity.Job-` in interface `IJobController`

Parameters:

- `job` - the job instance to cancel (required)

Returns:

- true if the job was cancelled, or false if the job was already finished

See Also:

- `IJobController.cancelJob(Job)`

---

pausedJob

`public boolean pausedJob(com.avoka.fc.core.entity.Job job)`

Description copied from interface: `IJobController#pausedJob-com.avoka.fc.core.entity.Job-`

Pause the job from performing any further processing.

Specified by:

- `IJobController#pausedJob-com.avoka.fc.core.entity.Job-` in interface `IJobController`

Parameters:

- `job` - the job instance to pause (required)

Returns:

- true if the job was paused

See Also:

- `IJobController.pausedJob(Job)`

---

resumeJob

`public boolean resumeJob(com.avoka.fc.core.entity.Job job)`

Description copied from interface: `IJobController#resumeJob-com.avoka.fc.core.entity.Job-`

Resume the job to perform further processing.

Specified by:

- `IJobController#resumeJob-com.avoka.fc.core.entity.Job-` in interface `IJobController`

Parameters:

- `job` - the job instance to pause (required)

Returns:

- true if the job was resumed

See Also:

- `IJobController.resumeJob(Job)`

---

isJobControllerEnabled

`public boolean isJobControllerEnabled()`

Description copied from interface: `IJobController#isJobControllerEnabled--`

Return true if the job controller service is enabled and should be processed by the scheduled job. This property can be used to prevent a controllers jobs from being processed while some other maintenance action is being performed.

Specified by:

- `IJobController#isJobControllerEnabled--` in interface `IJobController`

Returns:

- true if the controller service is enabled

See Also:

- `IJobController.isJobControllerEnabled()`

---

toString

`public java.lang.String toString()`

Return the string representation of this object.

Overrides:

- `toString` in class `java.lang.Object`

Returns:

- the string representation of this object

JobDeliveryService

Package:

- com.avoka.fc.core.service.job.impl

Class JobDeliveryService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- AbstractJobActionService
- com.avoka.fc.core.service.job.impl.JobDeliveryService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobActionService

---

```
public class JobDeliveryService
extends
     AbstractJobActionService
```

Provides a 'Job Delivery' Action Service.

Configuration

Below is a list of the Job Step Custom Properties and Service Definition Parameters which can be use to configure this action step.

Name	Description	Examples
Delivery Channel Name	Specify delivery channel to use to delivery the form submission data	{ "name": "Delivery Channel Name", "value": "Customer Email Delivery" }
Delivery Mode	Specify which Job submissions to deliver ['Last Submission' | 'All Submissions']	{ "name": "Delivery Mode", "value": "Last Submission" }
Delivery Abandon Incomplete Forms	Specify whether to automatically abandon submission which are not completed ['true' | 'false']	{ "name": "Delivery Abandon Incomplete Forms", "value": "true" }

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
`JobDeliveryService--()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`ActionResult`	`execute-com.avoka.fc.core.service.job.ActionContext-( ActionContext actionContext)` Execute the action and return the result.
`java.lang.String`	`validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-( ActionStepProperties actionStepProperties, com.avoka.fc.core.entity.Client client)` Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Methods inherited from class com.avoka.fc.core.service.job.impl. AbstractJobActionService

`AbstractJobActionService#getMaxErrorRetryAttempts--`, `AbstractJobActionService#getRetryDelayMins--`, `AbstractJobActionService#getServiceDefinition--`, `AbstractJobActionService#setMaxErrorRetryAttempts-java.lang.Integer-`, `AbstractJobActionService#setRetryDelayMins-java.lang.Integer-`, `AbstractJobActionService#setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-`

Methods inherited from class com.avoka.fc.core.service.CayenneService

`commitChanges, rollbackChanges`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

JobDeliveryService

```
public JobDeliveryService()
```

Method Detail

execute

```
public
     ActionResult execute(
     ActionContext actionContext)
```

Description copied from interface: `IJobActionService#execute-com.avoka.fc.core.service.job.ActionContext-`

Execute the action and return the result.

Parameters:

- `actionContext` - the job action execution context (required)

Returns:

- the action result

See Also:

- `IJobActionService.execute(ActionContext)`

---

validateProperties

```
public java.lang.String validateProperties(
     ActionStepProperties actionStepProperties,
                                        com.avoka.fc.core.entity.Client client)
```

Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Specified by:

- `IJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-` in interface `IJobActionService`

Overrides:

- `AbstractJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-` in class `AbstractJobActionService`

Parameters:

- `actionStepProperties` - the action step properties (required)
- `client` - the Job Controller client (optional)

Returns:

- null if the properties are valid or null otherwise

See Also:

- `IJobActionService.validateProperties(ActionStepProperties, Client)`

JobDeliveryWaitService

Package:

- com.avoka.fc.core.service.job.impl

Class JobDeliveryWaitService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- AbstractJobActionService
- com.avoka.fc.core.service.job.impl.JobDeliveryWaitService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobActionService

---

```
public class JobDeliveryWaitService
extends
     AbstractJobActionService
```

Provides a 'Job Delivery Wait' Action Service.
This action service will wait until all the jobs submissions have been delivered.

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
`JobDeliveryWaitService--()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`ActionResult`	`execute-com.avoka.fc.core.service.job.ActionContext-( ActionContext actionContext)` Execute the action and return the result.

Methods inherited from class com.avoka.fc.core.service.job.impl. AbstractJobActionService

`AbstractJobActionService#getMaxErrorRetryAttempts--`, `AbstractJobActionService#getRetryDelayMins--`, `AbstractJobActionService#getServiceDefinition--`, `AbstractJobActionService#setMaxErrorRetryAttempts-java.lang.Integer-`, `AbstractJobActionService#setRetryDelayMins-java.lang.Integer-`, `AbstractJobActionService#setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-`, `AbstractJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-`

Methods inherited from class com.avoka.fc.core.service.CayenneService

`commitChanges, rollbackChanges`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

JobDeliveryWaitService

```
public JobDeliveryWaitService()
```

Method Detail

---

execute

```
public
     ActionResult execute(
     ActionContext actionContext)
```

Description copied from interface: `IJobActionService#execute-com.avoka.fc.core.service.job.ActionContext-`

Execute the action and return the result.

Parameters:

- `actionContext` - the job action execution context (required)

Returns:

- the action result

See Also:

- `IJobActionService.execute(ActionContext)`

JobFormStartService

Package:

- com.avoka.fc.core.service.job.impl

Class JobFormStartService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- AbstractJobActionService
- com.avoka.fc.core.service.job.impl.JobFormStartService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobActionService

---

```
public class JobFormStartService
extends
     AbstractJobActionService
```

Provides a 'Job Form Start' Action Service.
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.

Configuration

Below is a list of the Job Action Properties which can be use to configure this step action.

Name	Description	Examples
Process Message Text	The submission processing status message text. As this is the form start the submission message will go against its associated submission.	{ "name": "Process Message Text", "value": "${formDataMap.name} your application has been approved" }
Process Message Send Email	Specify whether to send an processing message email.	{ "name": "Process Message Send Email", "value": "true" }
Process Message Email To	Specify whether to send an processing message email. If not specified then the submissions contact email address will be used.	{ "name": "Process Message Email To", "value": "${formDataMap.email}" }
Process Message Email Subject Template	Specify an alternative email subject template to the default Form and Organization email template: `'Email Submission Status Subject'`	{ "name": "Process Message Email Subject Template", "value": "Welcome Email Subject" }
Process Message Email Message Template	Specify an alternative email subject template to the default Form and Organization property email template: `'Email Submission Status Message'`	{ "name": "Process Message Email Message Template", "value": "Welcome Email Message" }
Conditional Route Name	This provides an alternate route name to that provided by the form. This my be based upon a dynamic value such as a velocity template or a property function. If a **non blank** value is returned then this will be used in place of the forms route name. @since 17.10	`{ "name": "Conditional Route Name", "value": "#if ($formDataMap.routeName == 'Approve' && $formDataMap.loanAmount >= 20000) Exceeds Threshold #end" } { "name": "Conditional Route Name", "value": "$func.invoke('Job Form Start Routing', 'RouteName:${formDataMap.routeName}\|LoanAmount:${formDataMap.loanAmount}')" }`

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description

JobFormStartService--()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
ActionResult	execute-com.avoka.fc.core.service.job.ActionContext-(ActionContext actionContext) Execute the action and return the result.
java.lang.String	validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-(ActionStepProperties actionStepProperties, com.avoka.fc.core.entity.Client client) Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Methods inherited from class com.avoka.fc.core.service.job.impl. AbstractJobActionService

AbstractJobActionService#getMaxErrorRetryAttempts--, AbstractJobActionService#getRetryDelayMins--, AbstractJobActionService#getServiceDefinition--, AbstractJobActionService#setMaxErrorRetryAttempts-java.lang.Integer-, AbstractJobActionService#setRetryDelayMins-java.lang.Integer-, AbstractJobActionService#setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-

Methods inherited from class com.avoka.fc.core.service.CayenneService

commitChanges, rollbackChanges

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JobFormStartService

```
public JobFormStartService()
```

Method Detail

execute

```
public
     ActionResult execute(
     ActionContext actionContext)
```

Description copied from interface: IJobActionService#execute-com.avoka.fc.core.service.job.ActionContext-

Execute the action and return the result.

Parameters:

- actionContext - the job action execution context (required)

Returns:

- the action result

See Also:

- IJobActionService.execute(ActionContext)

---

validateProperties

```
public java.lang.String validateProperties(
     ActionStepProperties actionStepProperties,
                                        com.avoka.fc.core.entity.Client client)
```

Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Specified by:

- IJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client- in interface IJobActionService

Overrides:

- AbstractJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client- in class AbstractJobActionService

Parameters:

- `actionStepProperties` - the action step properties (required)
- `client` - the Job Controller client (optional)

Returns:

- null if the properties are valid or null otherwise

See Also:

- `IJobActionService.validateProperties(ActionStepProperties, Client)`

JobFunctions

Package:

- com.avoka.fc.core.service.job.impl

Class JobFunctions

- java.lang.Object
- com.avoka.fc.core.service.job.impl.JobFunctions

```
public class JobFunctions
extends java.lang.Object
```

Provides Job Definition functions which can be called during the evaluation of a job definition.

Job Functions

Below is a listing of the job functions which are available for use in Job Definition property values, and in Step preConditions and Action preConditions.

---

formProperty

The `formProperty` function returns the specified form version property value (String) of the job's starting submission.

The property function example below looks up a form version property value named 'Managers Group' from the start submission form.

```
"properties": [
   { "name": "Task Assign Group",  "value": "$func.formProperty('Managers Group')" },
   ..
 ]
```

---

invoke

The `invoke` function invoke the specified Groovy Service name, with the given lists of args and return the result.

The example below invokes the Groovy Service named 'Manager Lookup' passing it an empty parameter map.

```
 "properties": [
   { "name": "Task Assign User",  "value": "$func.invoke('Manager Lookup')" },
   ..
 ]
```

The example below invokes the Groovy Service named 'Reviewer Lookup' passing it a list of arguments.

```
"properties": [
   { "name": "Task Assign User",  "value": "$func.invoke('Reviewer Lookup', $formDataMap.state, $formDataMap.
email)" },
   ..
 ]
```

In the example above the invoke function parameters are passed to the Groovy Service as an `args` list object. For example:

```
// Groovy Script args parameter.
 def state = args[0]
 def email = args[1]
```

The Groovy Service will also be passed the `job` (`Job`) and `jobAction` (`JobAction`) parameters, of the currently execution job and job action. This enables you to access all the properties of the currently executing job.

```
// Job and JobAction parameters.
 def jobName = job.getName()
 def stepName = jobAction.getJobStep().getName()
 def actionName = jobAction.getName()
```

```
println jobName + ' : ' + stepName + ' : ' + actionName
```

---

previousSubmission

The `previousSubmission` function returns the first submission (`Submission`) of the previously executing step.

The example below adds a processing status message to the previous submission.

```
"actions": [
   {
     "name": "Process Message",
     "type": "Job Process Message",
     "properties": [
       { "name": "Process Message Submission",  "value": "$func.previousSubmission()" },
       { "name": "Process Message Text",  "value": "Your application has been approved." }
     ]
   }
]
```

---

startForm

The `startForm` function returns the job start submission form (`Form`), or null if not available

The form task assign message example below uses the form name of the starting submission form.

```
"actions": [
   {
     "name": "Application Review",
     "type": "Job Task Assign",
     "properties": [
       { "name": "Task Message",  "value": "Please complete $func.startForm().formName." },
       ..
     ]
   }
]
```

---

startFormCode

The `startFormCode` function returns the form code of the job start submission form, or null if not available.

The form task assign example below uses the form code of the starting form submission for the review form.

```
"actions": [
   {
     "name": "Application Review",
     "type": "Job Task Assign",
     "properties": [
       { "name": "Task Form Code",  "value": "$func.startFormCode()" },
       { "name": "Task Subject",  "value": "Please complete direct debit form." },
       ..
     ]
   }
]
```

---

startSubmission

The `startSubmission` function returns the job's starting submission (`Submission`), or null if not found.

The example below adds a processing status message to the job start form submission.

```
"actions": [
   {
```

```
      "name": "Process Message",
      "type": "Job Process Message",
      "properties": [
        { "name": "Process Message Submission",  "value": "$func.startSubmission()" },
        { "name": "Process Message Text",  "value": "Your application has been approved." }
      ]
    }
 ]
```

---

startSubmissionXml

The `startSubmissionXml` function returns the job's starting submission XML (String), or null if not found.

The example below uses the job's start submission XML data as the Form XML Data for the assigned Form Task.

```
"actions": [
    {
      "name": "Application Review",
      "type": "Job Task Assign",
      "properties": [
        { "name": "Task Type",  "value": "Form" },
        { "name": "Task Form Code",  "value": "Debit Form" },
        { "name": "Task Form XML Data",  "value": "$func.startSubmissionXml()" }
        ..
      ]
    }
 ]
```

---

startUser

The `startUser` function returns the job's starting submission user (`UserAccount`), or null if not found.

The form task assign example below assigns the task to the initial form submission user.

```
"actions": [
    {
      "name": "Application Update",
      "type": "Job Task Assign",
      "properties": [
        { "name": "Task Assign User",  "value": "$func.startUser()" },
        { "name": "Task Subject",  "value": "Please complete update your application." },
        ..
      ]
    }
 ]
```

---

stepOrPreviousSubmissionXml

The `stepOrPreviousSubmissionXml` function returns the submission XML (String) for another submission in the same step, or the previous submission, or null if not found.

This function should be used where the steps `dynamicPreConditions` flag and `shareFormData` flag are enabled.

```
{
    "name": "Additional Products",
    "type": "",
    "dynamicPreConditions": true,
    "shareFormData": true,
    "allFormsEditable": true,
    "showPreviousForms": true,
    "redirectNext": true,
    "actions": [
      {
        "name": "Credit Cards Application",
        "type": "Job Task Assign",
        "preCondition": "$formDataMap.productCreditCards == 'true'",
        "redirectNext": true,
```

```
        "properties": [
          { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
          { "name": "Task Form Code",  "value": "onboard-credit-test" },
          { "name": "Task Message",  "value": "Please complete the Credit Card Application form." },
          { "name": "Task Subject",  "value": "Complete Credit Card Application" },
          { "name": "Task Input XML Prefill",  "value": "$func.stepOrPreviousSubmissionXml()" },
          { "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-insure-test" },
          { "name": "Task Message",  "value": "Please complete the Insurance Application form." },
          { "name": "Task Subject",  "value": "Complete Insurance Application" },
          { "name": "Task Input XML Prefill",  "value": "$func.stepOrPreviousSubmissionXml()" },
          { "name": "Task Type",  "value": "Anonymous" }
        ]
      }
    ]
```

---

stepOrStartSubmissionXml

The `stepOrStartSubmissionXml` function returns the submission XML (String) for another submission in the same step, if not found the or the job's starting submission, or null if not found.

This function should be used where the steps `dynamicPreConditions` flag and `shareFormData` flag are enabled.

```
{
    "name": "Additional Products",
    "type": "",
    "dynamicPreConditions": true,
    "shareFormData": true,
    "allFormsEditable": true,
    "showPreviousForms": true,
    "redirectNext": true,
    "actions": [
      {
        "name": "Credit Cards Application",
        "type": "Job Task Assign",
        "preCondition": "$formDataMap.productCreditCards == 'true'",
        "redirectNext": true,
        "properties": [
          { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
          { "name": "Task Form Code",  "value": "onboard-credit-test" },
          { "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": "$func.stepOrStartSubmissionXml()" },
          { "name": "Task Type",  "value": "Anonymous" }
        ]
      },
      {
        "name": "Insurance Application",
        "type": "Job Task Assign",
        "preCondition": "$formDataMap.productInsurance == 'true'",
        "redirectNext": true,
        "properties": [
          { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
          { "name": "Task Form Code",  "value": "onboard-insure-test" },
          { "name": "Task Message",  "value": "Please complete the Insurance Application form." },
          { "name": "Task Subject",  "value": "Complete Insurance Application" },
          { "name": "Task Input XML Prefill",  "value": "$func.stepOrStartSubmissionXml()" },
          { "name": "Task Type",  "value": "Anonymous" }
        ]
      }
    ]
```

---

stepOrSubmissionXml

The `stepOrSubmissionXml` function returns the submission XML (String) for another submission in the same step, or if not found the submission specified, or null if that submission cannot be found.

This function should be used where the steps `dynamicPreConditions` flag and `shareFormData` flag are enabled.

The example below uses the job's 2nd submission XML data as the Input XML Prefill data for the assigned Form Task.

```
{
    "name": "Additional Products",
    "type": "",
    "dynamicPreConditions": true,
    "shareFormData": true,
    "allFormsEditable": true,
    "showPreviousForms": true,
    "redirectNext": true,
    "actions": [
      {
        "name": "Credit Cards Application",
        "type": "Job Task Assign",
        "preCondition": "$formDataMap.productCreditCards == 'true'",
        "redirectNext": true,
        "properties": [
          { "name": "Task Assign Email",  "value": "$formDataMap.emailAddress" },
          { "name": "Task Form Code",  "value": "onboard-credit-test" },
          { "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": "$func.stepOrSubmissionXml($job.submissions.get(1))" },
          { "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-insure-test" },
          { "name": "Task Message",  "value": "Please complete the Insurance Application form." },
          { "name": "Task Subject",  "value": "Complete Insurance Application" },
          { "name": "Task Input XML Prefill",  "value": "$func.stepOrSubmissionXml($job.submissions.get(1))" },
          { "name": "Task Type",  "value": "Anonymous" }
        ]
      }
    ]
```

---

submissionXml

The `submissionXml` function returns the submission XML (String) of the specified submission, or null if not found.

The example below uses the job's 2nd submission XML data as the Input XML Prefill data for the assigned Form Task.

```
"actions": [
    {
      "name": "Application Review",
      "type": "Job Task Assign",
      "properties": [
        { "name": "Task Type",  "value": "Form" },
        { "name": "Task Form Code",  "value": "Debit Form" },
        { "name": "Task Input XML Prefill",  "value": "$func.submissionXml($job.submissions.get(1))" }
        ..
      ]
    }
 ]
```

---

userForEmail

The `userForEmail` function returns the user (`UserAccount`) for the specified email, or null if not found.

The form task assign example below assigns the task to TM `UserAccount` looked up from the specified email address. In the example below the input email address is resolved from a submitted form data extract.

```
"actions": [
    {
      "name": "Application Review",
      "type": "Job Task Assign",
      "properties": [
        { "name": "Task Assign User",  "value": "$func.userForEmail($formDataMap.managersEmail)" },
        { "name": "Task Subject",  "value": "Please review the application." },
        ..
      ]
    }
 ]
```

---

userForLogin

The `userForLogin` function returns the user (`UserAccount`) for the specified user login name, or null if not found.

The form task assign example below assigns the task to TM `UserAccount` looked up from the specified login user name. In the example below the input login user name is resolved from a submitted form data extract.

```
"actions": [
    {
      "name": "Application Review",
      "type": "Job Task Assign",
      "properties": [
        { "name": "Task Assign User",  "value": "$func.userForLogin($formDataMap.managerUsername)" },
        { "name": "Task Subject",  "value": "Please review the application." },
        ..
      ]
    }
 ]
```

Since:

- 4.1.0

Field Summary

Fields

Modifier and Type	Field and Description
`static java.lang.String`	`KEY` The function calling "func" key value.

Constructor Summary

Constructors

Constructor and Description
`JobFunctions-com.avoka.fc.core.entity.Job-(com.avoka.fc.core.entity.Job job)` Create a JobFunctions object with the given job.
`JobFunctions-com.avoka.fc.core.entity.JobAction-(com.avoka.fc.core.entity.JobAction jobAction)` Create a JobFunctions object with the given jobAction.

Method Summary

All Methods Static Methods Instance Methods Concrete Methods Deprecated Methods

Modifier and Type	Method and Description

static boolean	containsFuncCall-java.lang.String-(java.lang.String propertyValue) Return true if the given property value contains a func call reference.
static java.lang.Object	dispatch-java.lang.String-java.lang.String-com.avoka.fc.core.entity.JobAction-java.util.Map-boolean-(java.lang.String propertyName, java.lang.String propertyValue, com.avoka.fc.core.entity.JobAction jobAction, java.util.Map<java.lang.String,java.lang.String> formDataMap, boolean evaluateTemplate) Dispatch the function call and return the object result.
java.lang.String	formProperty-java.lang.String-(java.lang.String name) Return the named form version property value of the job start submission.
static java.util.List<java.lang.String>	getJobFuncInvokeServiceNames-com.avoka.fc.core.service.job.config.JobDef-(JobDef jobDef) Return the list of job func.invoke service names referenced in the jobDef
java.lang.Object	invoke-java.lang.String-java.lang.Object...-(java.lang.String serviceName, java.lang.Object... args) Invoke the specified named Groovy Service, with the given arguments and return the result.
static boolean	isObjectFuncCall-java.lang.String-(java.lang.String propertyValue) Return true if the given property value contains a direct object func call.
com.avoka.fc.core.entity.Submission	previousSubmission--()
com.avoka.fc.core.entity.Form	startForm--()
java.lang.String	startFormCode--()
com.avoka.fc.core.entity.Submission	startSubmission--()
java.lang.String	startSubmissionXml--()
com.avoka.fc.core.entity.UserAccount	startUser--()
java.lang.String	stepOrPreviousSubmissionXml--() Returns the submission XML data for the first submission in the current step, if there are no submissions then the XML Data from previous step submission, or null if not found or submission XML data has been deleted.
java.lang.String	stepOrStartSubmissionXml--() Returns the submission XML data for the first submission in the current step, if there are no submissions then the submission XML data for the start step, or null if not found or submission XML data has been deleted.
java.lang.String	stepOrSubmissionXml-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Returns the submission XML data for the first submission in the current step, if there are no submissions then the submission XML data for the given submission object, or null if not found or submission XML data has been deleted.
java.lang.String	submissionXml-com.avoka.fc.core.entity.Submission-(com.avoka.fc.core.entity.Submission submission) Return the submission XML data for the given submission object, or null if not found or submission XML data has been deleted.

com.avoka.fc.core.entity.UserAccount	userForEmail-java.lang.String-(java.lang.String email) Return the user account for the given email address.
com.avoka.fc.core.entity.UserAccount	userForLogin-java.lang.String-(java.lang.String loginName) Return the user account for the given login name.
static java.lang.String	validateFuncCall-com.avoka.fc.core.entity.Client-java.lang.String-(com.avoka.fc.core.entity.Client client, java.lang.String propertyValue) Validate the job function call property value, returning null if valid or the error message if not valid.
static java.lang.String	validateFuncCall-java.lang.String-(java.lang.String propertyValue) Deprecated. use validateFuncCall(Client, String) instead

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

KEY

```
public static final java.lang.String KEY
```

The function calling "func" key value.

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobFunctions.KEY

Constructor Detail

JobFunctions

```
public JobFunctions(com.avoka.fc.core.entity.JobAction jobAction)
```

Create a JobFunctions object with the given jobAction.

Parameters:

- jobAction - the currently executing jobAction

JobFunctions

```
public JobFunctions(com.avoka.fc.core.entity.Job job)
```

Create a JobFunctions object with the given job.

Parameters:

- job - the currently executing job

Method Detail

formProperty

```
public java.lang.String formProperty(java.lang.String name)
```

Return the named form version property value of the job start submission.

Parameters:

- `name` - the form version property name

Returns:

- the named form version property value of the job start submission, or null if not found

---

invoke

```
public java.lang.Object invoke(java.lang.String serviceName,
                               java.lang.Object... args)
```

Invoke the specified named Groovy Service, with the given arguments and return the result.

Parameters:

- `serviceName` - the Groovy Service definition name
- `args` - the service arguments

Returns:

- the executed Groovy Service result

---

previousSubmission

```
public com.avoka.fc.core.entity.Submission previousSubmission()
```

Returns:

- return the first submission of the previously executing step

---

startForm

```
public com.avoka.fc.core.entity.Form startForm()
```

Returns:

- the job start submission form, or null if not available

---

startFormCode

```
public java.lang.String startFormCode()
```

Returns:

- the form code of the job start submission form, or null if not available

---

startSubmission

```
public com.avoka.fc.core.entity.Submission startSubmission()
```

Returns:

- the starting submission of the job, or null if not found

---

startSubmissionXml

```
public java.lang.String startSubmissionXml()
```

Returns:

- the starting submission XML of the job, or null if not found or submission XML data has been deleted

---

submissionXml

```
public java.lang.String submissionXml(com.avoka.fc.core.entity.Submission submission)
```

Return the submission XML data for the given submission object, or null if not found or submission XML data has been deleted.

Parameters:

- `submission` - the submission object (optional)

Returns:

- the submission XML data for the given submission object, or null if not found or submission XML data has been deleted

---

stepOrSubmissionXml

```
public java.lang.String stepOrSubmissionXml(com.avoka.fc.core.entity.Submission submission)
```

Returns the submission XML data for the first submission in the current step, if there are no submissions then the submission XML data for the given submission object, or null if not found or submission XML data has been deleted.
Used where the step flags DynamicPreConditions and ShareFormData are enabled.

Parameters:

- `submission` - the submission object (optional)

Returns:

- the submission XML data for the first submission in the current step, if there are no submissions then the submission XML data for the given submission object, or null if not found or submission XML data has been deleted.

Since:

- 4.3.0

---

stepOrStartSubmissionXml

```
public java.lang.String stepOrStartSubmissionXml()
```

Returns the submission XML data for the first submission in the current step, if there are no submissions then the submission XML data for the start step, or null if not found or submission XML data has been deleted.
Used where the step flags DynamicPreConditions and ShareFormData are enabled.

Returns:

- the submission XML data for the first submission in the current step, if there are no submissions then the submission XML data for the start step, or null if not found or submission XML data has been deleted.

Since:

- 4.3.0

---

stepOrPreviousSubmissionXml

```
public java.lang.String stepOrPreviousSubmissionXml()
```

Returns the submission XML data for the first submission in the current step, if there are no submissions then the XML Data from previous step submission, or null if not found or submission XML data has been deleted.
Used where the step flags DynamicPreConditions and ShareFormData are enabled.

Returns:

- the submission XML data for the first submission in the current step, if there are no submissions then the XML Data from previous step submission, or null if not found or submission XML data has been deleted.

Since:

- 4.3.0

---

startUser

```
public com.avoka.fc.core.entity.UserAccount startUser()
```

Returns:

- the job start submission user, or null if not found

---

userForEmail

```
public com.avoka.fc.core.entity.UserAccount userForEmail(java.lang.String email)
```

Return the user account for the given email address.

Parameters:

- `email` - the user account email address

Returns:

- the user account for the given email address, or null if not found

---

userForLogin

```
public com.avoka.fc.core.entity.UserAccount userForLogin(java.lang.String loginName)
```

Return the user account for the given login name.

Parameters:

- `loginName` - the user account login name

Returns:

- the user account for the given login name, or null if not found

---

containsFuncCall

```
public static boolean containsFuncCall(java.lang.String propertyValue)
```

Return true if the given property value contains a func call reference.

Parameters:

- `propertyValue` - the property value to test

Returns:

- true if the given property value contains a func call reference

---

isObjectFuncCall

```
public static boolean isObjectFuncCall(java.lang.String propertyValue)
```

Return true if the given property value contains a direct object func call.

Parameters:

- `propertyValue` - the property value to test

Returns:

- true if the given property value contains a direct object func call

---

dispatch

```
public static java.lang.Object dispatch(java.lang.String propertyName,
                                        java.lang.String propertyValue,
                                        com.avoka.fc.core.entity.JobAction jobAction,
                                        java.util.Map<java.lang.String,java.lang.String> formDataMap,
                                        boolean evaluateTemplate)
```

Dispatch the function call and return the object result.

Parameters:

- `propertyName` - the property name (required)
- `propertyValue` - the property value (required)
- `jobAction` - the currently executing job action (required)
- `formDataMap` - the form data map (optional)
- `evaluateTemplate` - the evaluate function as a Velocity template

Returns:

- dispatch the function call and return the object result

---

getJobFuncInvokeServiceNames

```
public static java.util.List<java.lang.String> getJobFuncInvokeServiceNames(
     JobDef jobDef)
```

Return the list of job func.invoke service names referenced in the jobDef

Parameters:

- `jobDef` - the job definition (required)

Returns:

- the list of job func.invoke service names referenced in the jobDef

---

validateFuncCall

```
public static java.lang.String validateFuncCall(java.lang.String propertyValue)
```

Deprecated. use `validateFuncCall(Client, String)` instead

Validate the job function call property value, returning null if valid or the error message if not valid.

Parameters:

- `propertyValue` - the property value to test

Returns:

- null if valid, or the error message if not valid

---

validateFuncCall

```
public static java.lang.String validateFuncCall(com.avoka.fc.core.entity.Client client,
                                                java.lang.String propertyValue)
```

Validate the job function call property value, returning null if valid or the error message if not valid.

Parameters:

- `client` - the client of the job
- `propertyValue` - the property value to test

Returns:

- null if valid, or the error message if not valid

Since:

- 5.0.0

JobProcessMessageService

Package:

- com.avoka.fc.core.service.job.impl

Class JobProcessMessageService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- AbstractJobActionService
- com.avoka.fc.core.service.job.impl.JobProcessMessageService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobActionService

---

```
public class JobProcessMessageService
extends
     AbstractJobActionService
```

Provides a 'Job Processing Message' Action Service.
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

Configuration

Below is a list of the Job Action Properties which can be use to configure this step action.

Name	Description	Examples
Process Message Text	The submission processing status message text	{ "name": "Process Message Text", "value": "${formDataMap.name} your application has been approved" }
Process Message Submission	Specify the submission to update the processing status message on.	{ "name": "Process Message Submission", "value": "$func.startSubmission()" } { "name": "Process Message Submission", "value": "$funct.previousSubmission()" }
Process Message Submission Step	Specify the step name of the submission to update the processing status message on.	{ "name": "Process Message Submission Step", "value": "Application Start" }
Process Message Send Email	Specify whether to send an processing message email.	{ "name": "Process Message Send Email", "value": "true" }
Process Message Email To	Specify whether to send an processing message email. If not specified then the submissions contact email address will be used.	{ "name": "Process Message Email To", "value": "${formDataMap.email}" }
Process Message Email Subject Template	Specify an alternative email subject template to the default Form and Organization email template: `'Email Submission Status Subject'`	{ "name": "Process Message Email Subject Template", "value": "Welcome Email Subject" }
Process Message Email Message Template	Specify an alternative email subject template to the default Form and Organization property email template: `'Email Submission Status Message'`	{ "name": "Process Message Email Message Template", "value": "Welcome Email Message" }

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
`JobProcessMessageService--()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description

ActionResult	execute-com.avoka.fc.core.service.job.ActionContext-(ActionContext actionContext) Execute the action and return the result.
java.lang.String	validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-(ActionStepProperties actionStepProperties, com.avoka.fc.core.entity.Client client) Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Methods inherited from class com.avoka.fc.core.service.job.impl. AbstractJobActionService

AbstractJobActionService#getMaxErrorRetryAttempts--, AbstractJobActionService#getRetryDelayMins--, AbstractJobActionService#getServiceDefinition--, AbstractJobActionService#setMaxErrorRetryAttempts-java.lang.Integer-, AbstractJobActionService#setRetryDelayMins-java.lang.Integer-, AbstractJobActionService#setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-

Methods inherited from class com.avoka.fc.core.service.CayenneService

commitChanges, rollbackChanges

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JobProcessMessageService

```
public JobProcessMessageService()
```

Method Detail

execute

```
public
     ActionResult execute(
     ActionContext actionContext)
```

Description copied from interface: IJobActionService#execute-com.avoka.fc.core.service.job.ActionContext-

Execute the action and return the result.

Parameters:

- actionContext - the job action execution context (required)

Returns:

- the action result

See Also:

- IJobActionService.execute(ActionContext)

---

validateProperties

```
public java.lang.String validateProperties(
     ActionStepProperties actionStepProperties,
                                        com.avoka.fc.core.entity.Client client)
```

Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Specified by:

- IJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client- in interface IJobActionService

Overrides:

- AbstractJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client- in class AbstractJobActionService

Parameters:

- actionStepProperties - the action step properties (required)
- client - the Job Controller client (optional)

Returns:

- null if the properties are valid or null otherwise

See Also:

- `IJobActionService.validateProperties(ActionStepProperties, Client)`

JobReceiptWaitService

Package:

- com.avoka.fc.core.service.job.impl

Class JobReceiptWaitService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- AbstractJobActionService
- com.avoka.fc.core.service.job.impl.JobReceiptWaitService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobActionService

---

```
public class JobReceiptWaitService
extends
     AbstractJobActionService
```

Provides a 'Job Receipt Wait' Service.
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.

Configuration

There are no Job Action Properties which can be use to configure this action step.

Since:

- 4.1.0

Constructor Summary

Constructors

Constructor and Description
`JobReceiptWaitService--()`

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`ActionResult`	`execute-com.avoka.fc.core.service.job.ActionContext-( ActionContext actionContext)` Execute the action and return the result.

Methods inherited from class com.avoka.fc.core.service.job.impl. AbstractJobActionService

`AbstractJobActionService#getMaxErrorRetryAttempts--`, `AbstractJobActionService#getRetryDelayMins--`, `AbstractJobActionService#getServiceDefinition--`, `AbstractJobActionService#setMaxErrorRetryAttempts-java.lang.Integer-`, `AbstractJobActionService#setRetryDelayMins-java.lang.Integer-`, `AbstractJobActionService#setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-`, `AbstractJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-`

Methods inherited from class com.avoka.fc.core.service.CayenneService

`commitChanges, rollbackChanges`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

JobReceiptWaitService

```
public JobReceiptWaitService()
```

Method Detail

execute

```
public
     ActionResult execute(
     ActionContext actionContext)
```

Description copied from interface: `IJobActionService#execute-com.avoka.fc.core.service.job.ActionContext-`

Execute the action and return the result.

Parameters:

- `actionContext` - the job action execution context (required)

Returns:

- the action result

See Also:

- `IJobActionService.execute(ActionContext)`

JobTaskAssignActionBuilder

Package:

- com.avoka.fc.core.service.job.impl

Class JobTaskAssignActionBuilder

- java.lang.Object
- com.avoka.fc.core.service.job.impl.JobTaskAssignActionBuilder

---

```
public class JobTaskAssignActionBuilder
extends java.lang.Object
```

Provides a Job Task Assign Action Builder class using fluent interface design. Dynamically creates a JobAction or service type @see ServiceDefinition. SERVICE_TYPE_JOB_TASK_ASSIGN

Since:

- 4.3.4

Constructor Summary

Constructors

Constructor and Description
JobTaskAssignActionBuilder-com.avoka.fc.core.entity.JobAction-(com.avoka.fc.core.entity.JobAction jobAction) Creates a JobTaskAssignActionBuilder from a related JobTaskAssign JobAction in the same JobStep This will use the sequence of the JobAction.
JobTaskAssignActionBuilder-com.avoka.fc.core.entity.JobStep-(com.avoka.fc.core.entity.JobStep jobStep) Construct a JobTaskAssignActionBuilder

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
com.avoka.fc.core.entity.JobAction	execute--() Validates the inputs and builds the job action
JobTaskAssignActionBuilder	name-java.lang.String-(java.lang.String name) Sets the jobAction name
JobTaskAssignActionBuilder	property-java.lang.String-java.lang.String-(java.lang.String name, java.lang.String value) Adds a action property name-value pair to the propertyList for the Job Action
JobTaskAssignActionBuilder	property-java.lang.String-java.lang.String-java.lang.String-(java.lang.String name, java.lang.String value, java.lang.String template) Adds a action property name-value pair to the propertyList for the Job Action
JobTaskAssignActionBuilder	propertyList-java.util.List-boolean-(java.util.List<java.util.Map<java.lang.String,java.lang.String>> propertyList, boolean isReplace) Iterates over the propertyList parameter, list items are added to the actionProperty List for the Job Action
JobTaskAssignActionBuilder	redirectNext-boolean-(boolean redirectNext) Sets the redirectNext value for the JobAction
JobTaskAssignActionBuilder	redirectNext-java.lang.String-(java.lang.String preCondition) Sets the redirectNext Job Action
JobTaskAssignActionBuilder	repeatIndex-int-(int repeatIndex) Sets the assignRrepeatIndex for the Job Action
JobTaskAssignActionBuilder	sequence-int-(int sequence) Sets the sequence for the Job Action

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JobTaskAssignActionBuilder

```
public JobTaskAssignActionBuilder(com.avoka.fc.core.entity.JobAction jobAction)
```

Creates a JobTaskAssignActionBuilder from a related JobTaskAssign JobAction in the same JobStep This will use the sequence of the JobAction.

Parameters:

- `jobAction` - a JobAction of type @see com.avoka.fc.core.entity.ServiceDefinition.SERVICE_TYPE_JOB_TASK_ASSIGN jobAction.jobStep instance you want to create the Task Job Action in. The jobStep.status must be "In Progress" @see com.avoka.fc.core.entity.JobStep. STATUS_IN_PROGRESS. The new job action will be assigned the same sequence number as the JobAction passed in

---

JobTaskAssignActionBuilder

```
public JobTaskAssignActionBuilder(com.avoka.fc.core.entity.JobStep jobStep)
```

Construct a JobTaskAssignActionBuilder

Parameters:

- `jobStep` - the JobStep that you are wanting to create the new Job Action in. The jobStep.status must be "In Progress" @see com.avoka.fc.core. entity.JobStep.STATUS_IN_PROGRESS.

Method Detail

name

```
public
     JobTaskAssignActionBuilder name(java.lang.String name)
```

Sets the jobAction name

Parameters:

- `name` - the String name for the JobAction

Returns:

- the JobTaskAssignBuilder

---

redirectNext

```
public
     JobTaskAssignActionBuilder redirectNext(java.lang.String preCondition)
```

Sets the redirectNext Job Action

Parameters:

- `preCondition` - the String preCondition for the JobAction

Returns:

- the JobTaskAssignBuilder

---

sequence

```
public
     JobTaskAssignActionBuilder sequence(int sequence)
```

Sets the sequence for the Job Action

Parameters:

- `sequence` - Integer value (optional)

Returns:

- the JobTaskAssignBuilder

---

repeatIndex

```
public
     JobTaskAssignActionBuilder repeatIndex(int repeatIndex)
```

Sets the assignRrepeatIndex for the Job Action

Parameters:

- `repeatIndex` - Integer value (optional)

Returns:

- the JobTaskAssignBuilder

---

redirectNext

```
public
     JobTaskAssignActionBuilder redirectNext(boolean redirectNext)
```

Sets the redirectNext value for the JobAction

Parameters:

- `redirectNext` - boolean (Optional default value is false)

Returns:

- the JobTaskAssignBuilder

---

property

```
public
     JobTaskAssignActionBuilder property(java.lang.String name,
                                         java.lang.String value)
```

Adds a action property name-value pair to the propertyList for the Job Action

Parameters:

- `name` - the property name
- `value` - the property value

Returns:

- the JobTaskAssignBuilder

---

property

```
public
     JobTaskAssignActionBuilder property(java.lang.String name,
                                         java.lang.String value,
                                         java.lang.String template)
```

Adds a action property name-value pair to the propertyList for the Job Action

Parameters:

- `name` - the property name
- `value` - the property value
- `template` - the property template

Returns:

- the JobTaskAssignBuilder

---

propertyList

```
public
     JobTaskAssignActionBuilder propertyList(java.util.List<java.util.Map<java.lang.String,java.lang.String>>
propertyList,
                                             boolean isReplace)
```

Iterates over the propertyList parameter, list items are added to the actionProperty List for the Job Action

Parameters:

- `propertyList` - the provided List<Map<String,String>> that is added to the action property list for the Job Action
- `isReplace` - if set to true it will empty the contents of the action property list for the Job Action

Returns:

- the JobTaskAssignBuilder

---

execute

`public com.avoka.fc.core.entity.JobAction execute()`

Validates the inputs and builds the job action

Returns:

- the new JobTaskAssign JobAction

JobTaskAssignService

Package:

- com.avoka.fc.core.service.job.impl

Class JobTaskAssignService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- AbstractJobActionService
- com.avoka.fc.core.service.job.impl.JobTaskAssignService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobActionService

---

```
public class JobTaskAssignService
extends
     AbstractJobActionService
```

Provides a 'Job Task Assign' Action Service.

Configuration

Please note "Task Review Submission Step" must contain an action of type "Form Start" or "Task Assign" which will reference the form submission to be reviewed by this step. Also note either a user or a group must be assigned to as task.

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

The Job Task Assign service 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.

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 Service', $job)" }
Task Form Code	the form code of the form to assign in this Task. Since 5.1 Review Tasks do not need to specify a **Task Form Code**, in which case the form code and form version from the previous submission will be used. If a form code is specified without a form version then the current form version will be used.	{ "name": "Task Form Code", "value": "ABC123" } { "name": "Task Form Code", "value": "$func.startFormCode()" }
Task Form Version	this explicitly sets the form version of the form assigned in **Task Form Code**. Note this requires the **Task Form Code** to be specified. Since 5.1	{ "name": "Task Form Version", "value": "1" }
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": "$func.formProperty ('Manager Review Group')" }
Task Assign Groups	This property is only valid with 'Form' or 'Review' Task Types. The assigned task groups. A function can also specify the assigned groups.	{ "name": "Task Assign Groups", "value": "Blue-Reviewers, Red-Reviewers" } { "name": "Task Assign Groups", "value": "$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": "$formDataMap. assignGroups" } { "name": "Task Assign Group Create", "value": "true" } { "name": "Task Assign Repeating", "value": "true" }
Task Assign User	This property is only valid with 'Form' or 'Review' Task Types. The assigned task user's login name. The user has to be an active user in the databse.	{ "name": "Task Assign User", "value": "johnsmith" } { "name": "Task Assign User", "value": "$func.startUser()" } { "name": "Task Assign User", "value": "$formDataMap. managerLoginName" } { "name": "Task Assign User", "value": "$func.userForEmail ($formDataMap.managerEmail)" } { "name": "Task Assign User", "value": "$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')" }
Task User Deletable	specify whether the task can be deleted by the assignee [true | false]. If unset this is false	{ "name": "Task User Deletable", "value": "true" }
Task Scheduled	the task's scheduled time, format: yyyy-[m]m-[d]d	{ "name": "Task Scheduled", "value": "2014-04-10" }
Task Expiry	the task's expiry time, format: yyyy-[m]m-[d]d	{ "name": "Task Expiry", "value": "2014-04-10" }
Task Menu Label	the task menu label value for use in application bundle navigation menus	{ "name": "Task Menu Label", "value": "Credit Card" } { "name": "Task Menu Category", "value": "Products" }
Task Menu Category	the task menu category for use in application bundle navigation menus to group related tasks into a menu grouping	{ "name": "Task Menu Label", "value": "Credit Card" } { "name": "Task Menu Category", "value": "Products" }
Task Subject	the task subject line / title	{ "name": "Task Email Subject", "value": "Application Review Job" } { "name": "Task Email Subject", "value": "$job.referenceNumber for $formDataMap.firstName $formDataMap.lastName" }
Task Message	the task descriptive message	{ "name": "Task Email Message", "value": "$job.referenceNumber for $formDataMap.firstName $formDataMap.lastName }
Task Sequence	the sort order task sequence number	{ "name": "Task Sequence", "value": "1" }
Task Address	the task location address	{ "name": "Task Address", "value": "1a Rialto Lane, Manly, New South Wales" }

Task Latitude	the task location latitude in decimal degrees	{ "name": "Task Latitude", "value": "-33.79833" }
Task Longitude	the task location longitude in decimal degrees	{ "name": "Task Longitude", "value": "151.28723" }
Task Send Email	specify whether a notification email should be sent to task assignees [true | false]	{ "name": "Task Send Email", "value": "true" }
Task Email Message	the task email notification email message body (template). The following model is used: submission (Submission): the previous submission formDataMap: Map<String, String> the form data map for the submission task (Submission): the task submission format(EmailFormat): the email formatter portal (Portal): the Space (Portal), if the portal is specified or resolved taskUrl (String) the link to the newly created task, if portal specified or resolved environmentName (String): the environment name deployment property supportEmail (String): the email for the default sender Note: Most of the time this property should be removed in which case it will use the "Email Task Notification Message" template property will be looked up in this order: form -> organization -> space (portal) -> deployment property Adding large templates into the JobDefinitionJSON can be problematic. It may be better to set an "Email Task Notification Message" property.	{ "name": "Task Email Message", "value": "$job.referenceNumber for $formDataMap.firstName $formDataMap.lastName" } { "name": "Task Email Message", "value": '$job.referenceNumber for $formDataMap.firstName $formDataMap.lastName click here to open your task' }
Task Email Subject	the task email notification subject template. The following model is used: submission (Submission): the previous submission formDataMap: Map<String, String> the form data map for the submission task (Submission): the task submission format(EmailFormat): the email formatter portal (Portal): the Space (Portal), if the portal is specified or resolved taskUrl (String) the link to the newly created task, if portal specified or resolved environmentName (String): the environment name deployment property supportEmail (String): the email for the default sender Note: Most of the time this property should be removed in which case it will use the "Email Task Notification Subject" template property will be looked up in this order: form -> organization -> space (portal) -> deployment property Adding large templates into the JobDefinitionJSON can be problematic. It may be better to set an "Email Task Notification Subject" property.	{ "name": "Task Email Subject", "value": "Application Review Job" } { "name": "Task Email Subject", "value": "$job.referenceNumber for $formDataMap.firstName $formDataMap.lastName" }

Configuration: Status Updates and Email

Below is a list of the Job Action Properties which are used to configure this step action associated with Status Updates and Emails.

Name	Description	Examples
Process Message Text	The submission processing status message text	{ "name": "Process Message Text", "value": "${formDataMap.name} your application has been approved" }
Process Message Submission	Specify the submission to update the processing status message on.	{ "name": "Process Message Submission", "value": "$func.startSubmission()" } { "name": "Process Message Submission", "value": "$funct.previousSubmission()" }
Process Message Submission Step	Specify the step name of the submission to update the processing status message on.	{ "name": "Process Message Submission Step", "value": "Application Start" }
Process Message Send Email	Specify whether to send an processing message email.	{ "name": "Process Message Send Email", "value": "true" }
Process Message Email To	Specify whether to send an processing message email. If not specified then the submissions contact email address will be used.	{ "name": "Process Message Email To", "value": "${formDataMap.email}" }
Process Message Email Subject Template	Specify an alternative email subject template to the default Form and Organization email template: `'Email Submission Status Subject'`	{ "name": "Process Message Email Subject Template", "value": "Welcome Email Subject" }
Process Message Email Message Template	Specify an alternative email subject template to the default Form and Organization property email template: `'Email Submission Status Message'`	{ "name": "Process Message Email Message Template", "value": "Welcome Email Message" }

Since:

- 4.0.0

Field Summary

Fields

Modifier and Type	Field and Description
`static java.lang.String`	`PROPERTY_TASK_ADDRESS` The step/service property "Task Address"
`static java.lang.String`	`PROPERTY_TASK_ATTACHMENTS_PREVIOUS_STEP` The step/service property "Task Attachments Previous Step"
`static java.lang.String`	`PROPERTY_TASK_ATTACHMENTS_SUBMISSION_STEP` The step/service property "Task Attachments Submission Step"
`static java.lang.String`	`PROPERTY_TASK_EMAIL_MESSAGE` The step/service property "Task Email Message"
`static java.lang.String`	`PROPERTY_TASK_EMAIL_SUBJECT`

	The step/service property "Task Email Subject"
static java.lang.String	PROPERTY_TASK_EXPIRY The step/service property "Task Expiry"
static java.lang.String	PROPERTY_TASK_FORM_CODE The step/service property "Task Form Code"
static java.lang.String	PROPERTY_TASK_FORM_VERSION The step/service property "Task Form Version"
static java.lang.String	PROPERTY_TASK_FORM_XML_DATA The step/service property "Task Form XML Data"
static java.lang.String	PROPERTY_TASK_INPUT_XML_PREFILL The step/service property "Task Input XML Prefill"
static java.lang.String	PROPERTY_TASK_LATITUDE The step/service property "Task Latitude"
static java.lang.String	PROPERTY_TASK_LONGITUDE The step/service property "Task Longitude"
static java.lang.String	PROPERTY_TASK_MENU_CATEGORY The step/service property "Task Menu Category"
static java.lang.String	PROPERTY_TASK_MENU_LABEL The step/service property "Task Menu Label"
static java.lang.String	PROPERTY_TASK_MESSAGE The step/service property "Task Message"
static java.lang.String	PROPERTY_TASK_SEND_EMAIL The step/service property "Task Send Email"
static java.lang.String	PROPERTY_TASK_SEQUENCE The step/service property "Task Sequence"
static java.lang.String	PROPERTY_TASK_SUBJECT The step/service property "Task Subject"

Constructor Summary

Constructors

Constructor and Description
JobTaskAssignService--()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
ActionResult	execute-com.avoka.fc.core.service.job.ActionContext-(ActionContext actionContext) Execute the action and return the result.
java.lang.String	validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-(ActionStepProperties actionStepProperties, com.avoka.fc.core.entity.Client client) Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Methods inherited from class com.avoka.fc.core.service.job.impl. AbstractJobActionService

AbstractJobActionService#getMaxErrorRetryAttempts--, AbstractJobActionService#getRetryDelayMins--, AbstractJobActionService#getServiceDefinition--, AbstractJobActionService#setMaxErrorRetryAttempts-java.lang.Integer-, AbstractJobActionService#setRetryDelayMins-java.lang.Integer-, AbstractJobActionService#setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-

Methods inherited from class com.avoka.fc.core.service.CayenneService

commitChanges, rollbackChanges

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

PROPERTY_TASK_FORM_CODE

`public static final java.lang.String PROPERTY_TASK_FORM_CODE`

The step/service property "Task Form Code"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_FORM_CODE

PROPERTY_TASK_FORM_VERSION

`public static final java.lang.String PROPERTY_TASK_FORM_VERSION`

The step/service property "Task Form Version"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_FORM_VERSION

PROPERTY_TASK_EXPIRY

`public static final java.lang.String PROPERTY_TASK_EXPIRY`

The step/service property "Task Expiry"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_EXPIRY

PROPERTY_TASK_SEND_EMAIL

`public static final java.lang.String PROPERTY_TASK_SEND_EMAIL`

The step/service property "Task Send Email"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_SEND_EMAIL

PROPERTY_TASK_EMAIL_SUBJECT

`public static final java.lang.String PROPERTY_TASK_EMAIL_SUBJECT`

The step/service property "Task Email Subject"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_EMAIL_SUBJECT

PROPERTY_TASK_EMAIL_MESSAGE

`public static final java.lang.String PROPERTY_TASK_EMAIL_MESSAGE`

The step/service property "Task Email Message"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_EMAIL_MESSAGE

PROPERTY_TASK_SUBJECT

`public static final java.lang.String PROPERTY_TASK_SUBJECT`

The step/service property "Task Subject"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_SUBJECT

PROPERTY_TASK_MESSAGE

`public static final java.lang.String PROPERTY_TASK_MESSAGE`

The step/service property "Task Message"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_MESSAGE

PROPERTY_TASK_SEQUENCE

```
public static final java.lang.String PROPERTY_TASK_SEQUENCE
```

The step/service property "Task Sequence"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_SEQUENCE

PROPERTY_TASK_ADDRESS

```
public static final java.lang.String PROPERTY_TASK_ADDRESS
```

The step/service property "Task Address"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_ADDRESS

PROPERTY_TASK_LATITUDE

```
public static final java.lang.String PROPERTY_TASK_LATITUDE
```

The step/service property "Task Latitude"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_LATITUDE

PROPERTY_TASK_LONGITUDE

```
public static final java.lang.String PROPERTY_TASK_LONGITUDE
```

The step/service property "Task Longitude"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_LONGITUDE

PROPERTY_TASK_FORM_XML_DATA

```
public static final java.lang.String PROPERTY_TASK_FORM_XML_DATA
```

The step/service property "Task Form XML Data"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_FORM_XML_DATA

PROPERTY_TASK_INPUT_XML_PREFILL

```
public static final java.lang.String PROPERTY_TASK_INPUT_XML_PREFILL
```

The step/service property "Task Input XML Prefill"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_INPUT_XML_PREFILL

PROPERTY_TASK_ATTACHMENTS_PREVIOUS_STEP

```
public static final java.lang.String PROPERTY_TASK_ATTACHMENTS_PREVIOUS_STEP
```

The step/service property "Task Attachments Previous Step"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_ATTACHMENTS_PREVIOUS_STEP

PROPERTY_TASK_ATTACHMENTS_SUBMISSION_STEP

`public static final java.lang.String PROPERTY_TASK_ATTACHMENTS_SUBMISSION_STEP`

The step/service property "Task Attachments Submission Step"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_ATTACHMENTS_SUBMISSION_STEP

PROPERTY_TASK_MENU_LABEL

`public static final java.lang.String PROPERTY_TASK_MENU_LABEL`

The step/service property "Task Menu Label"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_MENU_LABEL

PROPERTY_TASK_MENU_CATEGORY

`public static final java.lang.String PROPERTY_TASK_MENU_CATEGORY`

The step/service property "Task Menu Category"

See Also:

- Constants#com.avoka.fc.core.service.job.impl.JobTaskAssignService.PROPERTY_TASK_MENU_CATEGORY

Constructor Detail

JobTaskAssignService

`public JobTaskAssignService()`

Method Detail

execute

```
public
     ActionResult execute(
     ActionContext actionContext)
```

Description copied from interface: `IJobActionService#execute-com.avoka.fc.core.service.job.ActionContext-`

Execute the action and return the result.

Parameters:

- `actionContext` - the job action execution context (required)

Returns:

- the action result

See Also:

- `IJobActionService.execute(ActionContext)`

---

validateProperties

```
public java.lang.String validateProperties(
     ActionStepProperties actionStepProperties,
                                        com.avoka.fc.core.entity.Client client)
```

Validate the action service using the given action step service properties, returning null if valid or an error string otherwise.

Specified by:

- `IJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-` in interface `IJobActionService`

Overrides:

- `AbstractJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-` in class `AbstractJobActionService`

Parameters:

- `actionStepProperties` - the action step properties (required)
- `client` - the Job Controller client (required)

Returns:

- null if the properties are valid or null otherwise

See Also:

- `IJobActionService.validateProperties(ActionStepProperties, Client)`

JobTaskWaitService

Package:

- com.avoka.fc.core.service.job.impl

Class JobTaskWaitService

- java.lang.Object
- com.avoka.fc.core.service.BaseService
- com.avoka.fc.core.service.CayenneService
- AbstractJobActionService
- com.avoka.fc.core.service.job.impl.JobTaskWaitService

All Implemented Interfaces:

- com.avoka.fc.core.service.IServiceDefinitionAware, IJobActionService

---

```
public class JobTaskWaitService
extends
     AbstractJobActionService
```

Provides a 'Job Task Wait' Action Service.
The is usually configured in the same step as a Job Task Assign action service which has creates and assigns an Task. This service waits for the Task to be submitted. It then route the task based upon the the route name specified in the form data **//SystemProfile/Job/RouteName**.

Configuration

Below is a list of the Job Action Properties which can be use to configure this action step.

Name	Description	Examples
Conditional Route Name	This provides an alternate route name to that provided by the form. This my be based upon a dynamic value such as a velocity template or a property function. If a **non blank** value is returned then this will be used in place of the forms route name.	`{ "name": "Conditional Route Name", "value": "#if ($formDataMap.routeName == 'Approve' && $formDataMap.loanAmount >= 20000) Exceeds Threshold #end" } { "name": "Conditional Route Name", "value": "##EXECUTE_GROOVY_SERVICE('2 Step Review Applicant Update Routing', 'RouteName:${formDataMap.routeName} |LoanAmount:${formDataMap.loanAmount}')" }`

Since:

- 4.0.0

Constructor Summary

Constructors

Constructor and Description
JobTaskWaitService--()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
ActionResult	execute-com.avoka.fc.core.service.job.ActionContext-(ActionContext actionContext) Execute the action and return the result.

Methods inherited from class com.avoka.fc.core.service.job.impl. AbstractJobActionService

AbstractJobActionService#getMaxErrorRetryAttempts--, AbstractJobActionService#getRetryDelayMins--, AbstractJobActionService#getServiceDefinition--, AbstractJobActionService#setMaxErrorRetryAttempts-java.lang.Integer-, AbstractJobActionService#setRetryDelayMins-java.lang.Integer-, AbstractJobActionService#setServiceDefinition-com.avoka.fc.core.entity.ServiceDefinition-, AbstractJobActionService#validateProperties-com.avoka.fc.core.service.job.impl.ActionStepProperties-com.avoka.fc.core.entity.Client-

Methods inherited from class com.avoka.fc.core.service.CayenneService
`commitChanges, rollbackChanges`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

JobTaskWaitService

```
public JobTaskWaitService()
```

Method Detail

execute

```
public
     ActionResult execute(
     ActionContext actionContext)
```

Description copied from interface: `IJobActionService#execute-com.avoka.fc.core.service.job.ActionContext-`

Execute the action and return the result.

Parameters:

- `actionContext` - the job action execution context (required)

Returns:

- the action result

See Also:

- `IJobActionService.execute(ActionContext)`

Constants

Constant Field Values

com.avoka.*

com.avoka.fc.core.service.job.config. JobDef

Modifier and Type	Constant Field	Value
public static final java.lang.String	JobDef#DESCRIPTION	"description"
public static final java.lang.String	JobDef#NAME	"name"
public static final java.lang.String	JobDef#PRE_CONDITION_SUBMISSION_STEP	"Pre Condition Submission Step"

com.avoka.fc.core.service.job.config. StepDef

Modifier and Type	Constant Field	Value
public static final java.lang.String	StepDef#DEFAULT_ROUTE_NAME	"Default"
public static final java.lang.String	StepDef#EXPIRY_ROUTE_NAME	"Expiry"
public static final java.lang.String	StepDef#NEXT_STEP_PREVOUS_STEP	"##PREVIOUS_STEP"
public static final java.lang.String	StepDef#TYPE_ENDPOINT	"endpoint"
public static final java.lang.String	StepDef#TYPE_START	"start"

com.avoka.fc.core.service.job.impl. JobActionUtils

Modifier and Type	Constant Field	Value
public static final java.lang.String	JobActionUtils#PROPERTY_PROCESS_MESSAGE_EMAIL_MESSAGE_TEMPLATE	"Process Message Email Message Template"
public static final java.lang.String	JobActionUtils#PROPERTY_PROCESS_MESSAGE_EMAIL_SUBJECT_TEMPLATE	"Process Message Email Subject Template"
public static final java.lang.String	JobActionUtils#PROPERTY_PROCESS_MESSAGE_EMAIL_TO	"Process Message Email To"
public static final java.lang.String	JobActionUtils#PROPERTY_PROCESS_MESSAGE_SEND_EMAIL	"Process Message Send Email"
public static final java.lang.String	JobActionUtils#PROPERTY_PROCESS_MESSAGE_SUBMISSION	"Process Message Submission"
public static final java.lang.String	JobActionUtils#PROPERTY_PROCESS_MESSAGE_SUBMISSION_STEP	"Process Message Submission Step"
public static final java.lang.String	JobActionUtils#PROPERTY_PROCESS_MESSAGE_TEXT	"Process Message Text"

com.avoka.fc.core.service.job.impl. JobFunctions

Modifier and Type	Constant Field	Value
public static final java.lang.String	JobFunctions#KEY	"func"

com.avoka.fc.core.service.job.impl. JobTaskAssignService

Modifier and Type	Constant Field	Value
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_ADDRESS	"Task Address"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_ATTACHMENTS_PREVIOUS_STEP	"Task Attachments Previous Step"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_ATTACHMENTS_SUBMISSION_STEP	"Task Attachments Submission Step"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_EMAIL_MESSAGE	"Task Email Message"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_EMAIL_SUBJECT	"Task Email Subject"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_EXPIRY	"Task Expiry"
	JobTaskAssignService#PROPERTY_TASK_FORM_CODE	"Task Form Code"

public static final java.lang.String		
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_FORM_VERSION	"Task Form Version"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_FORM_XML_DATA	"Task Form XML Data"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_INPUT_XML_PREFILL	"Task Input XML Prefill"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_LATITUDE	"Task Latitude"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_LONGITUDE	"Task Longitude"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_MENU_CATEGORY	"Task Menu Category"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_MENU_LABEL	"Task Menu Label"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_MESSAGE	"Task Message"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_SEND_EMAIL	"Task Send Email"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_SEQUENCE	"Task Sequence"
public static final java.lang.String	JobTaskAssignService#PROPERTY_TASK_SUBJECT	"Task Subject"

Collaboration Jobs - Current Release Documentation Home

Welcome to your new space!

Confluence spaces are great for sharing content and news with your team. This is your home page. Right now it shows recent space activity, but you can customize this page in anyway you like.

Complete these tasks to get started

- [ ] Edit this home page - Click Edit in the top right of this screen to customize your Space home page
- [ ] Create your first page - Click the Create button in the header to get started
- [ ] Brand your Space - Click Configure Sidebar in the left panel to update space details and logo
- [ ] Set permissions - Click Space Tools in the left sidebar to update permissions and give others access

Recent space activity

Chad Bakeman

Transact Manager Form Configuration commented Mar 20, 2019

Maestro commented Mar 20, 2019

Maestro commented Mar 20, 2019

Anonymous

Chaining and Form Bundles commented Oct 19, 2018

Sindy Tang

Customer Onboarding Job - Form Bundle commented Aug 21, 2018

Space contributors

- Unknown User (nsell) (1813 days ago)