Course Exercises Guide

# Developing workflow solutions using IBM Business Automation Workflow V20.0.0.1

Course code WB835 / ZB835 ERC 1.0

IBM Training

**December 2020 edition**

## Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*United States of America*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

### Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

# Contents

# Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

| | | |
|---|---|---|
| Bluemix® | Cognitive Era™ | Cognos® |
| SPSS® | Watson Avatar® | Worklight® |

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

# Exercises description

This course includes the following exercises:

- Exercise 1. Creating your first process application
- Exercise 2. Playback 0: Creating the To-Be process
- Exercise 3. Playback 0: Controlling process flow
- Exercise 4. Playback 1: Controlling process flow with business data
- Exercise 5. Playback 1: Business data, services, and coaches
- Exercise 6. Playback 1: User interface design and implementation
- Exercise 7. Playback 1: Conducting the Playback session
- Exercise 8. Playback 2: Integrations
- Exercise 9. Playback 3: Handling errors and deploying your process application
- Exercise 10. Implementing a custom routing solution
- Exercise 11. Implementing the "four eyes" policy by using a team filter
- Exercise 12. Building a cancellation pattern
- Exercise 13. Building web service connections
- Exercise 14. Handling content events in a process

In the exercise instructions, you can check off the line before each step as you complete it to track your progress.

Most exercises include required sections, which should always be completed. It might be necessary to complete these sections before you can start later exercises. Some exercises also include optional sections that you might want to complete when you have sufficient time and want an extra challenge.

This course has been tested in Firefox. Firefox is the default browser. However, Chrome is available. Keep in mind, the behavior may be different. If Firefox prompts to install updates, decline the updates.

**Important**

The exercises in this course use a set of lab files that might include scripts, applications, files, solution files, project interchange files, and others. The course lab files can be found in the following directory:

`C:\labfiles` for the Windows platform

The exercises point you to the lab files as you need them.

# How to follow the exercise instructions

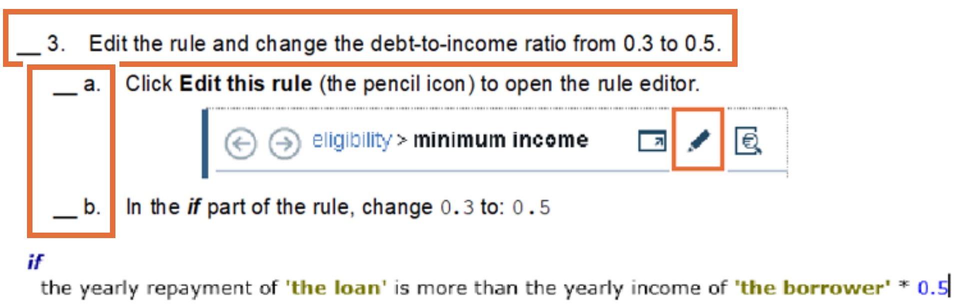
## Exercise structure

Each exercise is divided into sections with a series of steps and substeps. The step represents an action to be completed. If required, the substeps provide guidance on completing the action:

- The numbered steps (1, 2, 3) describe what actions to do.
- The lettered substeps (a, b, c) provide detailed guidance on how to complete the action.

1+1=2 **Example**

As shown in this example, the numbered step (“3”) tells you to change the value in a rule. Substeps “a” and “b” provide details on how to edit.




If you already know how to edit rules and change values, you can skip the substeps.

### Text highlighting in exercises

Different text styles indicate various elements in the exercises.

Words that are highlighted in **bold font** represent GUI items that you interact with, such as:

- Menu items
- Field names
- Icons

Words that are highlighted with a `code font` include the following items:

- Text that you type or enter as a value
- System messages
- Directory paths
- Code

## Tracking your progress

As shown in the example step, you can see that an underscore precedes each numbered step and lettered substep.

You are encouraged to use these markers to track your progress. As you complete a step, place an **X** or a check mark on the underscore to indicate that it is completed. By tracking your progress in this manner, you can stay focused when you experience interruptions during a lengthy exercise.

## Required exercise sections

Most exercises include required sections that should always be completed. It might be necessary to complete these sections before you can start subsequent exercises.

Dependencies between exercises are listed in the exercise introduction.

## Optional exercise sections

Some exercises might also include optional sections that you can complete when you have sufficient time and want an extra challenge.

---

**Stop**

***Course updates and errata***


A Course Corrections document might be available for this course.

If you are taking the class with an instructor, the instructor can provide this document to you.

If you are taking the course in a self-paced environment, the course corrections document is provided with the other manuals.

To check whether a Course Corrections document exists for this course:

1. Go to the following URL: https://ibm-learning-skills-dev.github.io/
2. On the web page, click the **IBM Cloud courses** category.
3. Find your product in one of the categories and click the link to open the category.
4. Find your course in the list and click the link to see whether an errata document exists with updated instructions.
5. To save the file to your computer, click the document link and follow the dialog box prompts.

# User IDs and passwords

*Table 1. User accounts and passwords*

| Type | Username | Password |
|---|---|---|
| Windows Server 2016 | `administrator` | `passw0rd` |
| IBM Business Automation Workflow | `author1` | `passw0rd` |
| WebSphere Application Server cell administrator | `administrator` | `passw0rd` |

# Exercise 1. Creating your first process application

## Estimated time

01:00

## Overview

This exercise introduces Business Automation Workflow and the creation of your first process application. You are also introduced to various tools that can be used for troubleshooting. After creating your process application, you learn how to export and import snapshots.

## Objectives

After completing this exercise, you should be able to:

- Start IBM Business Automation Workflow
- Create a process application in Workflow Center
- Use the Process Admin console and other tools to aid in troubleshooting
- Export and import process applications

## Introduction

IBM Business Automation Workflow is a platform to create workflow applications to improve productivity. Workflow applications coordinate work between tasks that are performed by humans and automated tasks to improve daily business operations.

IBM Business Automation Workflow is part of the IBM Digital Business Automation platform, which includes various product offerings that you can use to digitize all aspects of business operations, while providing real-time insight into your business.

## Requirements

None

## Index

# Exercise instructions

## *Part 1: Start the IBM Workflow Center deployment environment*

Before you can start the IBM Workflow Center deployment environment, three server configurations must be started. After logging on to the lab environment, start the Deployment Manager profile, the Node Agent profile, and the Deployment Environment.

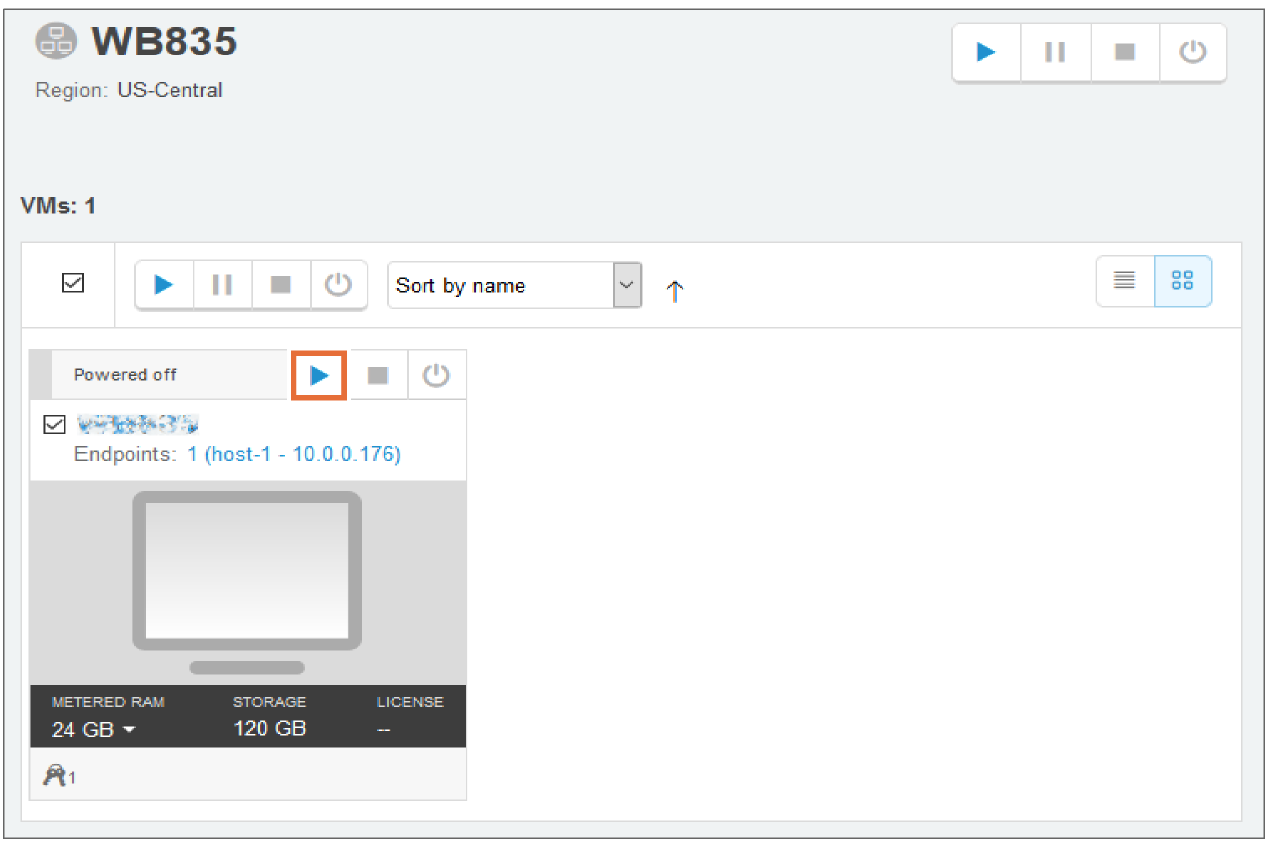
**Important**

All three server configurations must be started in order, starting with the Deployment Manager profile, then the Node Agent profile, and followed last by the Deployment Environment. To accomplish this task, the product provides a Quick Start routine to run the server start in that order.

__ 1. Start the lab environment.

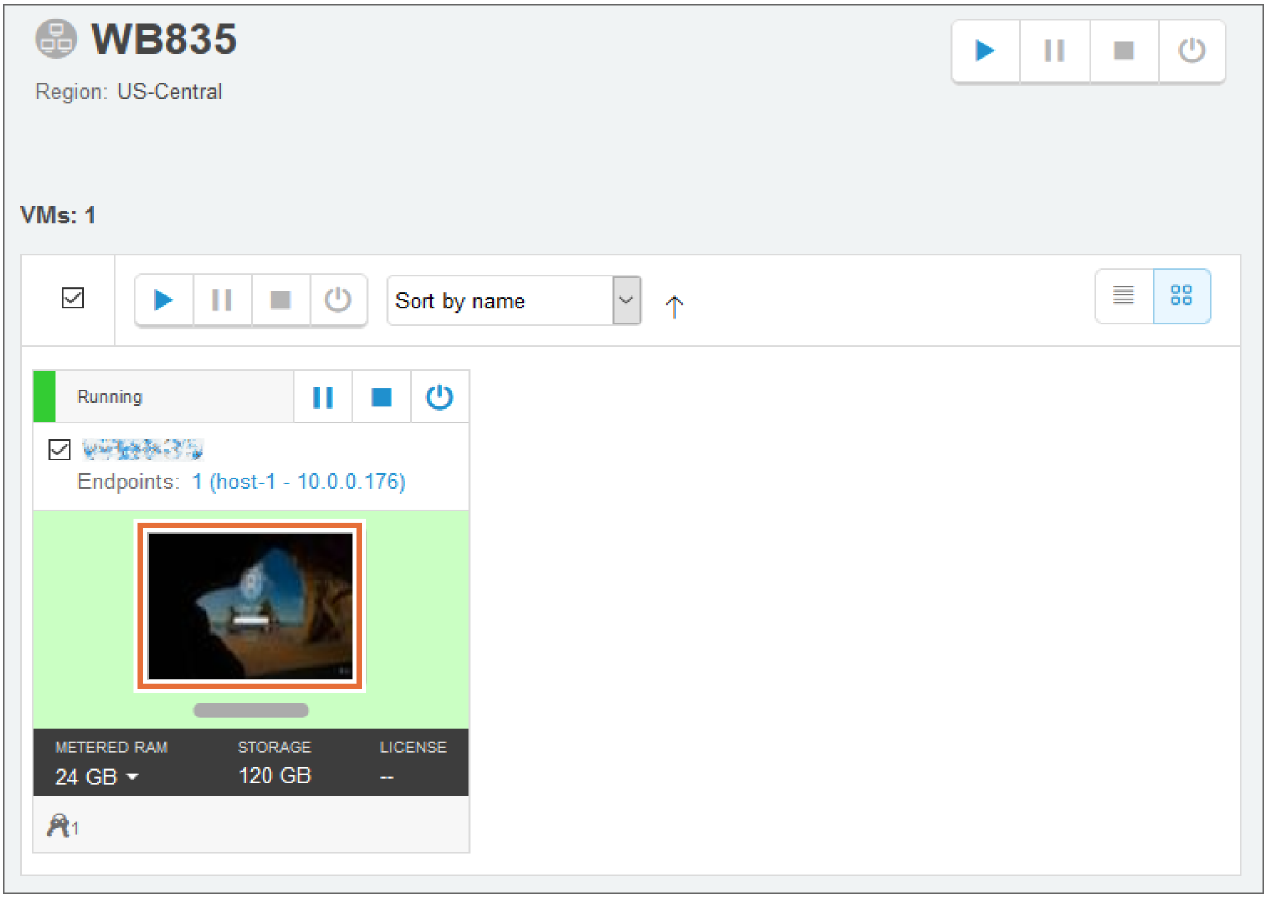
When you receive your lab environment, the environment is in a Stopped state.

__ a. To start the lab environment, click **Run this VM**.




It may take several minutes for the lab environment to start.

__ b. When the environment is in a started state, it turns green and the status changes to Running. Click the computer icon to enter the environment.



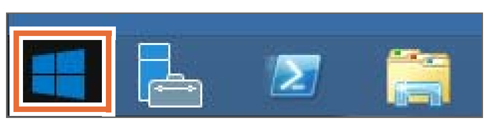

The image in the computer might be different.

__ 2. Start the **Workflow Center** server.

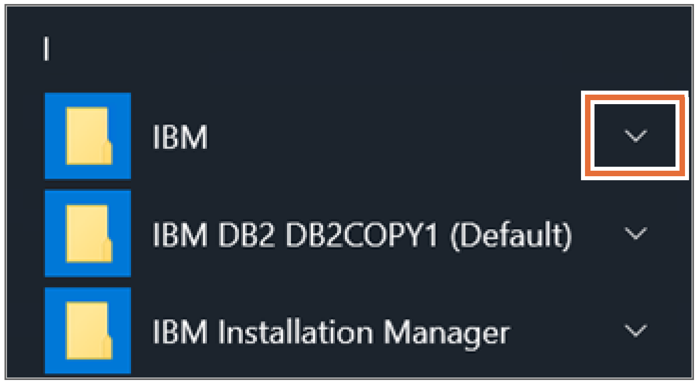
__ a. Log in to the Windows environment using the following credentials:

**Username**: `administrator` **Password**: `passw0rd`

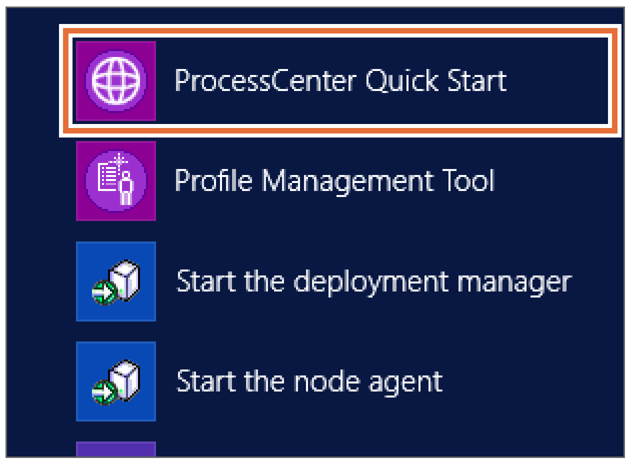
__ b. Click the **Windows Icon**.



__ c. In the Start menu, scroll down to the **IBM** group and click the **Down Arrow** link next to it.



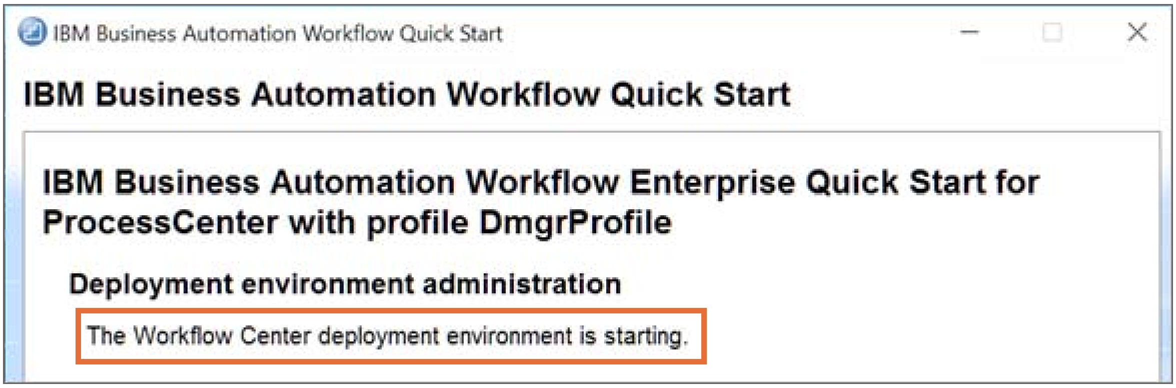

__ d. Click **ProcessCenter Quick Start**.



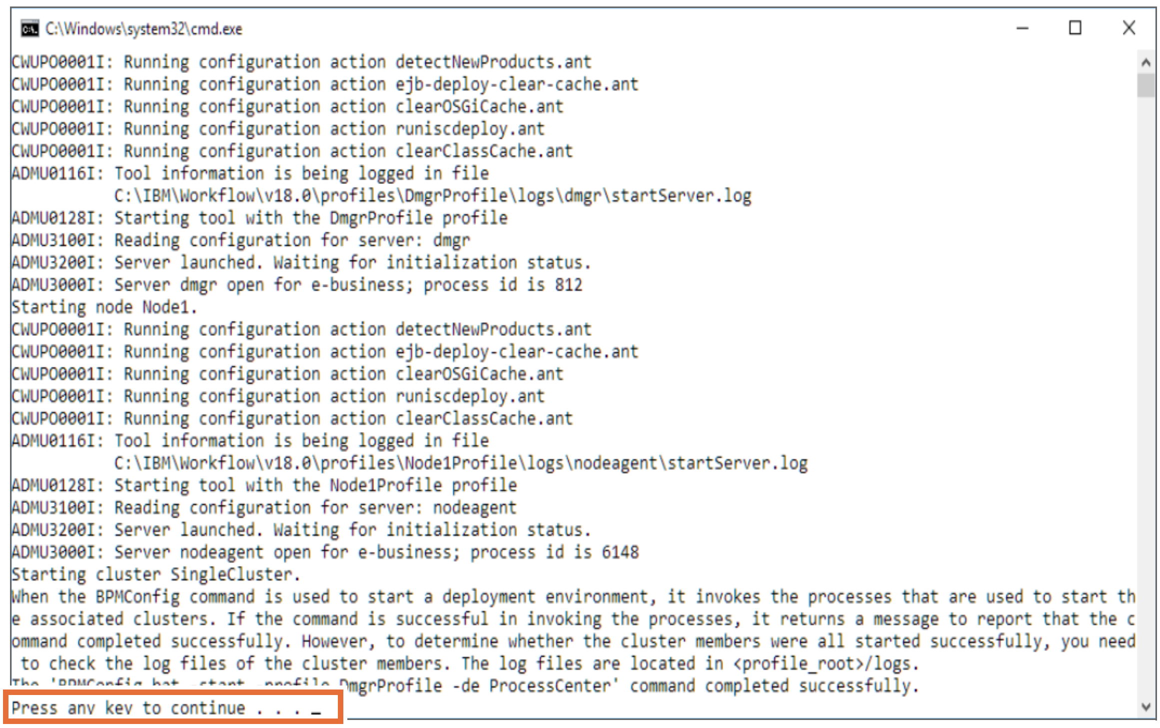

__ e. In the **IBM Business Automation Workflow Quick Start** window that opens, click the **Start the Workflow Center Deployment Environment** link.


It takes a while for the deployment environment to get started. A command prompt window runs through the start of the Deployment Manager profile, Node Agent profile, and Deployment Environment. Allow the entire start process to complete. The Quick

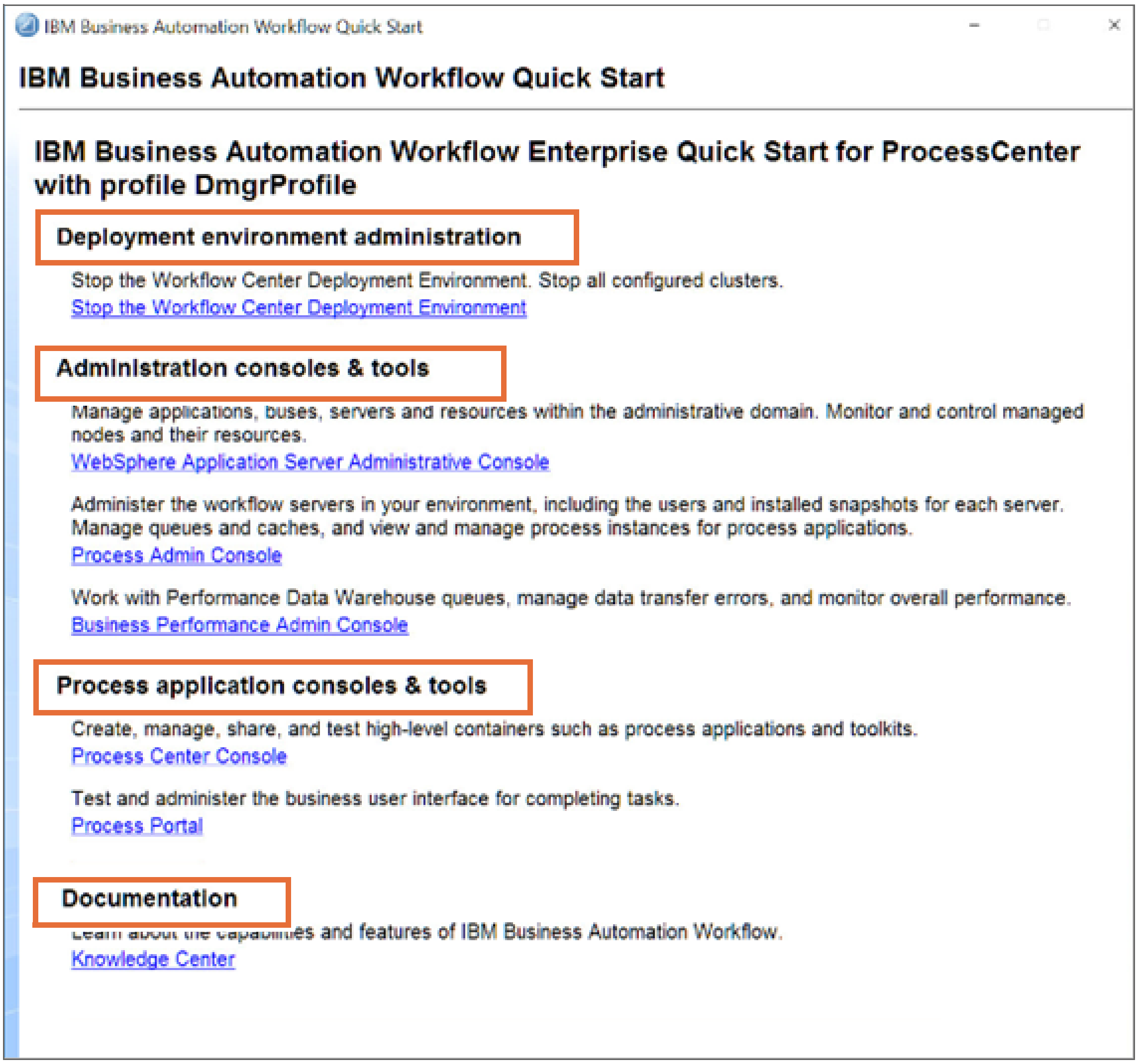
Start window displays that the environment is starting. It can take about 10 – 20 minutes, so make sure that you provide ample time for this initial start.




__ f. Switch to the newly open command window. When the environment starts, the **Press any key to continue** message is displayed. When prompted, press any key. You might have to press a key multiple times, at which point the command window closes.



__ g. The Quick Start window refreshes and updates the display. It might take up to 10 – 20 minutes for the servers to fully engage depending on the hardware configuration. Wait for the four sections to be populated in the **IBM Business Automation Workflow Quick Start** window (Deployment environment administration, Administration consoles and tools, Process application consoles and tools, and Documentation).




## *Part 2: Log in to the IBM Workflow Center and set preferences*

__ 1. The Workflow Center includes a repository for all processes, services, and other assets created in IBM Process Designer. The Workflow Center provides the tools that you need to maintain the repository. You now connect to the **IBM Workflow Center**.

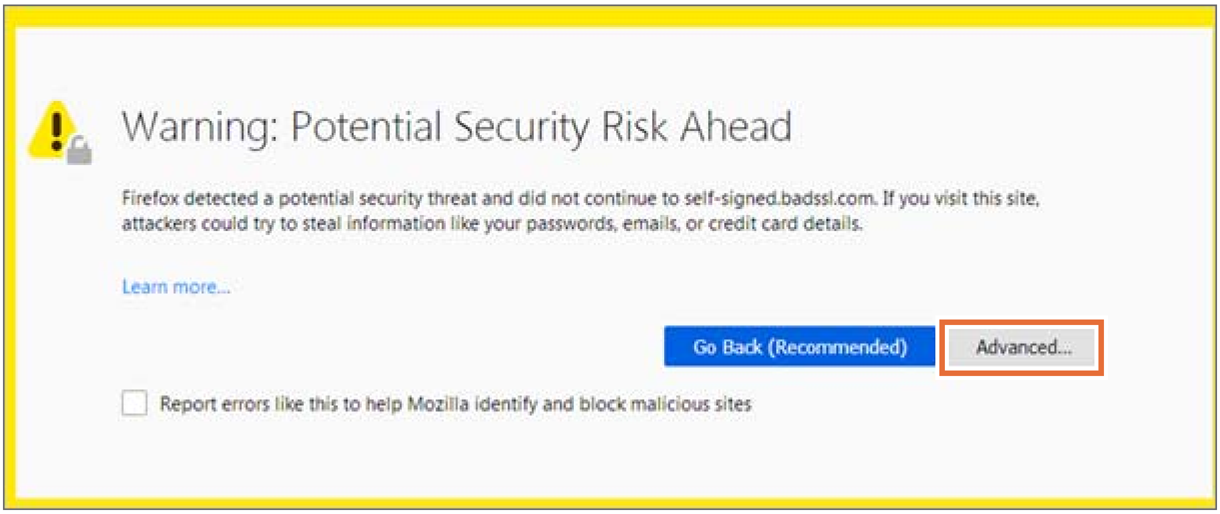
__ a. Minimize both the **IBM Business Automation Workflow Quick Start** window and the **ProcessCenter Quick Start** command window.

__ b. Open Firefox and click the link on the Bookmark toolbar for **IBM Workflow Center**

**Information**

The new terminology for Process Center is Workflow Center. While the new term is used throughout the product, there are times when the product still makes a reference to Process Center.

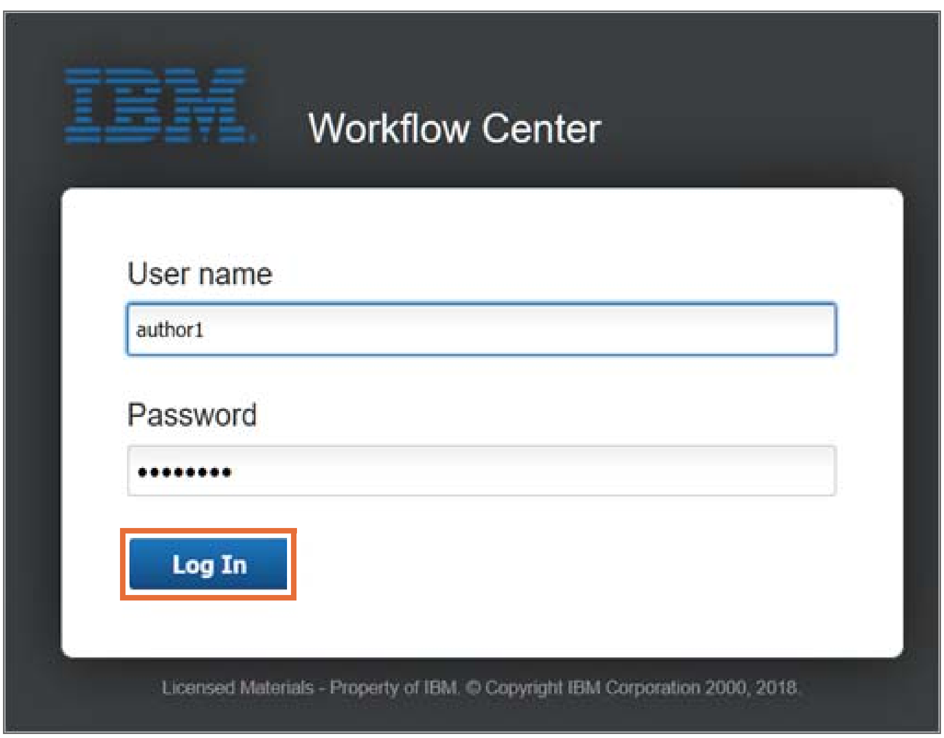
__ c. If a Firefox security warning message is displayed, it is safe to ignore that. Click **Advanced** to continue. Depending on the security message, you might have to click a few times.




A loading screen and then a login screen are displayed for Workflow Center.

__ d. Enter `author1` in the **User name** field and `passw0rd` in the **Password** field.

__ e. Click **Log In**.

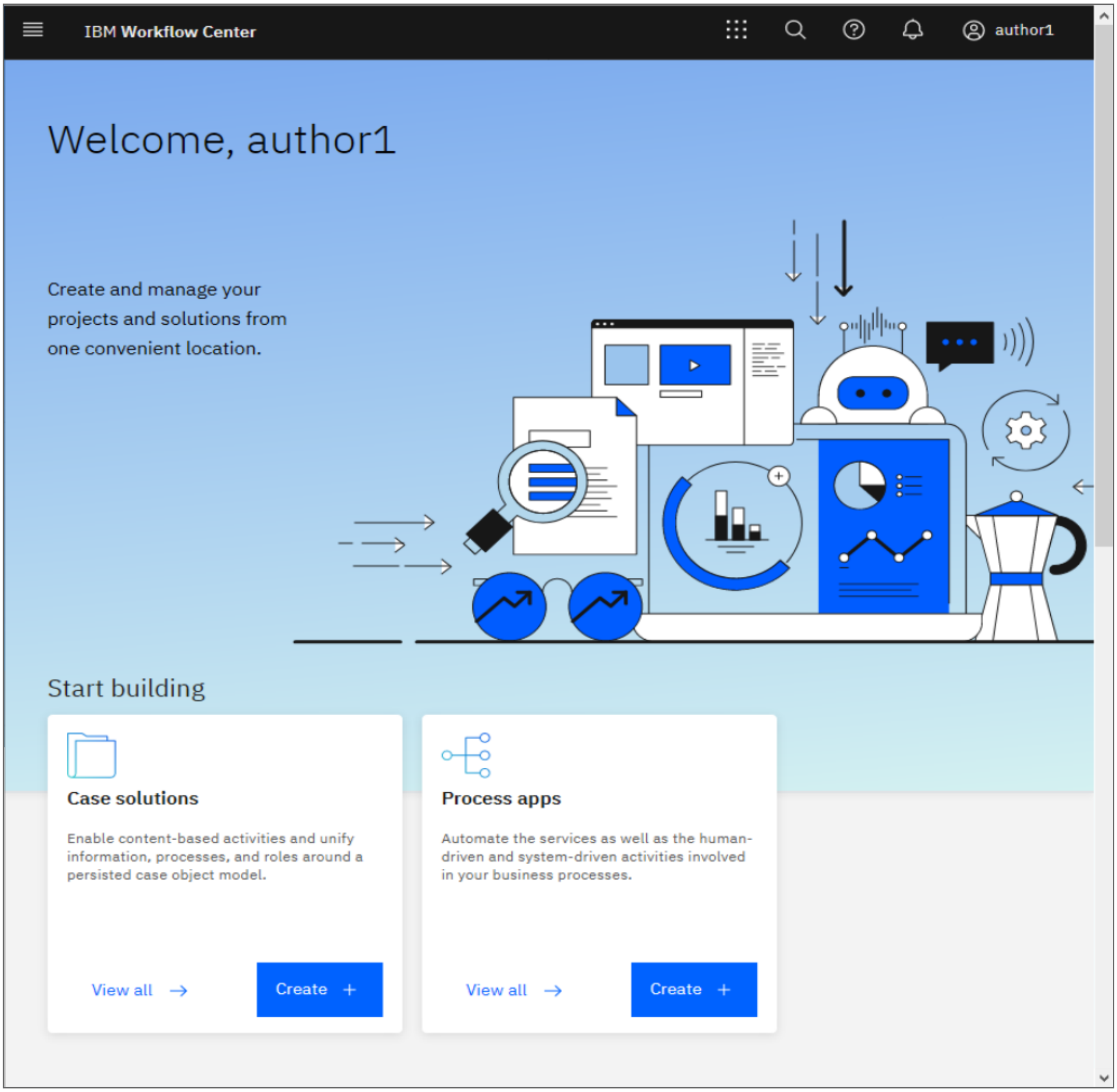


**Information**

The **Workflow Center** repository URL is: `https://ws2016x64:9443/WorkflowCenter/`

The **Classic Workflow Center** repository URL is: `https://ws2016x64:9443/ProcessCenter/`

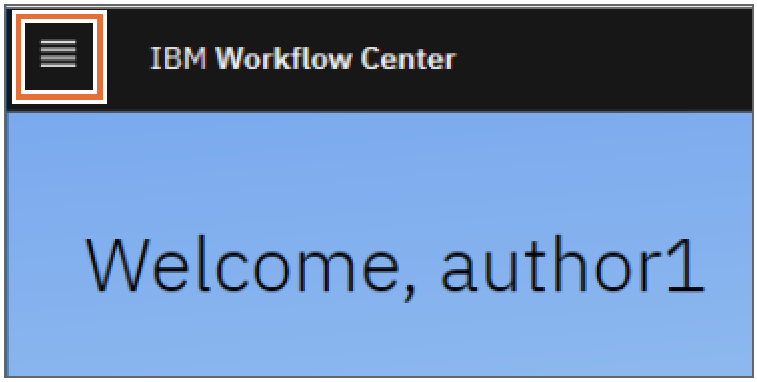
You are now logged in to the Workflow Center as `author1`.



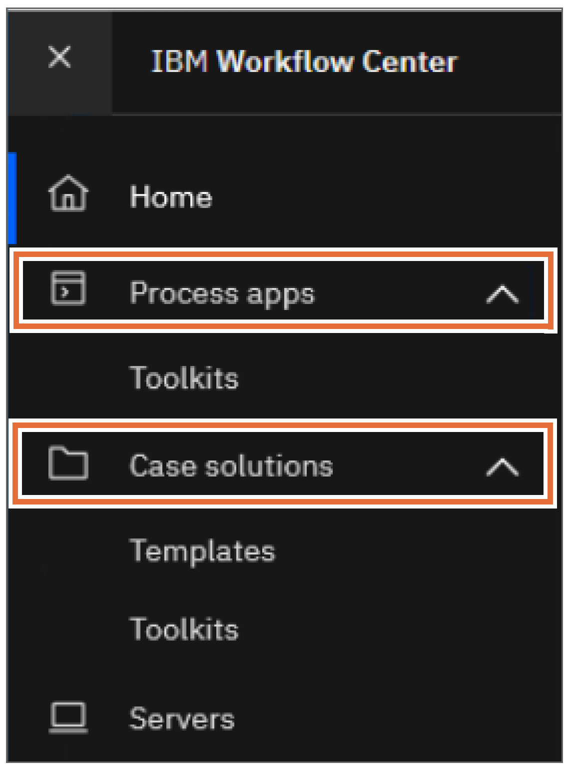
__ 2. Review and set preferences.

When initially logging in to Workflow Center, it defaults to displaying the Basic configuration.

__ a. To display the functions associated with the Basic configuration, click the navigation icon in the upper left.



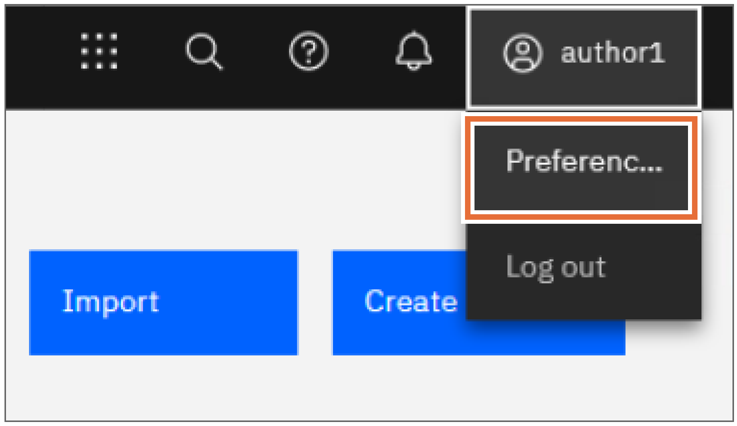

__ b. Expand the **Process apps** and **Case solutions** sections.



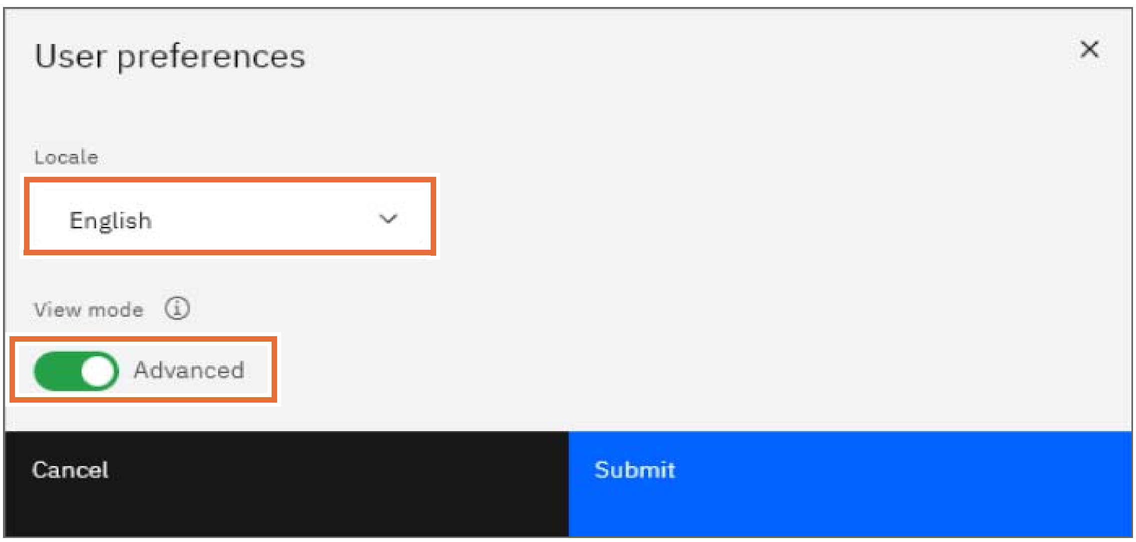

Notice that you can access Process apps, Toolkits, Case solutions and Case Templates. You can also create servers by using the Server tab.

__ c. Click the **X** in the upper left corner to close the navigation bar.

__ d. Open the Preferences settings by clicking **author1 > Preferences**

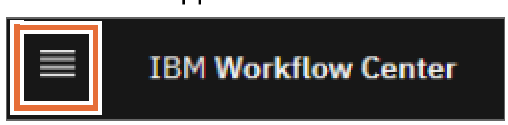



__ e. In the User Preferences dialog box, change the **Locale** to your language and the **View mode** to **Advanced**.

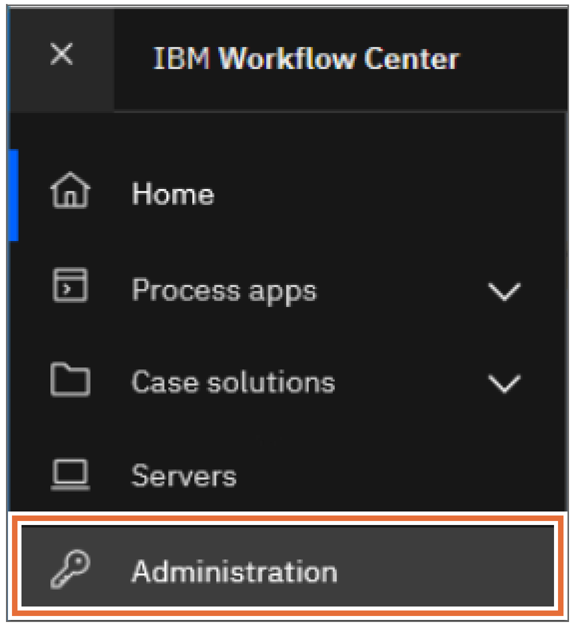



__ f. Click **Submit**

__ g. Click the navigation icon in the upper left.

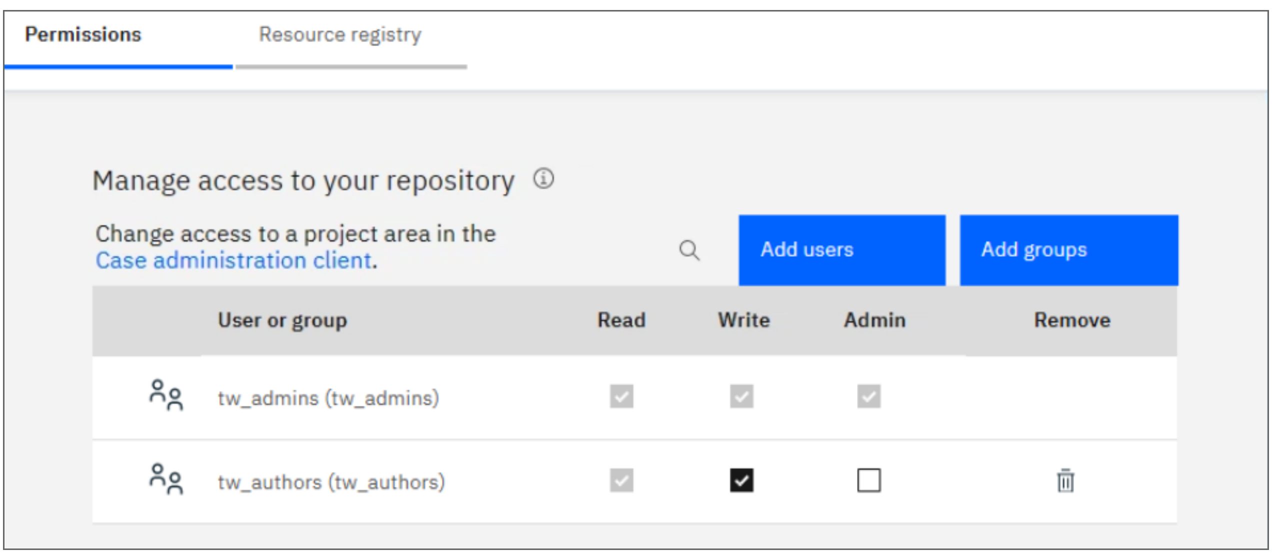


__ h. Select **Administration**



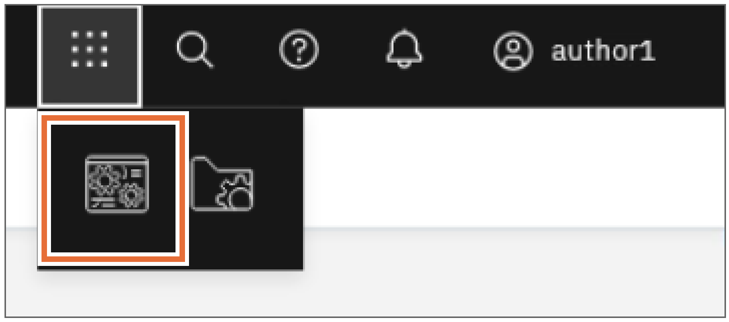

After setting preferences to Advanced, you can now see the Administration option.

You are presented with the Permissions page displaying the default groups for Workflow Center. You can access this page because `author1` is a member of the `tw_authors` group. From this page, you can add users and groups.



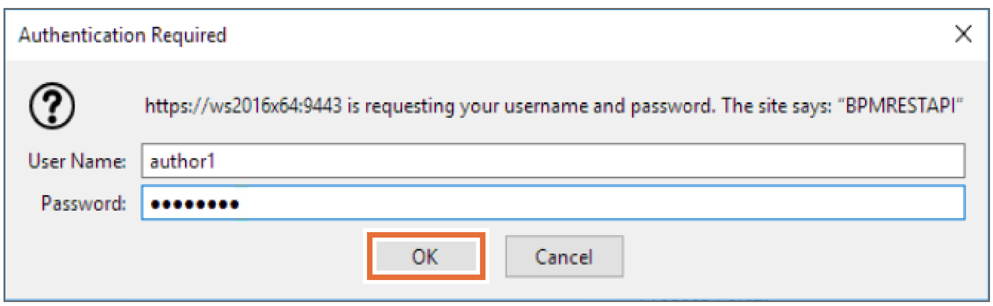
__ i. Click the grid in the menu bar

In case you need to access the Classic Workflow Center, you can click the **Classic Workflow Center** icon




### Troubleshooting

If you leave your workstation for a period of time, BAW logs you out and provides a dialog box to log back in. If this happens, enter your credentials and press **OK**.

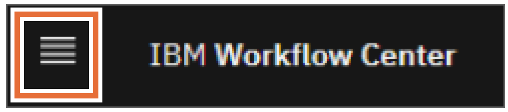


## *Part 3: Create and manage a process application*

In this course, you build a Hiring Request Process application. You create that process application in this section. In subsequent lab exercises, you build all the required artifacts, specifying more details as you progress.

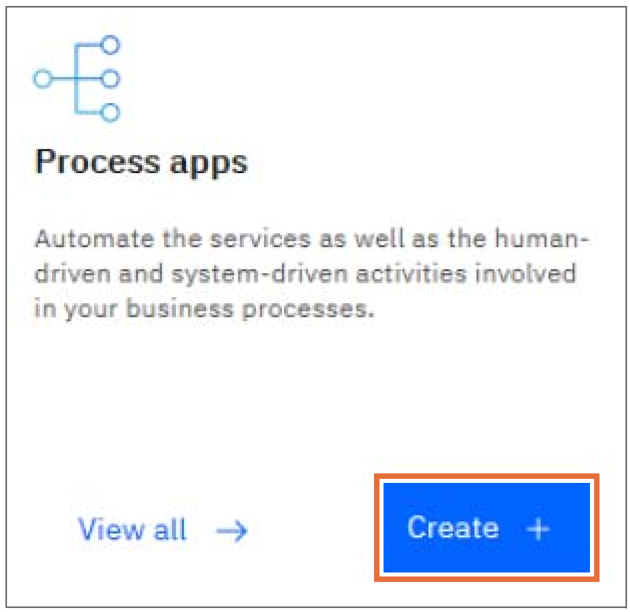
__ 1. Create a process application.

__ a. Click the navigation icon in the upper left.




__ b. Select **Home**.

__ c. Click **Create** in the Process apps tile.




---

**Hint**

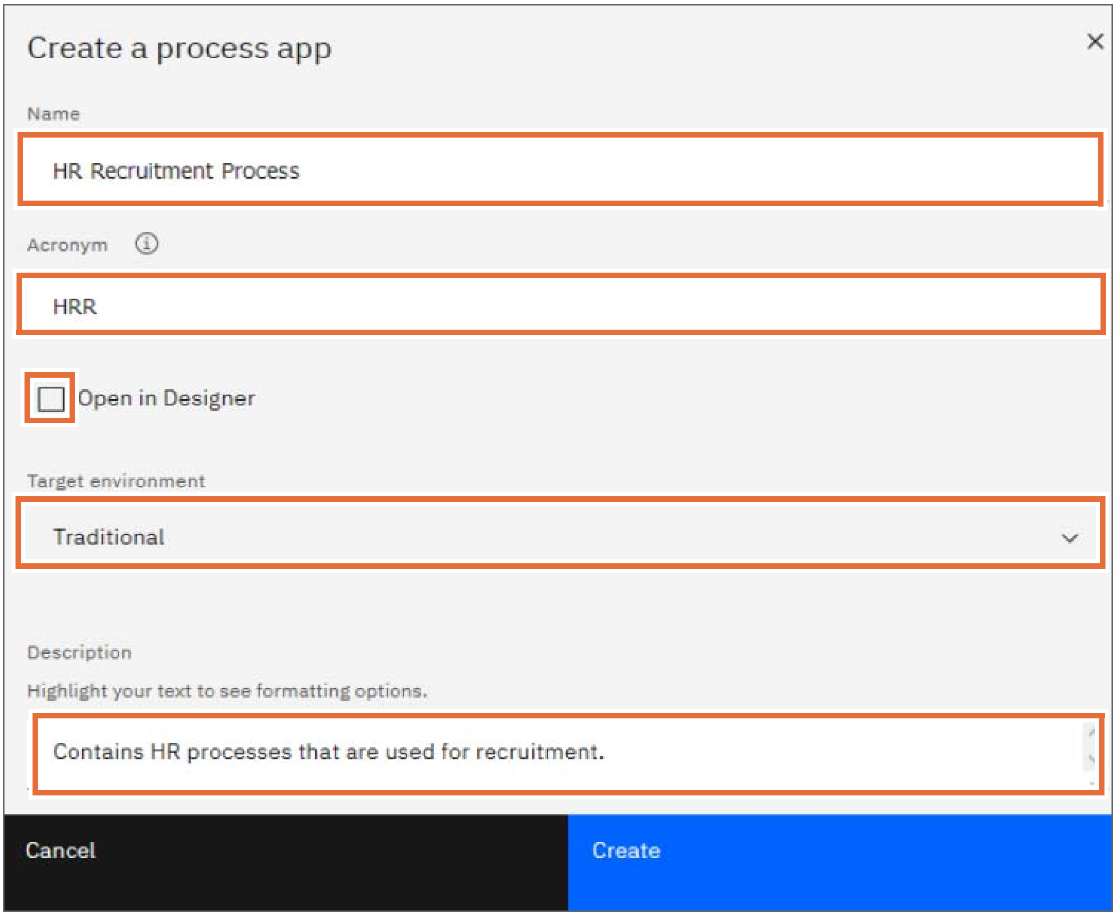
While completing the form, be sure not to click outside of it. Once the focus is taken away, the form is withdrawn. If this happens, click **Create** again to initiate the dialog.

---

__ d. When the **Create a process app** dialog box is displayed, enter the following values:

- **Name**: `HR Recruitment Process`
- **Acronym**: `HRR`
- De-select (clear) the **Open in Process Designer** option
- **Target Environment:** `Traditional`

- **Description**: `Contains HR processes that are used for recruitment.`




__ e. Click **Create**.

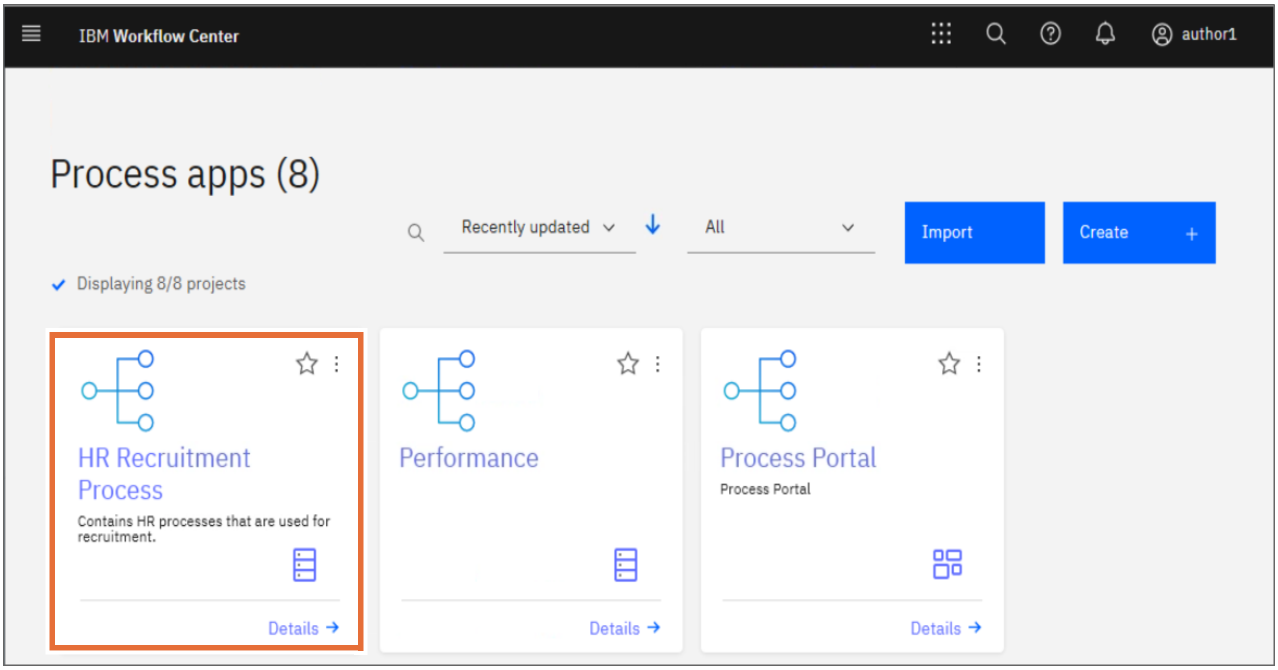
---

### Troubleshooting

If you do not see the Acronym text box, be sure you correctly set the View Mode for the `author1` username to **Advanced**.

---

__ f. Verify that the process application is created and is now in your list of process apps.

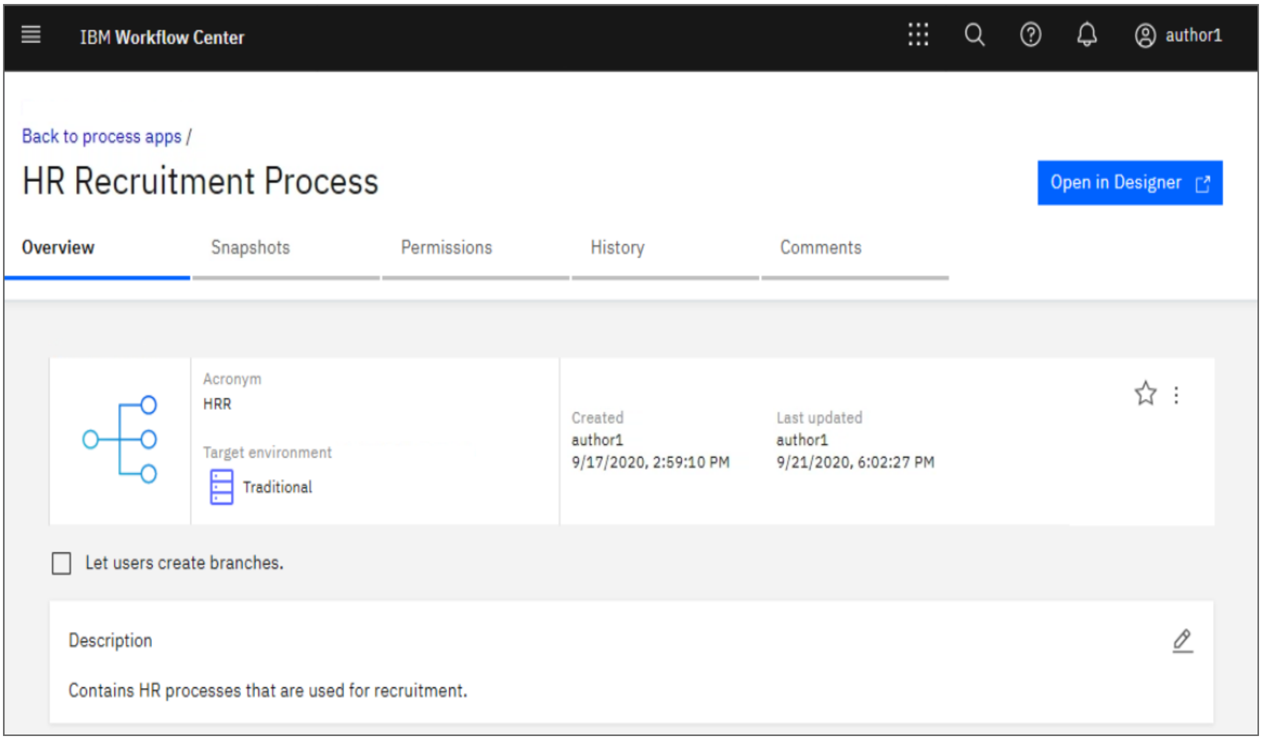



Notice the **Traditional** icon is displayed denoting a Traditional (on-premises) implementation.

__ 2. Manage the HR Recruitment Process process application.

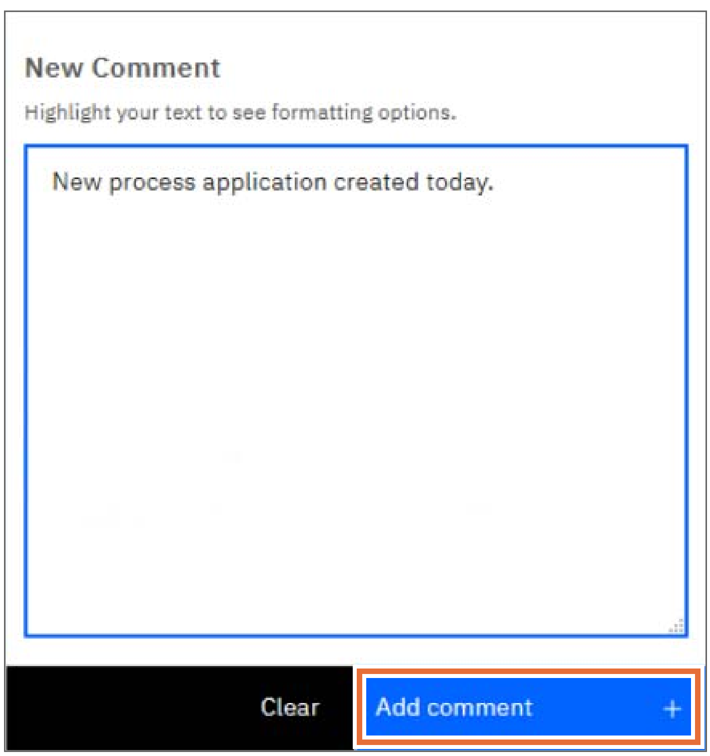
__ a. To view the details of the **HR Recruitment Process** process application, click **Details** on the tile.

__ b. By default, the **Overview** tab is displayed. Other tabs that are displayed include Snapshots, Permissions, History, and Comments. You can create snapshots in Workflow Center or in IBM Process Designer. From this page, you can also change permissions, review the history of the application, and provide comments.




__ c. Click the **Comments** tab and enter a comment.

__ d. Click **Add comment**.



The Comments entry is added to the left with a date and time stamp.

__ e. Review the **History** and **Permissions** tab to become familiar with the information presented. These sections are specific to the process application.

In the next section, you are introduced to tools that are helpful in troubleshooting development issues.

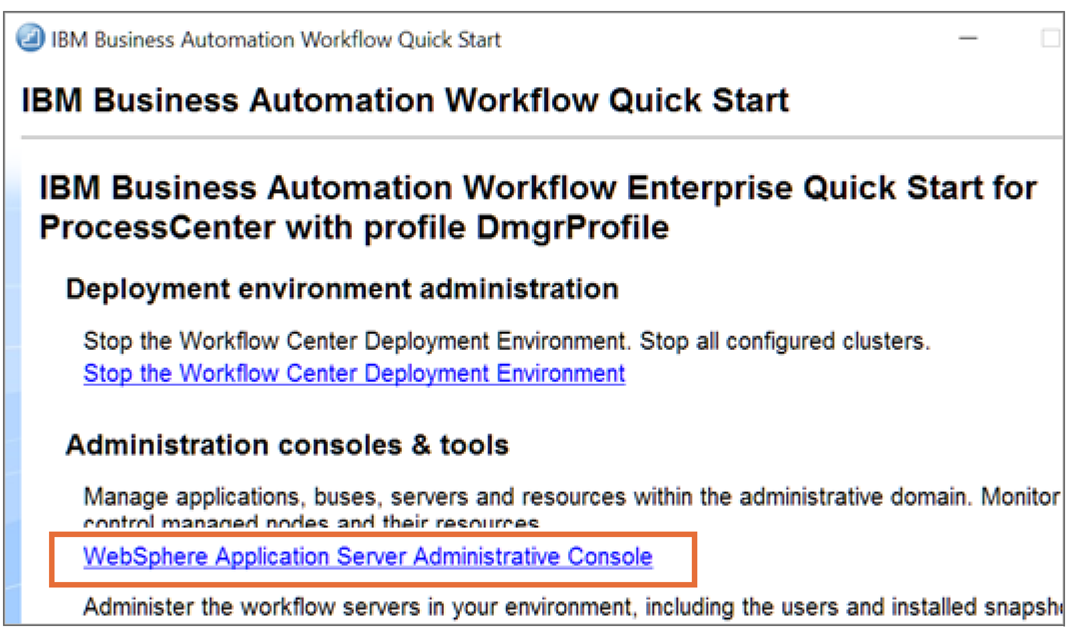
In the final section, you export and import the process application. At the beginning of each exercise, you have the option to import an existing solution file or to continue working with your solution. If you run into issues in your process application, you have the option of importing a working solution to keep progressing in the class.

## *Part 4: Explore other tools*

In addition to working with the Workflow Center console, there are several other tools that one works with when developing applications in IBM Business Automation Workflow. You explore some of those tools in this section. You can use these tools throughout the course to aid in troubleshooting.

__ 1. Explore the Administration console. The Administrative console is used to administer applications, services, and other resources at a cell, node, server, or cluster scope. You can use the console with stand-alone servers and with deployment managers that manage all servers in a cell in a networked environment.

__ a. To access the administration console, go back to the minimized Quick Start window and click the **WebSphere Application Server Administrative Console** link.

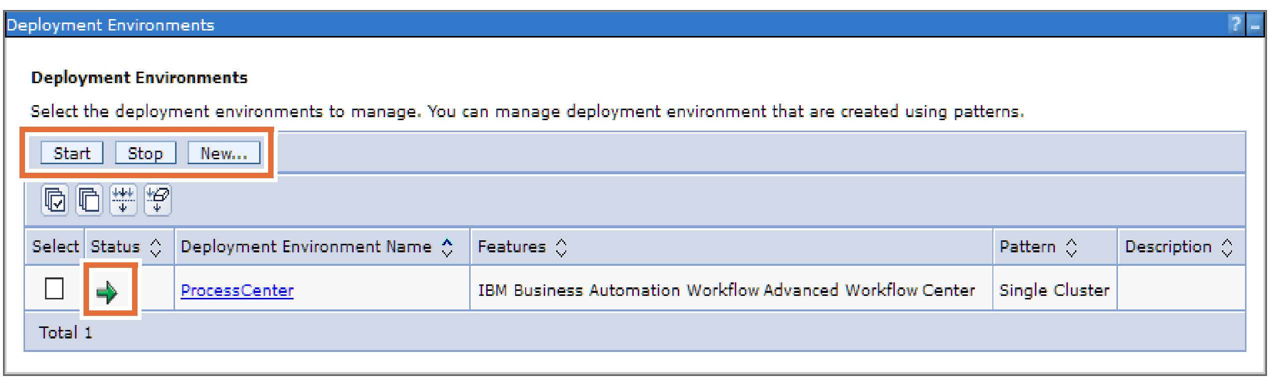


__ b. If prompted, log in by entering `author1` in the **User ID** field and `passw0rd` in the **Password** field. Click **Log In**.


### Troubleshooting

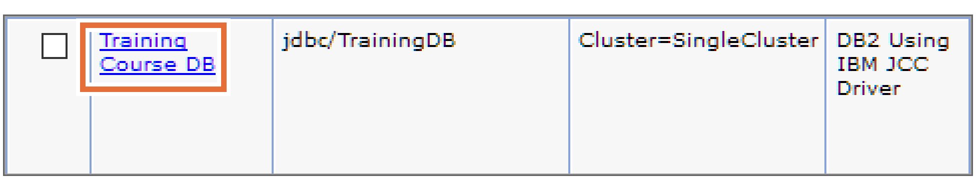
If you receive a message stating another user is currently logged in with the same user ID, leave the default selection (to log out the other user) and click **OK**.

__ c. Expand the Servers panel and click **Deployment Environments**. On this screen, you can start, stop, and create new deployment environments. You can also verify the status of a deployment environment.

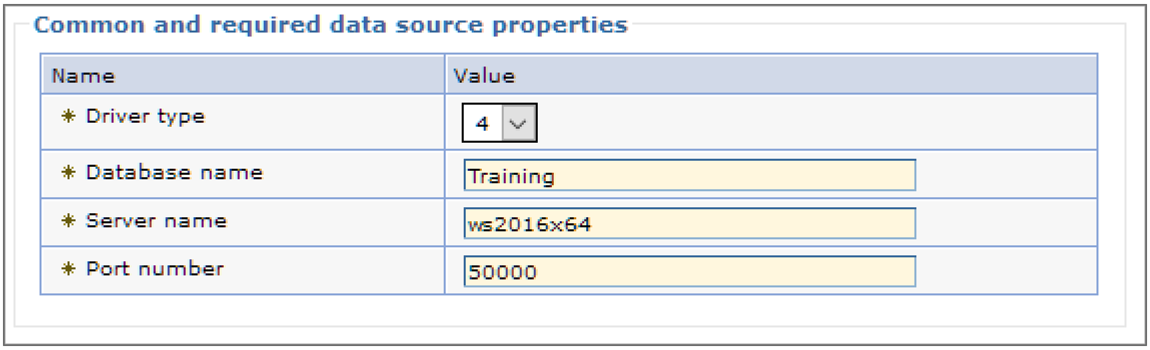



__ d. Expand the **Resources** panel, then expand the **JDBC** section and click **Data sources**.

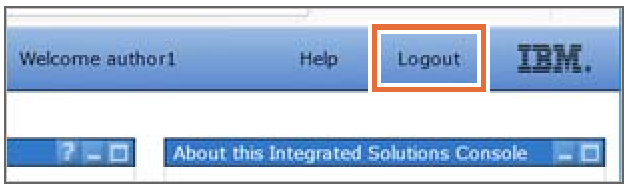
__ e. Scroll down to the bottom and click **Training Course DB**. This is the JDBC driver that is used later in the course to access a DB2 database.



__ f. Scroll down to the bottom of the page and the data source properties are displayed.

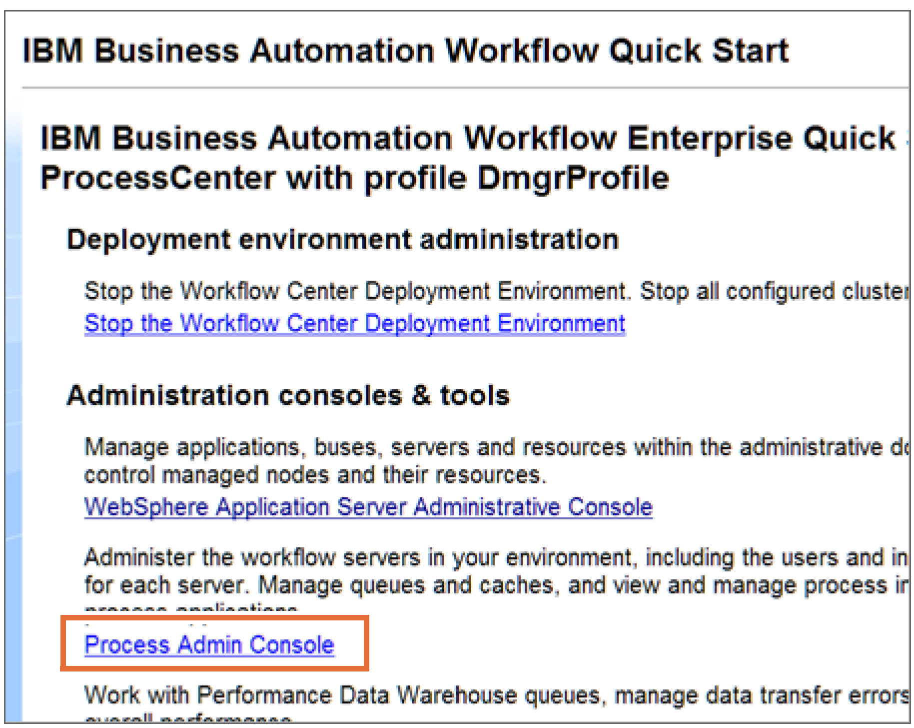



__ g. While logged in, feel free to explore the administrative console further. Optionally, click **Logout** to log out of the console.



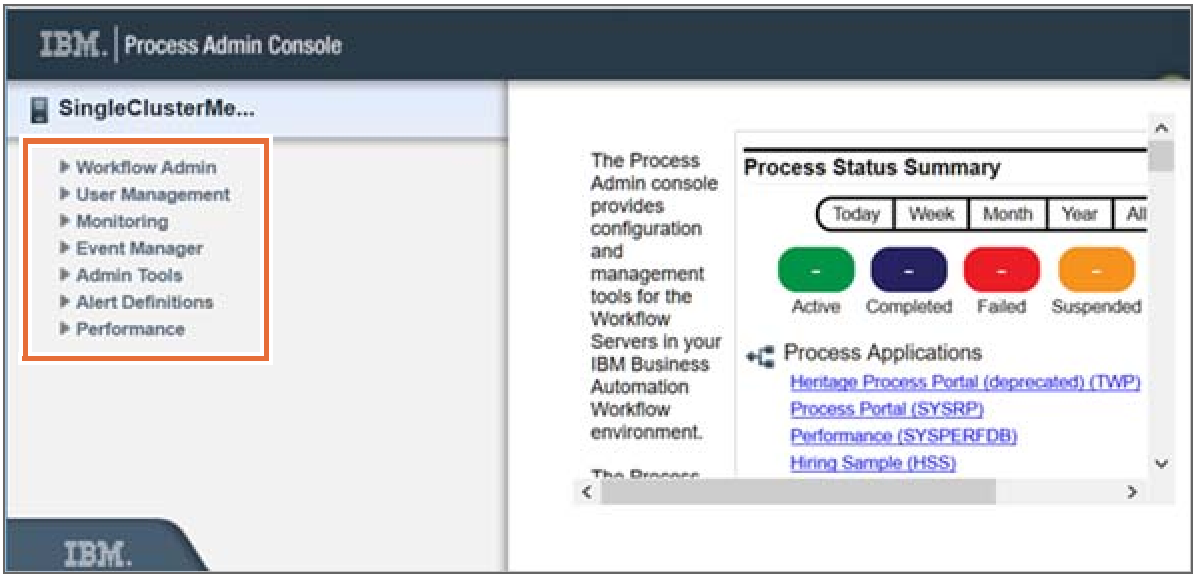

__ 2. Explore the Process Admin console. The Process Admin Console is used to administer the process servers in your environment, including the users and installed snapshots for each server. In addition, it provides tools to help you manage queues and caches.

__ a. To access the administration console, go back to the Quick Start window and click the **Process Admin Console** link.

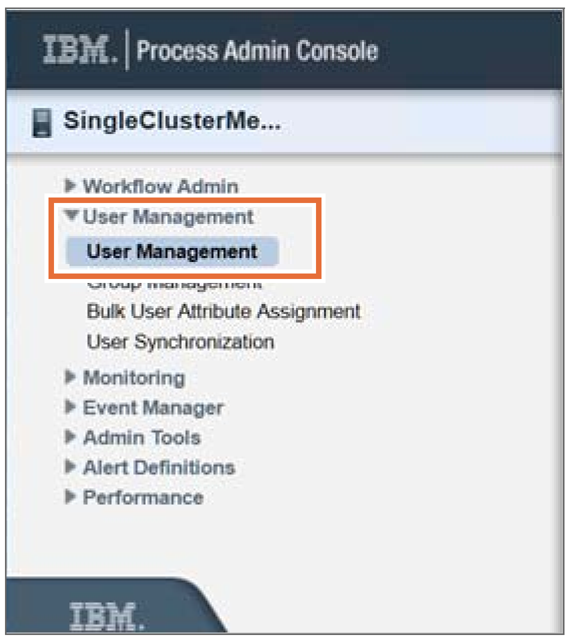



__ b. If you logged out of the Administrative console, then you are prompted to log in to the Process Admin Console. Enter `author1` in the **User Name** field and `passw0rd` in the **Password** field. Click **Log In**.

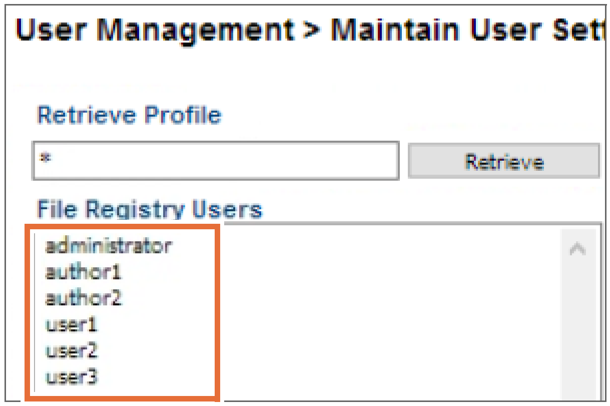
__ c. When the Process Admin console is displayed, note the various links in the left pane.




__ d. Expand **User Management** and then click **User Management**.



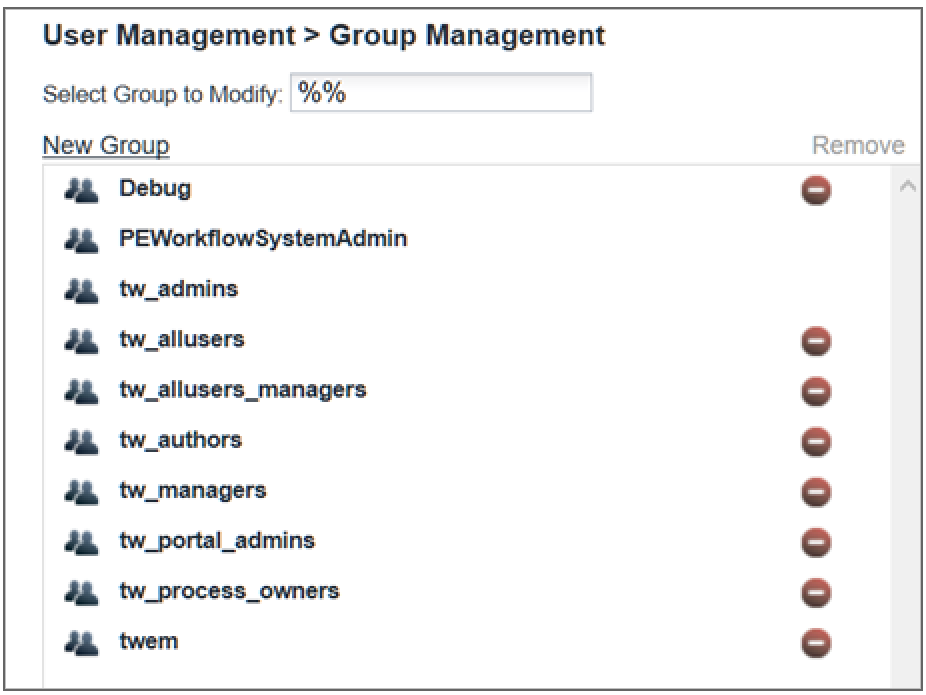
__ e. In the **Retrieve Profile** field, enter * and click **Retrieve** to view a list of users.




__ f. Click **Group Management** in the left menu.

__ g. In the **Select Group to Modify** field, enter %%

__ h. Examine the groups listed. Do not make any changes as it impacts the exercises.

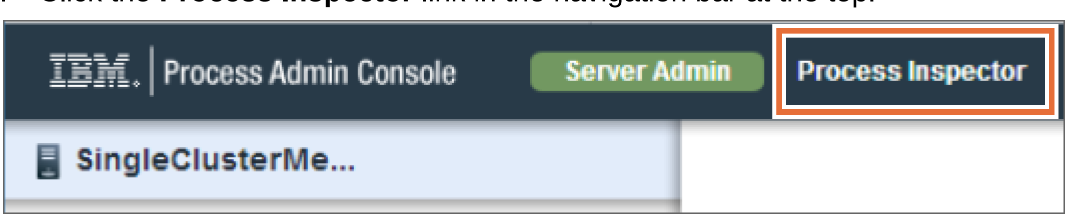



__ i. As time allows, continue to explore the Process Admin Console.

__ 3. Explore the Process Inspector.

Use the Process Inspector to view the task, activity, and data details of your active process instances.

__ a. Click the **Process Inspector** link in the navigation bar at the top.



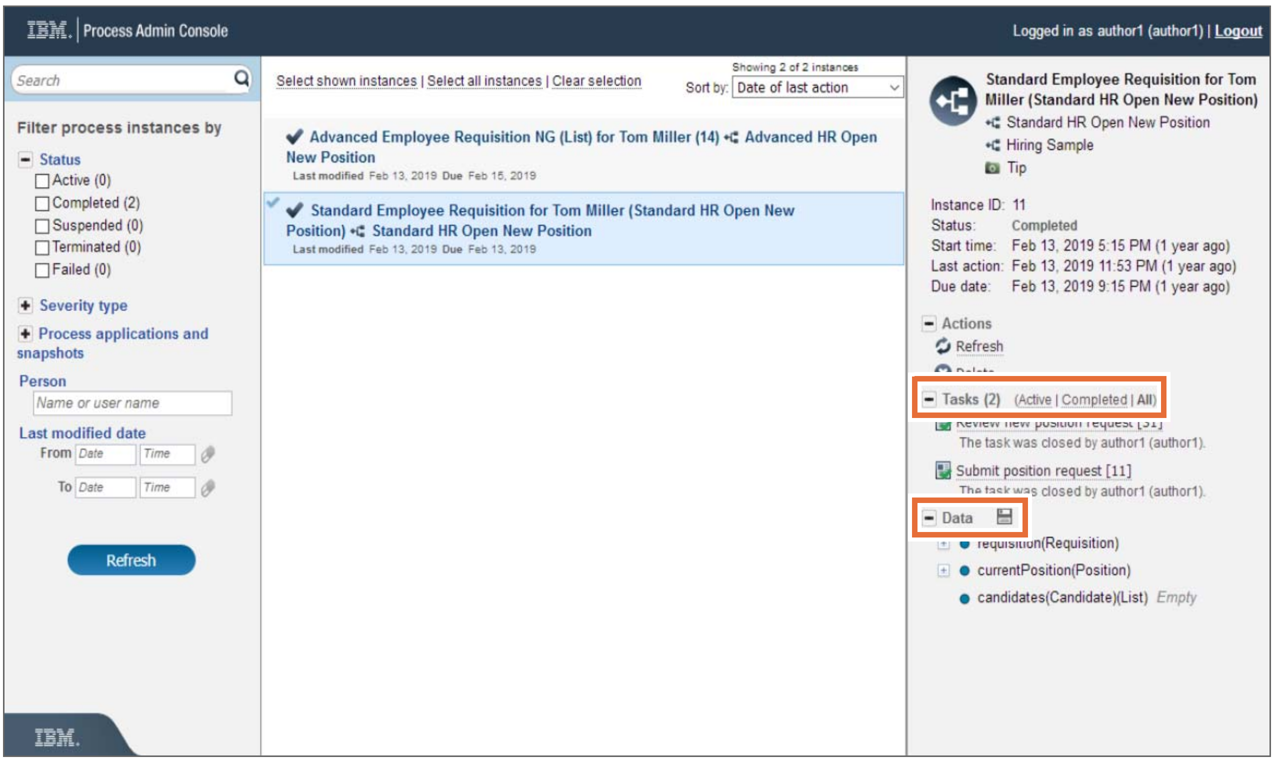
The Process Inspector opens in a new tab. If you run into issues in your development, you can use the Process Inspector to view the status of all running process applications.

__ b. Click **Search**

The **Filter process instances by** section is updated and all process applications in the repository are listed on the right with their status. The Search button changes to Refresh.

__ c. Click the **Standard Employee Requisition for Tom Miller** instance.

__ d. On the right, expand **Tasks** and **Data**.




As you develop your solution, you can use the Process Inspector to inspect the status of active, completed, suspended, terminated, and failed process applications. You can also view the task last executed and the data that is associated with the task.

If you expect to see a task or process display in Process Portal and it does not, you can use the Process Inspector to determine what state the process application is in.

## Part 5: Export the process application

Now that this exercise is complete, you are ready to export the process application. This export saves your application as a `twx` file in a folder for future use. If you wish to keep your solution at the end of the course, you can export it following this procedure and upload it to a web-based file sharing program such as Box.

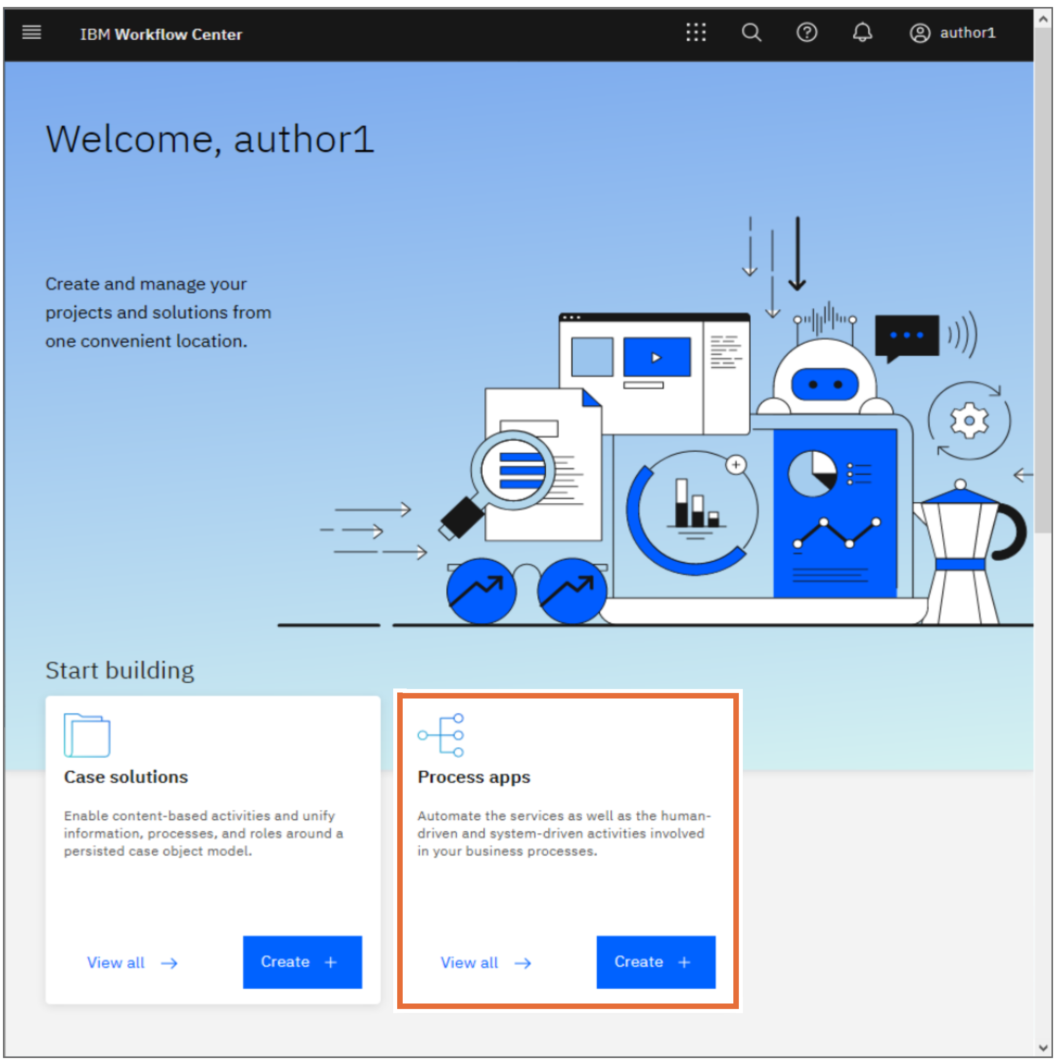
Each exercise in this course builds on the prior one. The progression resembles a real-world implementation of the IBM playback methodology.

There are working solutions for each exercise listed in Appendix A. Solutions. If you run into issues in an exercise, you can export your solution and import the working solution to perform troubleshooting. If you are unable to determine the issue, you can import the exercise solution for the current exercise to continue working. This process helps you to continue working with exercises without any interruption from an unresolved error in your previous exercise.

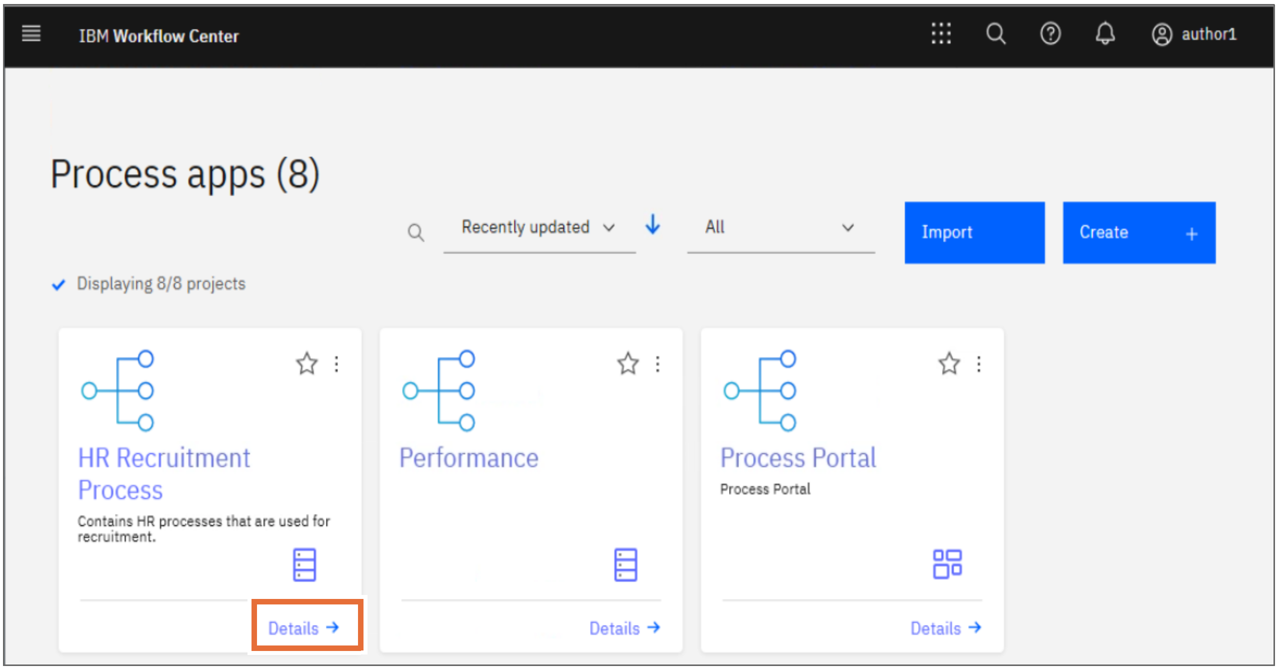
__ 1. Create a snapshot

Only process applications with snapshots can be exported.

__ a. Return to the Workflow Center and click the **Process apps** tile.



__ b. Click **Details** on the HR Recruitment Process tile.

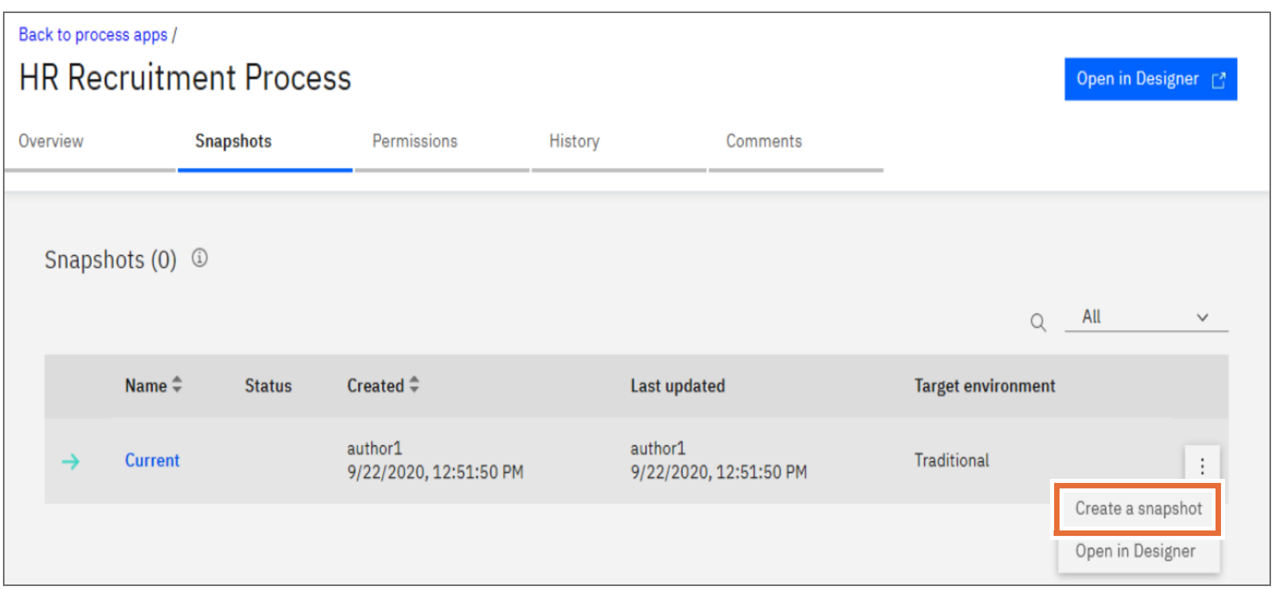



__ c. Select the **Snapshots** tab.

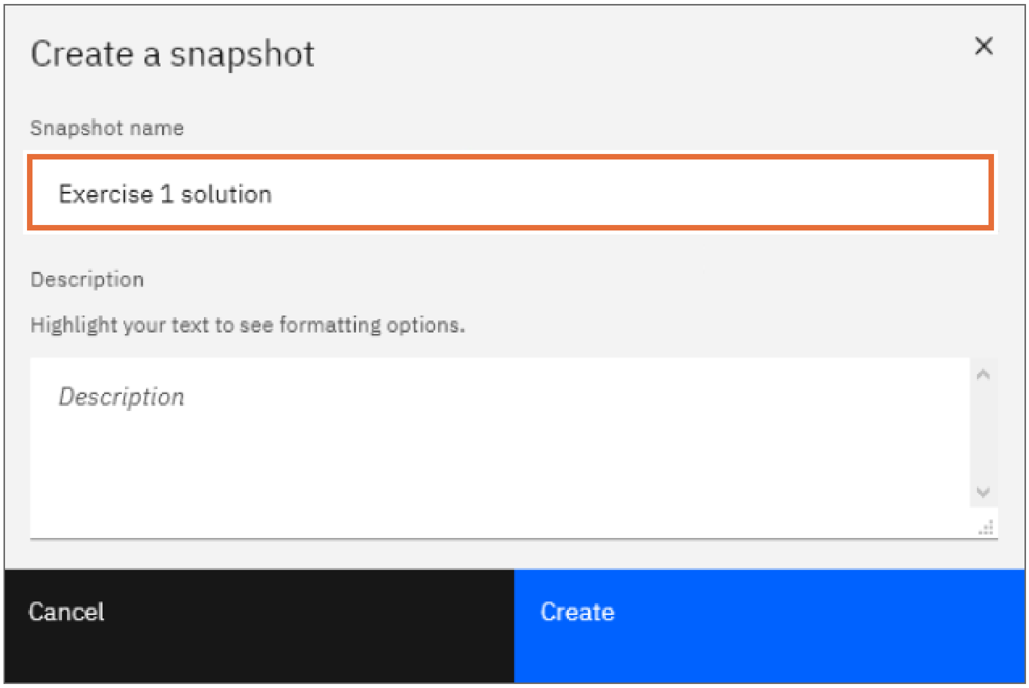
__ d. Open the list of Options by clicking the three dots on right side of the **Current** tip.

The tip is the current working version of a project.

__ e. Select **Create a snapshot**



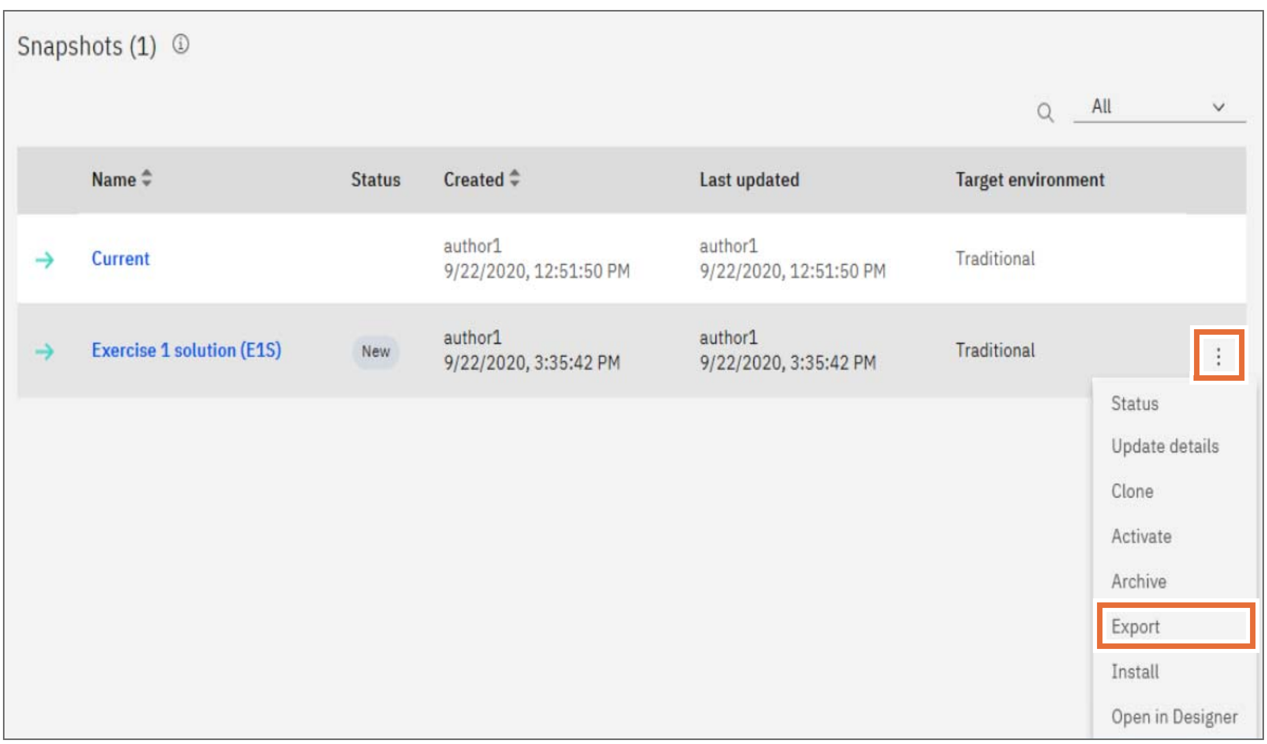
__ f. Enter `Exercise 1 solution` for the name.




__ g. Click **Create**.

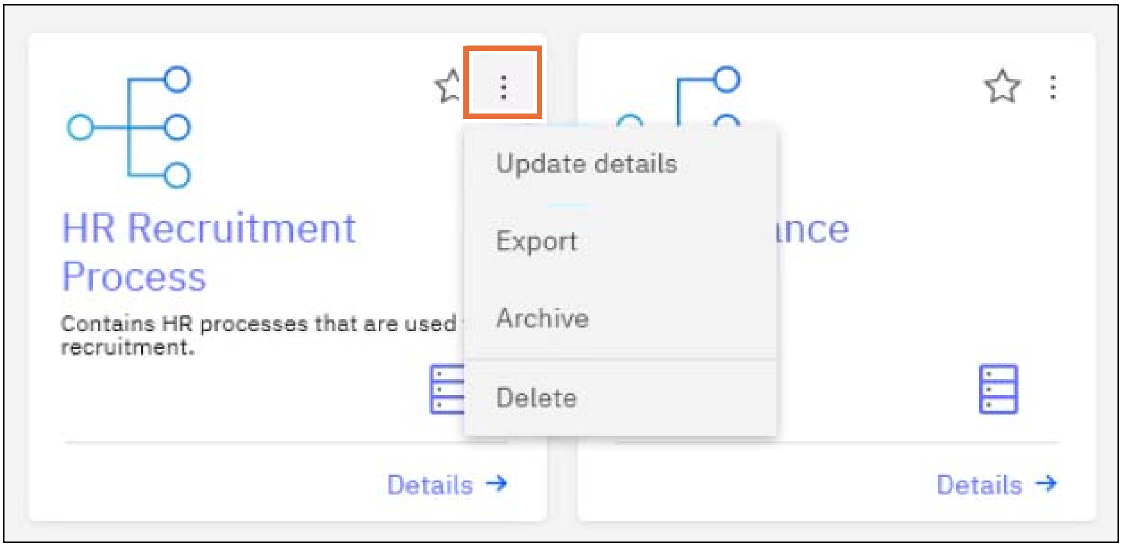
__ 2. Export the application. Exporting the application also exports the toolkit that it depends on. You can only export snapshots. Since there are no associated toolkits yet, only the process application is exported.

__ a. Open the list of options for the snapshot and select **Export**

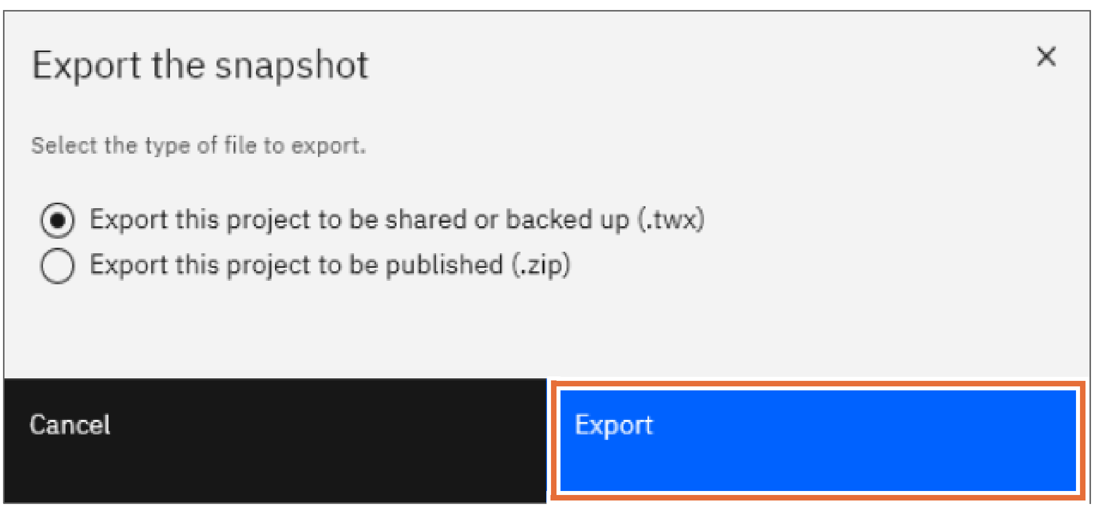


**Hint**

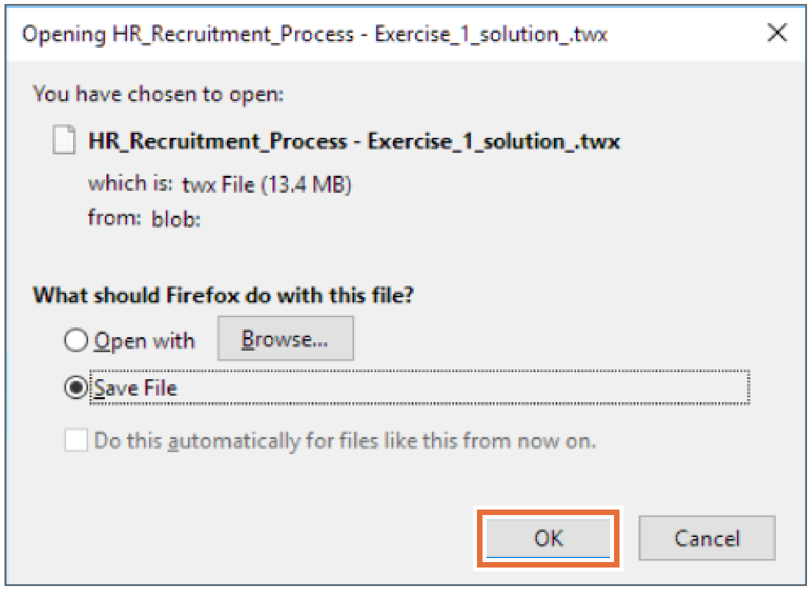
You can also export, archive, and delete process applications directly from the **Process apps** page by accessing the list of options on the tile.




__ b. In the **Export the snapshot** window, click **Export**. It may take a minute or two to generate the `.twx` file.



__ c. Click **OK** to save the file. The application is saved to the default folder, which is the Downloads directory.




__ d. Verify that the application is listed in the downloads directory.

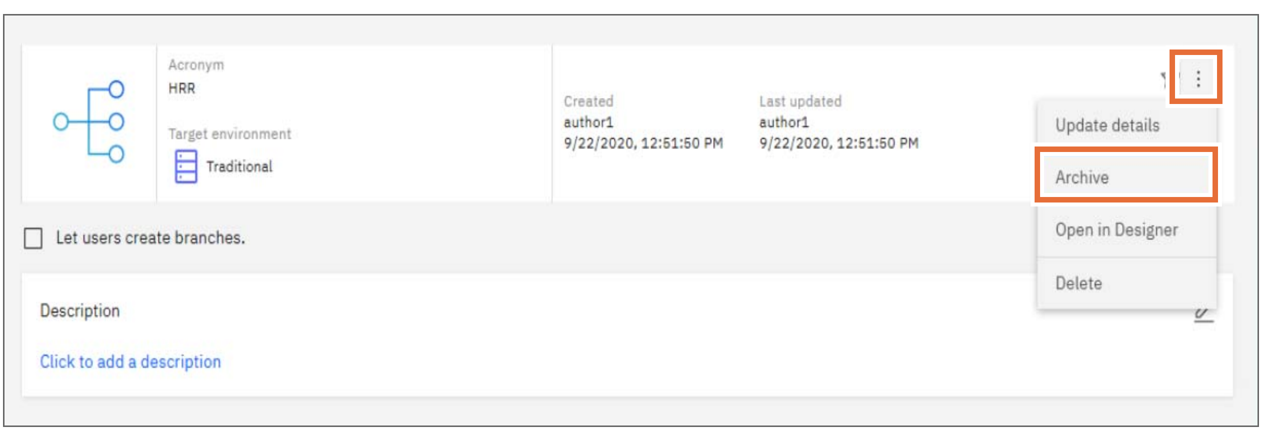
__ 3. Archive the process application and delete it. Although you don't have to archive an application before deleting it, it is a good practice. Archiving an application also archives the toolkit that it depends on. Since there are no associated toolkits yet, only the process application is archived.

**Important**

Don't archive a toolkit that other projects depend on nor use archived toolkits when creating toolkit dependencies in Process Designer. Archived items are not part of the active library.

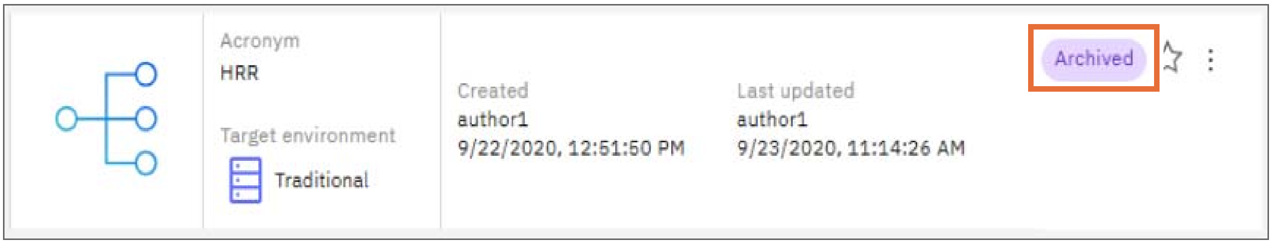
__ a. Return to the HR Recruitment Process application in the Workflow Center and select the **Overview** tab.

__ b. Open the list of options for the application and select **Archive**.

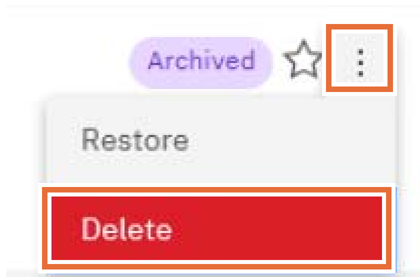


__ c. In the window that opens, click **Archive** to confirm the action. Clicking **Archive** does not delete the application; it removes it from the available **Process Apps** and makes it ready for deletion.

An **Archived** tag appears in the application Overview tile.




__ d. Open the list of options for the application and select **Delete**.




__ e. In the window that opens, click **Delete** to confirm the deletion. The process application is now deleted.

You are returned to the list of Process apps.

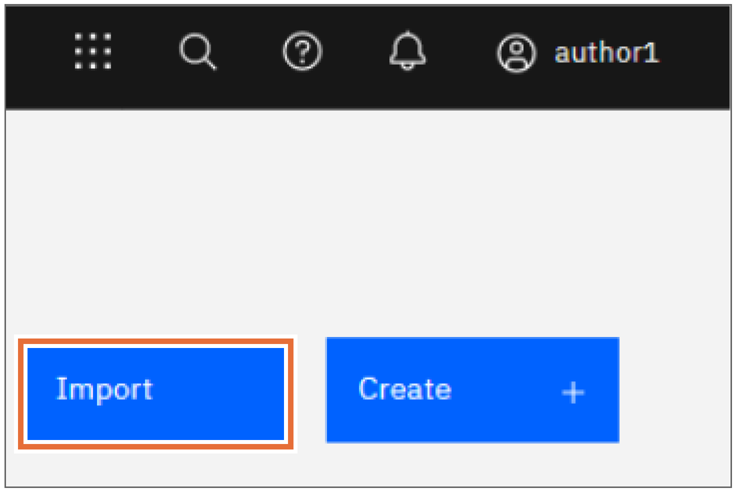
__ f. Verify that **HR Recruitment Process** is no longer listed.

## *Part 6: Import the process application*

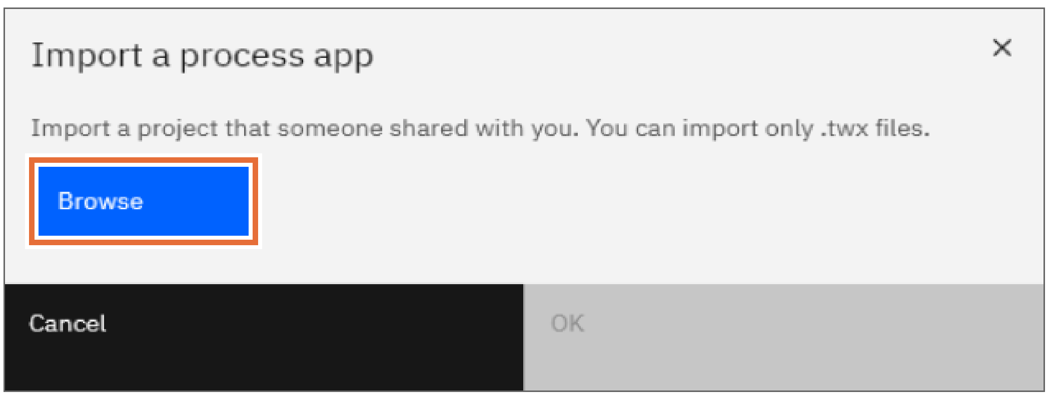
In this section, you import the process application solution that you exported in the previous section. If you perform this import procedure in the beginning of an exercise by using the solution for the prior exercise, this ensures that you have a good copy of the process application at the start of the exercise. If you run into issues in your exercise, you can import the working solution. This process helps you to continue working with exercises without any interruption from an unresolved error in your previous exercise.

__ 1. Import the process application.

__ a. On the list of Process apps, click **Import**.




__ b. Click **Browse**.



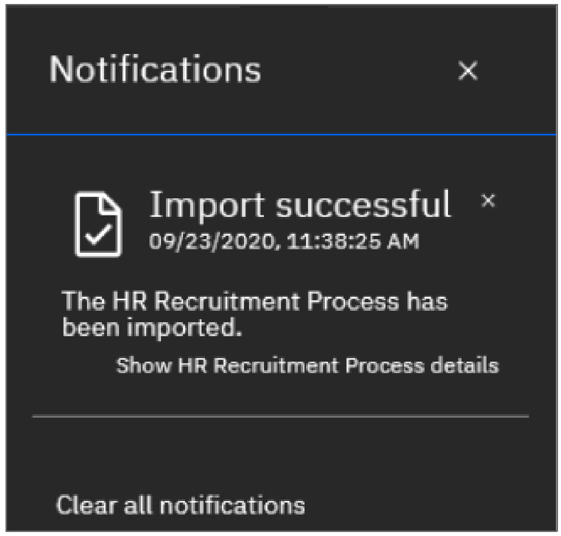

__ c. Navigate to the **Downloads** folder and select the file:

`HR_Recruitment_Process - Exercise_1_solution.twx`

__ d. Click **Open**.

__ e. Click **OK**.

__ f. The Notifications window displays the message



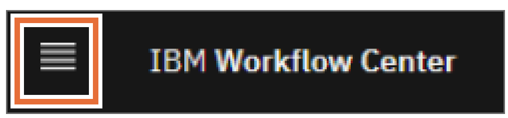

__ g. Close the **Notifications** window by clicking the **X** in the upper right corner.

__ h. Verify that the **HR Recruitment Process** application is listed.

__ 2. Let users update the toolkit.

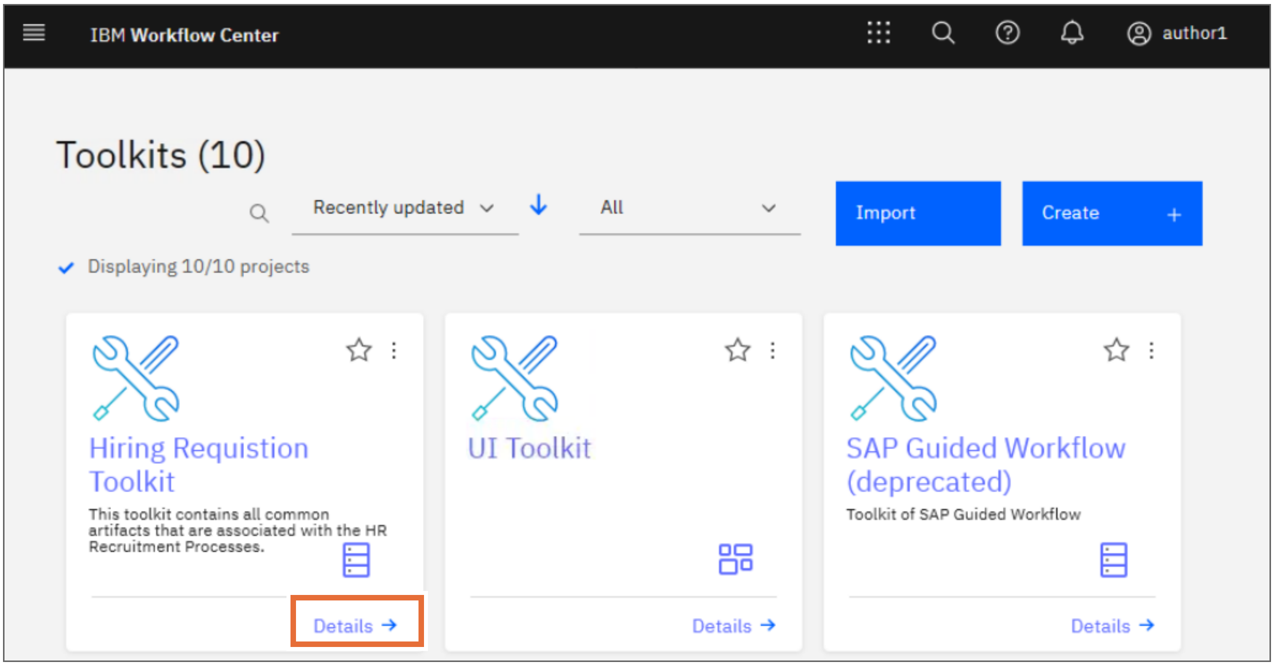
This step is only pertinent after Exercise 7 when you start using toolkits. When importing a toolkit, it does not default to let others use it.

__ a. Click the navigation icon in the upper left.

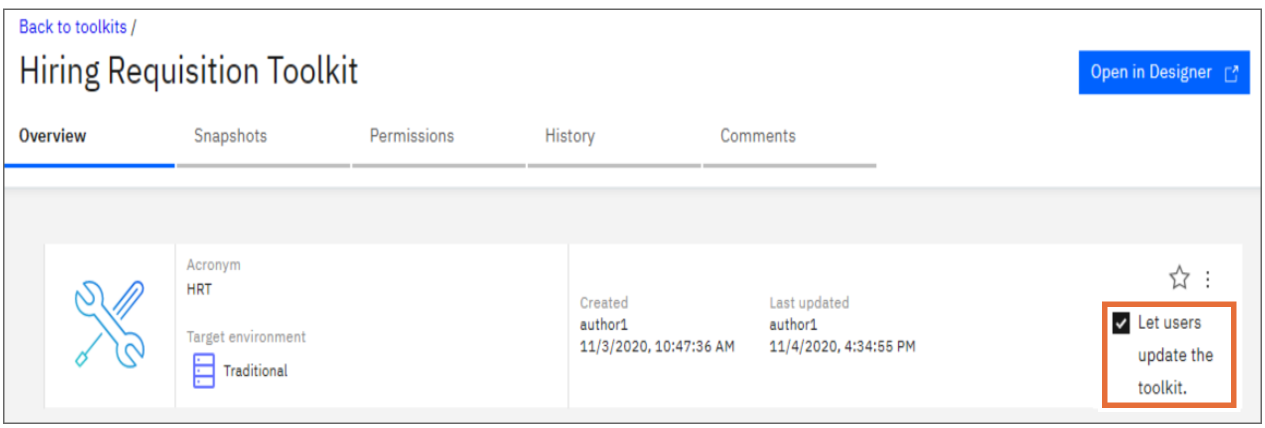



__ b. Select **Toolkits**.

__ c. Click **Details** on the Hiring Requisition Toolkit.




__ d. Verify the box is checked next to **Let users update the toolkit**.




You have completed this exercise.

You are now ready to start building your process application. Each exercise shows step-by-step instructions that you can follow to complete the tasks. In IBM Business Automation Workflow, you find many different ways to complete modeling tasks. The step-by-step instructions show one way to accomplish these tasks in the exercises.

## End of exercise

# Exercise 2. Playback 0: Creating the To-Be process

## Estimated time

02:00

## Overview

This exercise covers how to create a process by using IBM Process Designer. The Hiring Requisition process owner provided detailed information about the process and its current state to the BPM analyst, who in turn documented the information. This step completed the process discovery and initial analysis, and now the process model can be created. To accomplish the task of creating the initial process model, you model it with a pool, lanes, and flow objects such as activities, and events, and nested processes. You take the information that is provided for the use case and translate that into a process. Your first task is to create a process and name it according to naming conventions. You then add the activities in the appropriate lanes and use sequence flow to connect the activities. Be sure to model the happy path (critical path) first. You also complete decomposition on your process and create a linked process where you see the opportunity.

## Objectives

After completing this exercise, you should be able to:

- Translate business process workflow steps that are documented in the process discovery and analysis into process model tasks
- Create the foundation for a process by adding the appropriate lanes to the default pool
- Model the expected process flow for the initial process model
- Decompose business process workflow steps that are documented in the process discovery and analysis into process model tasks
- Create a linked process

## Introduction

In the previous exercise, you created a process application. In this exercise, you model the process to be used for the application.

In this exercise, you create a to-be process model. To accomplish the task of creating the initial process model, you define a pool, lanes, and flow objects such as activities, events, and a linked process.

In this exercise, you add activities to the appropriate lanes and use sequence flow to connect the activities. As mentioned in the unit, you model the happy path first. Gateways and various flows are presented in the next exercise.

Finally, you create a subprocess and link it to the main process.

This exercise is long, so a break is included in the middle of the instructions.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

# Exercise instructions

## Hiring Requisition process

The company expanded from the simple (as-is) hiring process to a more detailed (to-be) hiring process. The detailed process now includes more information. The process that you are going to examine and model is called the Hiring Requisition Process. The company also wants to reduce rework by introducing a more structured process for the Hiring Requisition process. This process covers a new job position through submission, approval, and completion so applicants can apply for the job position.

## Core requirements

- 1.1: A Hiring Manager submits a hiring requisition to the HR Department. The request contains the following information:
  - Customer details:
    - Requisition number
    - Date of request
    - Requester
    - Date position available
    - Job title
    - Job description
    - Job level
    - Number of people managed
    - Division
    - Department
    - Salary to offer
    - Bonus amount
    - Hiring Manager comments
    - New position
- 2.1: If the answer to “New position” is Yes, the request is forwarded to a General Manager. After the General Manager receives the request, the General Manager indicates approval or disapproval.
- 2.2: If the request is not approved, the General Manager specifies a reason and the request is closed. If the request is approved, a salary compliance check is conducted.
- 2.3: The Hiring Manager is notified of the General Manager’s decision after the General Manager approval step.
- 2.4: After the requisition is submitted, an automated system checks for salary compliance. If the request meets salary compliance, the hiring request is automatically posted to the HR Positions database and made available for dissemination.

- 2.5: When a request violates the established salary guidelines of the company, the HR Administrator can approve or reject the requested salary override.
- 2.6: If the salary override is approved, the request is posted to the HR Positions database and made available for dissemination.
- 2.7: If the HR Administrator rejects the requested salary, the HR Administrator must provide comments for the violation, add a proposed salary, and send the request back to the Hiring Manager who originated the request.
- 2.8: When the Hiring Manager gets the request back because of a rejection, the Hiring Manager can negotiate an adjusted salary or can cancel the request. If the negotiation is successful, the request is resubmitted back to the same HR Administrator.
- 2.9: All hiring requests must be added to the HR Positions database regardless of the disposition at the end of the process during a finalization activity.
- 2.10: The HR Administrator has 4 hours to complete the review. If the review is not completed within 4 hours, an email is sent to the HR Administrator. The email notifies the HR Administrator of the missed deadline.

The process owner provides detailed information about the process and its current state to the IBM Process Analyst, who in turn documents the information and analyzes the process for improvement. The process discovery and initial analysis are already completed for you to proceed with the creation of the to-be process model.

## *Part 1: Create a process*

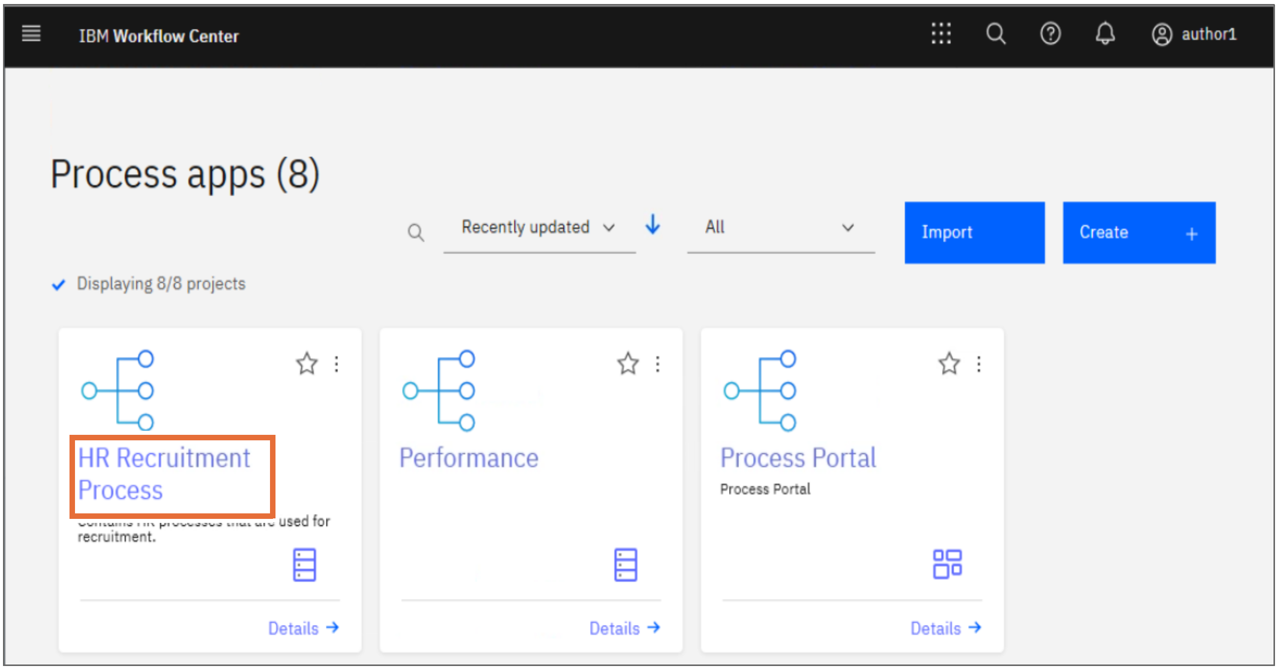
To add the process model to the HR Recruitment Process application you created in the last exercise, return to the Process Designer window.

A process defines the activities that are needed to resolve a specific business problem. A process also defines who works on these activities and the steps they take to resolve the problem. At run time, a business user works with an instance of a process.

__ 1. Create a process.

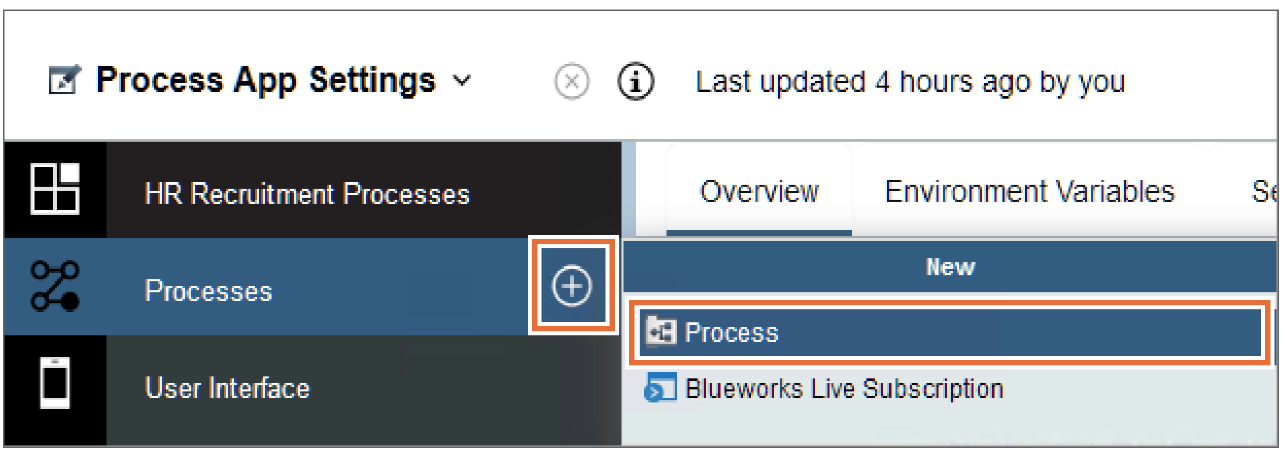
__ a. From the Workflow Center Home page, click **Process Apps**.

__ b. Click **HR Recruitment Process**



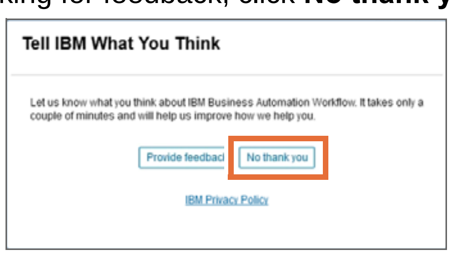

The process application is opened in Process Designer.

__ c. From the process library on the left, click the **(+) plus sign** next to Processes and select **Process** from the list.




**Information**

If you see a pop up window asking for feedback, click **No thank you**.

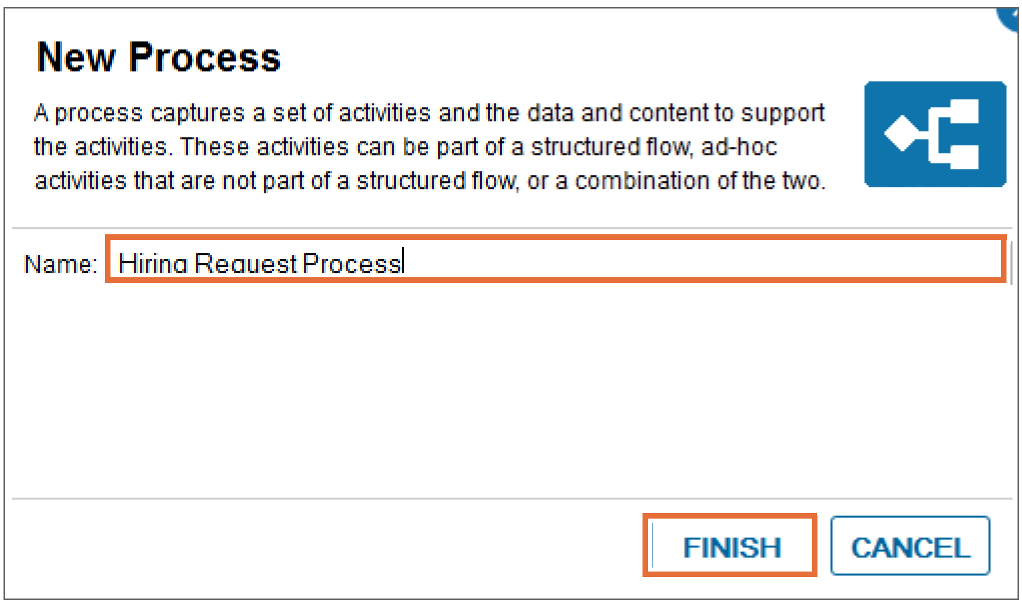


**Note**

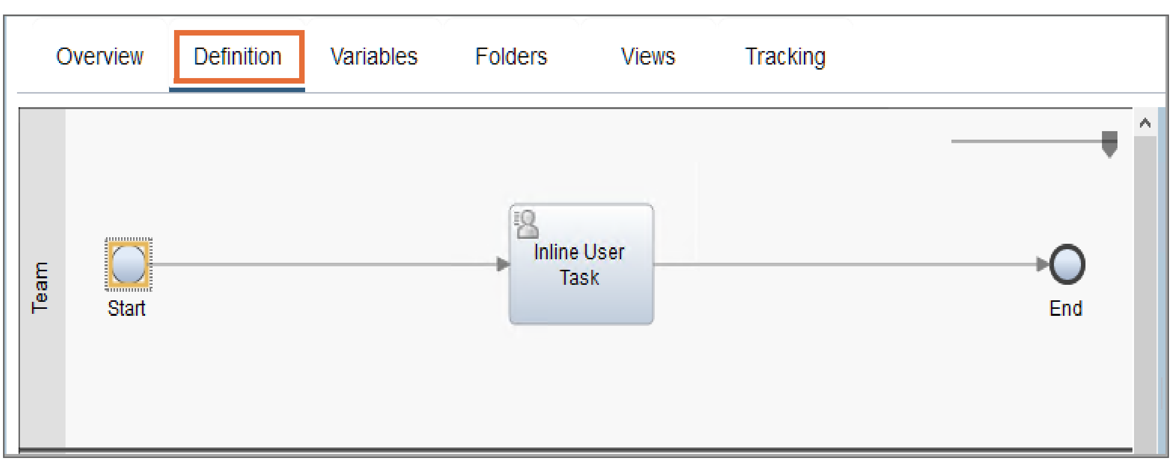
When developing web-based user interfaces, it is important to keep in mind the different ways in which different browsers render JavaScript. The screen that you see might vary slightly from the screen captures depending on how the JavaScript is being rendered in the browser. For instance, in the Chrome browser, it places a box around the highlighted tab.


Although the development environment might vary slightly, the most important thing to keep in mind is how the user interface is displayed in the browser to be used with the solution.

__ d. Enter `Hiring Request Process` in the **Name** field. Click **FINISH**.



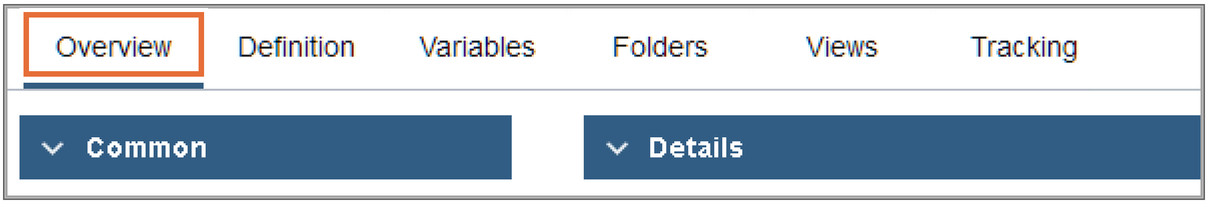

The process is created, and the initial model is shown under the **Definition** tab. By default you have a process with a team lane, a system lane, a start event, an inline user task, and an end event.



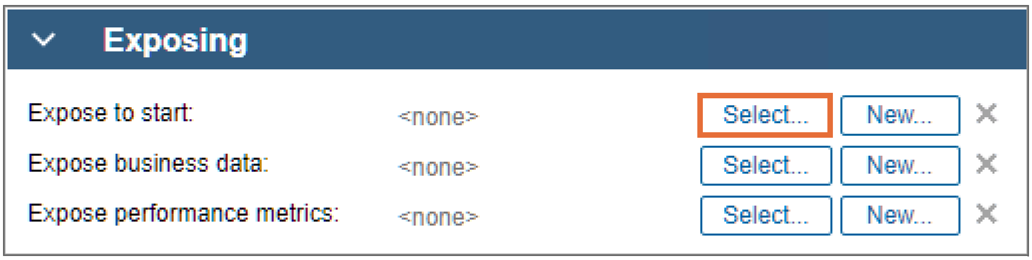
__ 2. Expose the process to **All Users** so that all the users can start the process.

For now, you assign all users to the Hiring Request process. In a later unit and lab exercise in this course, you learn how to define and assign a team and users to your process.

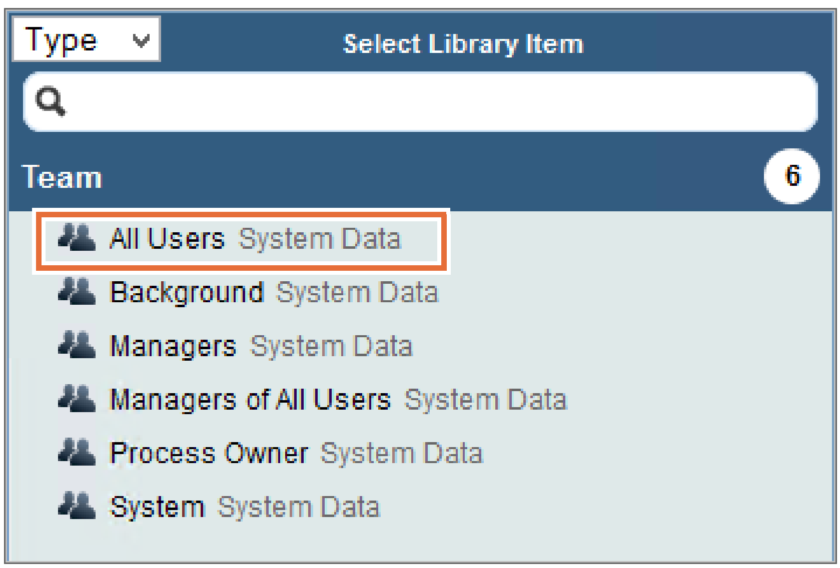
__ a. Click the **Overview** tab.




__ b. Under the Exposing section on the right, click **Select** for the **Expose to start** option.




__ c. Select **All Users**.




__ d. Click **Finish Editing** to save your changes.


**Hint**

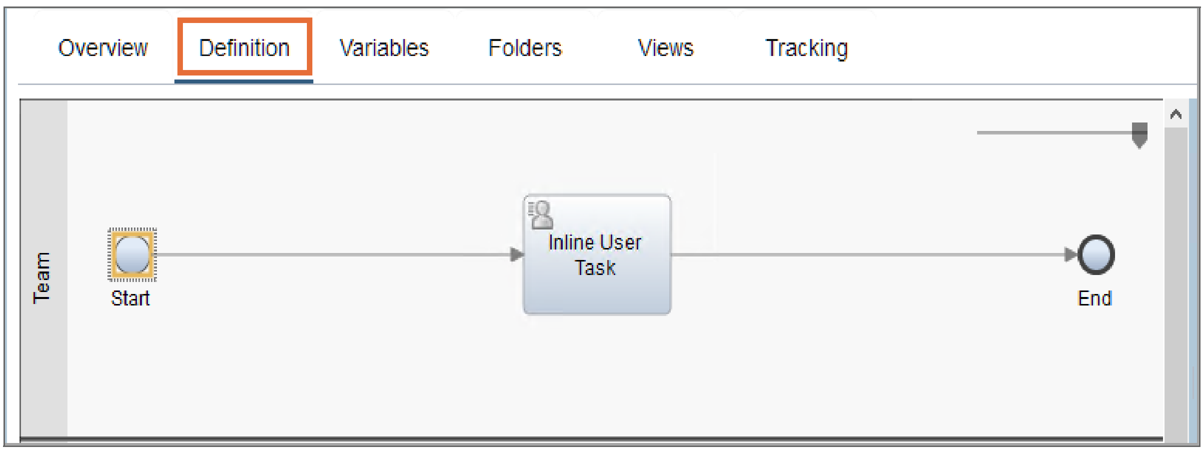
A number appears next to the check mark corresponding to the number of artifacts with unsaved changes. You can also use **Ctrl+S** to save your work.

## Part 2: Create process activities

To add activities and events to the process model, use the process requirements that are provided at the beginning of this exercise. Read the main process requirements and determine the activities from the requirements. Because the requirements were written down, it might be easier to write down the activities on paper before proceeding. Remember, activities use a verb-noun naming convention. If you read the requirements carefully, notice that the main process is described from item 2.1 through item 2.4. In the process requirements, four activities are in the main process: Submit Hiring Request, Approve New Hire Request, Approve Hire Request, and Complete Hire Request.

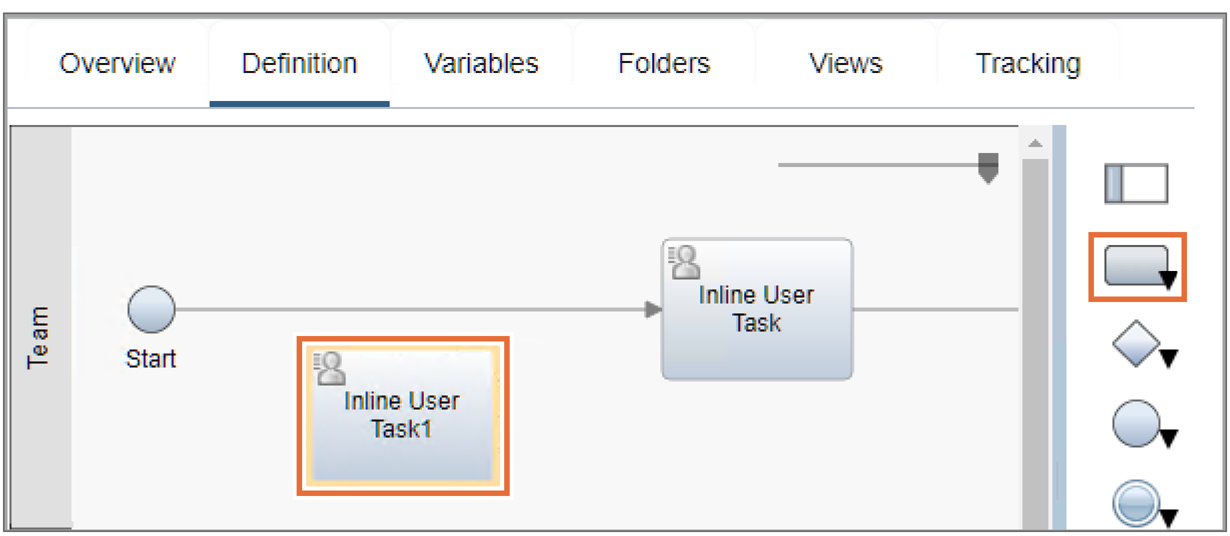
__ 1. Create an activity that is named **Submit Hiring Request**.

__ a. Click the **Definition** tab to reopen the model.




__ b. While holding down the left mouse button, drag an **Activity** from the palette on the right side of the canvas into the Team lane to the right of the start event. When you drag an object to the canvas, the cursor turns into a green plus when you can drop it.

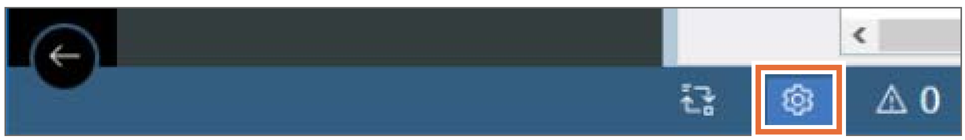
__ c. Release the left mouse button to place the activity on the canvas to the left of Inline User Task.



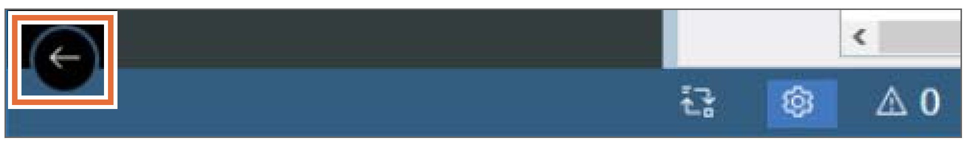
**Hint**

As you model more artifacts, you can increase the size of the canvas by dragging the double lines on the line separating the model definition from the properties section.

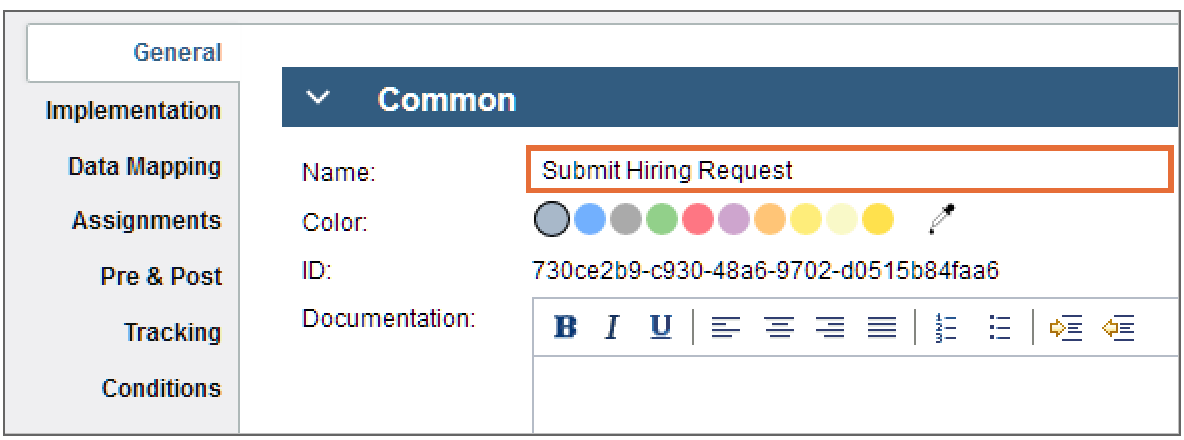

Alternatively, you can toggle the Properties section to display by selecting the Properties icon in the bottom status bar of Process Designer.




To open and close the Process Library, click the arrow at the bottom left.




__ d. While **Inline User Task1** is selected, rename it to `Submit Hiring Request` under the **General > Common** properties section.

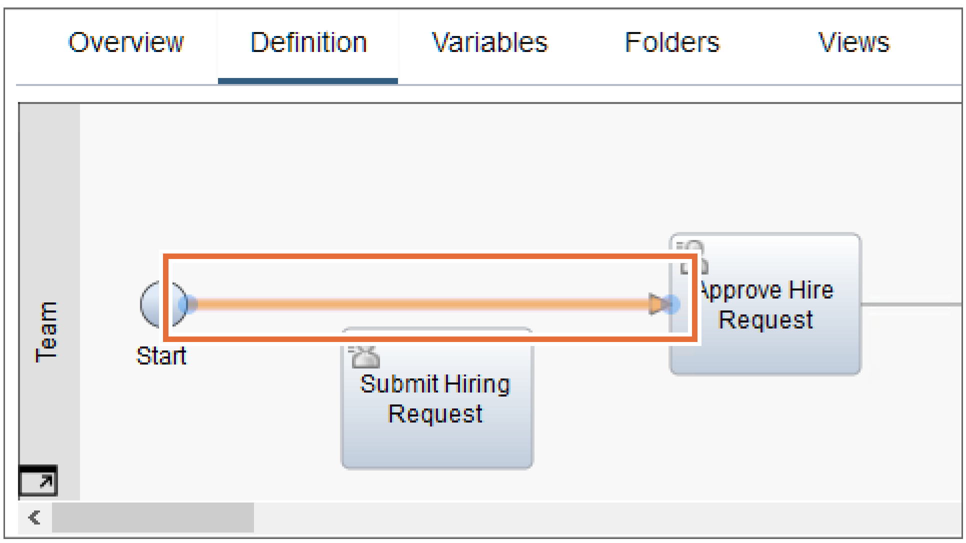



__ 2. Create an activity that is named **Approve Hire Request**.

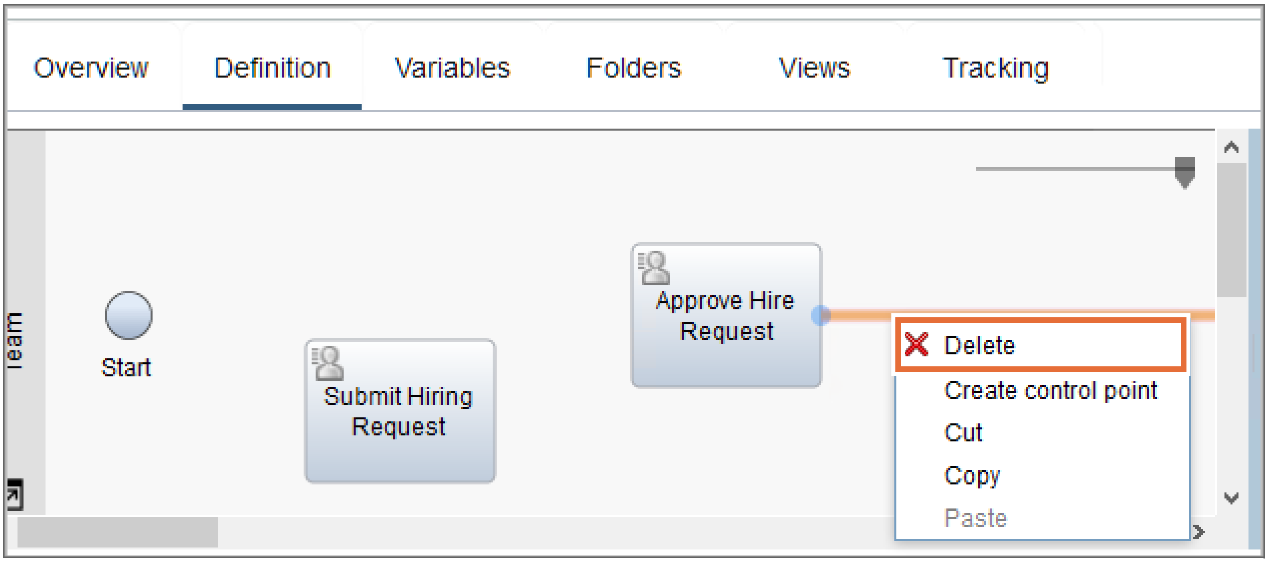
__ a. Click the **Inline User Task** to the right of the Submit Hiring Request activity.

__ b. While highlighted, click the text on the activity to rename it as `Approve Hire Request` and press the **Enter** key to apply the name change.

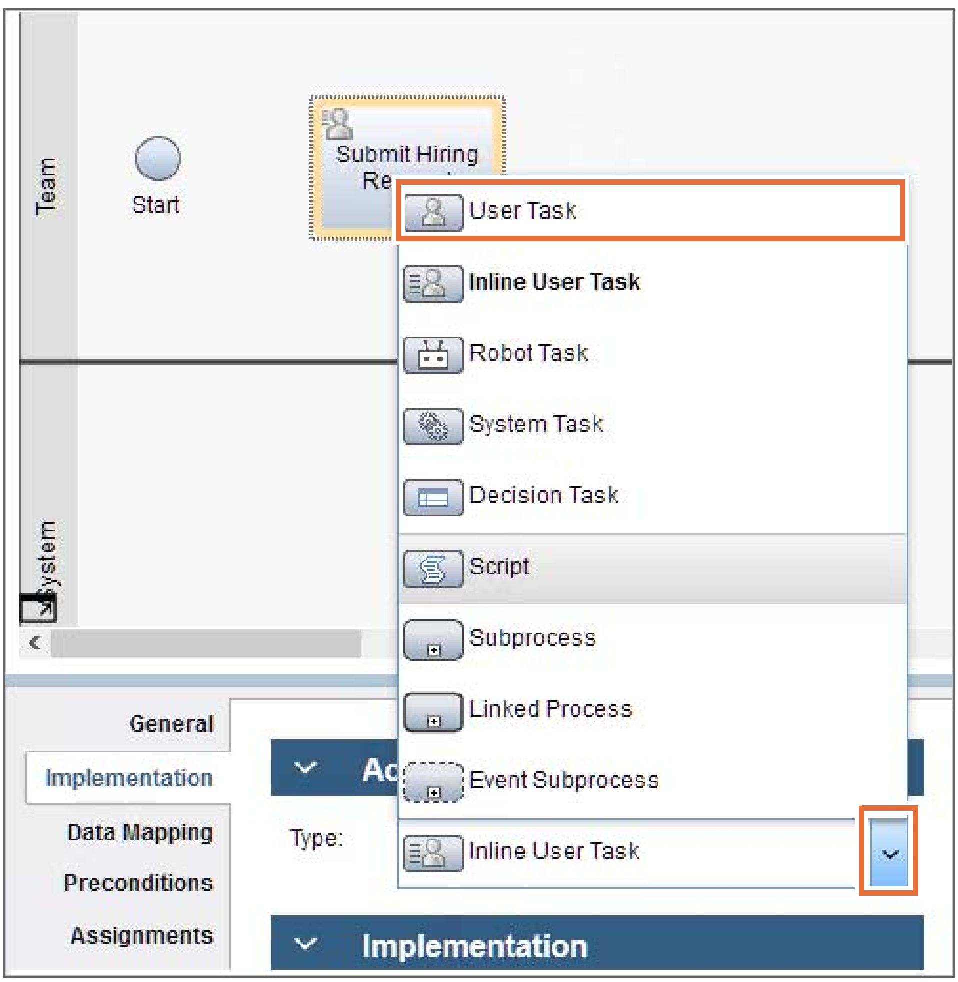

__ c. Click the **flow line** between the Start event and the Approve Hire Request activity. Press the **Delete** key on your keyboard to delete the flow line.



__ d. Click the **flow line** between the Approve Hire Request and the End Event. This time, right-click the flow line and click **Delete.**

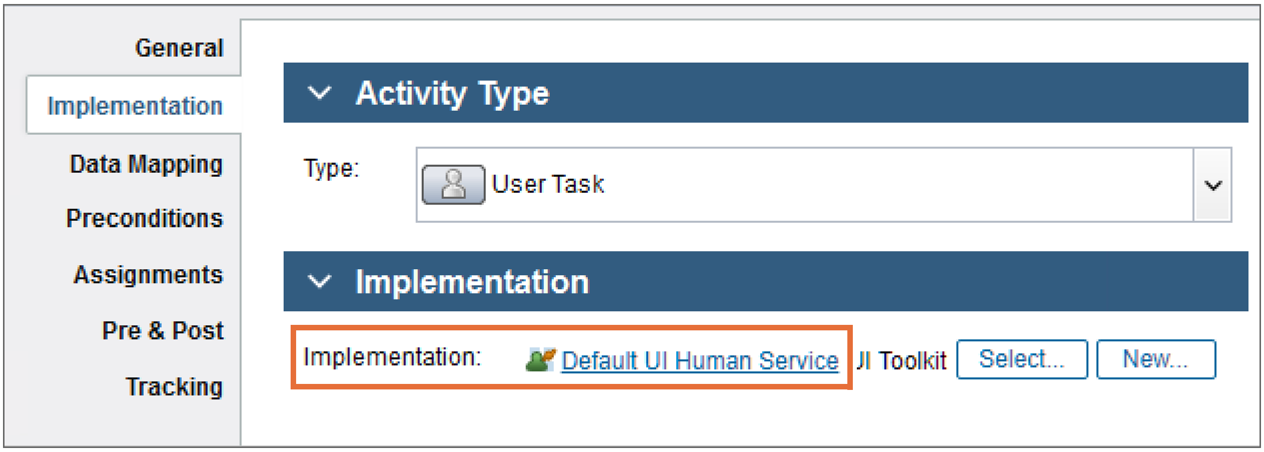



__ e. Later in the course, you build custom user interfaces for these tasks. Click the **Submit Hiring Request** activity. In the properties panel at the bottom, go to the **Implementation > Activity Type** menu, expand **Type**, and change the setting to **User Task**.



__ f. Under the **Implementation** section, click the **Select** button and select **Default UI Human Service**.

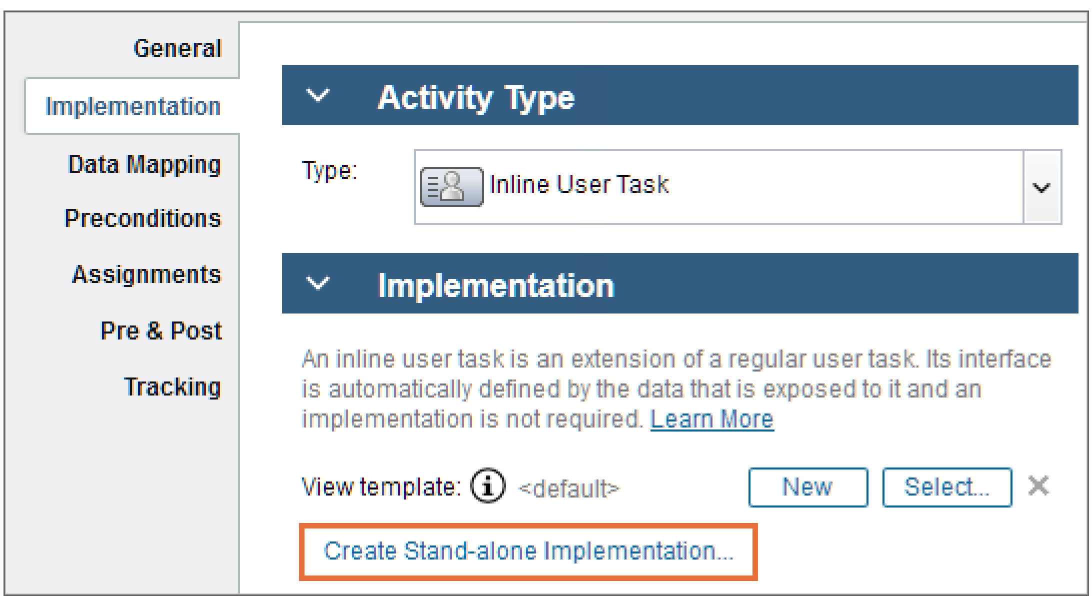
__ g. Verify that the **Default UI Human Service** implements your User Task.



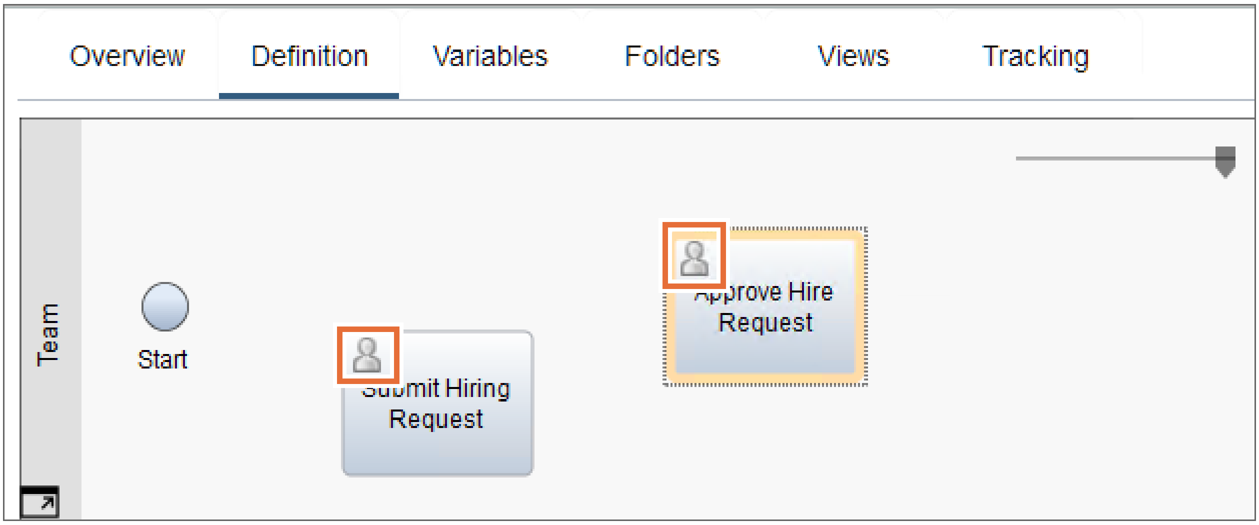

**Information**

Alternatively, you could continue to use the inline user tasks. An inline user task is an extension of a regular user task and automatically exposes the data that is mapped to the user task. Inline user tasks can be used to run the process immediately after an IBM Blueworks Live import.

You also have the option to extend the inline user task by creating a stand-alone implementation. For the purposes of this training, you use the default UI human service until you create the user interfaces in a later exercise.

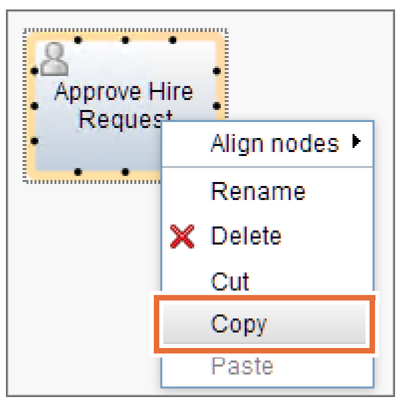


__ h. Click the **Approve Hire Request** activity. In the properties panel below, go to the **Implementation > Activity Type** menu, change this activity to **User Task.** Both activities now have the same icon.




__ 3. Create an activity that is named **Approve New Hire Request**.

__ a. Right-click the **Approve Hire Request** task and select **Copy**.

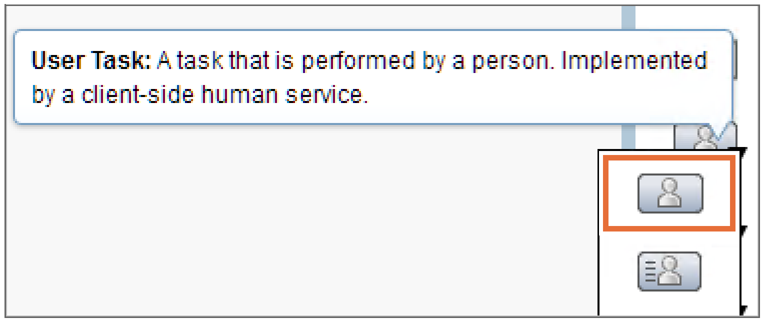



__ b. Right-click on the canvas and select **Paste**.


__ c. Rename the task to: **Approve New Hire Request.**

__ 4. Create an activity named **Complete Hire Request**.

__ a. Expand the activity icon in the palette by clicking the down arrow and drag a **User Task** to the canvas.



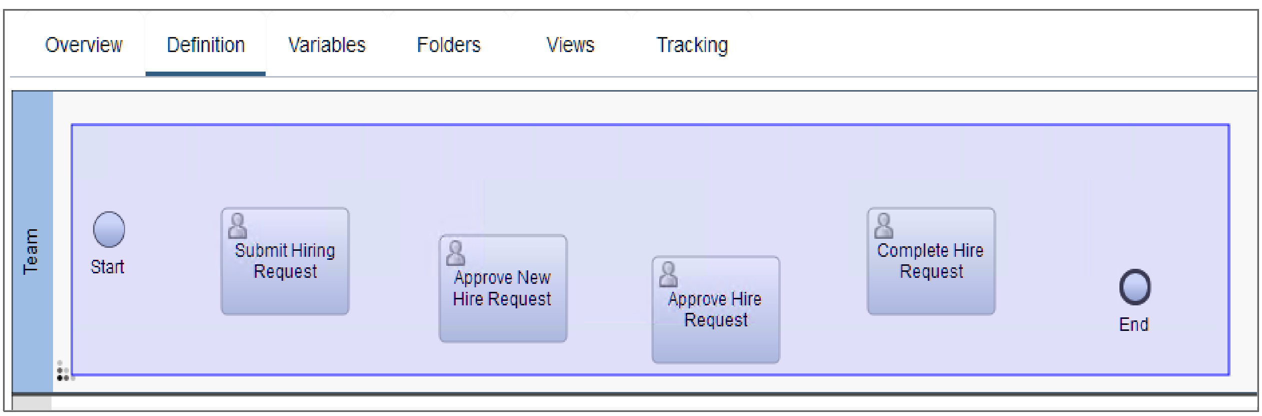

__ b. Rename the activity to **Complete Hire Request**.

__ 5. Sequence the activities and add sequence flow lines.

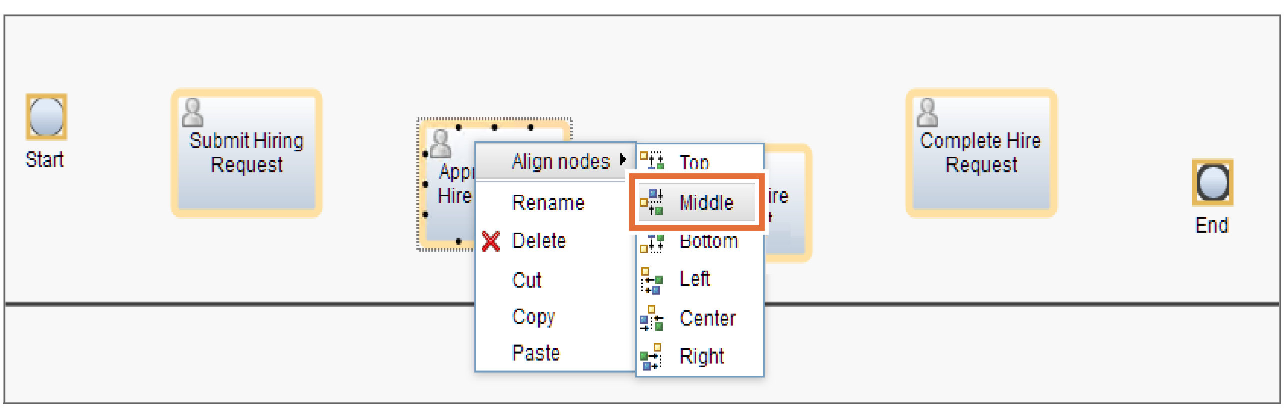
__ a. Place the activities in line by using the following sequence:

Submit Hiring Request > Approve New Hire Request > Approve Hire Request > Complete Hire Request

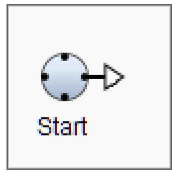
__ b. Lasso all objects by holding down the left mouse button and highlighting all objects including the start and end events. After all are highlighted, release the mouse button.



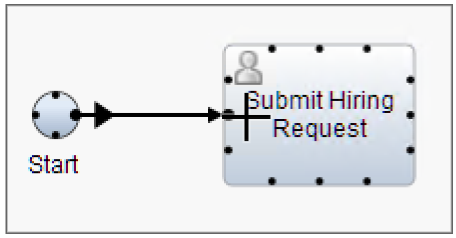

__ c. With all items highlighted (yellow halo denotes selection), right-click one of the items and select **Align > Middle**. The result is all artifacts are aligned.



__ d. Hover over the right portion of the **Start** event until you see the right-arrow connector.

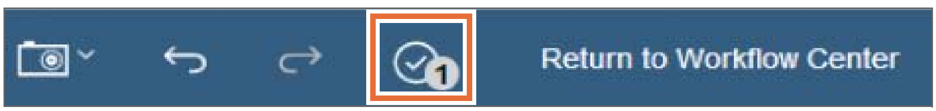



__ e. Click on the arrow. While holding down the **left mouse button**, drag the **arrow** until it connects to the first activity. When the cross displays and the activity is highlighted, you can release the mouse button.

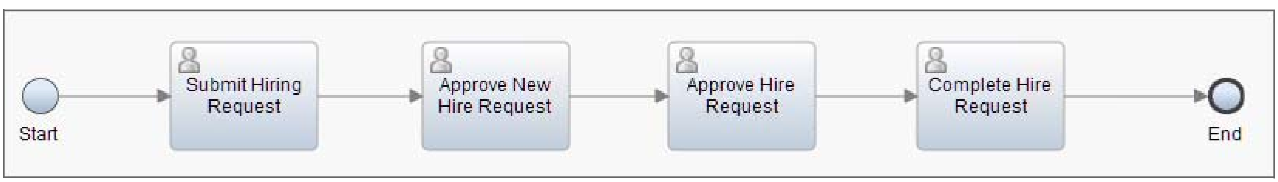



__ f. Repeat adding sequence flows for the remaining activities.

__ g. Click **Finish Editing** to save your changes.




__ h. Verify your process resembles the screen capture below.




Although this is the correct sequence of activities in the process, the roles that complete the activities have not been defined yet. You do this in the next section.

**Optional**

You completed parts 1-3 out of 6, so you are now approximately half done with this exercise. Now would be a good time to take a 10-minute break.

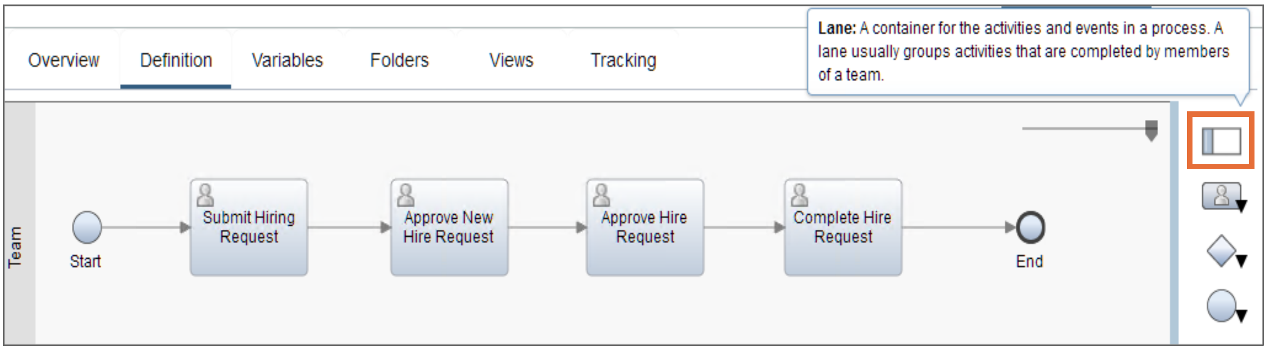
## *Part 3: Model the process teams*

When creating a new process, by default you have a team lane, a system lane, a start event, and an end event. In this section, you add the necessary team lanes to the two default lanes (team and system) and rename the lanes.

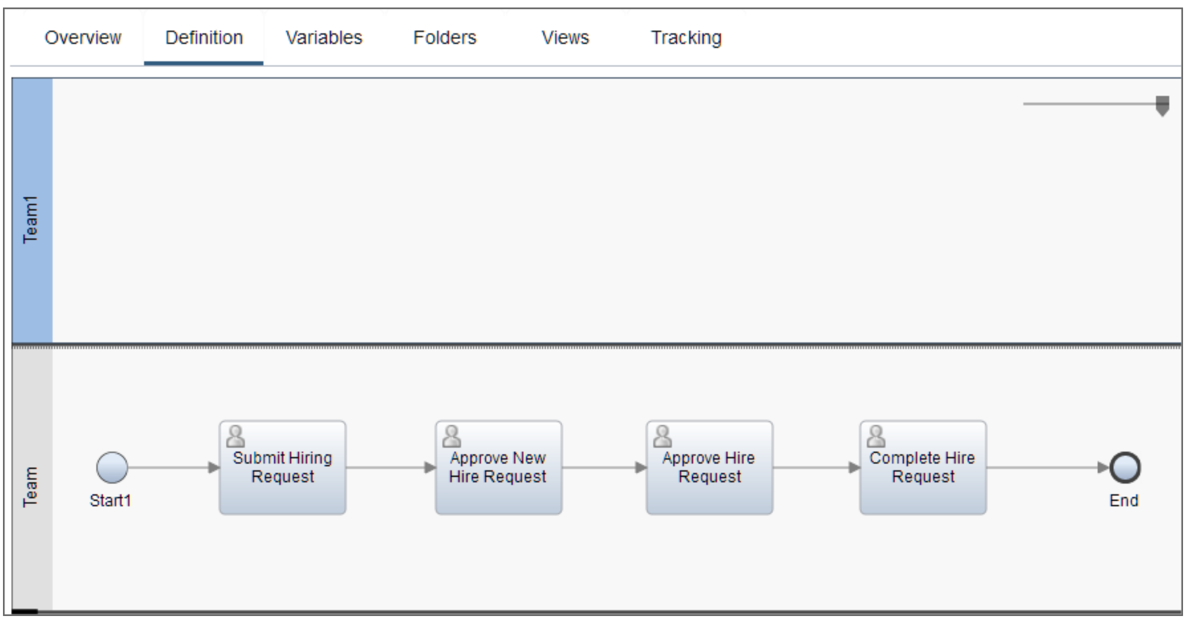
__ 1. Add a team lane to the process.

Read the core requirements in this exercise, pages 2-3 and 2-4, and identify the teams. In the process requirements, two teams are in the main process: Hiring Manager and General Manager. Also, the process has one System lane for a total of three lanes.

__ a. Click the **Lane** icon and drag one lane from the palette to the canvas above the Team lane until you see the green (**+**) symbol. When you release your mouse button, the new lane appears.




__ b. Verify a lane with name **Team1** is added to the process.

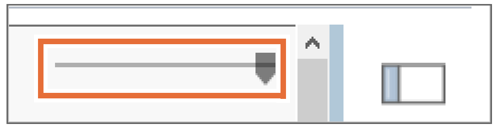



**Hint**

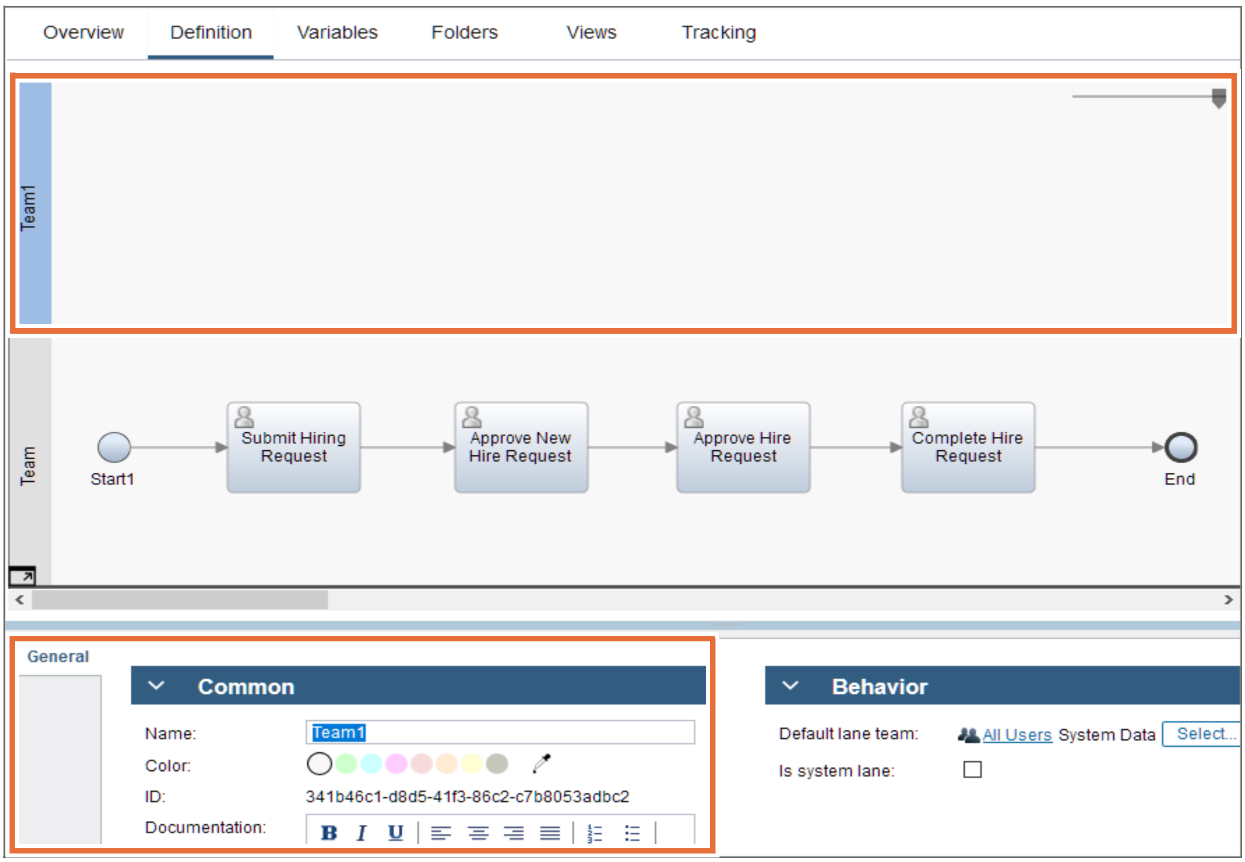
If the lane is below, you can drag it above the Team lane using the mouse. To drag the lane, left-click the Team lane label and drop it where appropriate.

__ 2. Set the properties of the team lanes.

__ a. Use the zoom slider at the upper right of the canvas to view the whole process if it doesn't fit on the screen.



__ b. Click the top lane to select the entire lane. The **General** properties tab populates at the bottom of the screen.




__ c. In the **General > Common** section, change the name of the top lane from **Team1** to: `Hiring Manager`

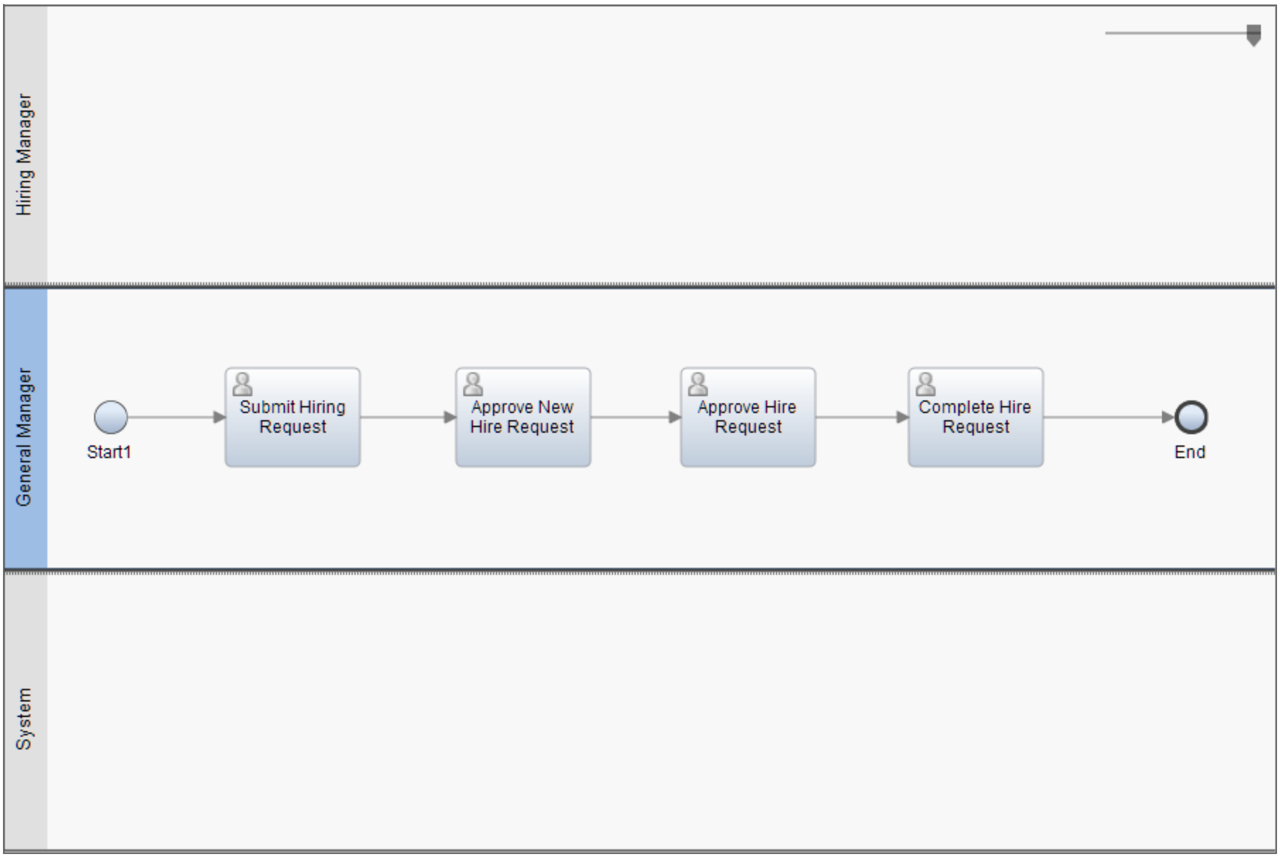
__ 3. Rename the second lane labeled **Team**.

__ a. Select the lane that is labeled **Team** and double-click the name.

__ b. Change the name of the middle lane from **Team** to: `General Manager`

This step leaves the bottom lane as the **System** lane. It is a good practice to keep the **System** lane at the bottom of your process, even if you have no system lane activities

Your process model now has the appropriate Team lanes defined. However, the activities are not assigned to the appropriate Team lanes. You complete this activity in the next section.



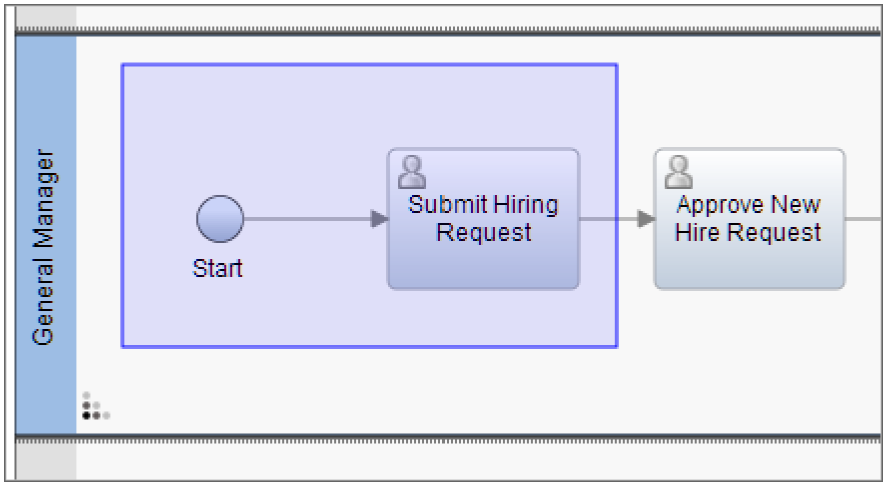

## *Part 4: Assign the activities to the appropriate Team*

You identify the teams that are associated with the lanes instead of allowing all users to claim the tasks that are associated with each activity. Determine which teams conduct each of the four activities. From the process requirements, you determine the following assignments:

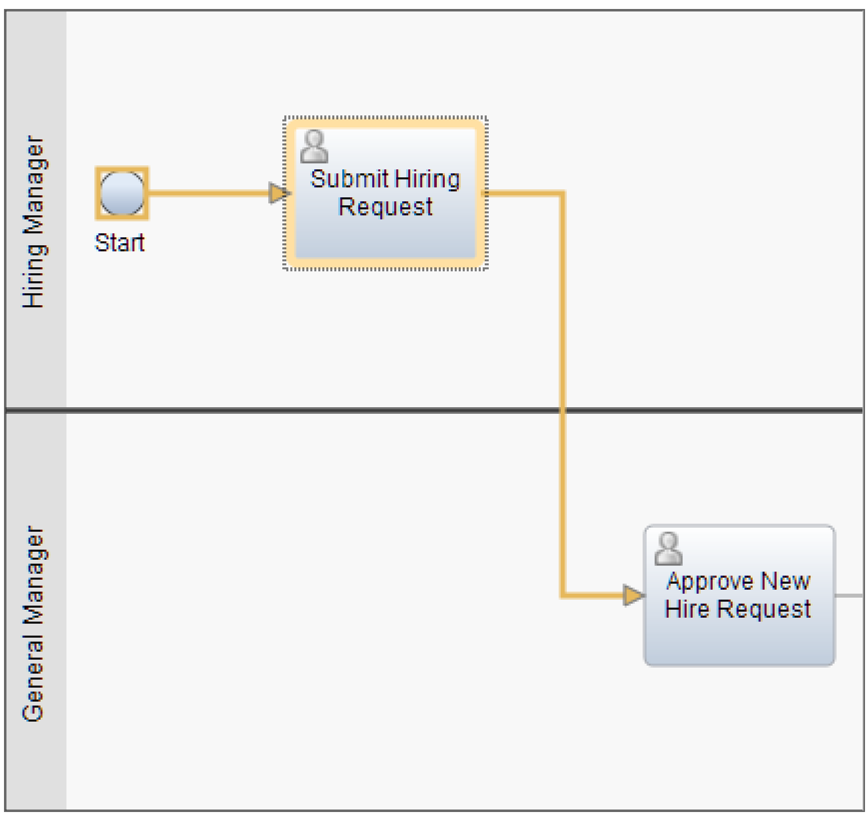
- Hiring Manager (team): Submit Hiring Request
- General Manager (team): Approve New Hire Request
- System (team): Approve Hire Request and Complete Hire Request

__ 1. Arrange the activities to the team lanes to establish the correct activity assignment.

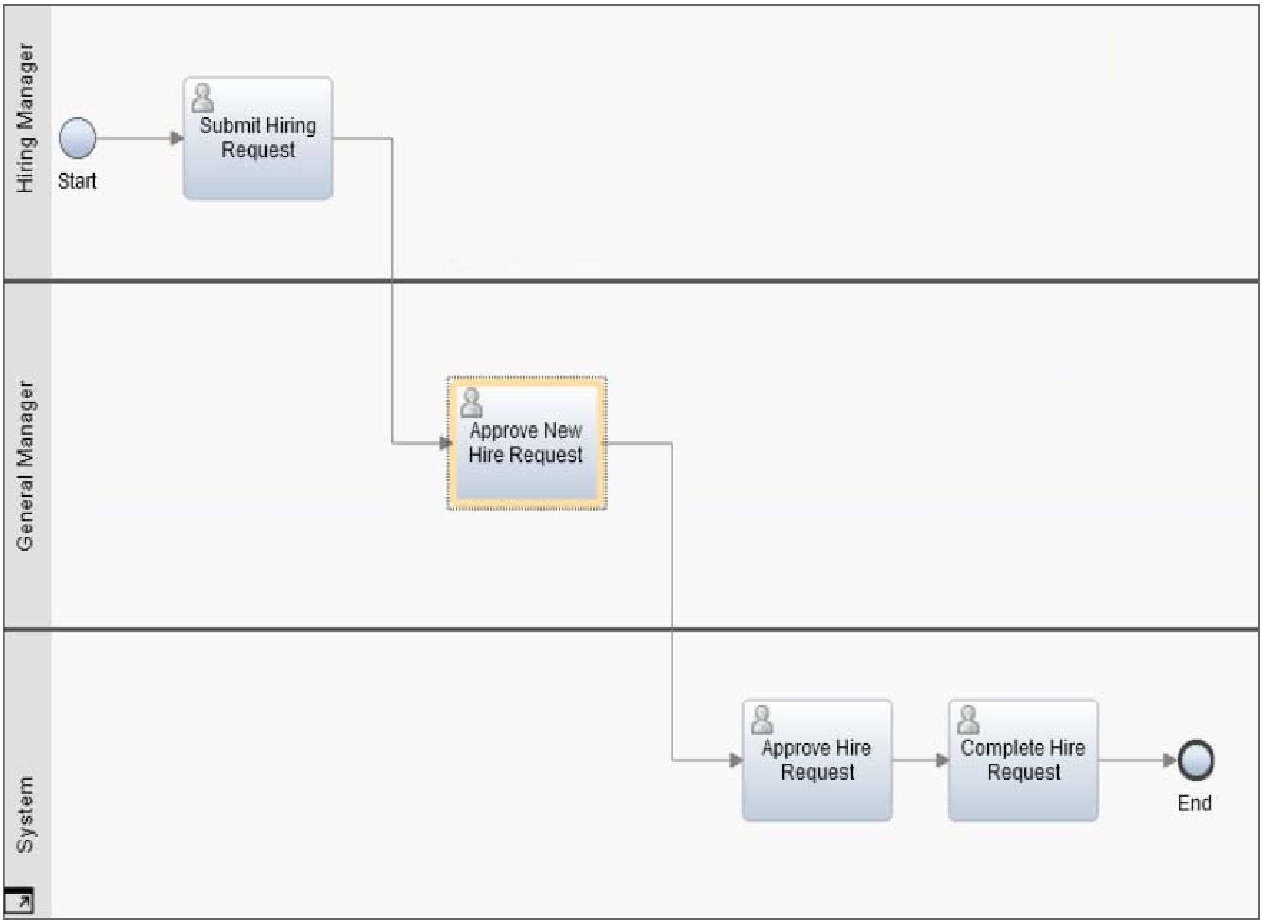
__ a. Select the first two items in the process. To drag the Start event and the Submit Hiring Request activity to the **Hiring Manager** lane, lasso both items by highlighting a box around both items with your cursor.




__ b. With the two items highlighted (yellow halo), drag the selected activities to the **Hiring Manager** lane.



__ c. Leave the Approve New Hire Request activity in the General Manager Team lane and drag the **Approve Hire Request** and **Complete Hire Request** activities, along with the **End** event, to the System lane. Later in the exercise you rearrange this activity in a new lane: Approvers.

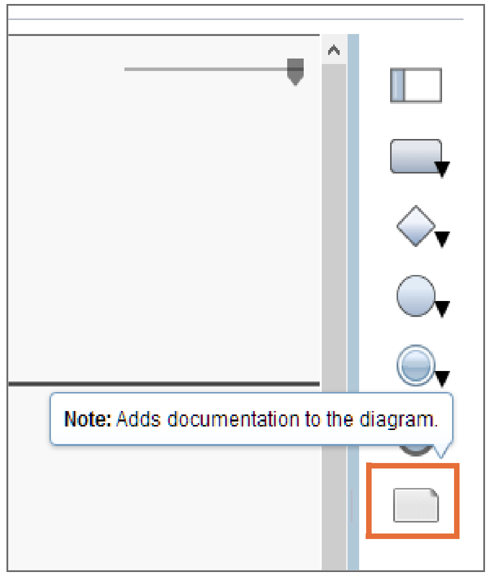



__ 2. Add an annotation

It is common to place annotations in the process when first modeling it. They can serve as placeholders for questions or concerns while the process is in development. The process owner expressed that the approval process might involve multiple steps and departments. Thus, an annotation is placed in the model mentioning this step might be a potential subprocess. At this point, it might not be known whether it is to be implemented as a linked process or subprocess.

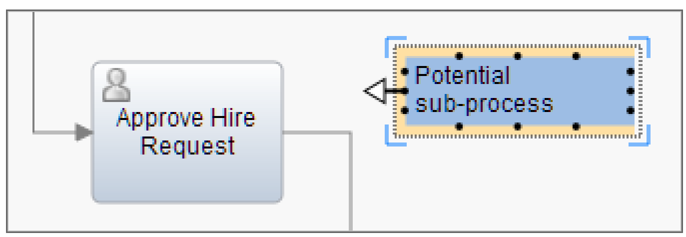
__ a. Verify you are displaying the Definition tab of the Hiring Request Process.

__ b. Drag an **annotation** to the canvas next to the activity **Approve Hire Request**.

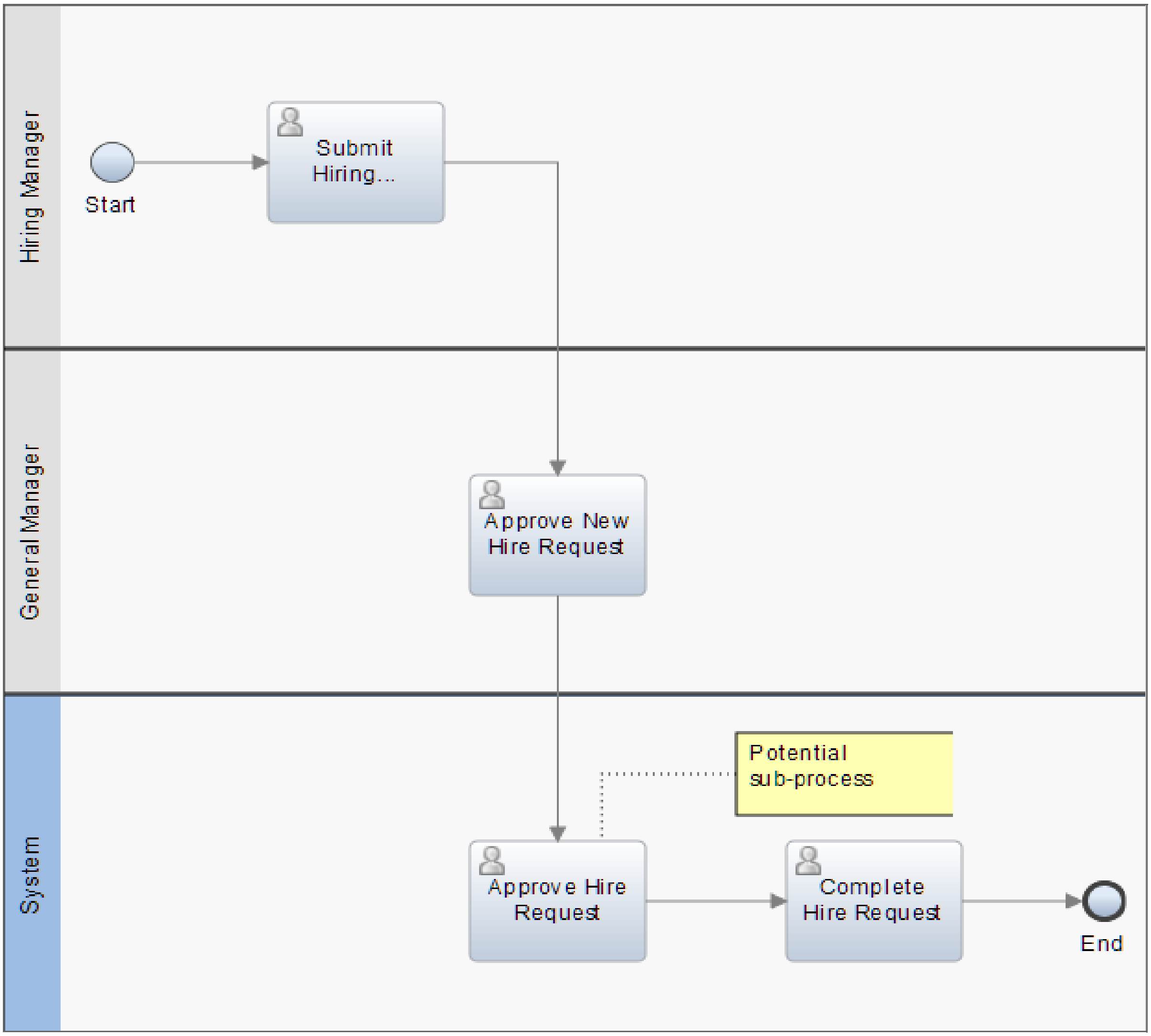



__ c. Double-click the **annotation** and add the text: `Potential sub-process`.

__ d. Highlight the left portion of the **annotation** until an arrow appears and draw a line from the annotation to the activity Approve Hire Request.



__ e. Save your process. The first process is complete for this exercise. Next, you create another process, a linked process.




**Hint**

Your sequence lines might be different. You can drag sequence endpoints from one activity to the other. To do this, highlight the endpoint until it displays as bright blue. Then, drag the endpoint to another activity and release.


It can be tricky to highlight the endpoint correctly. As you become familiar with the Process Designer tool, you can choose the methods that work best for you. For instance, if it becomes too cumbersome to drag sequence lines from one activity to another, you can delete them and rebuild them.

## *Part 5: Create a linked process*

Determine whether any additional work is required for each activity and if so, what the work involves.

- **Submit Hiring Request**: Generally no rework is needed.
- **Approve New Hire Request**: Extra work that is required for the General Manager to record a rejection reason.
- **Approve Hire Request**: Extra team work that the HR Administrator and the Hiring Manager require.
- **Complete Hire Request**: Generally no rework is needed.

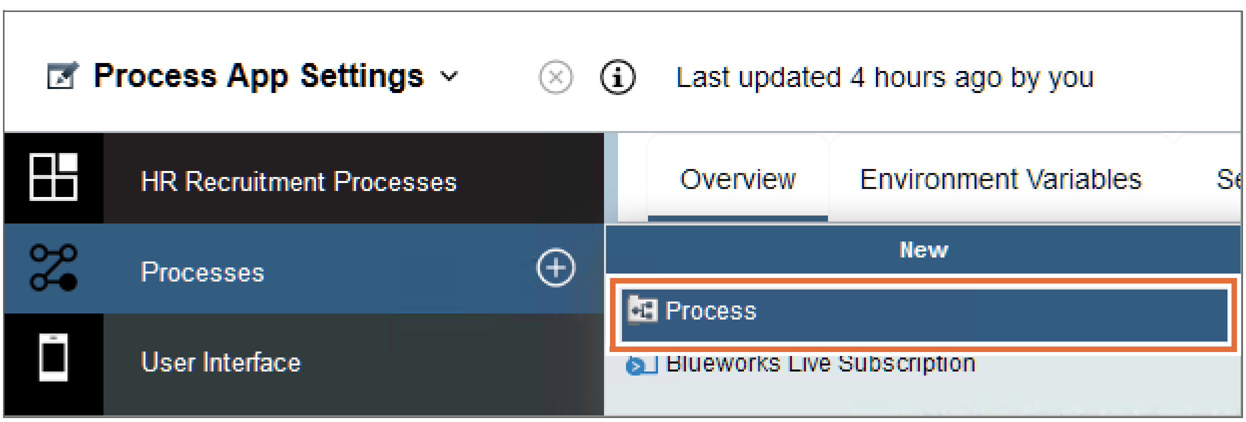
Because more than one team is involved in the **Approve Hire Request** activity, it is no longer a "task" but a linked process. The linked process has lanes, such as System, HR Administrator, and Hiring Manager. Activities include Check Hire Request, Override Hire Request, and Negotiate Hire Request.

There are two ways to implement a nested process. If the process is not to be reused, the use of a subprocess is appropriate. The subprocess is part of the parent process and shares all data. This is done if the process is becoming too large and complex. A subprocess cannot be shared. However, a linked process is a completely separate process that can be shared. You must determine whether you want to use a subprocess or linked process. Because **Approve Hire Request** might be reused in other HR processes, you create a linked process for this scenario.

__ 1. Create a process that is called **Approve Hire Request**.

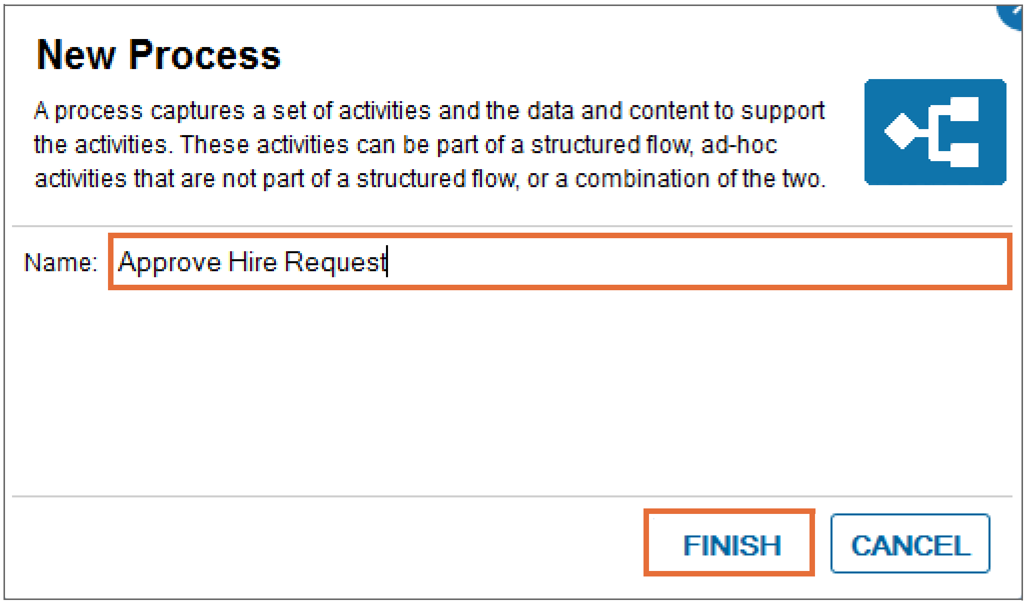
__ a. From the process library, hover over **Processes** category and then click the **(+) plus sign**.

__ b. From the **New** menu, click **Process**.




__ c. Enter `Approve Hire Request` for the **Name**.

__ d. Click **FINISH**.

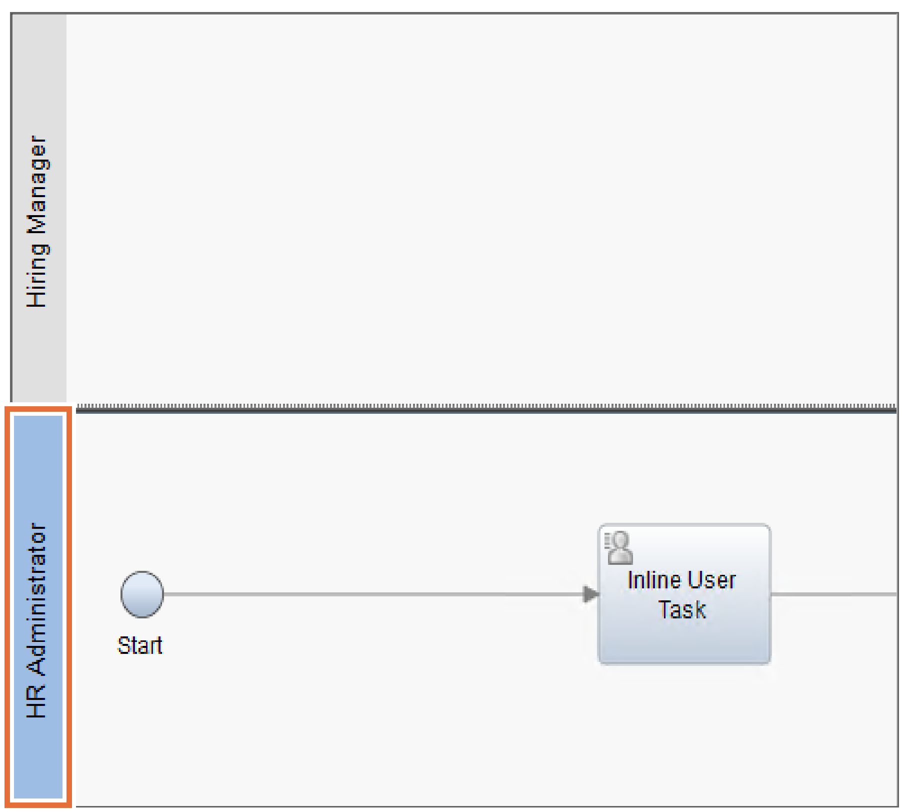



__ 2. Create the System, HR Administrator, and Hiring Manager lanes.

__ a. Click the **Lane** icon from the palette and drag one lane from the palette to the canvas above the default Team lane.

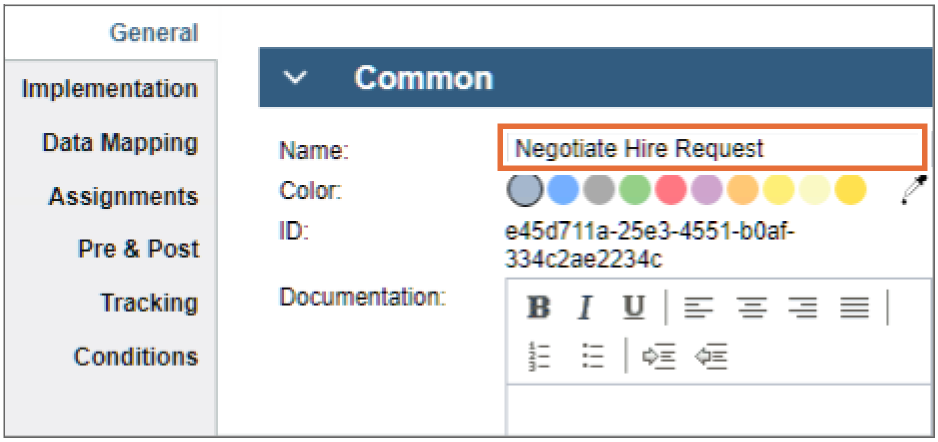
__ b. In the **General > Common** section for the new Team1 lane, change the name to: `Hiring Manager`

__ c. Highlight the second lane and rename it to: `HR Administrator`

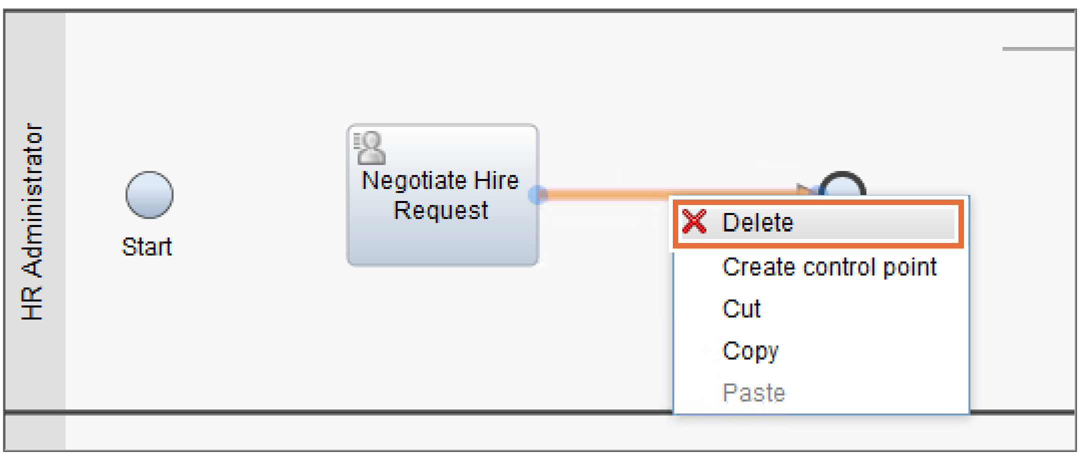


__ 3. Create the Check Hire Request, Override Hire Request, and Negotiate Hire Request activities.

__ a. Rename the **Inline User Task** activity to: `Negotiate Hire Request`




__ b. Delete all the **flows** on the canvas.



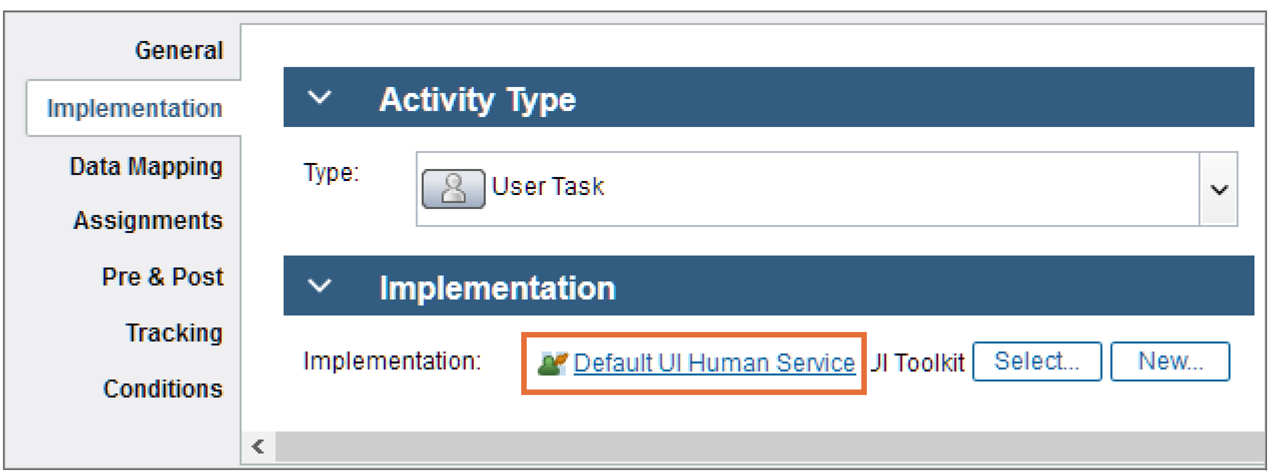

__ c. Expand the activity icon in the palette by clicking the down arrow and drag a **User Task** to the right of Negotiate Hire Request on the canvas.


__ d. Name the activity `Override Hire Request.`

__ e. Move the **Negotiate Hire Request** activity to the Hiring Manager lane.

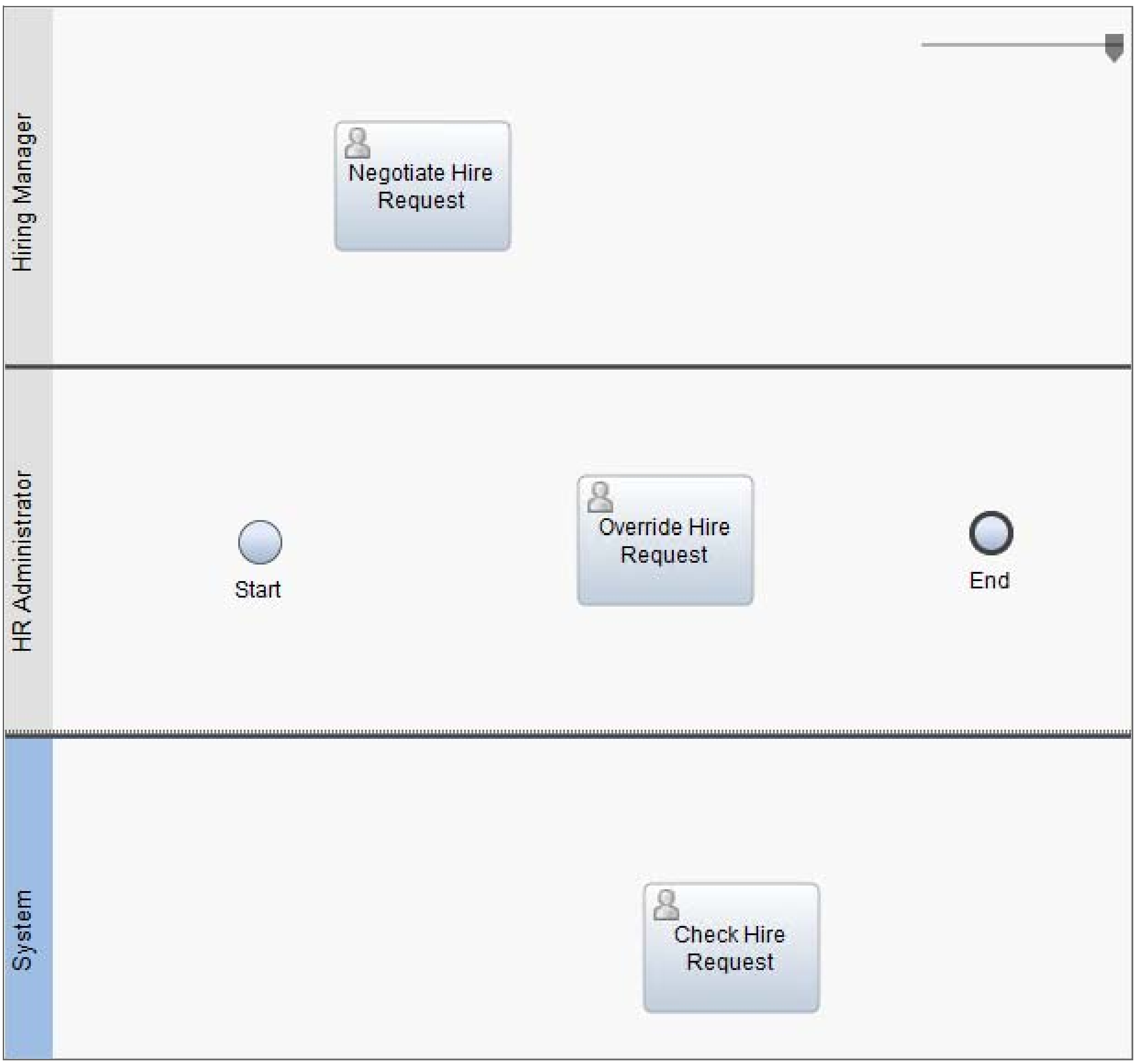

__ f. While Negotiate Hire Request is selected, change the activity to a **User Task i**n the **Implementation > Activity Type** panel.

__ g. In the Implementation section, change the implementation to **Default UI Human Service**.



__ h. Drag another User Task to the System lane and name the task: `Check Hire Request`. The selection that you chose last becomes the default for each artifact in the palette. To drag the User Task now, you don't need to select the down arrow since it now defaults to the User Task selection.

__ i. The locations of your activities should match the screen capture below.

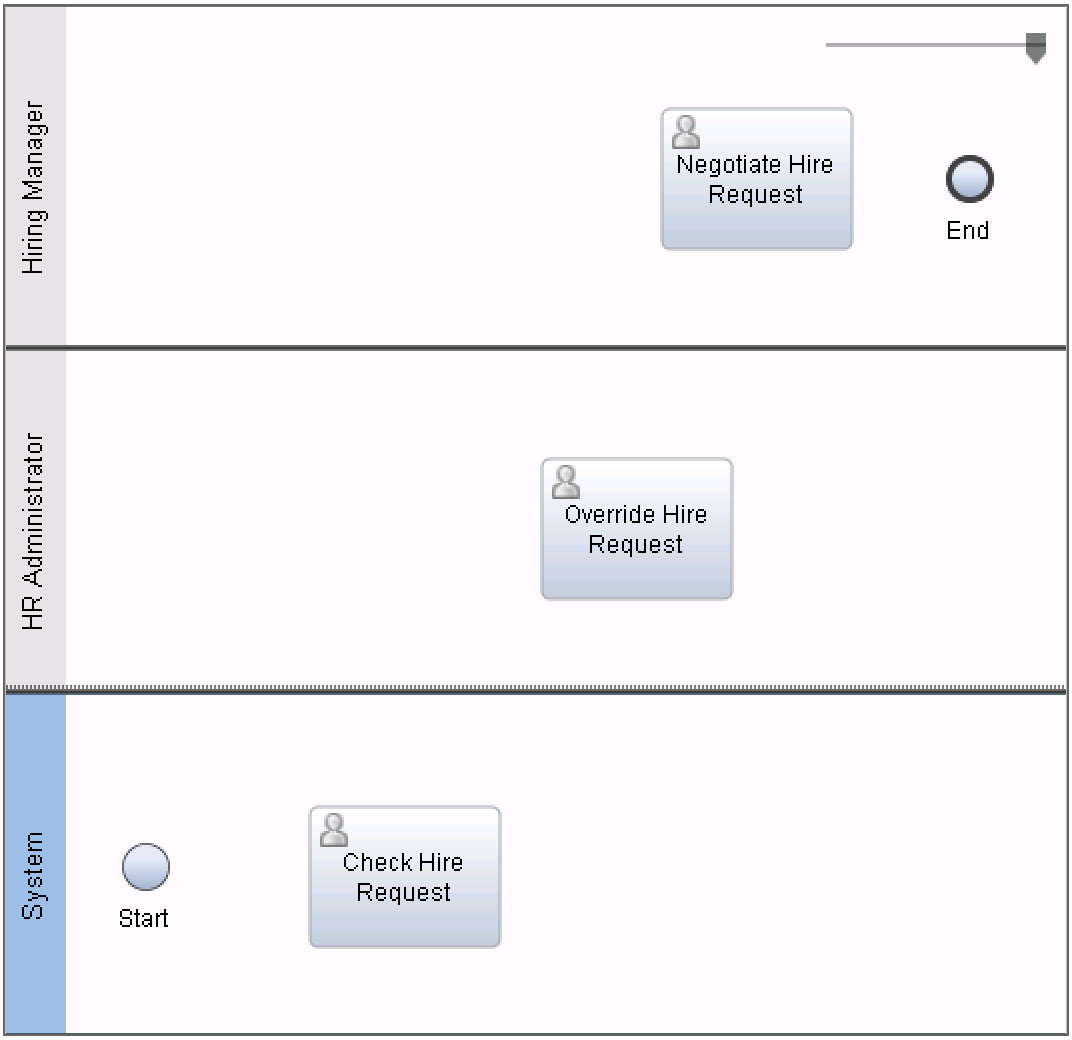



__ 4. Wire the activities

__ a. Move your **Start** event before the first activity, **Check Hire Request** in the System lane

__ b. Move the **End** event after the **Negotiate Hire Request** activity in the Hiring Manager lane.

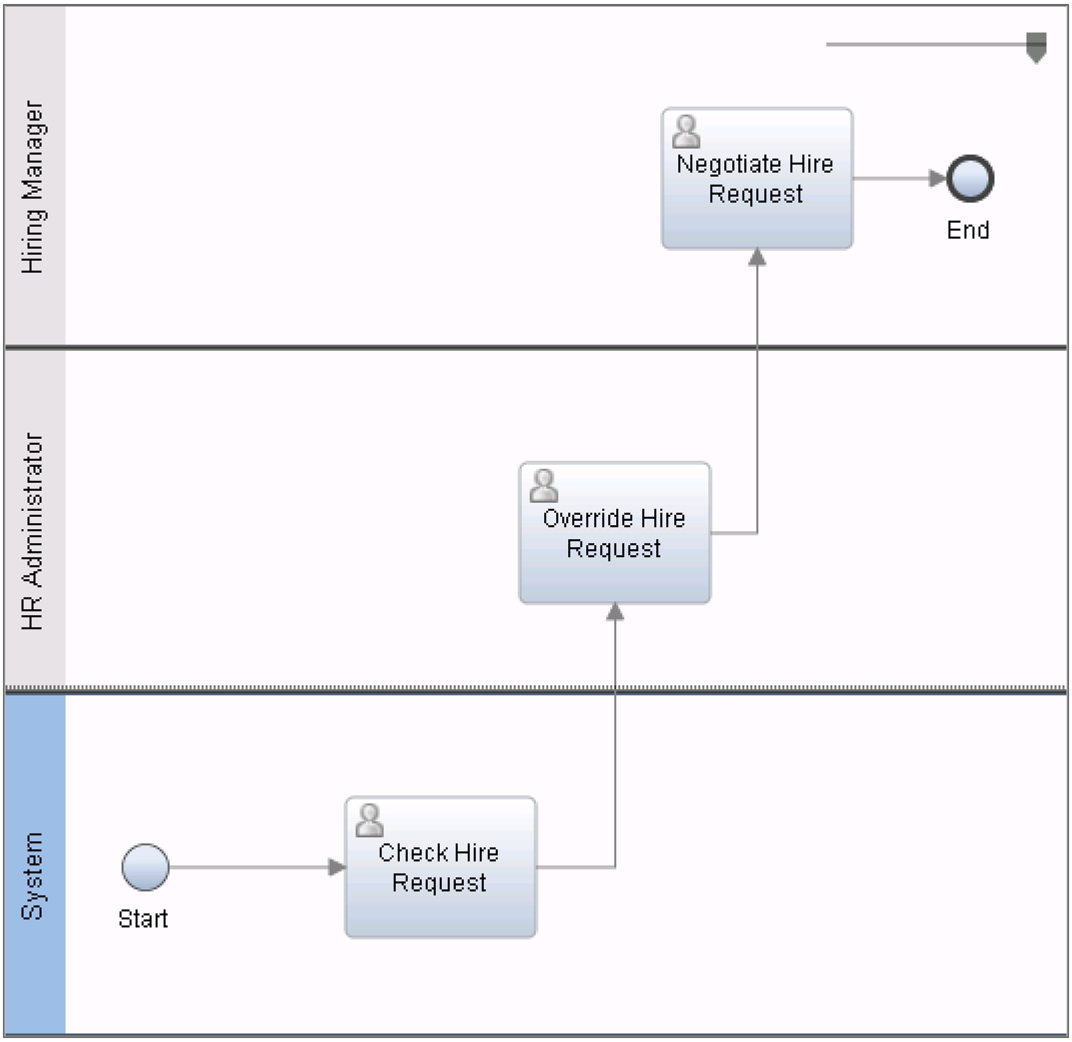
__ c. Arrange the activities in an upward diagonal line.




Although the layout breaks the preferred left-to-right, top-to-bottom approach because the process starts in the system lane, most developers keep the system lane as the bottom lane. Communicating a clear, concise model is the most important goal, and even though this model breaks the preferred practice, it meets the readability goal.

__ d. Create a sequence flow from the **Start** event to **Check Hire Request.**

__ e. Repeat adding sequence flows for the remaining activities.



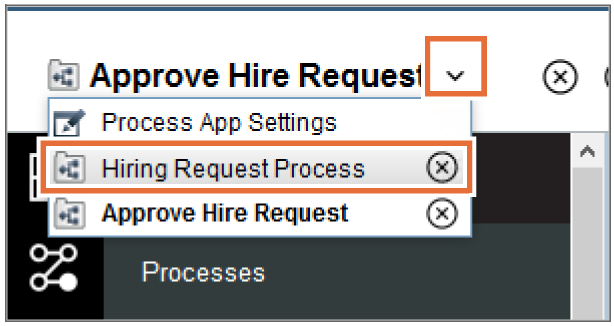

__ f. Save your process application.

## Part 6: Attach the linked process

__ 1. Add the **Approvers** lane to the parent process.

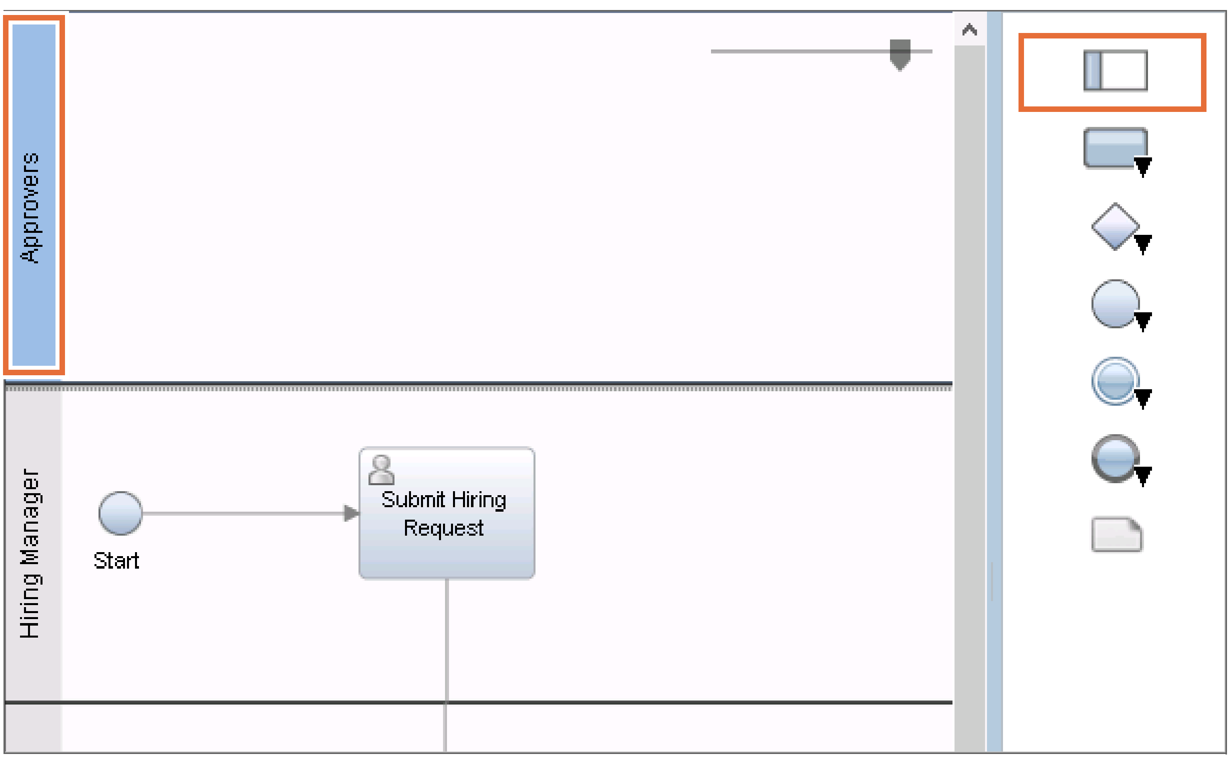
To accommodate a subprocess with multiple lanes, a new lane is added in the parent process to encompass the roles depicted in the subprocess.

__ a. Return to the Hiring Request Process by opening the **History** menu at the top and selecting **Hiring Request Process**.



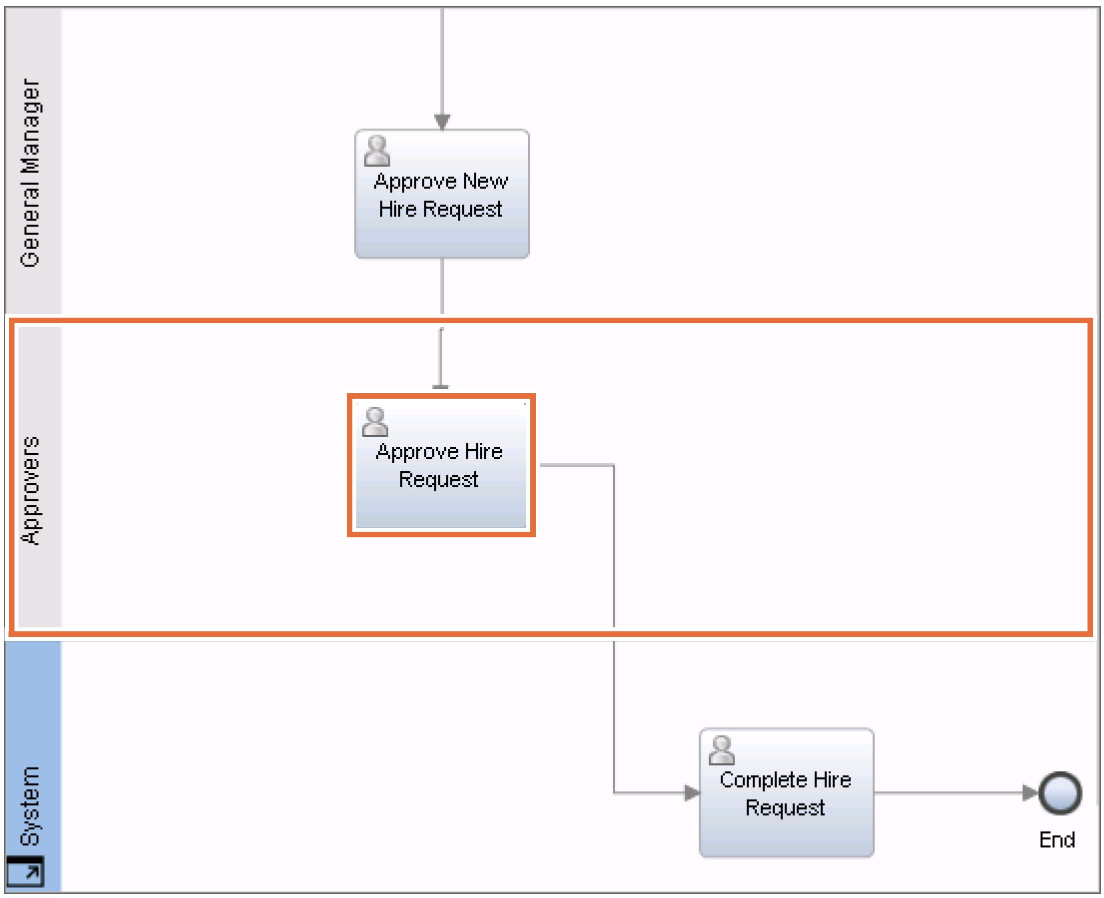

__ b. Delete the **annotation** mentioning the potential subprocess. This annotation served as a temporary placeholders to help in understanding the process. Now that the linked process is being implemented, there is no need for the annotation. In a real-world implementation, this would likely occur over a period of time.

__ c. Drag a new lane above the Hiring Manager lane in the diagram and name it: `Approvers`



__ d. Right-click the **Approvers** lane and click **Move lane down** twice. Continue doing this until it is above the **System** lane.


__ e. Drag the existing **Approve Hire Request** activity that is in the System lane into the **Approvers** lane.



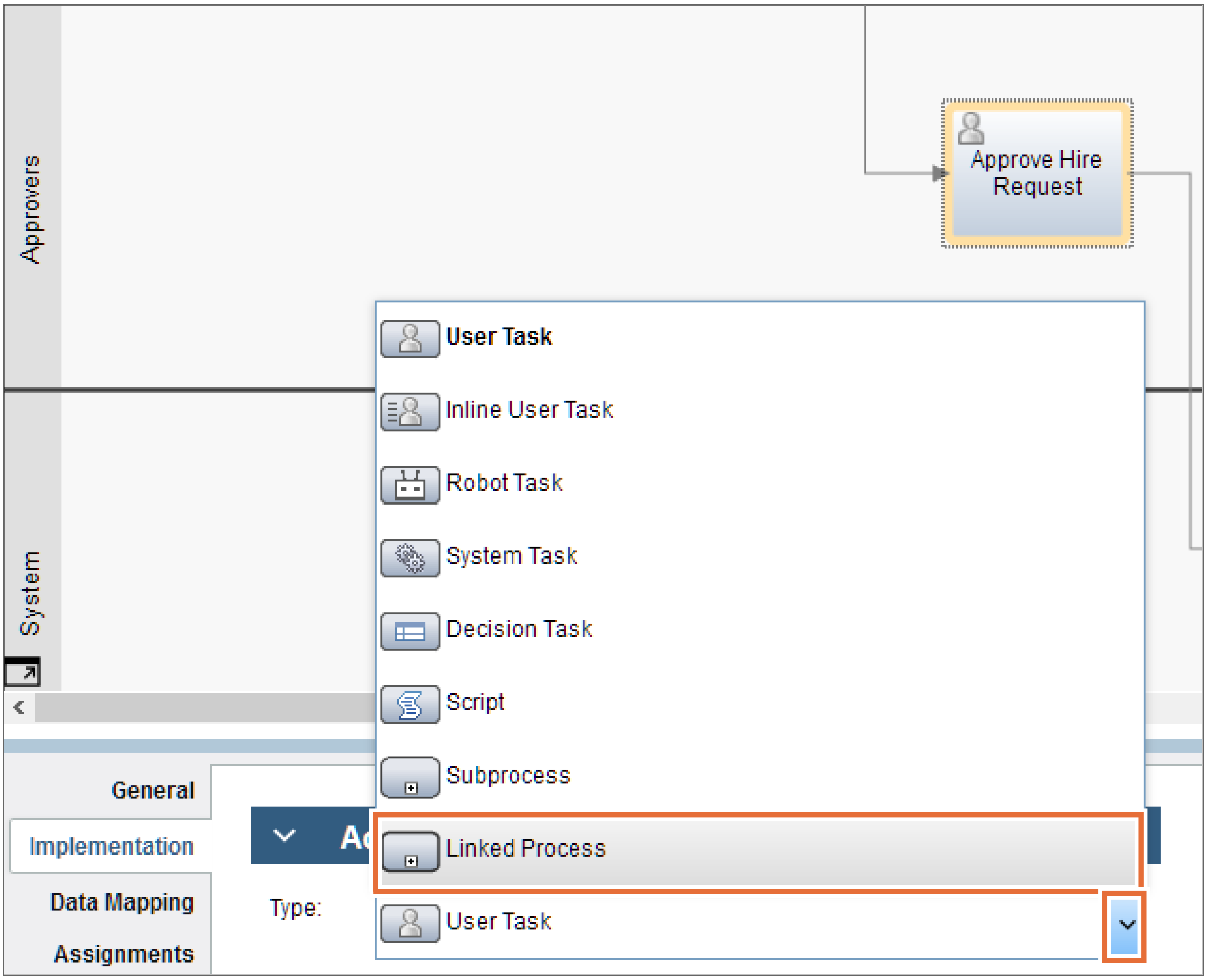

__ f. Realign the activities in their respective lanes so that the sequence flow lines are straight.

__ g. Save your work.

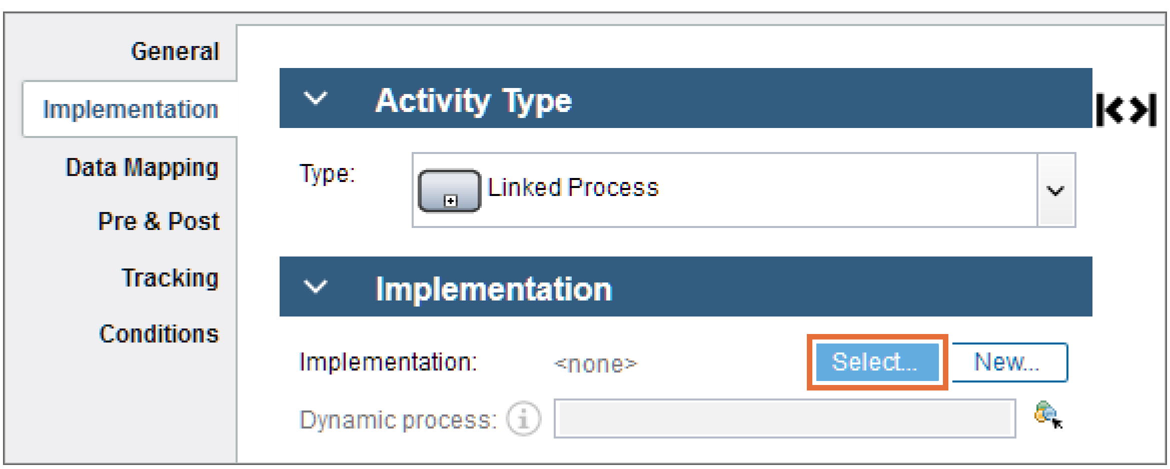
__ 2. Change the implementation of the existing **Approve Hire Request** activity to a linked process.

__ a. With the **Approve Hire Request** activity selected, open the **Implementation** menu.

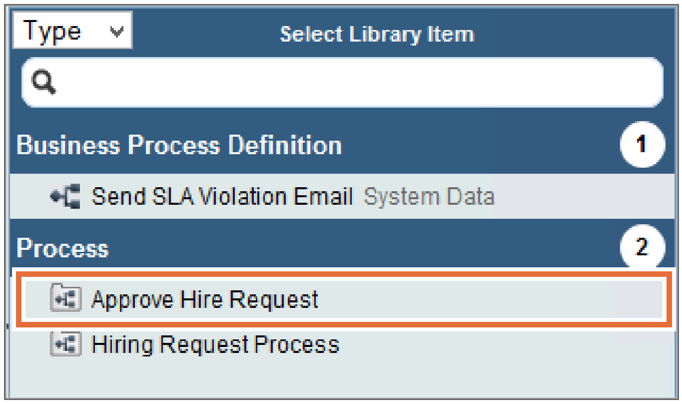
__ b. In the Activity Type section, select the Type as **Linked Process**.



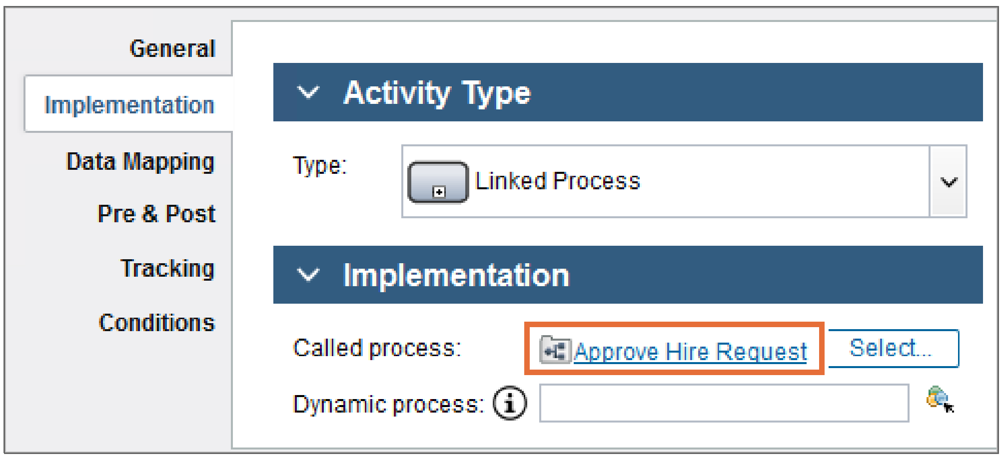

__ c. Under the Implementation section, click **Select** next to Implementation.



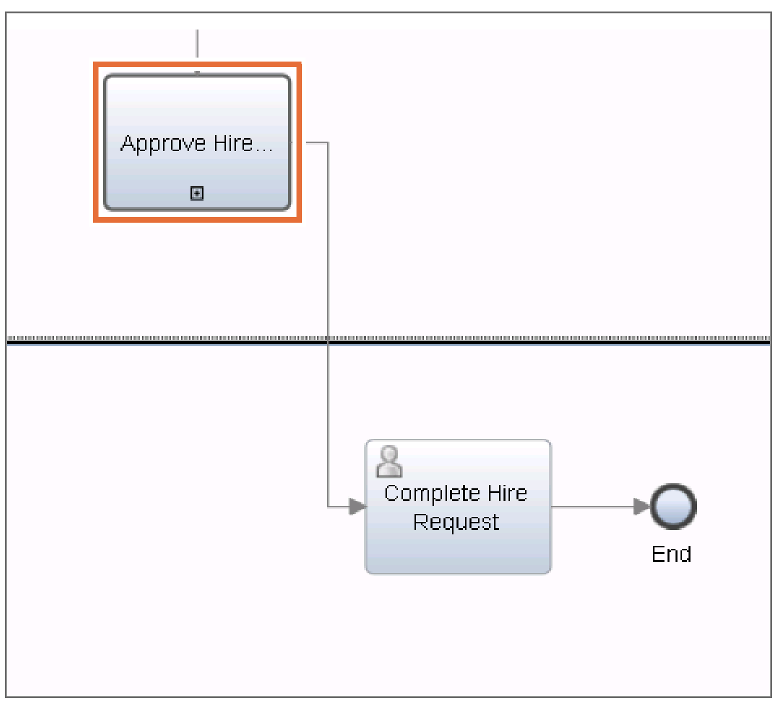
__ d. Select **Approve Hire Request** from the Process menu.




__ e. The process now shows **Approve Hire Request** as a linked process.




__ f. The Approve Hire Request activity is now changed to a linked process. In Business Process Model and Notation, this is denoted with a **plus-sign**.



__ g. Save your work.

You have completed this exercise.

In this exercise, you modeled the parent and linked processes in the Workflow Center for the HR Recruitment Processes solution. First, you converted business process workflow steps that are documented in the process discovery and analysis into process model tasks. Then, you defined and modeled team lanes in the business process and assigned the tasks to the appropriate teams. Finally, you created a subprocess and linked it to the parent process. The next step is to add gateways to direct the process. This is covered in the next exercise.

## End of exercise

# Exercise 3. Playback 0: Controlling process flow

## Estimated time

01:30

## Overview

This exercise covers how to create gateways in a business process, and how to create timer intermediate events. Validation is accomplished through a review session with all business stakeholders, business users, and the BPM development team. This unit describes the Playback 0 validation goals and requirements, explains how to validate that a process model meets the goals and requirements, and describes how to reach consensus on the process model. At the end of this exercise, you complete Playback 0.

## Objectives

After completing this exercise, you should be able to:

- Add gateways to a process
- Model the appropriate sequence flows for each gateway
- Add a timer intermediate event to a process based on business requirements
- Model an escalation path in a process with IBM Process Designer
- Add a new swimlane and activity for legal review to meet additional requirements.
- Validate the process application and create a snapshot

## Introduction

The purpose of this exercise is to add all the gateways necessary to model the flow control for the process.

In this exercise, you add a timer intermediate event that helps satisfy newly identified requirements for the Hiring Requisition process. You are also provided new requirements regarding a legal review. After developing from the new requirements, you validate the process application and create a snapshot.

Throughout the exercise, you switch back and forth several times between the Hiring Request Process and Approve Hire Request processes. Take additional care when working through the lab to make sure that you work in the right process.

This exercise is long, so a break is included in the middle of the instructions.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

# Exercise instructions

## *Part 1: Create gateways for parent process*

A gateway is represented as a diamond, and can be thought of as a question at a point in the process flow.

Gateways are added to the parent process, the Hiring Request Process. In this part of the exercise, you create a gateway that is called Is Position New.

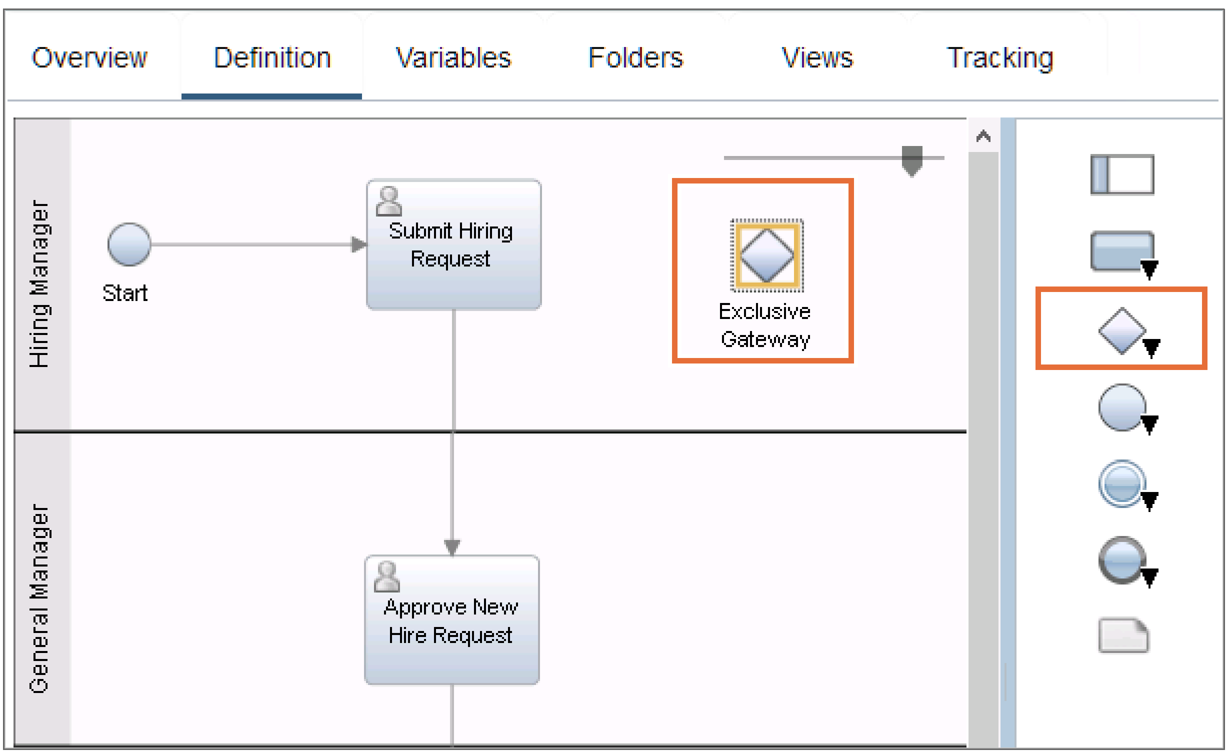
In the Hiring Request Process, you need a gateway to direct the process for the General Manager to review the salary after the Submit Hiring Request activity.

__ 1. Create the **Is Position New** gateway

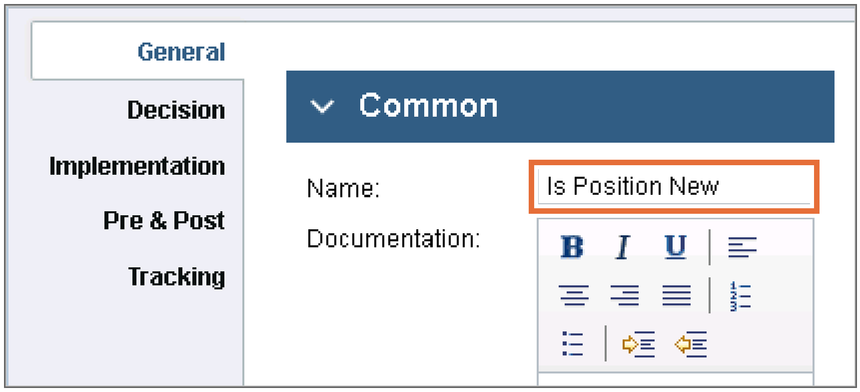
__ a. Open the **HR Recruitment Process** process application.

__ b. Open the **Hiring Request Process**.

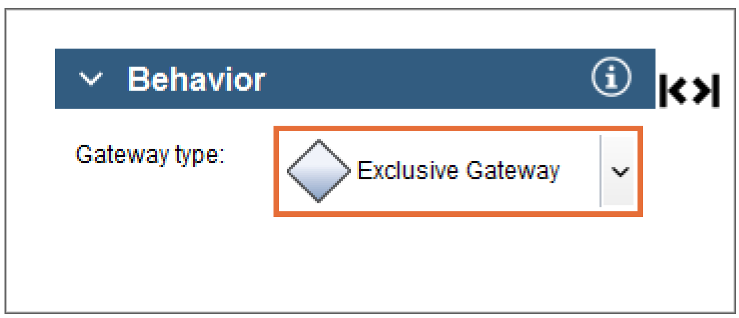
__ c. In the Hiring Request Process, drag an **Exclusive Gateway** from the palette onto the canvas to the right of the **Submit Hiring Request** activity. You do not need to open the menu to drag an exclusive gateway to the canvas, as the exclusive gateway is the default.



__ d. Select the gateway and in the **General > Common** section below, enter `Is Position New` as the **Name**. Because exclusive gateways imply a question, the question mark is unnecessary, and anyone who views the process automatically assumes the question mark.



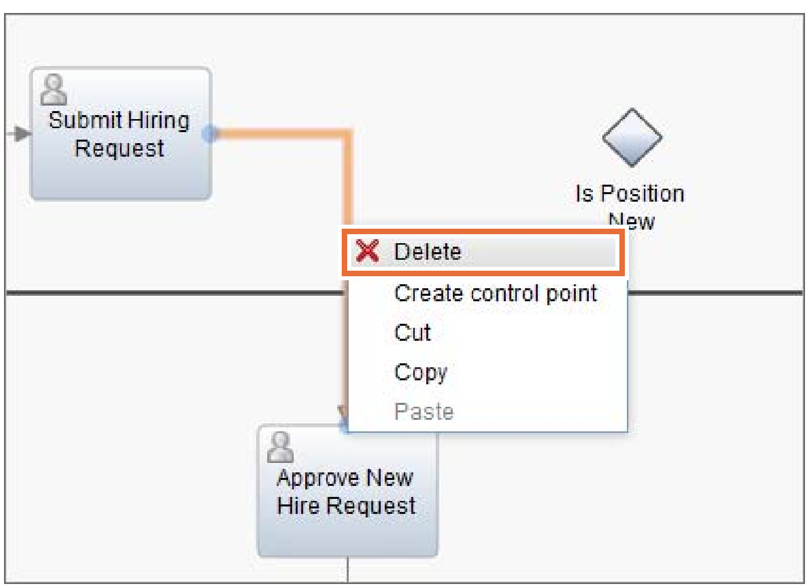

__ e. In the **General > Behavior** section, verify that the Gateway type is: **Exclusive Gateway**.



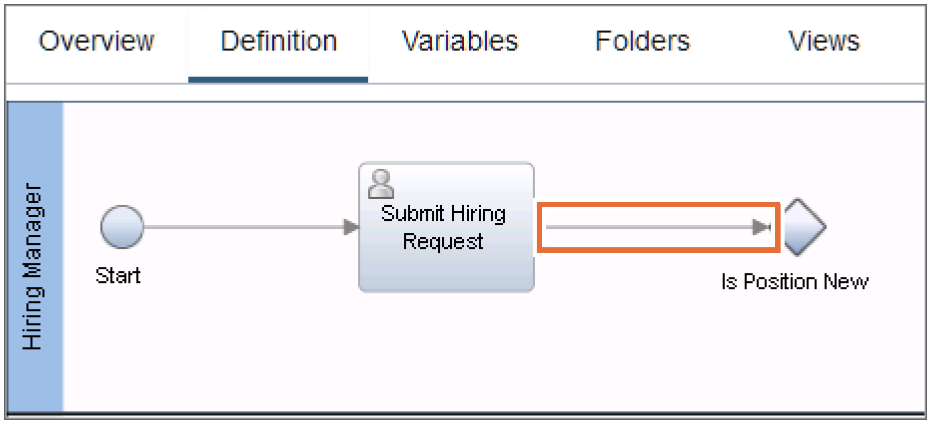

__ 2. Add sequence flows to connect the gateway.

__ a. Select the sequence flow between **Submit Hiring Request** and **Approve Hew Hire Request**.

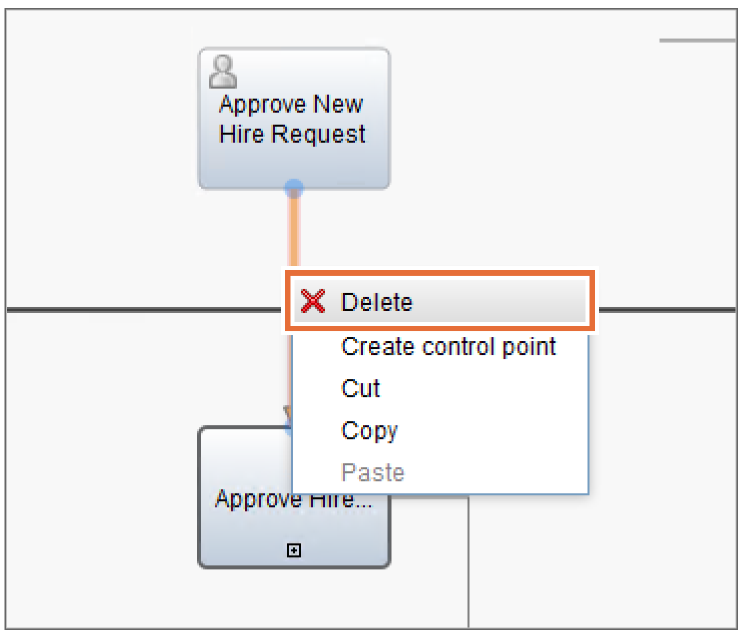
__ b. While selected, right-click the sequence flow and click **Delete**.



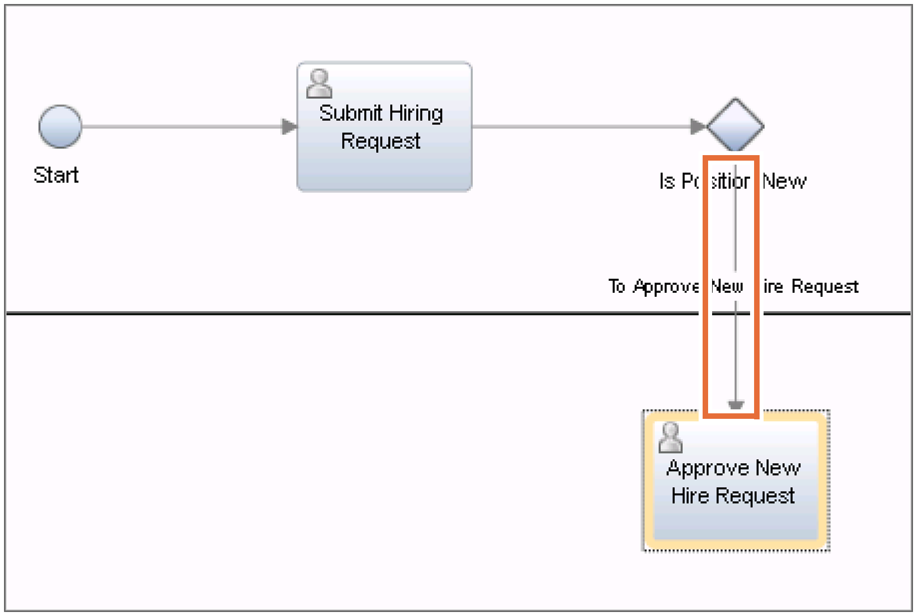
__ c. Create a sequence flow between the **Submit Hiring Request** activity and the **Is Position New** gateway. Hover over the right anchor point on the activity until an arrow appears, then drag the arrow to the left anchor point on the gateway.



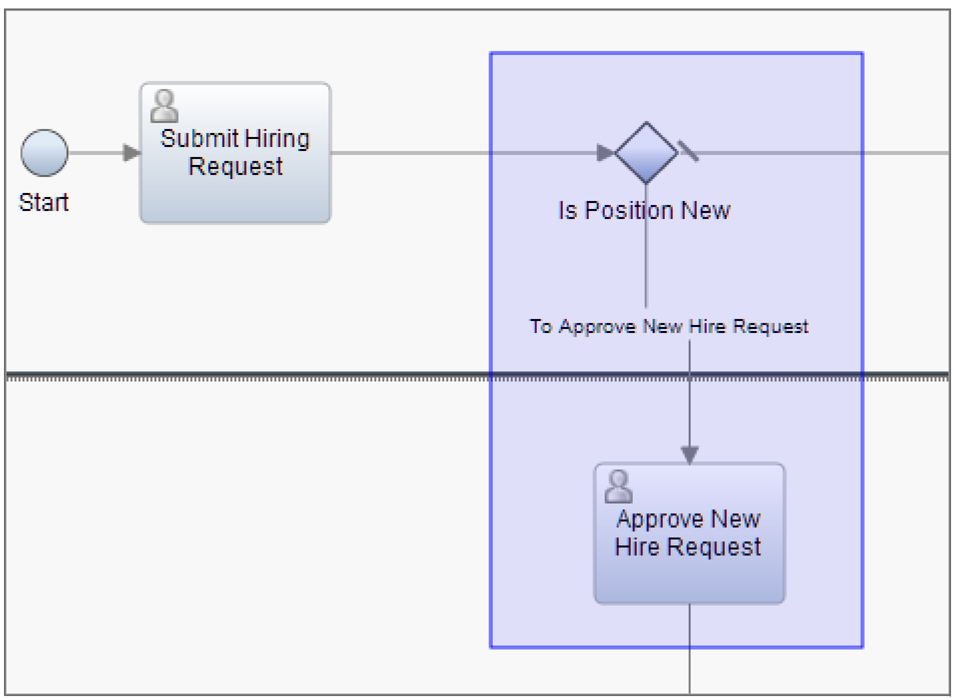

__ d. Delete the sequence flow between **Approve New Hire Request** and **Approve Hire Request**.



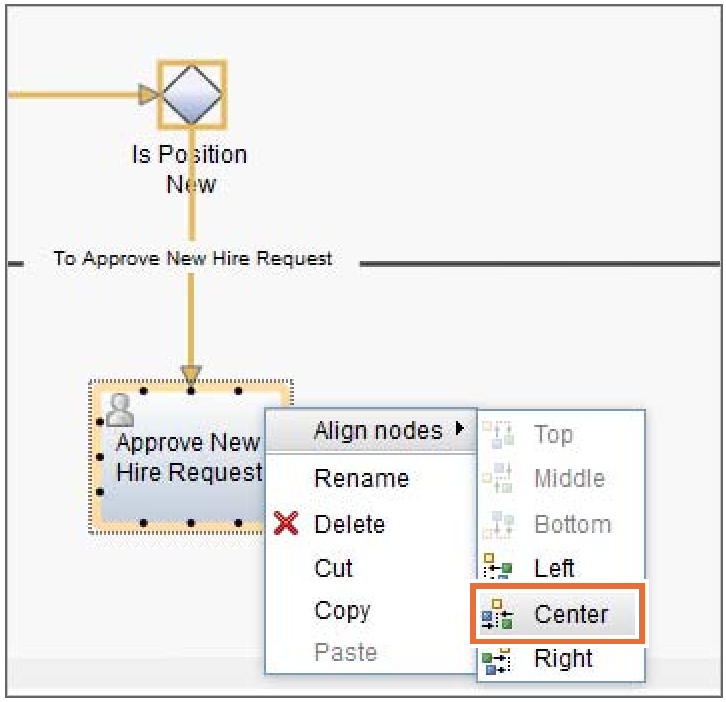
__ e. Connect the **Is Position New** gateway to the top of **Approve New Hire Request**.



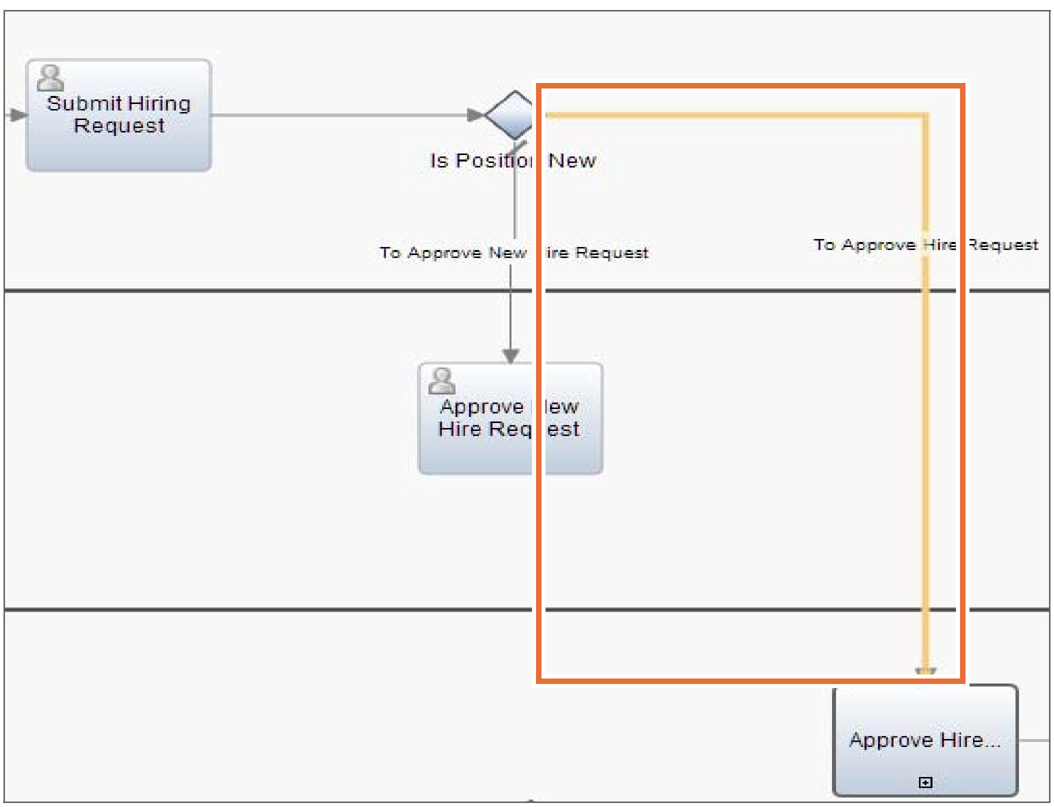

__ f. Lasso the **Is Position New** gateway and the **Approve New Hire Request** nodes.



__ g. Although the sequence flow is highlighted, Process Designer will only align the nodes. After highlighting the nodes (depicted by yellow halo), right-click the mouse on one of the nodes to bring up the Align nodes menu. Select **Center** from the Align nodes menu.




__ h. Connect the **Is Position New** gateway to the top of **Approve Hire Request**.

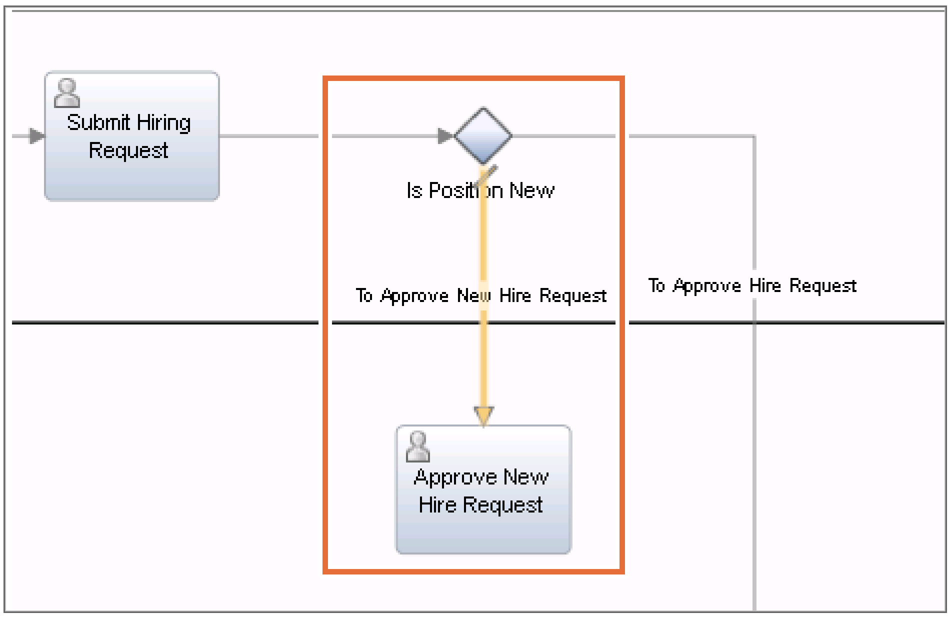


**Note**

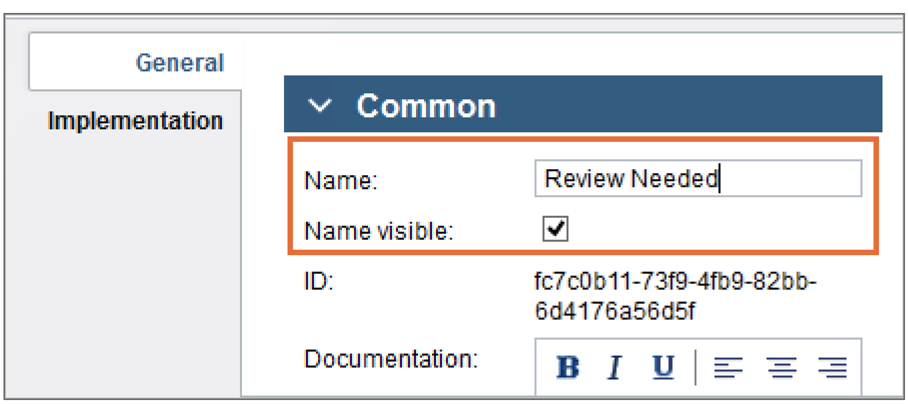
Your sequence flows do not have to exactly match the exercise diagrams in this lab. The sequence flows change after implementation of the gateways. The flows become default or conditional (represented by a diagonal hash marker) according to the order you draw them, so your flows might vary from the diagrams in the labs. You learn how to set the default flow in a later exercise.

__ 3. Label the sequence flows.

__ a. Select the flow between the **Is Position New** gateway and **Approve New Hire Request**.

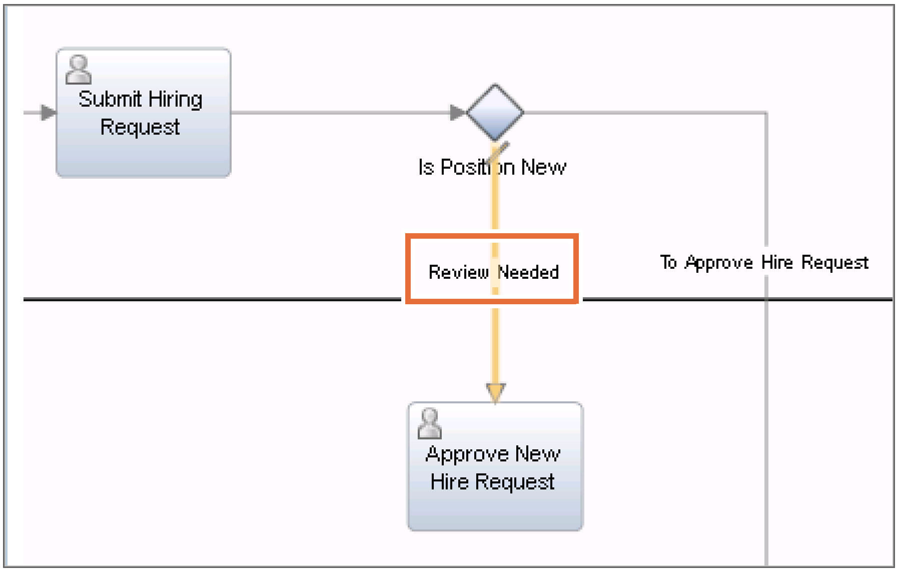



__ b. In the **General > Common** section, enter `Review Needed` as the **Name** and verify **Name visible** is selected.



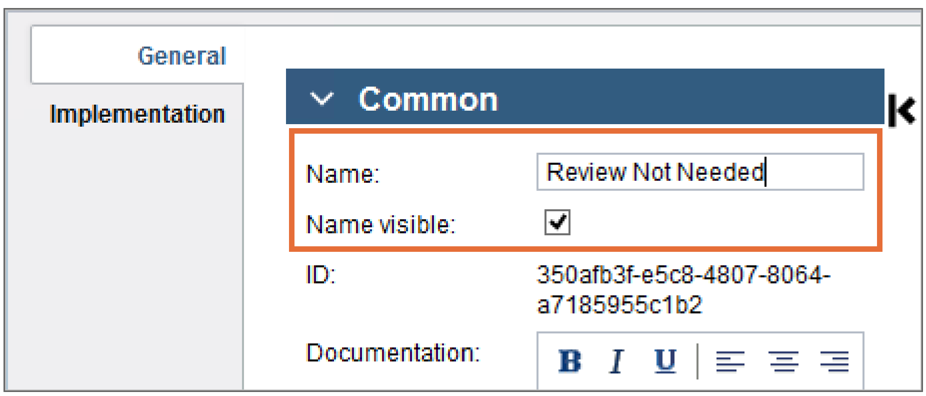

 

The flow displays the label.

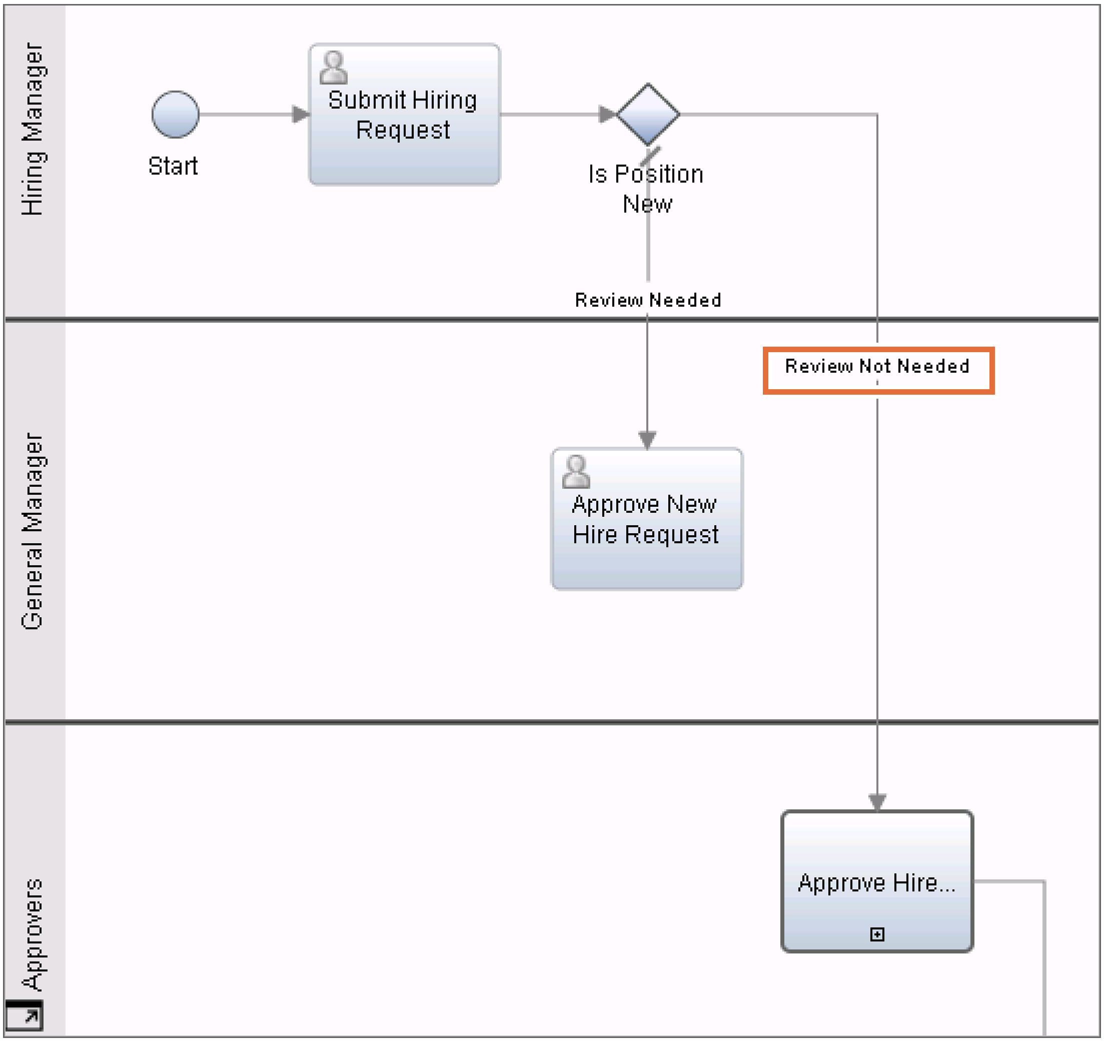



__ c. Select the flow between the **Is Position New** gateway and **Approve Hire Request**.

__ d. In the **General > Common** section, enter `Review Not Needed` as the Name and verify **Name visible** is selected.

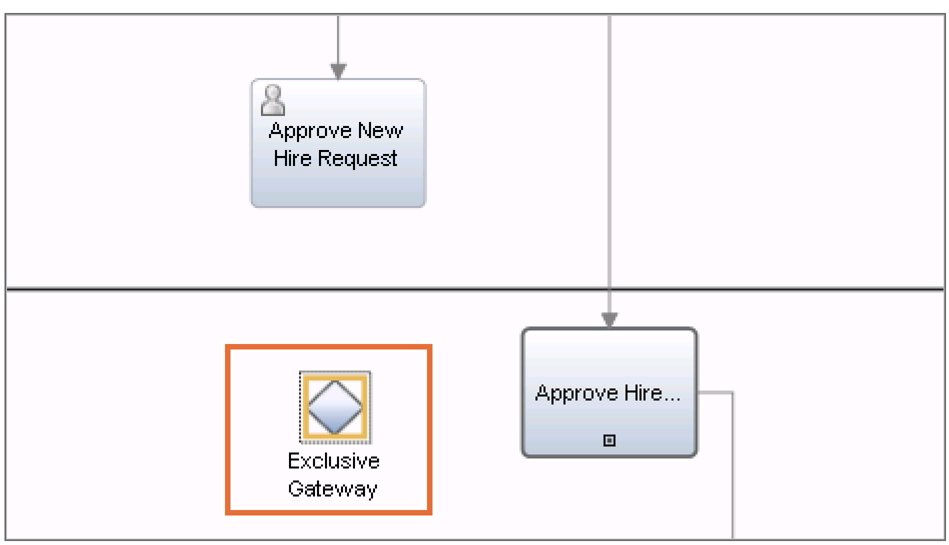


__ e. Save your changes. The flow displays the label.

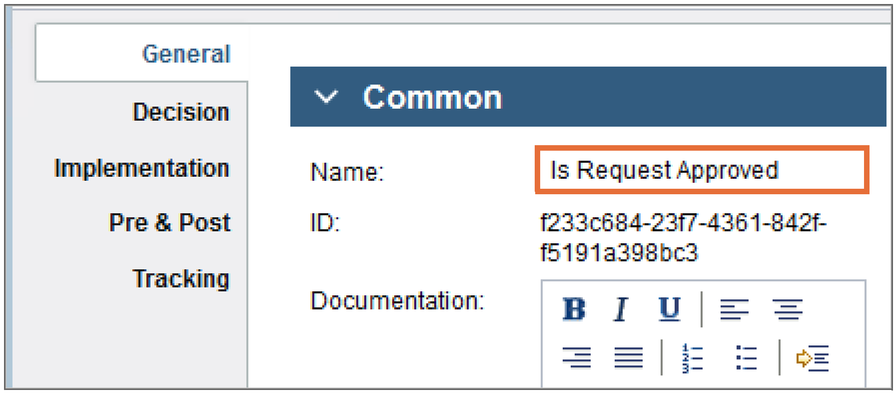



__ 4. Create the **Is Request Approved** gateway

__ a. Drag a **Gateway** from the palette onto the canvas to the left of the **Approve Hire Request** activity.



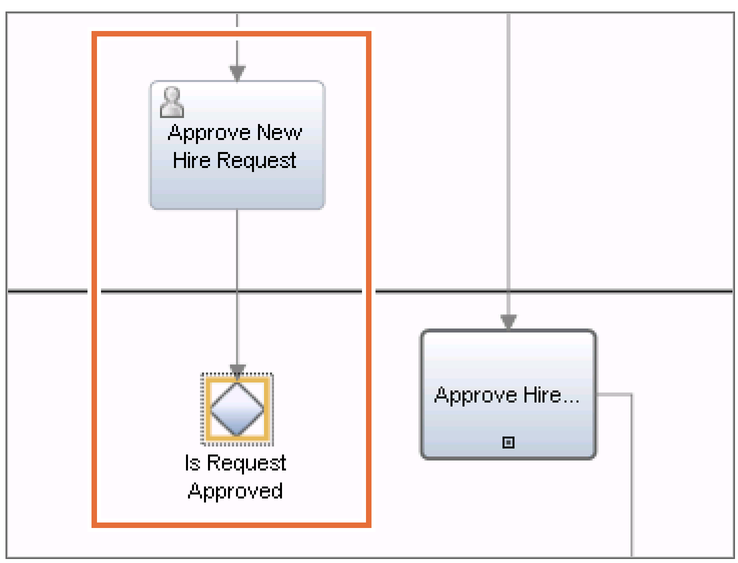
__ b. Select the **Exclusive Gateway** and in the **General > Common** section, enter `Is Request Approved` as the Name.




__ c. In the **General > Behavior** section, verify **Exclusive Gateway** as the Gateway type.

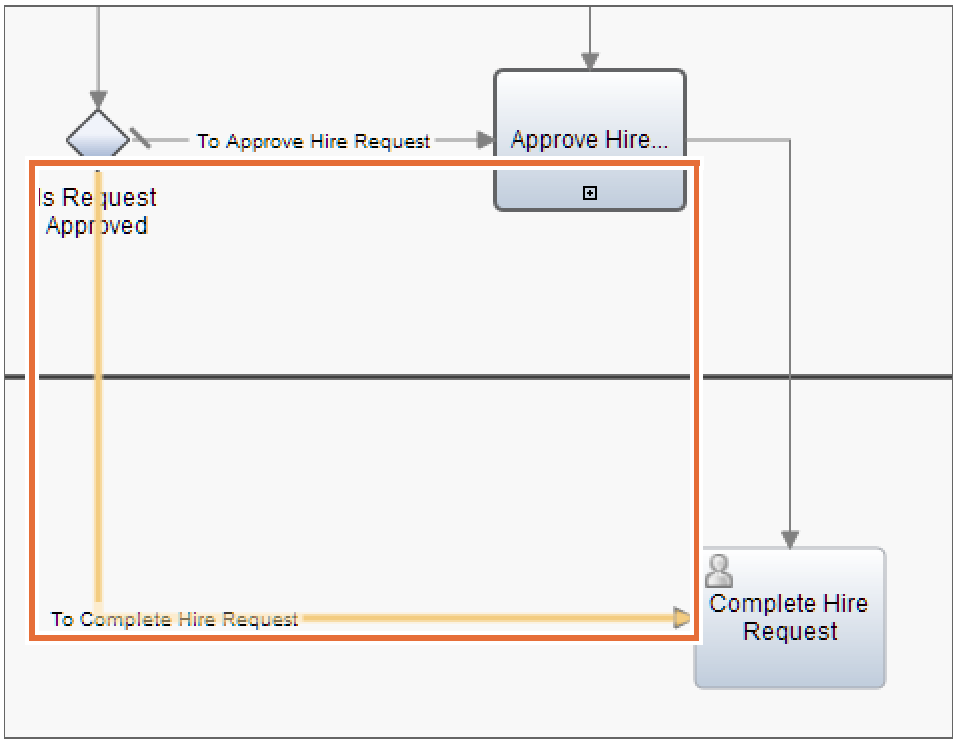

5. Connect and label the sequence flows.

__ a. Connect the **Approve New Hire Request** to the top of the **Is Request Approved** gateway.



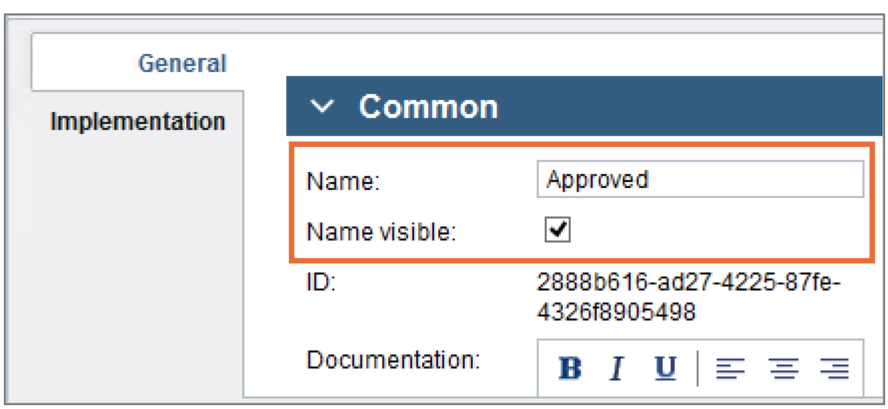
__ b. Connect the **Is Request Approved** gateway to the left anchor point of **Approve Hire Request**.


__ c. Connect the **Is Request Approved** gateway to the **Complete Hire Request** activity.



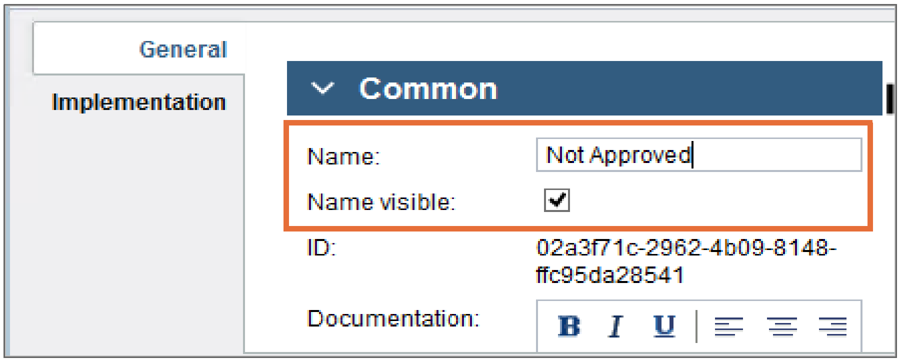

__ d. Select the flow between the **Is Request Approved** gateway and the **Approve Hire Request** linked process.

__ e. Rename the default name to `Approved` and verify **Name visible** is selected.

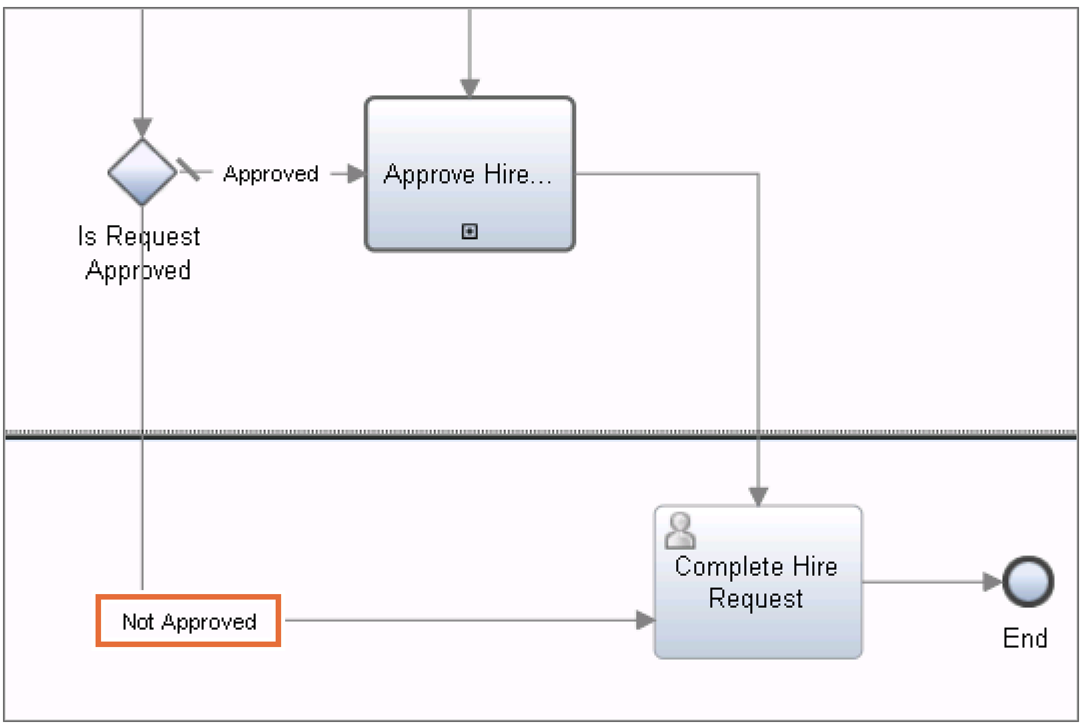


__ f. Select the flow between the **Is Request Approved** gateway and **Complete Hire Request**.

__ g. Rename the default name to `Not Approved` and verify **Name visible** is selected.




The flow displays the label:

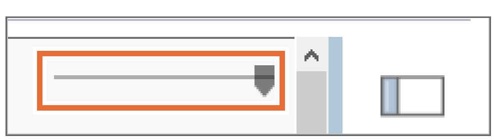



__ h. Save your work.

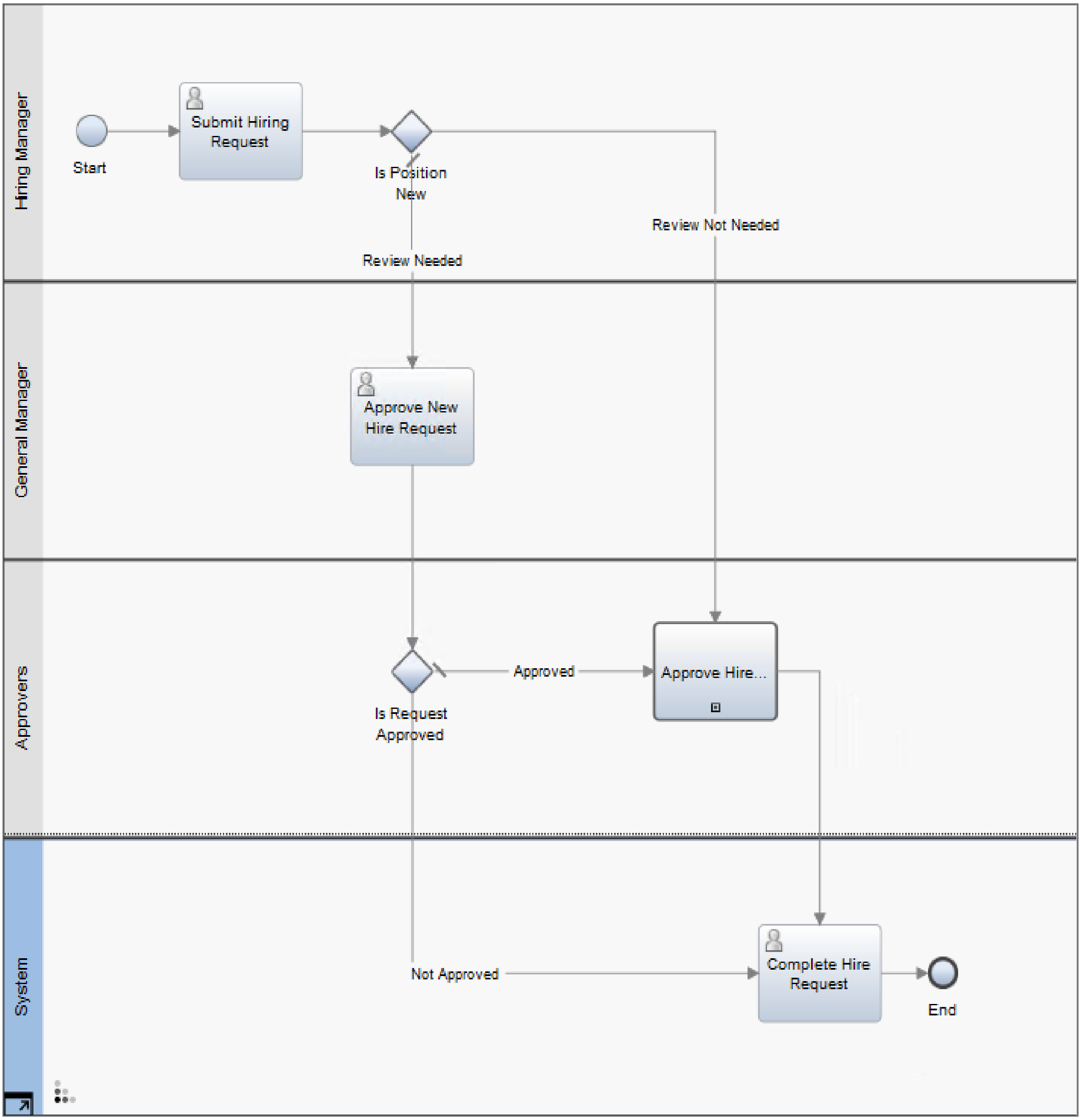
__ 6. Verify the process model

If you have a colleague or others nearby, see whether you can explain your new model to them to simulate a playback. Being able to clearly communicate the process is key when implementing business process management.

__ a. Use the zoom slider at the upper right of the canvas to view the whole process if it doesn't fit on the screen.



__ b. Ensure that the **Hiring Request Process** process looks similar to the screen capture before moving on:

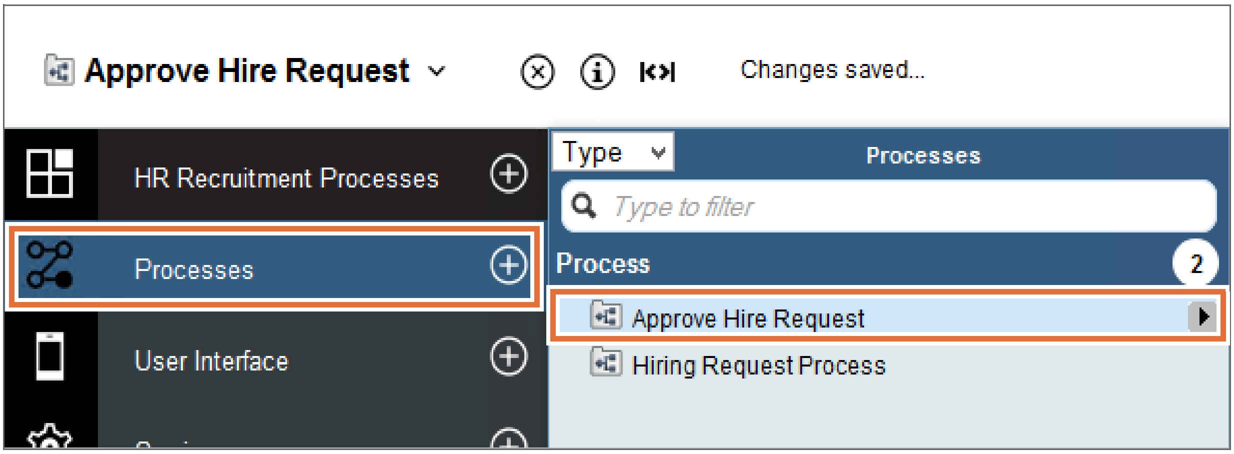


## *Part 2: Create gateways for the linked process*

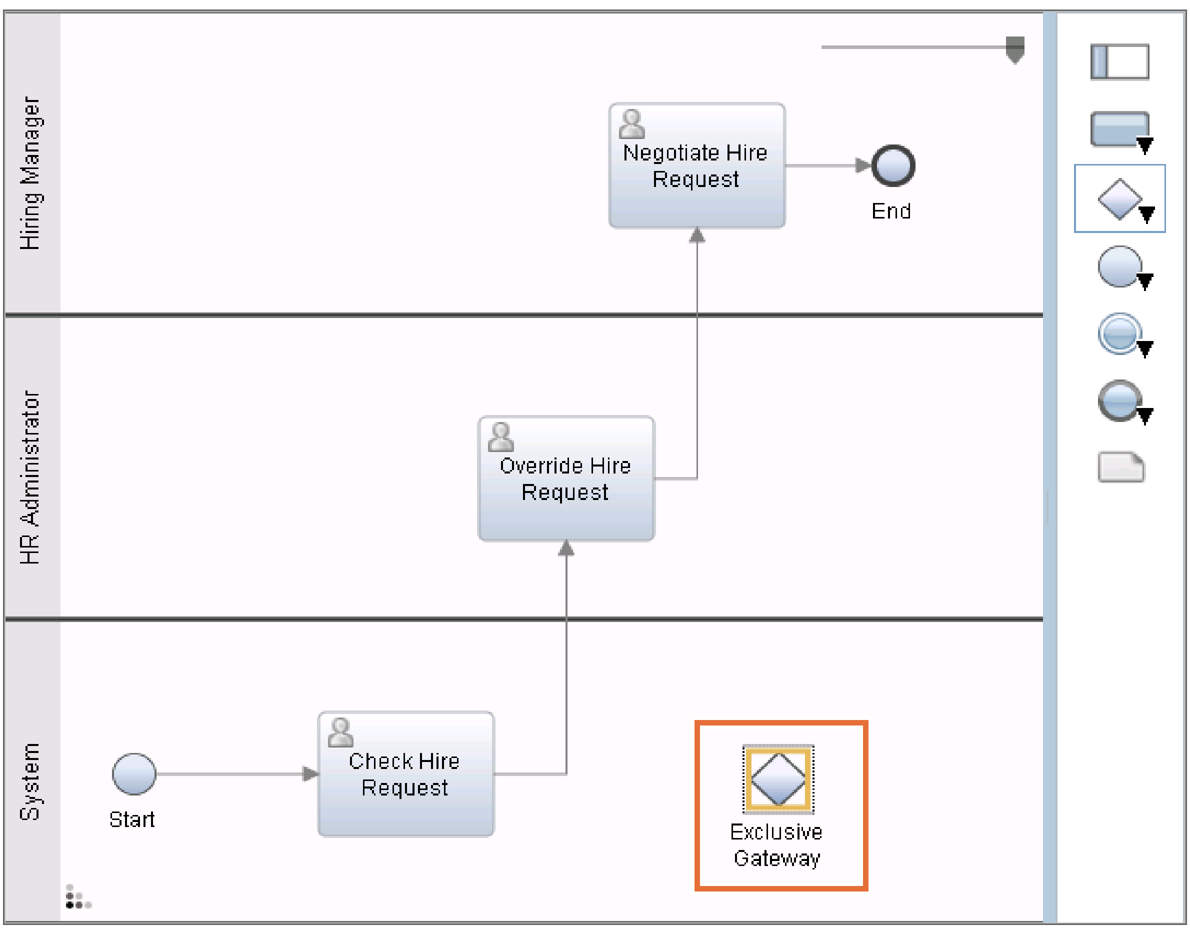
Add gateways to the nested process Approve Hire Request.

__ 1. Create the **Is Salary Compliant** gateway.

__ a. Click **Processes** in the library and click **Approve Hire Request** from the menu.




__ b. Drag a **Gateway** from the palette onto the canvas to the right of the **Check Hire Request** activity.



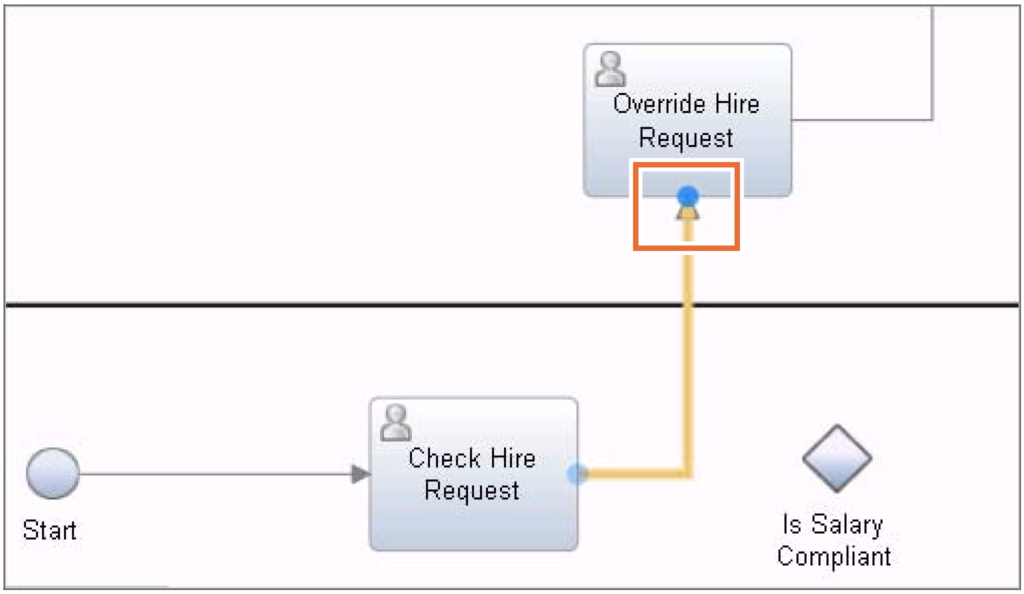

__ c. Rename the gateway to: `Is Salary Compliant`

__ d. Verify that the Gateway type is **Exclusive Gateway**.

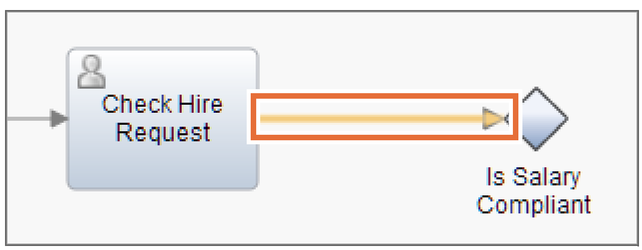
2. Connect and label sequence flows.

__ a. Select the sequence flow between **Check Hire Request** and **Override Hire Request**.

__ b. Hover over the tip of the sequence flow to see a blue point.




__ c. Select the blue point and drag the flow to connect **Check Hire Request** to the **Is Salary Compliant** gateway.

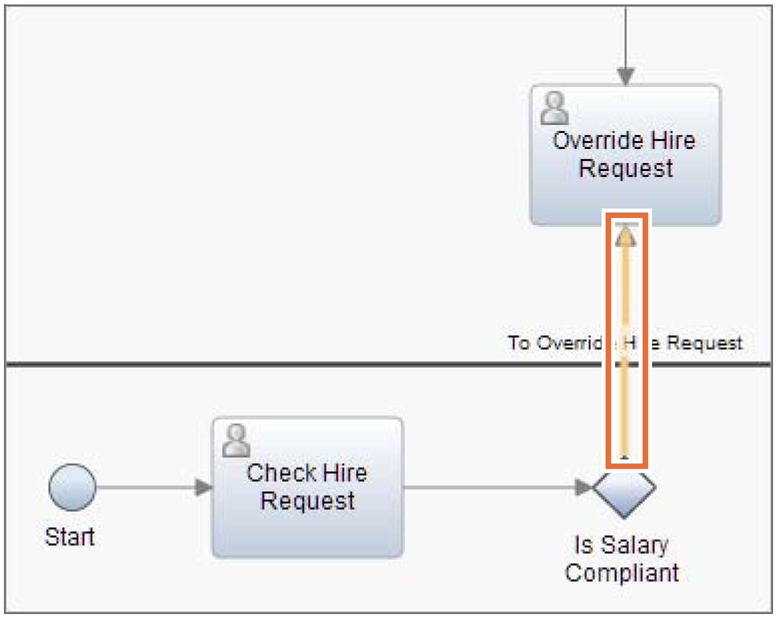



**Hint**

You can drag sequence endpoints from one activity to the other. To do this, highlight the endpoint until it displays as bright blue. Then, drag the endpoint to another activity and release.

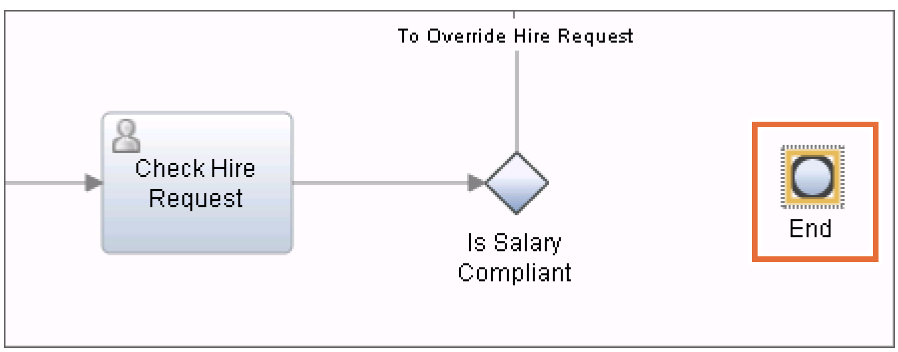

It can be tricky to highlight the endpoint correctly. As you become familiar with the Process Designer tool, you can choose the methods that work best for you. For instance, if it becomes too cumbersome to drag sequence lines from one activity to another, you can delete them and rebuild them.

__ d. Connect **Is Salary Compliant** to the bottom of **Override Hire Request**.

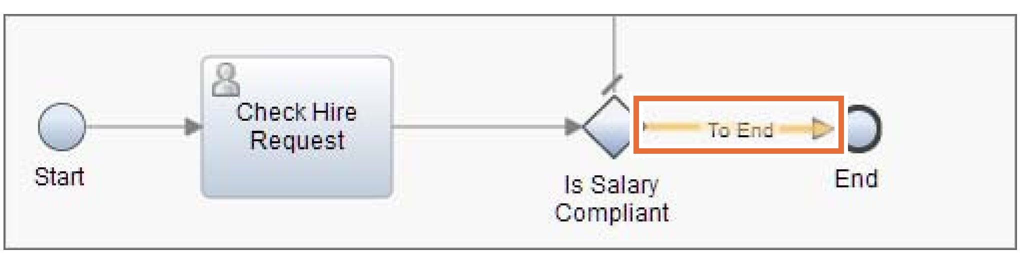



__ e. Remove the existing sequence flow between **Negotiate Hire Request** and the **End** event by highlighting the flow then pressing the Delete button on the keyboard.

__ f. Move the **End** event from the top of the diagram and place it to the right of **Is Salary Compliant**.




__ g. Connect **Is Salary Compliant** to the left anchor point of the **End** event.

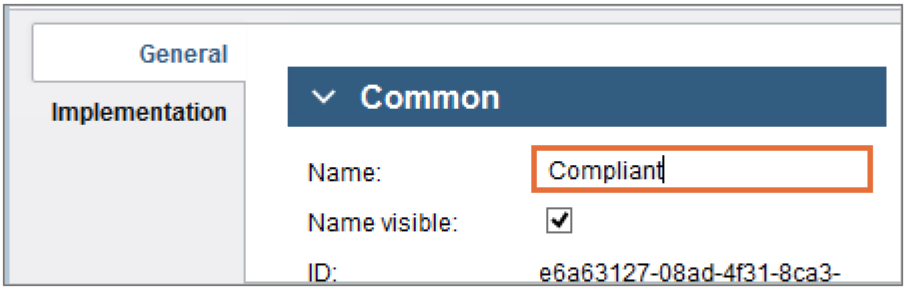



__ h. Select the flow between the **Is Salary Compliant** gateway and **Override Hire Request**.

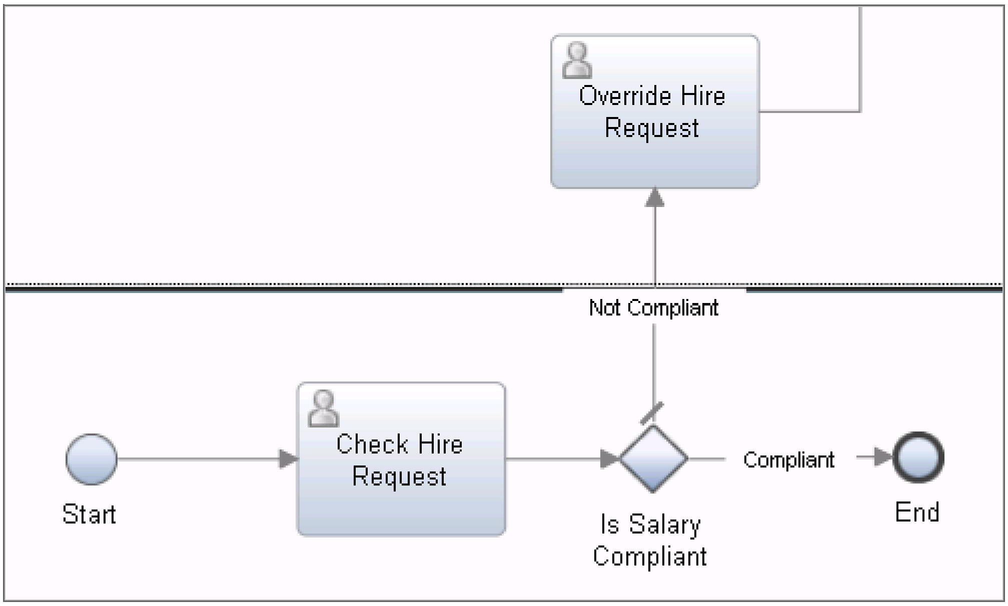
__ i. Rename the flow: `Not Compliant`

__ j. Select the flow between the **Is Salary Compliant** gateway and the **End** event.

__ k. In the **General > Common** section, rename the flow: `Compliant`




__ l. Verify your process. Your process should look similar to this screen capture.

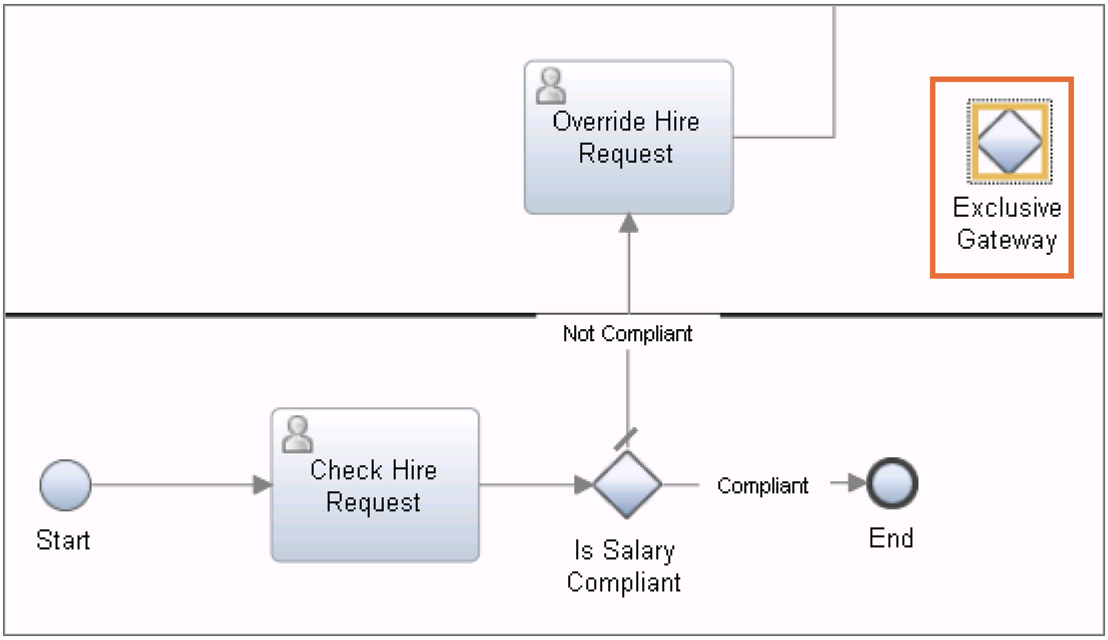



__ m. Save your work.

It is okay if the default sequence flows do not match. These are implemented later.

__ 3. Create the **Is Salary Acceptable** gateway.

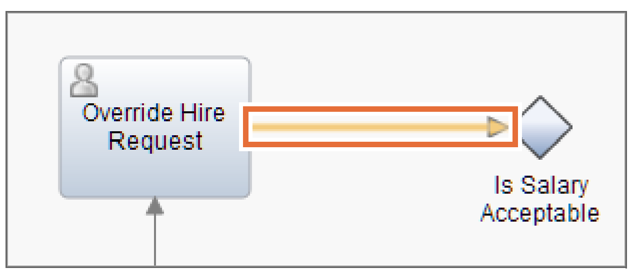
__ a. Drag a **Gateway** from the palette onto the canvas to the right of the **Override Hire Request** activity.




__ b. Set the gateway name to: `Is Salary Acceptable`

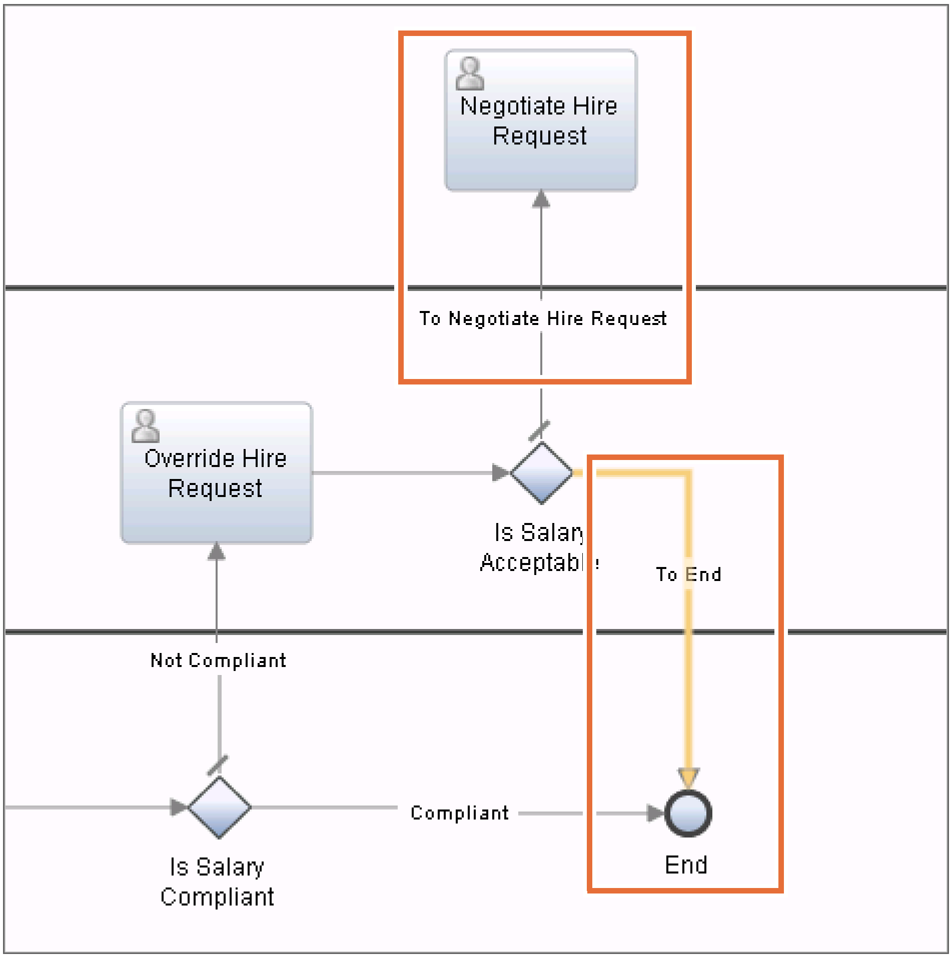
__ c. Delete the flow between the **Override Hire Request** and the **Negotiate Hire Request** activities.

__ d. Connect the **Override Hire Request** to the left anchor of the **Is Salary Acceptable** gateway.



__ e. Connect the **Is Salary Acceptable** gateway to the bottom anchor of **Negotiate Hire Request** and to the top anchor of the **End** event.

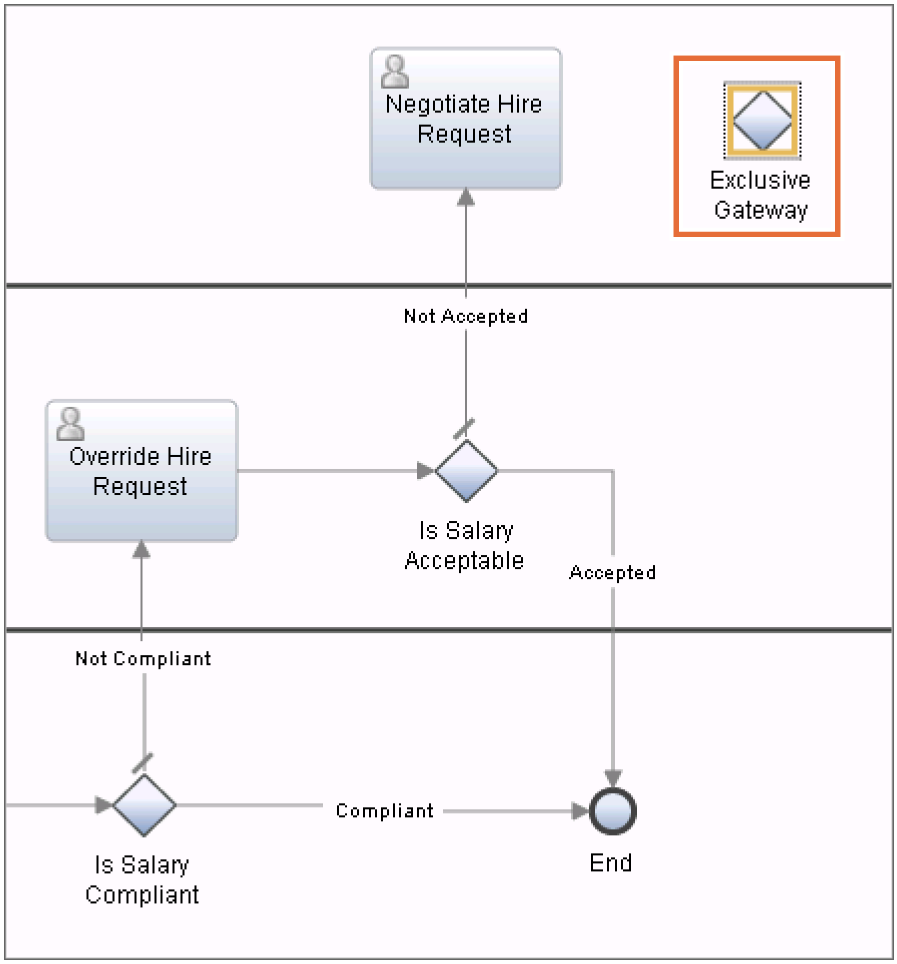



__ f. Rename the flow between the **Is Salary Acceptable** gateway and **Negotiate Hire Request** as: `Not Accepted`

__ g. Rename the flow between the **Is Salary Acceptable** gateway and the end event as: `Accepted`

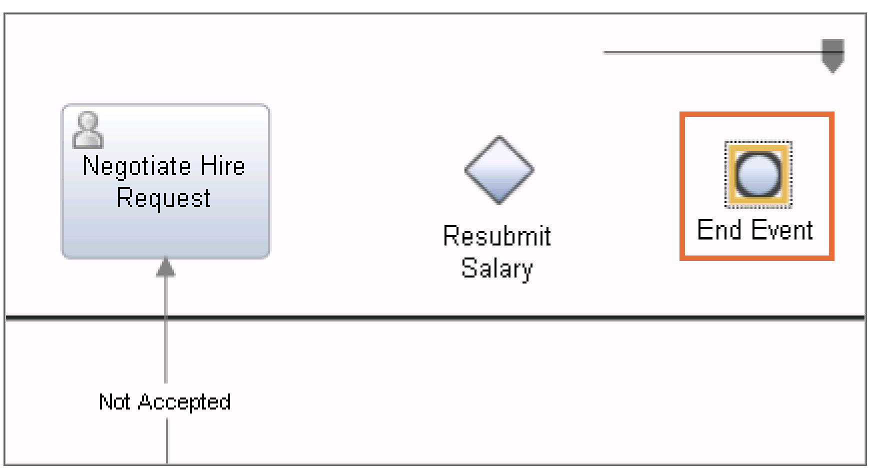
__ 4. Create the **Resubmit Salary** gateway.

__ a. Drag a **Gateway** from the palette onto the canvas to the right of the **Negotiate Hire Request** activity.




__ b. Rename the gateway to `Resubmit Salary` and verify that the Gateway type is **Exclusive Gateway**.

__ c. Drag an **End event** from the palette onto the canvas to the right of the **Resubmit Salary** gateway.

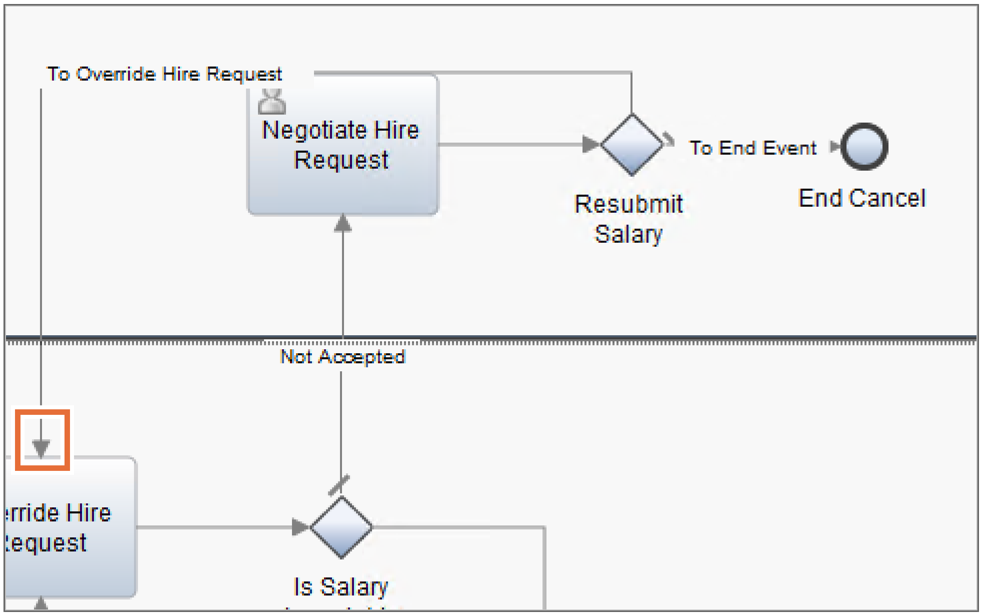


__ d. Rename the **End** event as: `End Cancel`

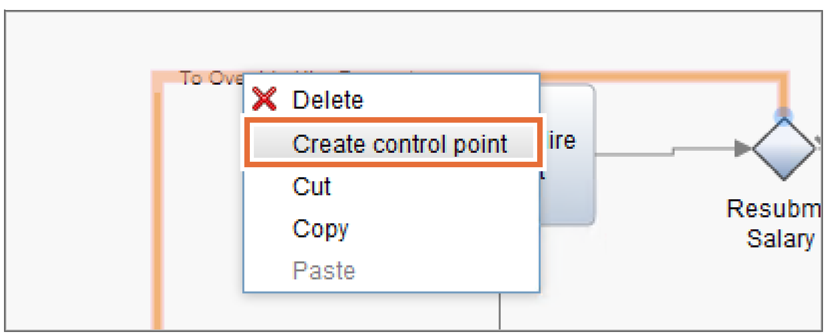
__ e. Connect **Negotiate Hire Request** to the **Resubmit Salary** gateway.

__ f. Connect the **Resubmit Salary** gateway to the **End Cancel** event.

__ g. Connect the **Resubmit Salary** gateway to the top of **Override Hire Request**.

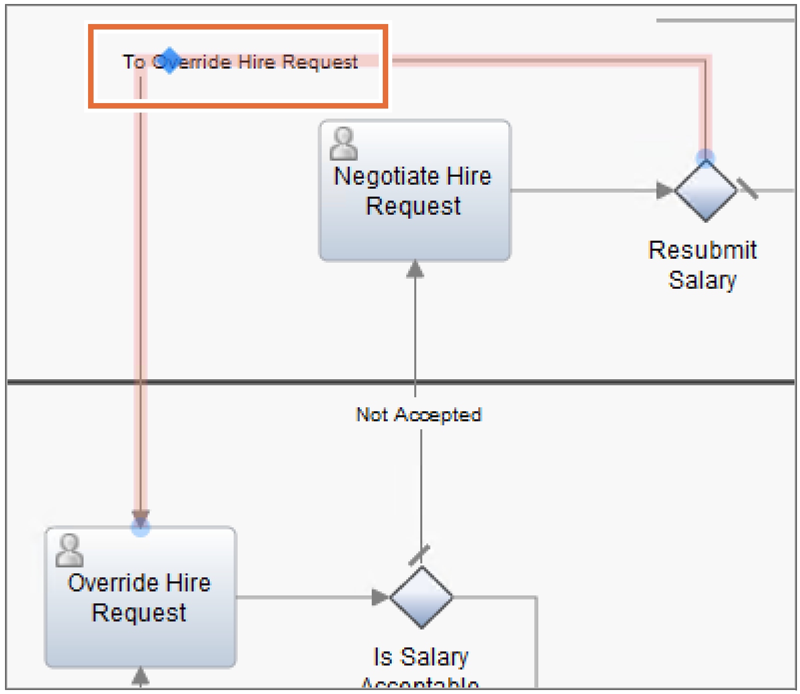



__ h. To fix the flow that overlaps the Negotiate Hire Request activity, select the **To Override Hire Request** flow and right-click the flow. Click **Create control point**.



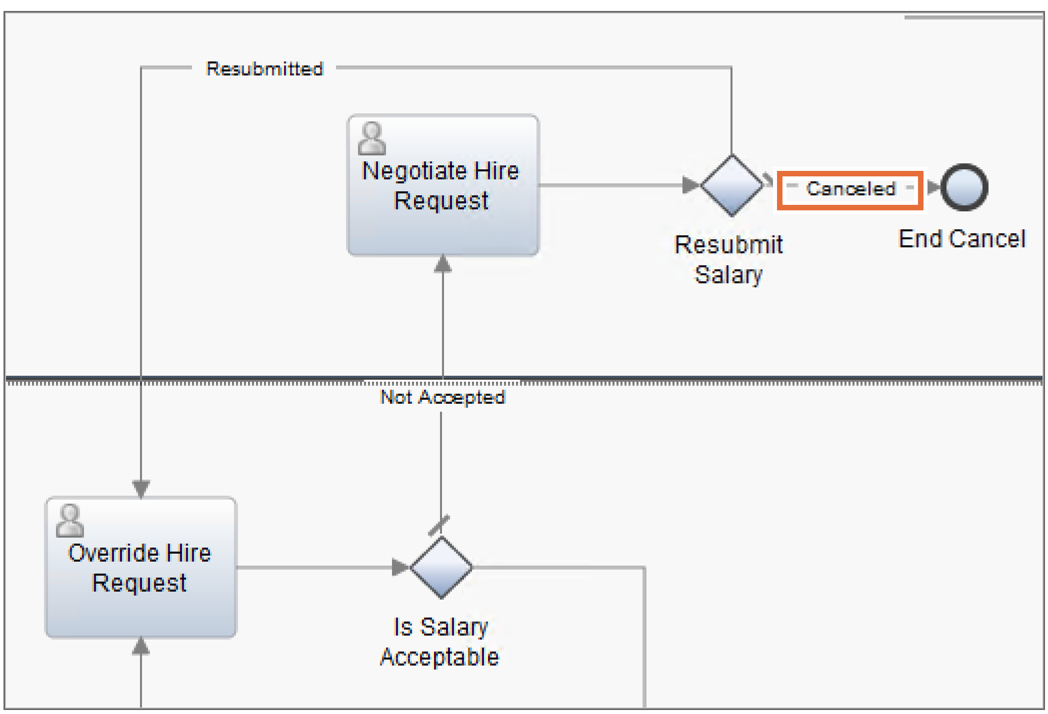

You use control points to manipulate the sequence flow path between two connection points. This helps to make the diagram more readable.

__ i. Drag the blue diamond to move the flow above the activity.




__ j. Name the flow between the **Resubmit Salary** gateway and **Override Hire Request** as `Resubmitted`.

__ k. Label the flow between the **Resubmit Salary** gateway and the **End** event as `Canceled`.




__ l. Straighten the sequence flows as necessary.

__ m. Save your work.

**Optional**

You completed part 2 of the 5, so you are now approximately half done with this exercise. Now would be a good time to take a 10-minute break.

## *Part 3: Modeling timer intermediate events*

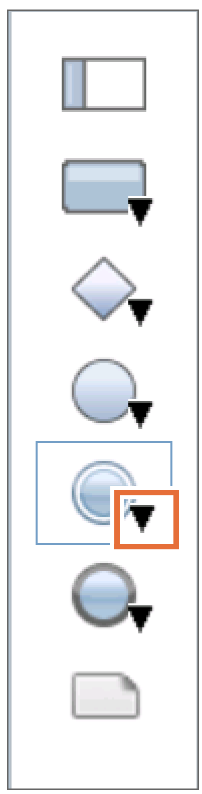
Examine the process requirements and determine where an escalation exists.

Refer to the core requirements for an escalation in section 2.10 that is listed in Exercise 2. The HR Administrator has 4 hours to complete the review. If the review is not completed within 4 hours, an email is sent to the HR Administrator notifying the HR Administrator of the missed deadline.

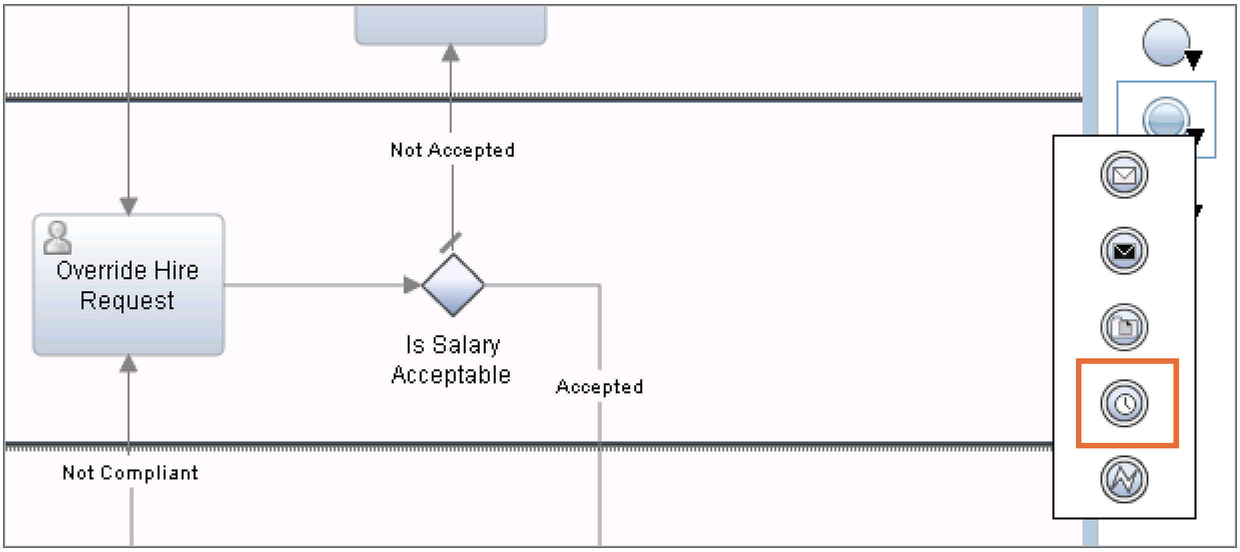
In this part of the exercise, you use the timer implementation option to model an escalation path in your processes. Using a timer intermediate event, you can specify a time interval after or before which some activity is conducted. The timer implementation option is available for events that are included in the process flow and events that are attached to an activity.

__ 1. Attach a four-hour timer event to the Override Hire Request activity.

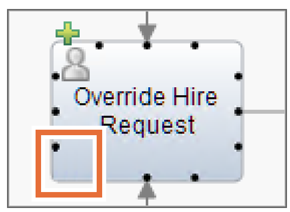
__ a. In the **Approve Hire Request** process, select the **drop-down icon** beside the **Intermediate** event.



__ b. Select the **Timer event** from the set of intermediate events.

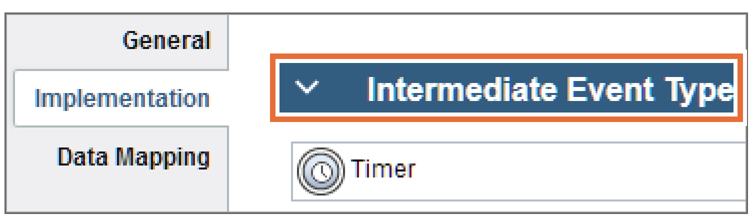



__ c. Drag the **Timer event** from the palette onto the lower-left anchor of the **Override Hire Request** activity.




## Important

If **Boundary Event Type** is not displayed and you see **Intermediate Event Type** in its place, you have not successfully anchored the Timer Event to the boundary of the activity.

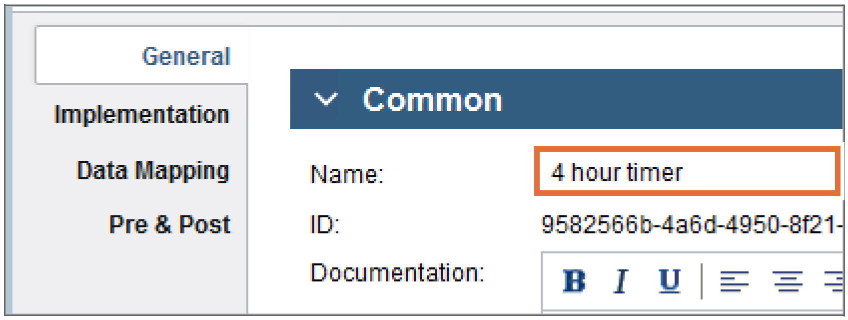



Sometimes attaching the timer event to the activity can be tricky. Look for the anchor points to appear on the activity while the mouse is hovering over the activity, then drag the timer to one of the anchor points. You see the green plus if you hover over one of the anchor points.

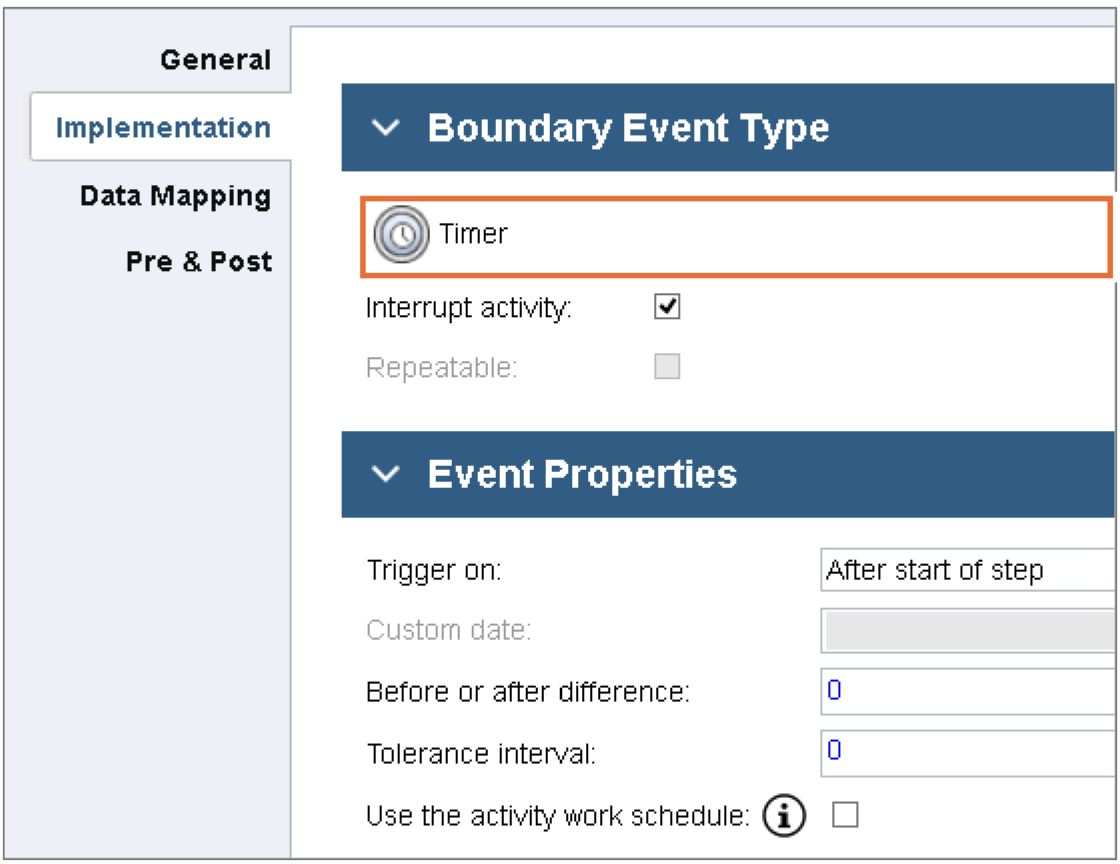
For this exercise, make sure that you drag the timer event onto the activity instead of outside of the activity. In this case, you want a boundary or attached intermediate event instead of a sequence flow intermediate event. If you correctly create an attached timer event, the name for the attached intermediate event is also changed to Boundary Event.

__ d. Select the **Timer** event.

__ e. In the **General > Common** section, change the name to: `4 hour timer`

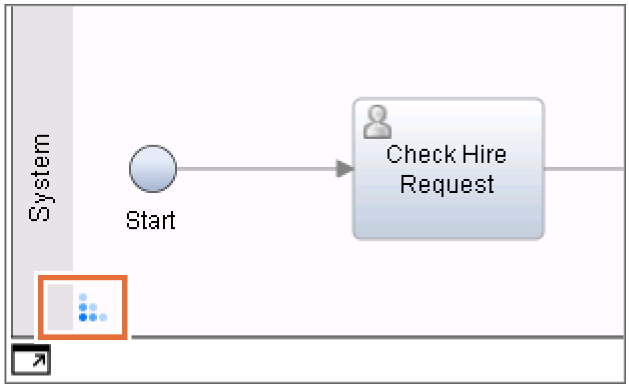



__ f. Select the **Implementation** menu. In the **Boundary Event Type** section, make sure that the Timer boundary event is selected. Keep the remaining default properties that were set during implementation.



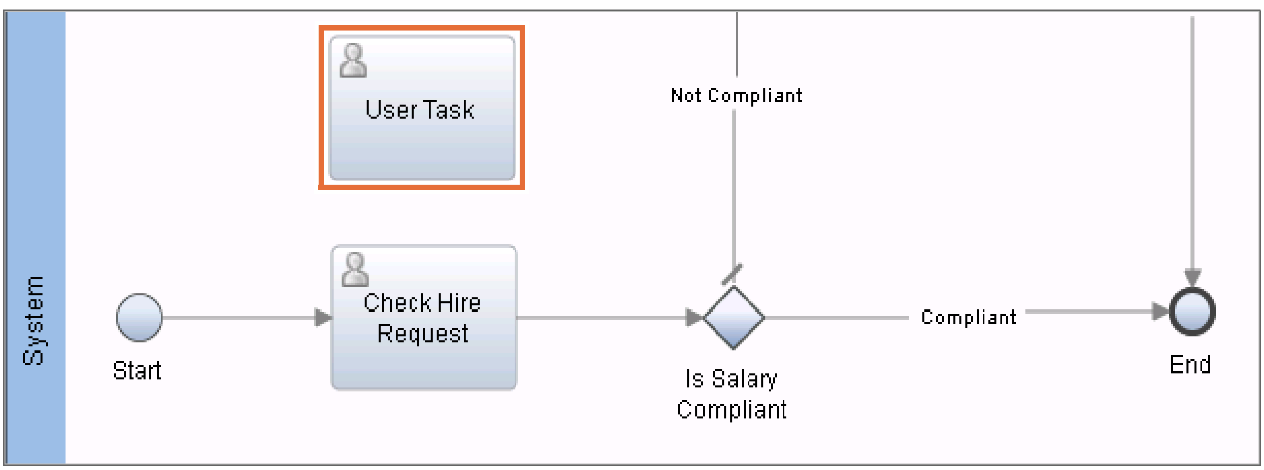
__ 2. Add the Notify HR Administrator activity.

__ a. Resize the **System** lane. To do this, click and hold the left mouse button down, then drag the mouse outside of the lane to increase the height of the lane.



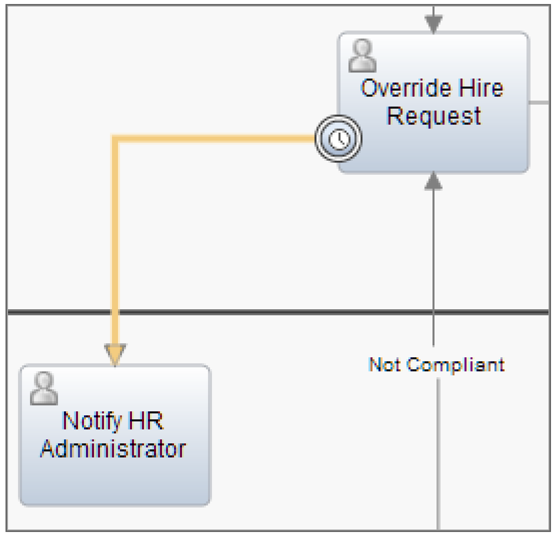

__ b. Drag a box around the activities, events, and the gateway in the System lane (lasso), and move them down together to make space for this new activity. Once items are highlighted, you can also use the arrow keys on the keyboard to move them.

__ c. Expand the Activity icon in the palette and drag **User Task** from the palette to the **System** lane. Place it above the **Check Hire Request** activity.

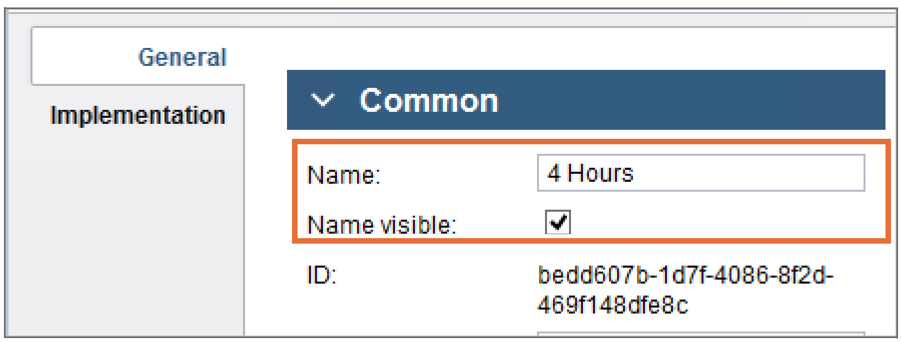



__ d. Rename the activity to: `Notify HR Administrator`

__ e. Connect the **Timer Intermediate Event** to the **Notify HR Administrator** activity.

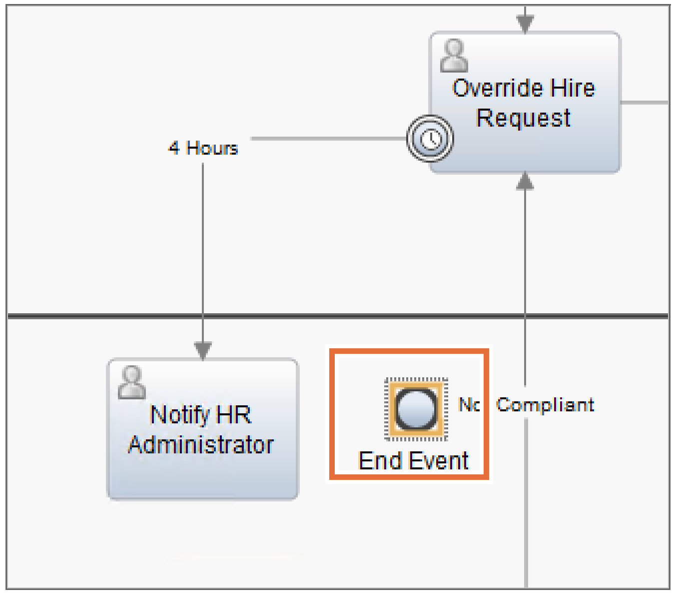



__ f. With the sequence line highlighted, name the flow `4 Hours` and select **Name visible** in the **General > Common** properties section.

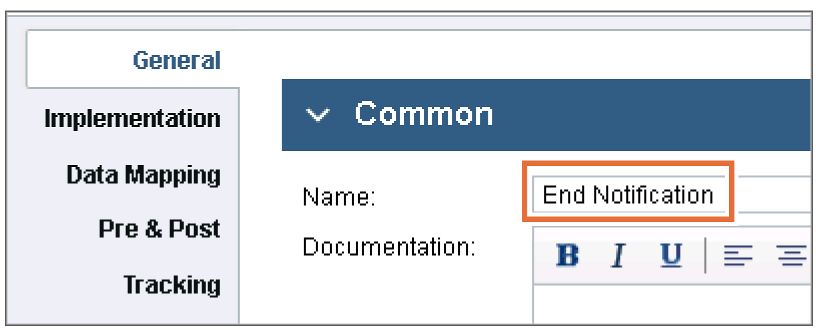


__ 3. Add an End Notification event.

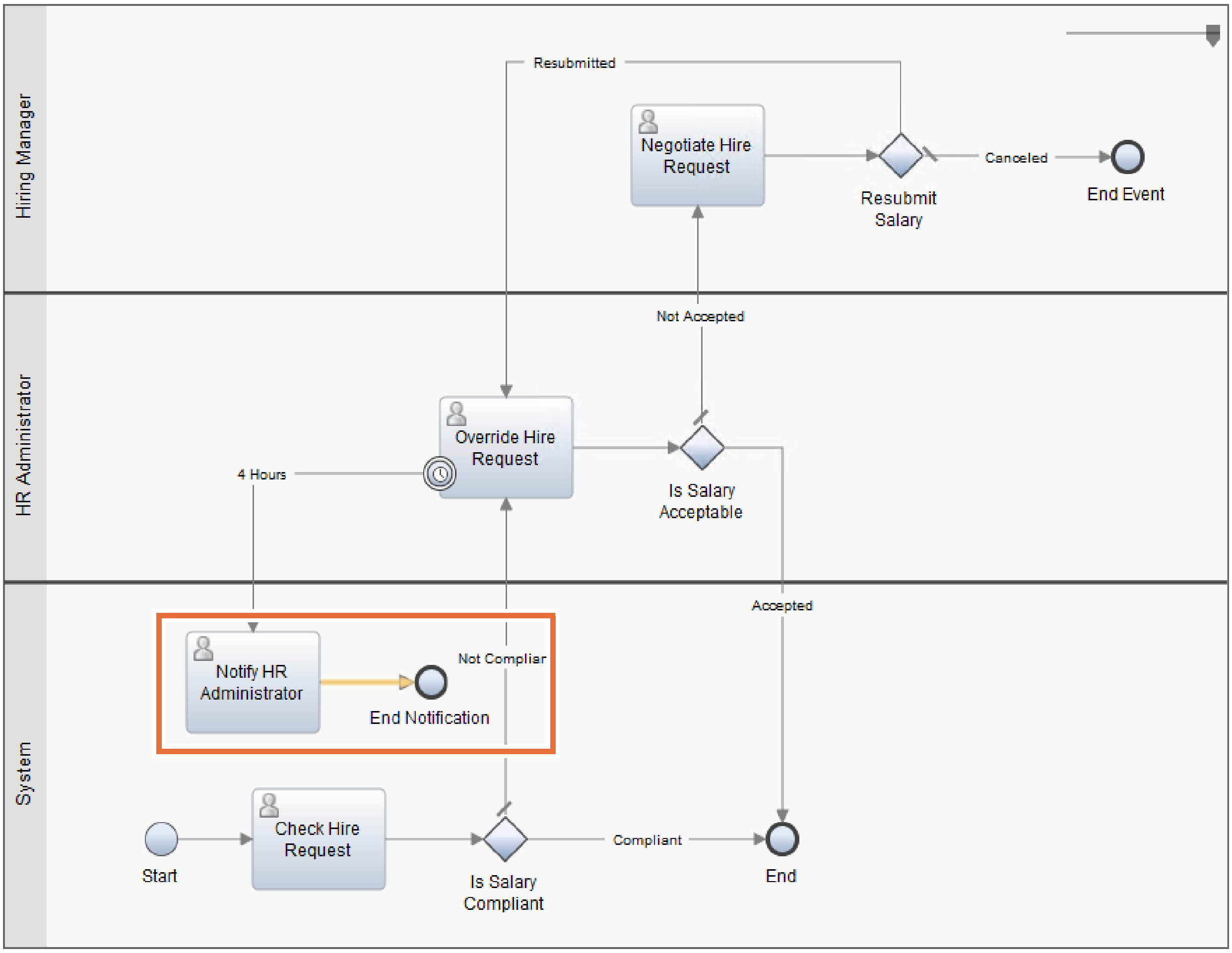
__ a. Drag an **End Event** from the palette onto the canvas to the right of **Notify HR Administrator**.




__ b. Rename the End event to: `End Notification`



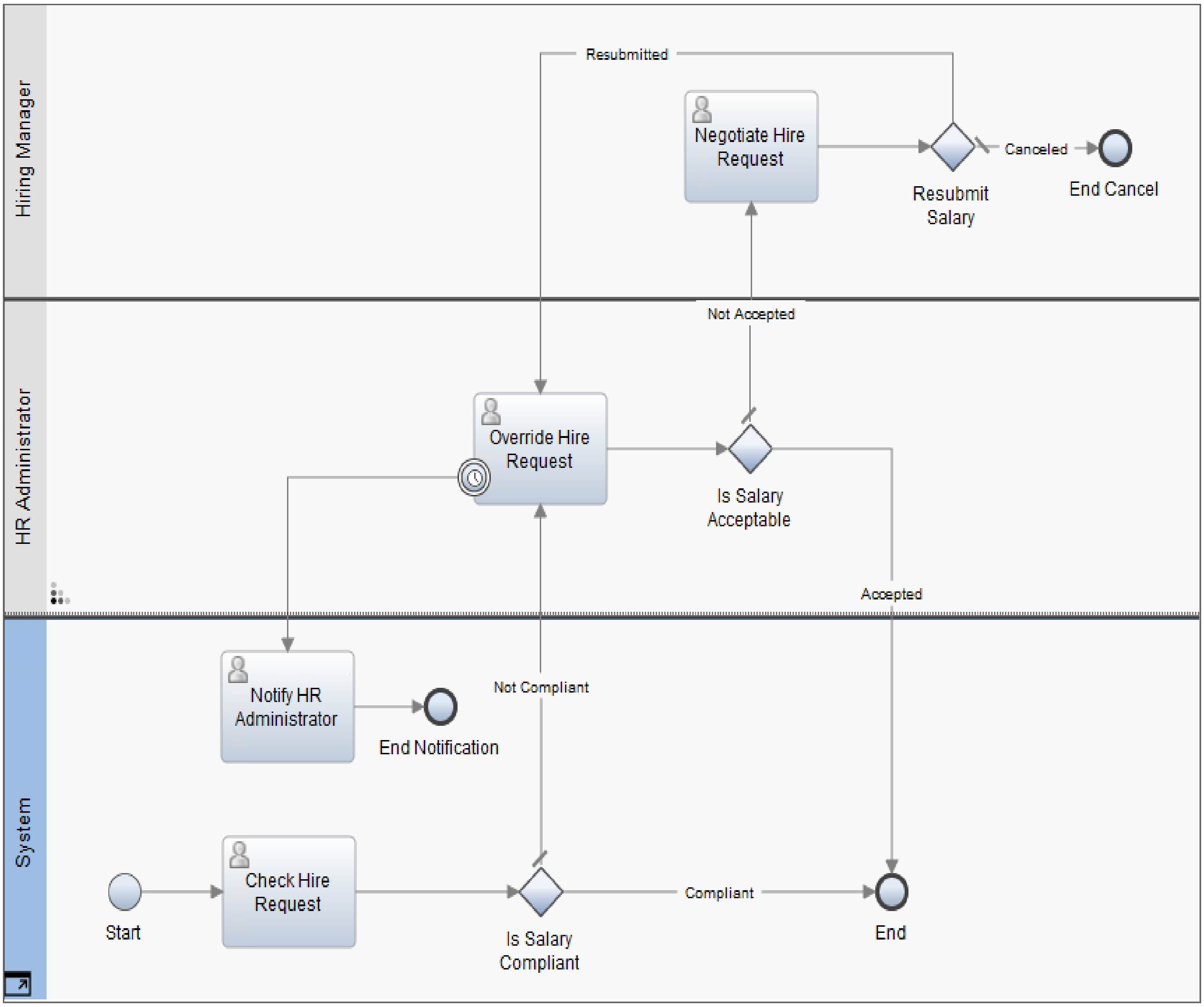
__ c. Connect the flow from **Notify HR Administrator** to **End Notification**.




__ d. Save your work.

__ 4. Verify the process model

__ a. Ensure that the **Approve Hire Request** process looks similar to the diagram before moving on:



## *Part 4: Add new process requirements to the process*

It is common to clarify requirements throughout the playback sessions. During the initial playback, the vice president mentions new requirements.

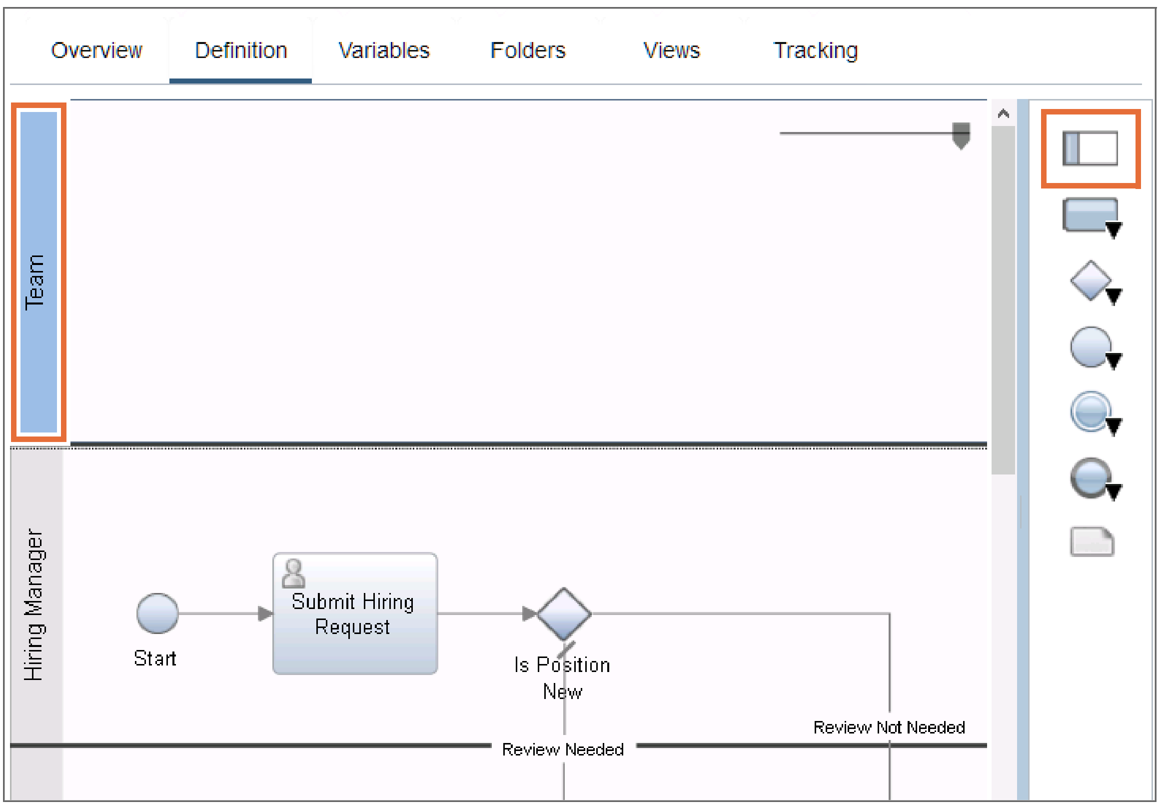
Several job postings contain wording that violates a new set of hiring laws. To address this issue, the vice president suggests that a lawyer must review every job post at some point before the job opening is released and posted to the public. The lawyer checks for legal compliance and edits if necessary.

Examine the requirement and come up with the solution for this requirement.

__ 1. Add a lane in the Hiring Request Process for the legal team.

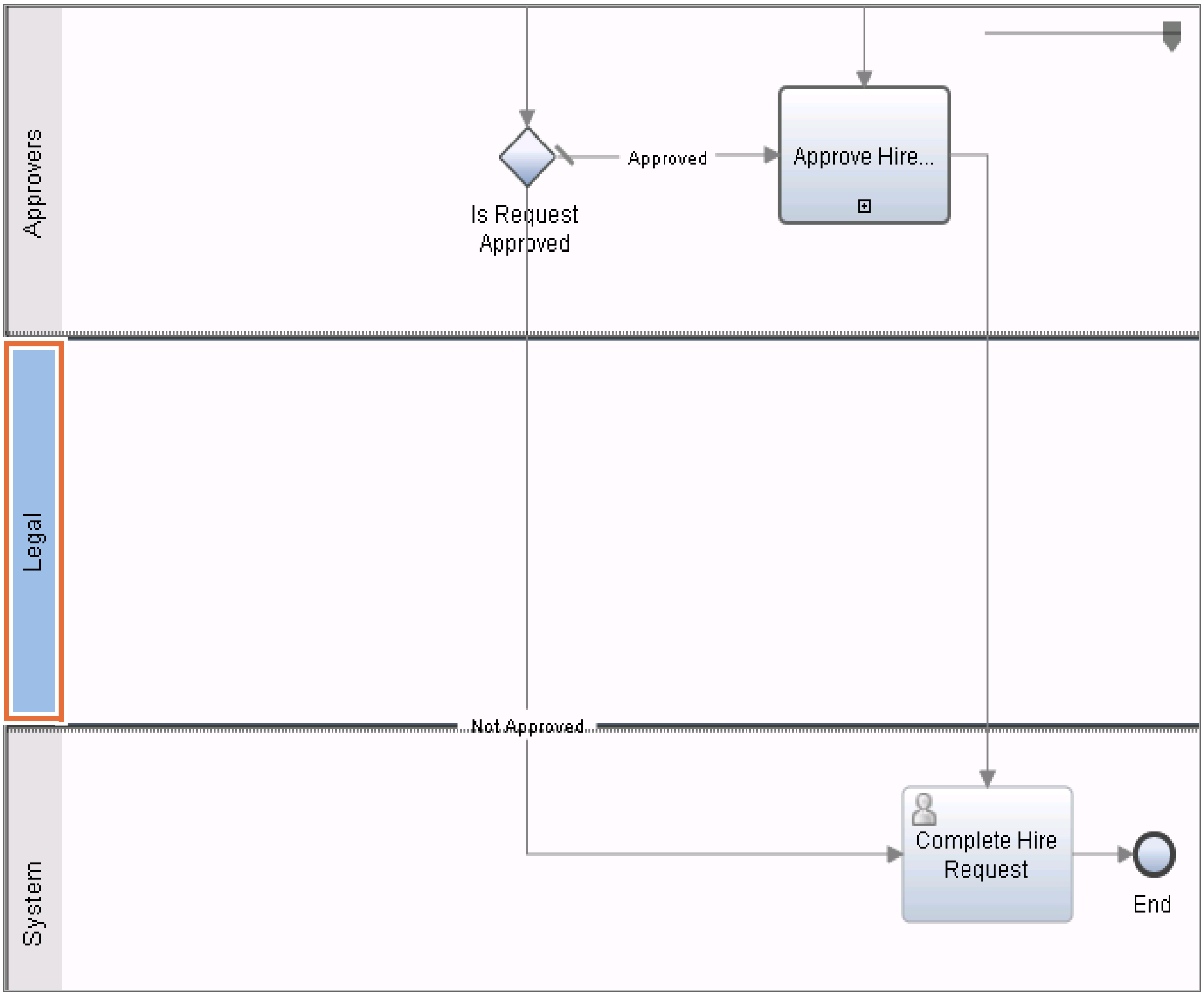
__ a. Open the **Hiring Request Process**.

__ b. Drag a **Lane** from the palette to the canvas above the Hiring Manager lane.




__ c. Name the lane: `Legal`

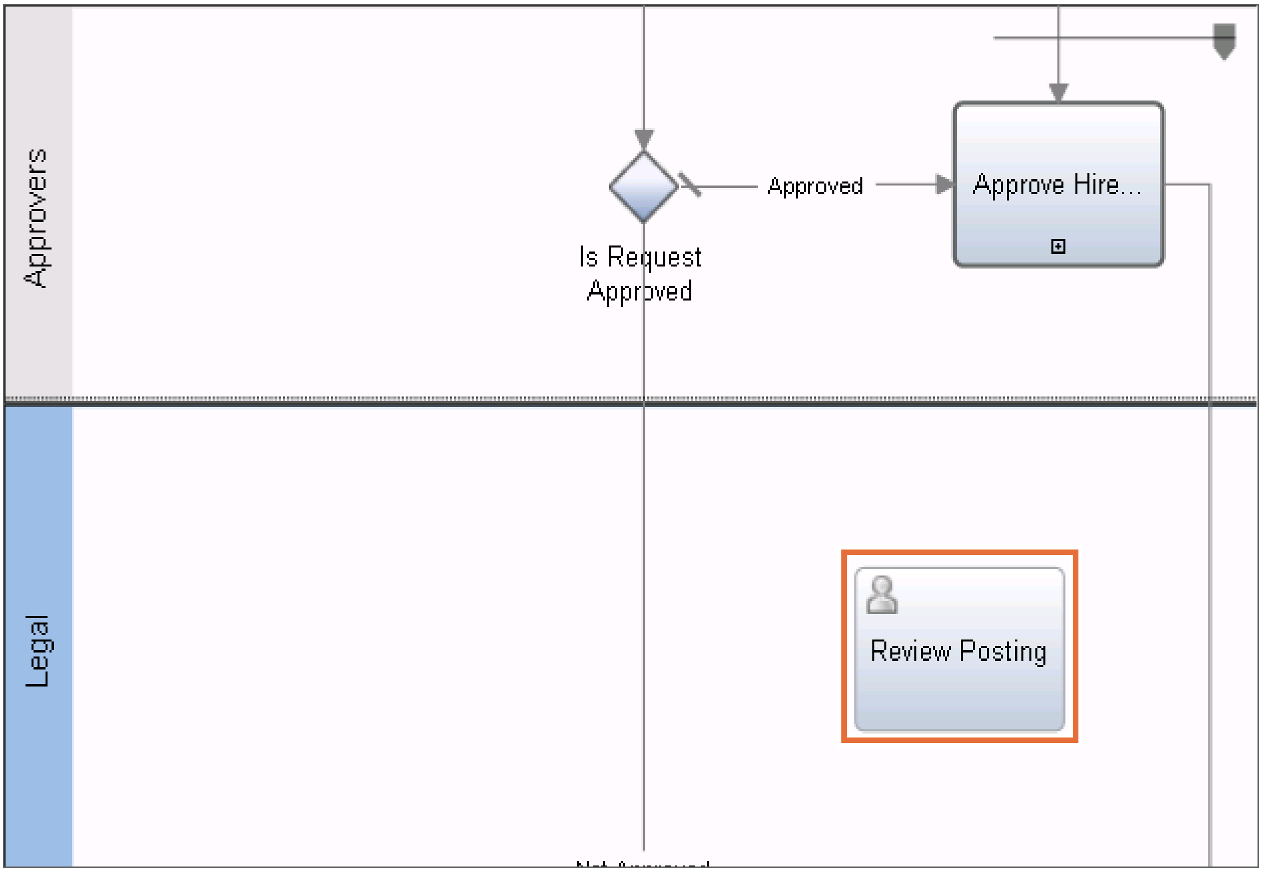
__ d. Right-click the **Legal** lane. Click **Move lane down.** Repeat moving the lane down three times until the Legal lane is between the **Approvers** and **System** lanes.




__ 2. Create an activity for a lawyer to review the hire request.

__ a. Drag **User Task** from the palette to the **Legal** lane in the process.

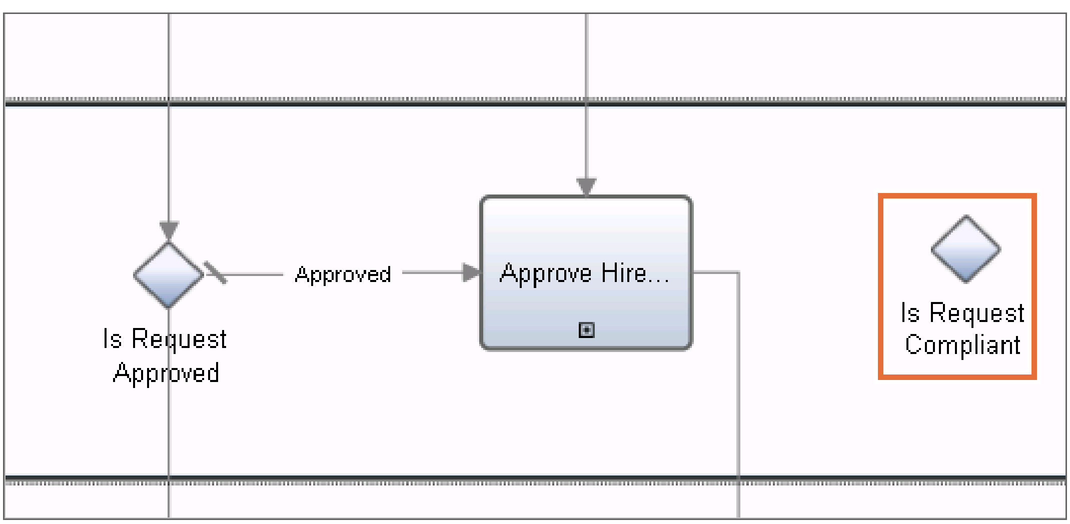
__ b. Name the activity: `Review Posting`




__ 3. Add a gateway to check the compliance of the request.

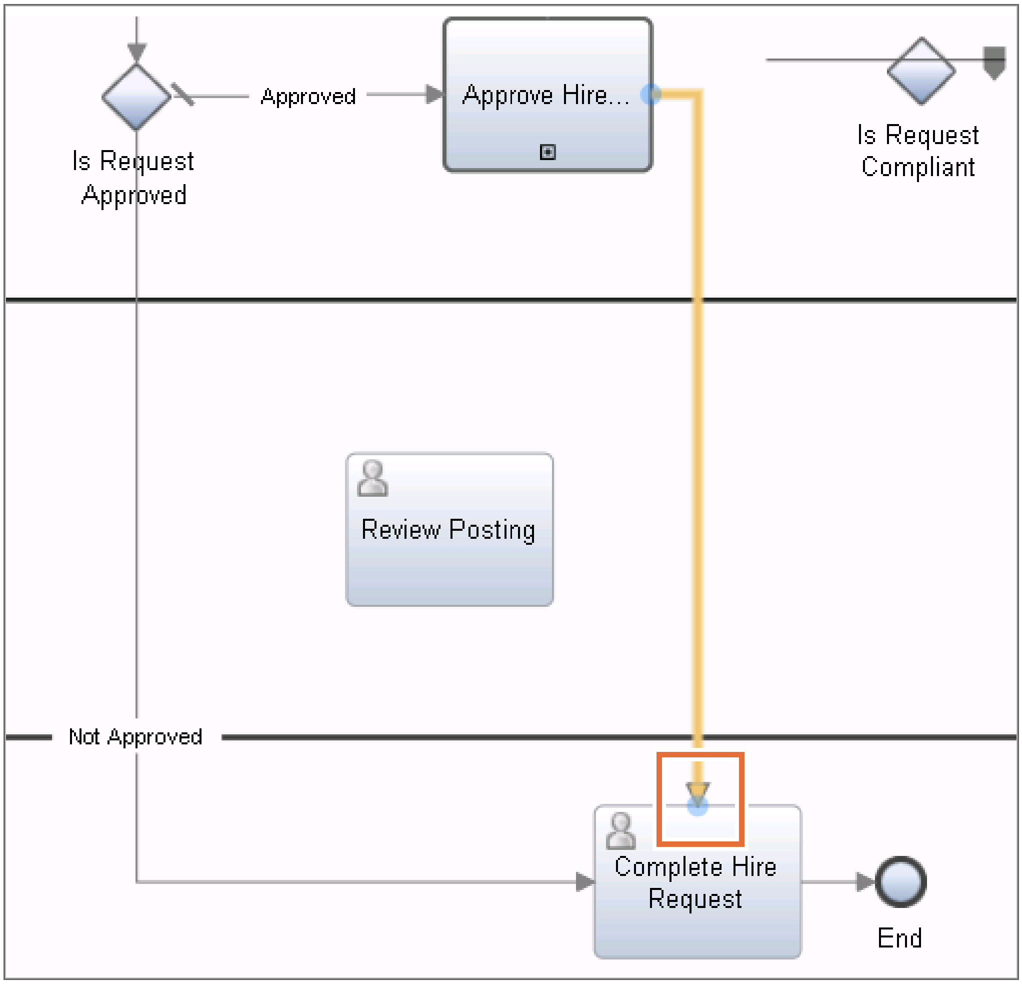
__ a. Drag a **Gateway** to the right of **Approve Hire Request**.

__ b. Rename the gateway to: `Is Request Compliant`

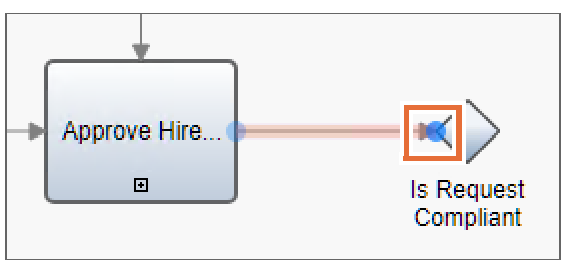


__ 4. Connect the **Review Posting** activity.

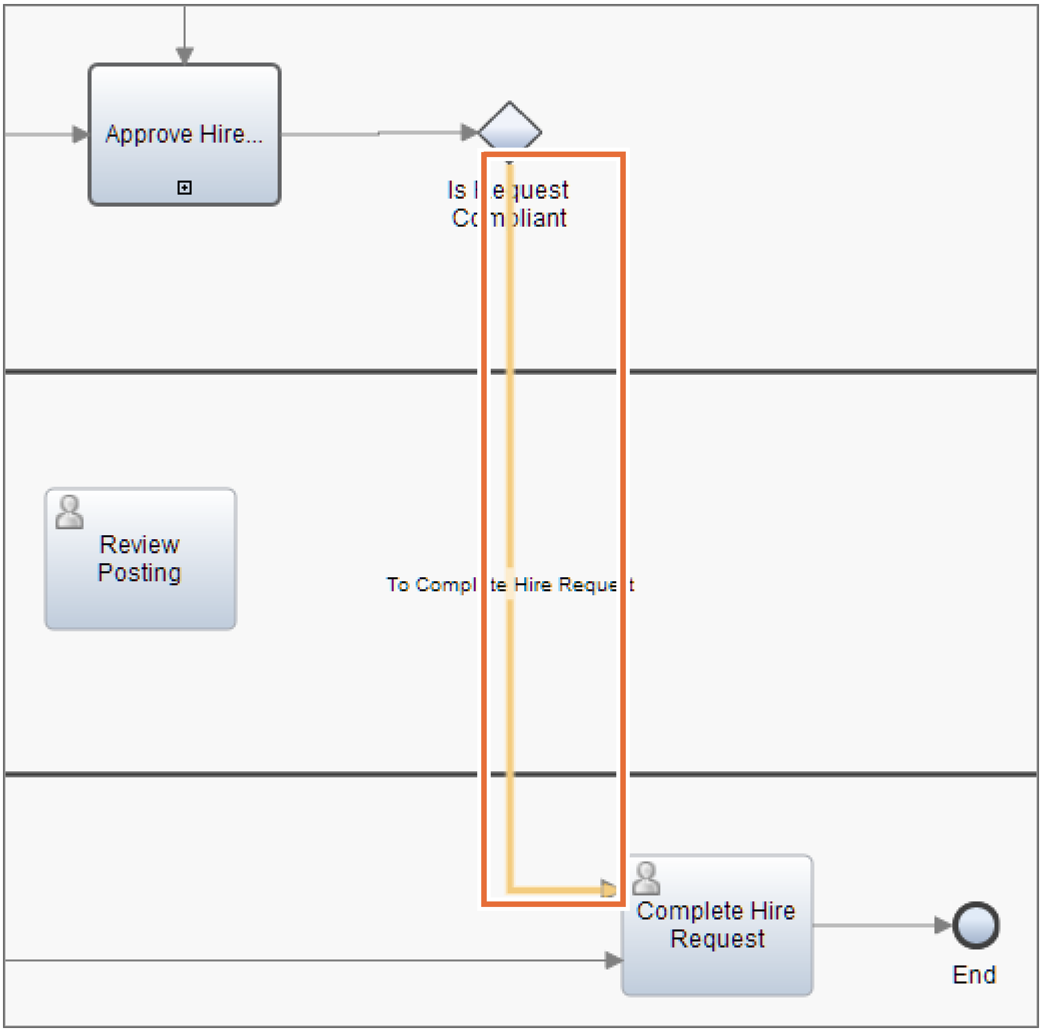
__ a. Select the sequence flow between **Approve Hire Request** and **Complete Hire Request** activities.




__ b. Drag the blue point that appears at the tip of the flow line at the **Complete Hire Request** activity to the **Is Request Compliant** gateway.

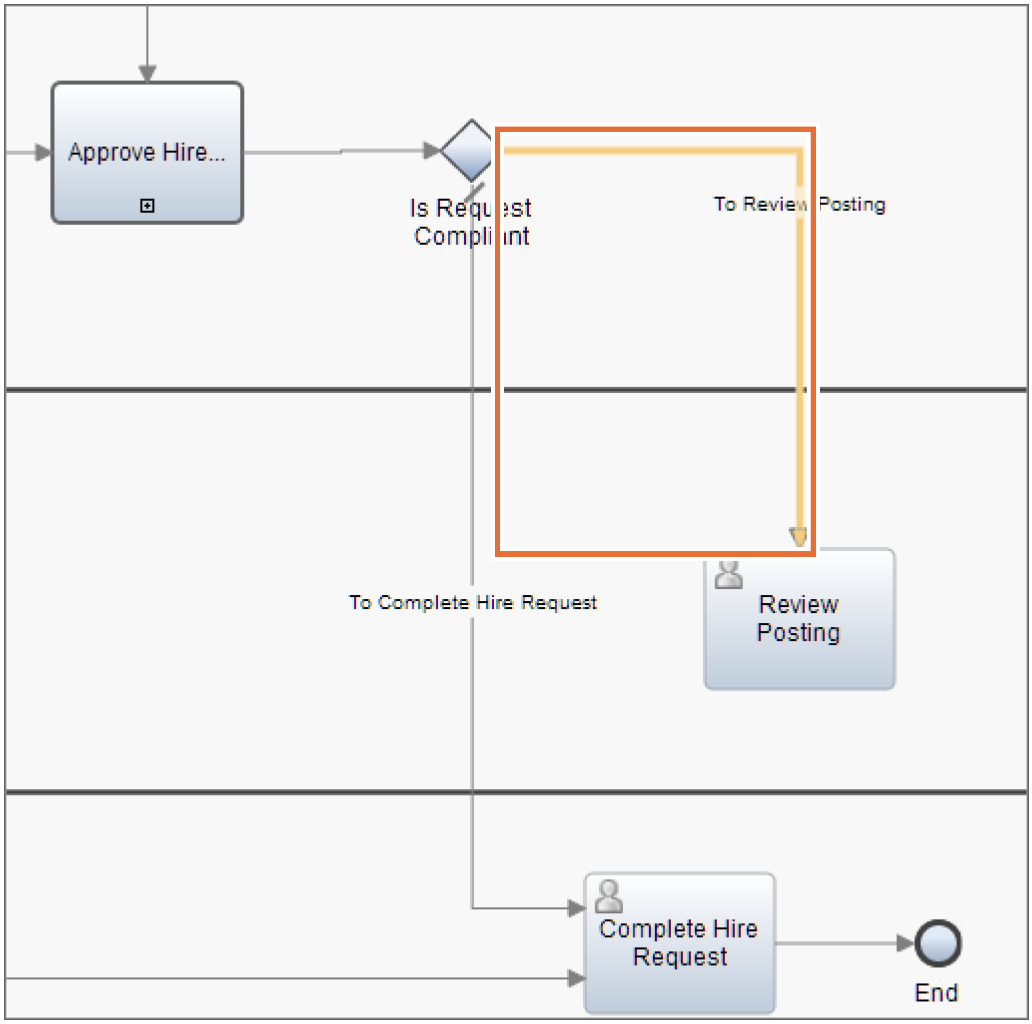


__ c. Connect the **Is Request Compliant** gateway to **Complete Hire Request**.

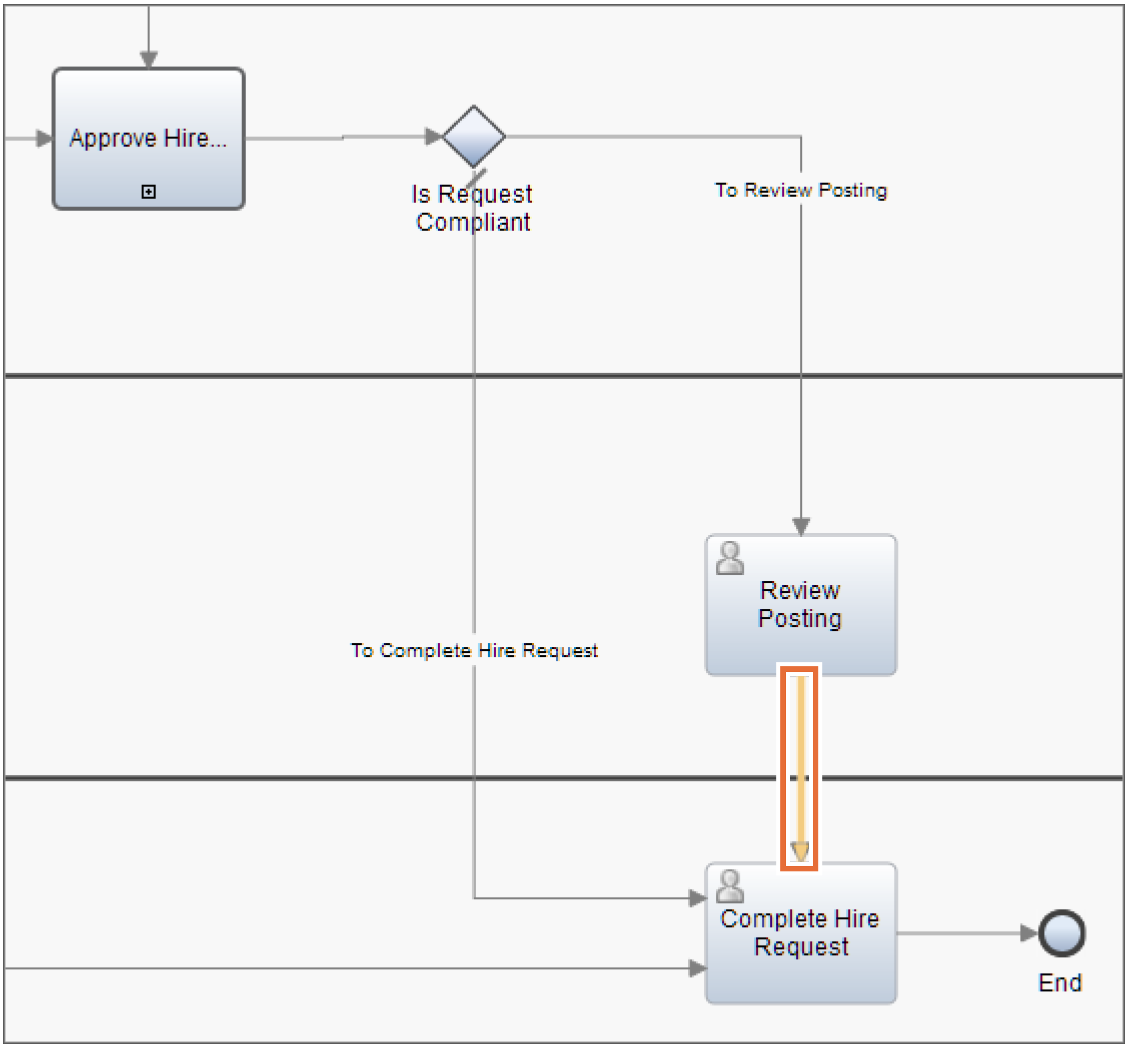



__ d. Drag the **Review Posting** activity above the **Complete Hire Request** activity.

__ e. Connect the **Is Request Compliant** gateway to the **Review Posting** activity.



__ f. Create a flow from **Review Posting** to **Complete Hire Request**.



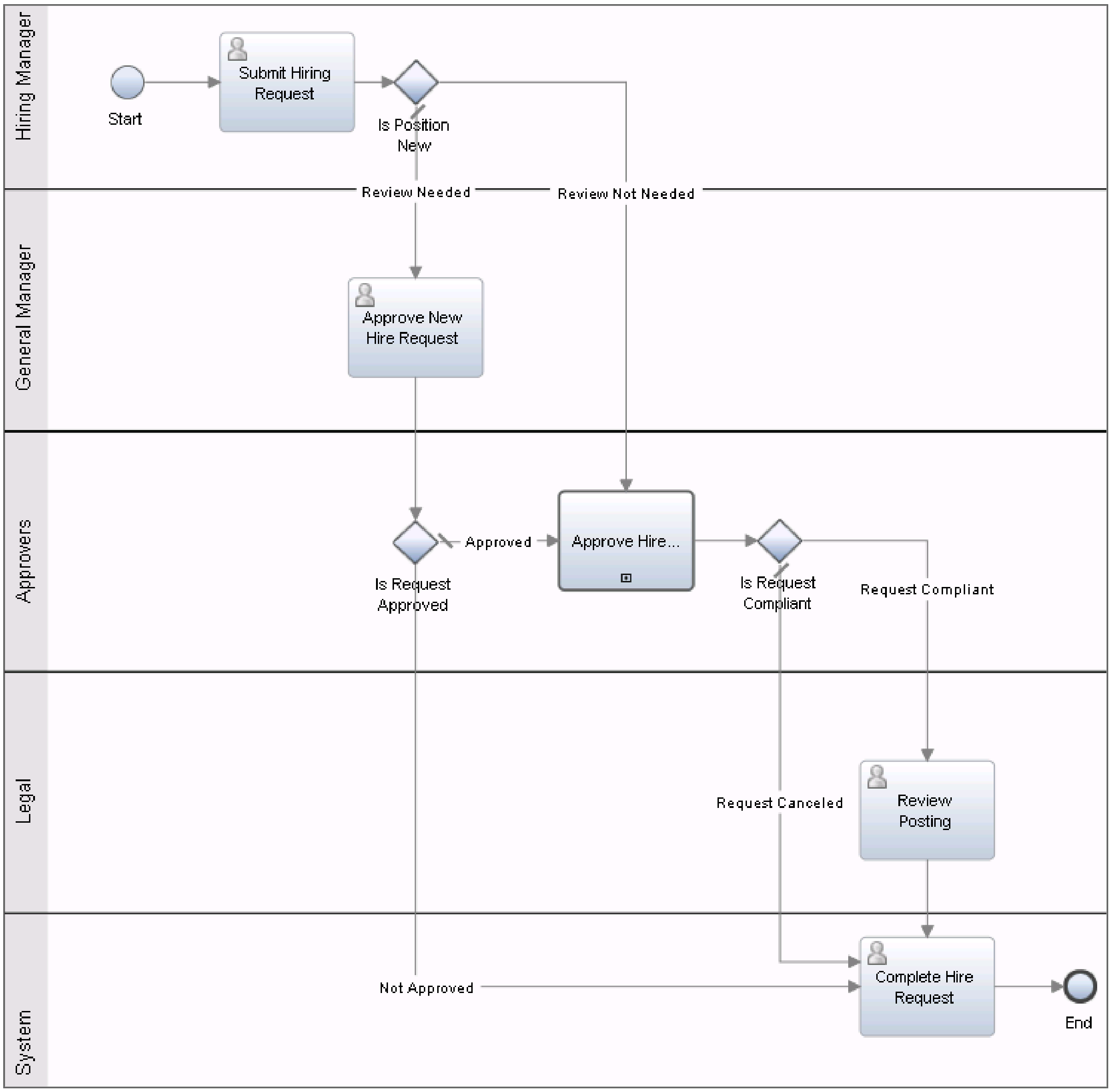

__ g. Rename the sequence flow between **Is Request Compliant** and **Complete Hire Request** as: `Request Canceled`

__ h. Rename the sequence flow between **Is Request Compliant** and **Review Posting** as: `Request Compliant`

__ i. Save your work.

__ 5. Verify the process model

__ a. Ensure that the new process looks similar to the diagram before moving on:




**Important**

After adding an element to your process, it is a good practice to verify the process a second time. When you encounter process problems while developing for the Playback 1 process phase, it is common to return to Playback 0 to correct the problems.

Playback 0 is complete.

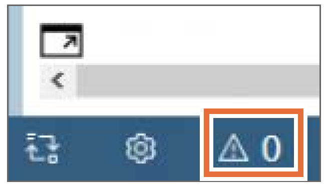
## *Part 5: Validate the process model and take a snapshot*

Now that playback 0 is complete, it is a good time to validate the process application and take a snapshot.

__ 1. Validate the process application.

While you are developing, Process Designer keeps track of any validation errors and displays the number of errors in the status bar at the bottom.

__ a. Review the status bar validation icon for any errors.

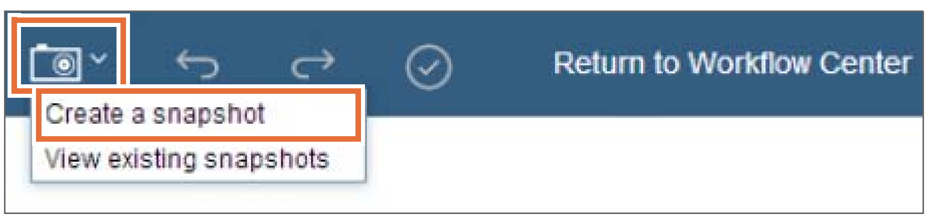



__ b. If you have validation errors, you should resolve them before moving onto the next playback. As you add coaches and services, this functionality becomes more important. For now, there should be no validation errors for the processes.

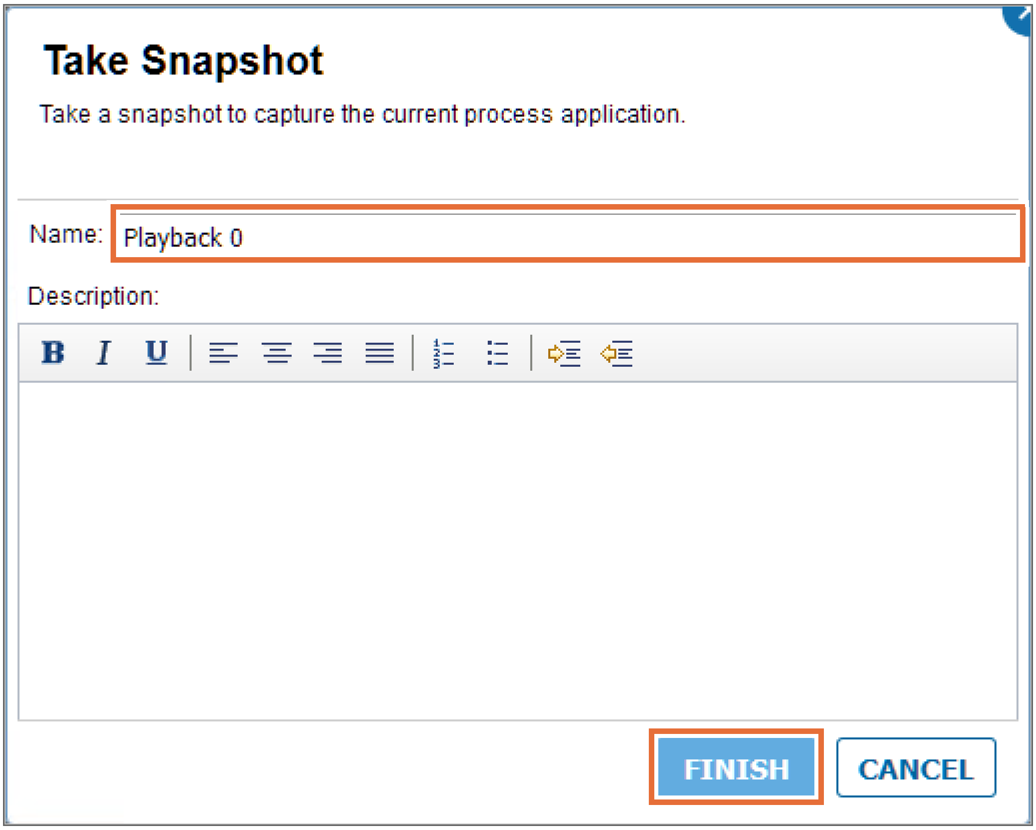
__ 2. Take a snapshot.

A snapshot captures the state of the library items within a process application at a specific point in time. Snapshots usually represent a milestone or are used for playbacks or for installation.

__ a. Click the **camera icon** in the upper right corner and select **Create a new snapshot**.



__ b. Enter `Playback 0` as the name and click **FINISH**.




__ c. Click **OK** when the snapshot is created.

You have completed this exercise.

The first part of this exercise involved adding gateways to the process in IBM Process Designer. Next, you modeled the appropriate sequence flows for each gateway. Then, you added a timer intermediate event for a process requirement. Next, you validated the business process, and you took the playback feedback and added a requirement to the process. All the stakeholders agreed to the revised version of the process, and the process owner along with the lead developer suggested that the team transition to Playback 1.

## End of Exercise

# Exercise 4. Playback 1: Controlling process flow with business data

## Estimated time

01:30

## Overview

In this exercise, you create assets that are required during Playback 1 controlling the process flow of the process lifecycle. You create variables, implement timer intermediate events, establish routing, and implement exclusive gateways.

## Objectives

After completing this exercise, you should be able to:

- Create simple variables in a process
- Implement timer intermediate events in a process
- Implement gateways for a process
- Implement routing for an activity

## Introduction

In this part of Playback 1, the process variables (also known as flow data) are built and implemented in the gateways. Flow data is used throughout the life of the process. It is used along with gateways to determine the flow of the process based on the values of the flow data. When a token is at a gateway, the value of the variable is evaluated to determine the next path to take. Flow data also includes the following circumstances:

- Data that is used to determine which activities to run
- Data that is used to determine who starts each activity
- Data that is used to determine when an activity is due or when an activity must be escalated

The flow data ensures that the business process gets the right activities to the right people at the right time.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

# Exercise instructions

## *Part 1: Implement the intermediate timer event*

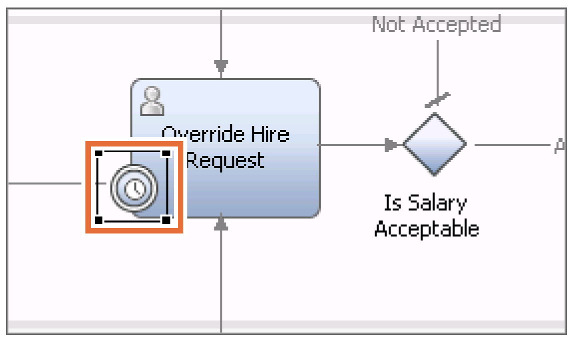
In Playback 0 of the Hiring Request Process, an attached timer intermediate event was modeled. The HR Administrator has 4 hours to complete the review. If the review is not completed within 4 hours, an email is sent to the HR Administrator. The email notifies the HR Administrator of the missed deadline. Now your responsibility is to implement the timer event that was modeled.

__ 1. Implement the attached timer intermediate event.

__ a. Open the **HR Recruitment Process** process application.

__ b. Open the **Approve Hire Request** process and select the **Definition** tab.

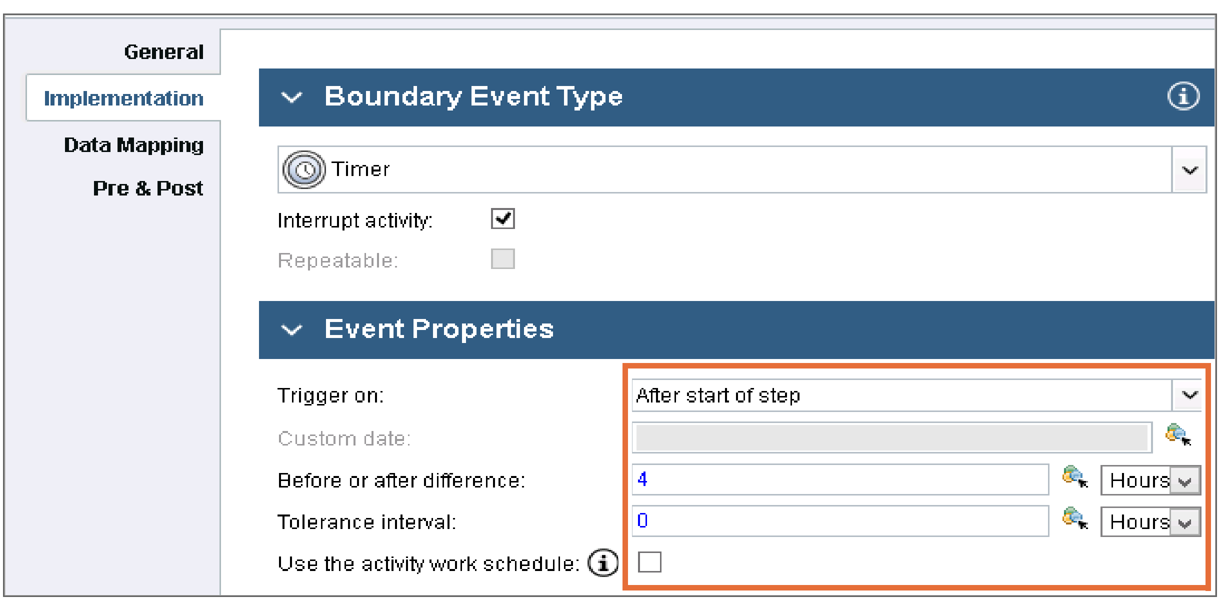
__ c. Click the **timer intermediate event** that is attached to the `Override Hire Request` activity.




__ d. Below in the properties section, click **Implementation**.

__ e. Change the **Event Properties** to match the following values:

- **Trigger on**: `After start of step`
- **Before or after difference**: `4 Hours`
- **Tolerance interval**: `0 Hours`

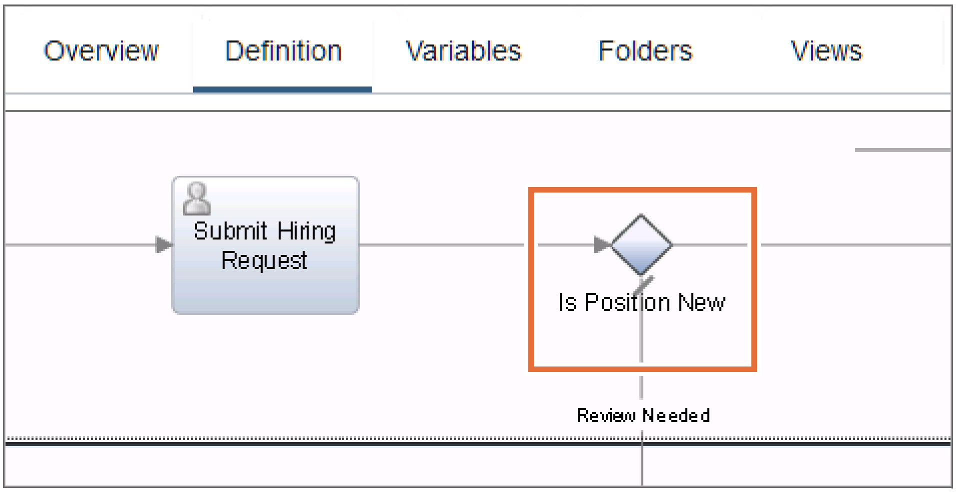


__ f. Save your work.

## Part 2: Create the process flow variables for the Hiring Request Process

The process model for the Hiring Request Process is in place. Data flow implementation is one of the goals of Playback 1. Now you must start to implement the process, first by creating simple variables to implement the logic behind the decision gateways on the current process.

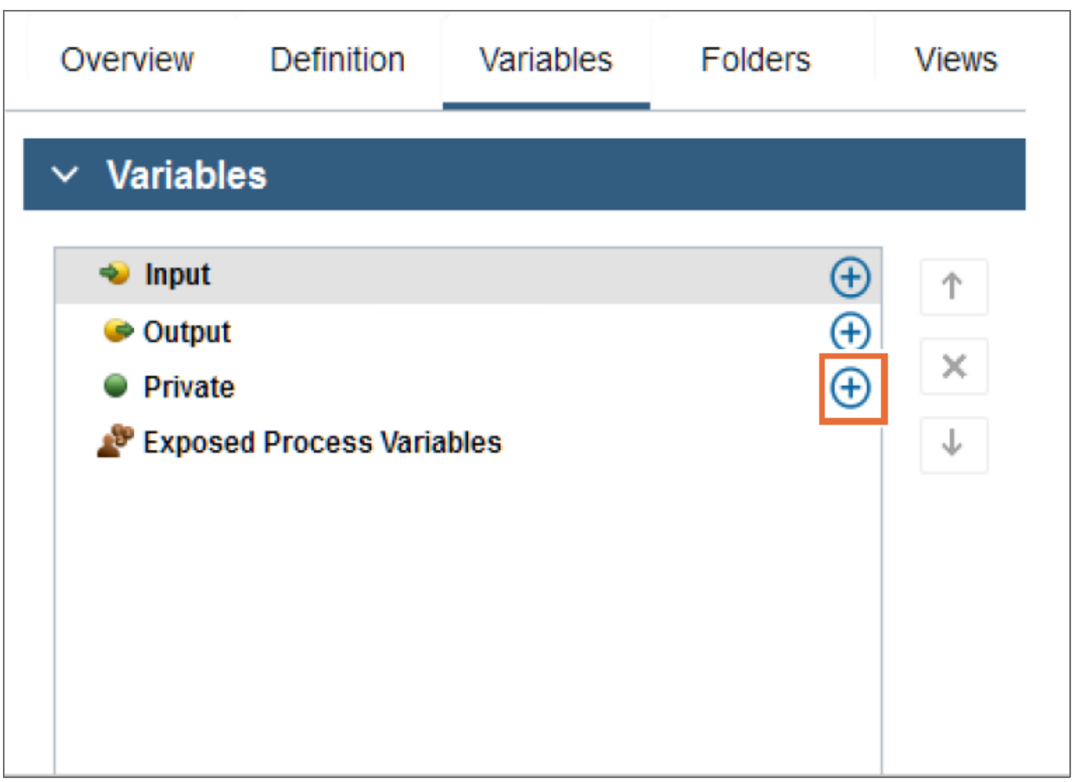
__ 1. Create a private variable to control the **Is Position New** exclusive gateway.



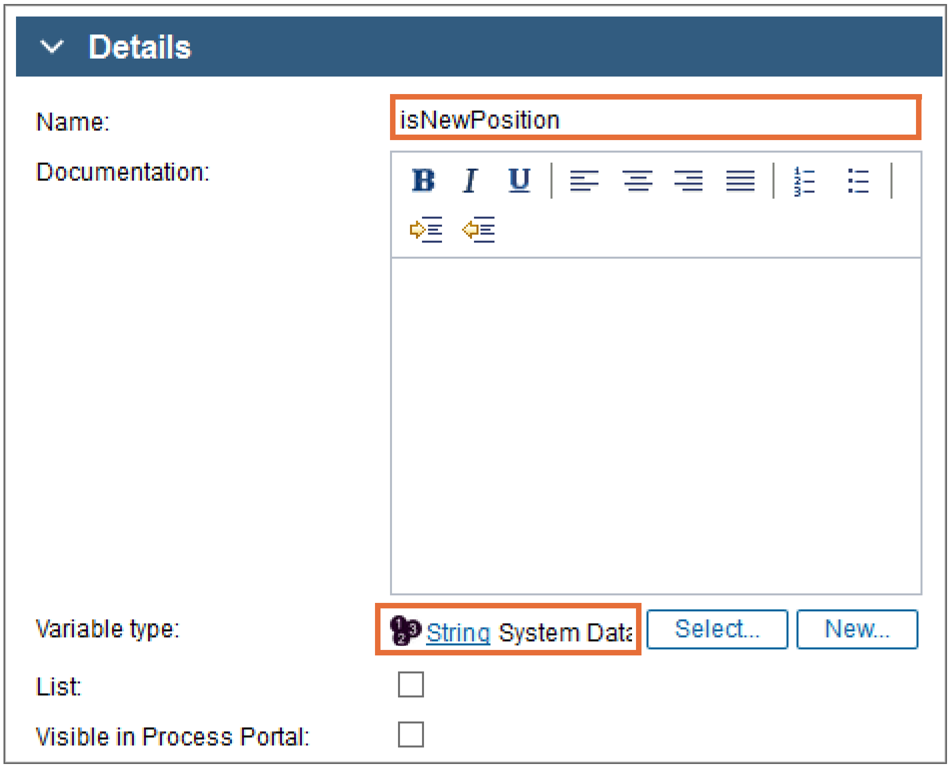

__ a. Open the **Hiring Request Process**.

__ b. Click the **Variables** tab.

__ c. Click the **(+) plus sign** next to Private.



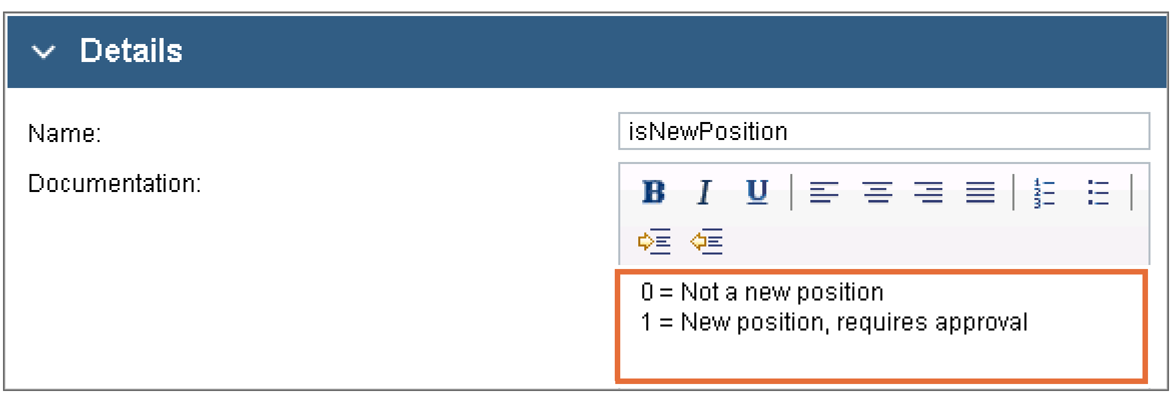
__ d. Name the private variable: `isNewPosition`
Confirm that the Variable Type is **String**.




String variables are used here instead of Boolean for flexibility of the implementation. If requirements change later, and a third outflow is added to the gateway, a String is easier to implement the change than a Boolean would be.

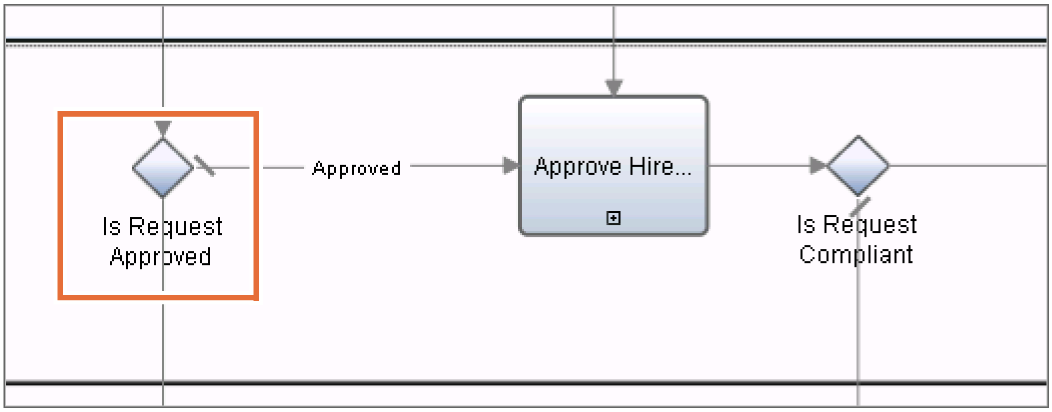
__ e. In the **Documentation** field, insert the following text:

```
0 = Not a new position
1 = New position, requires approval
```




__ f. Do not change the other options. Save your changes.

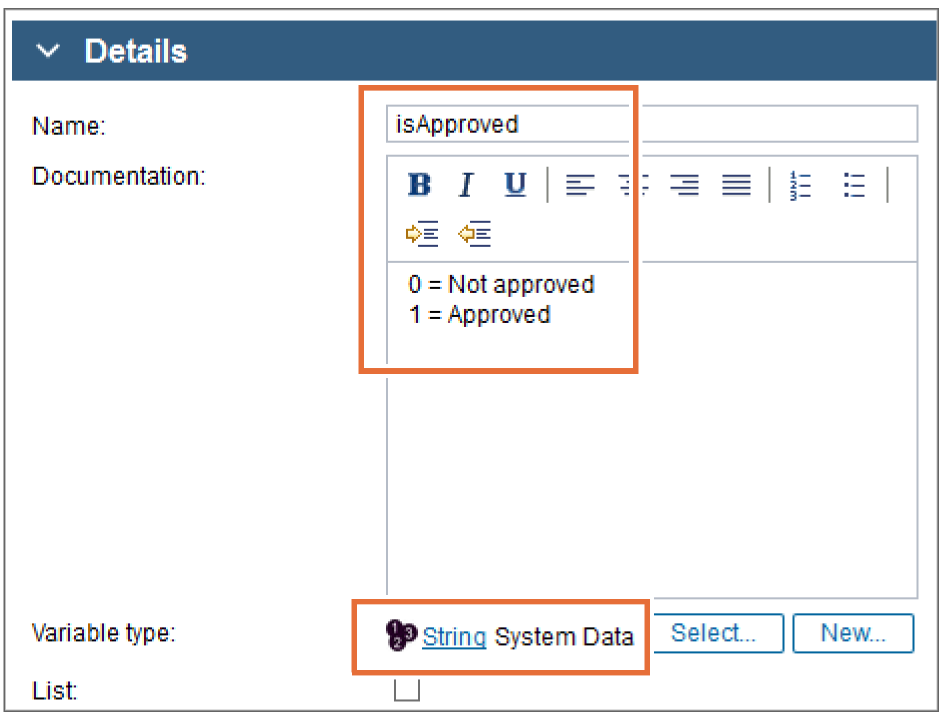
__ 2. Create a variable to control the **Is Request Approved** exclusive gateway.



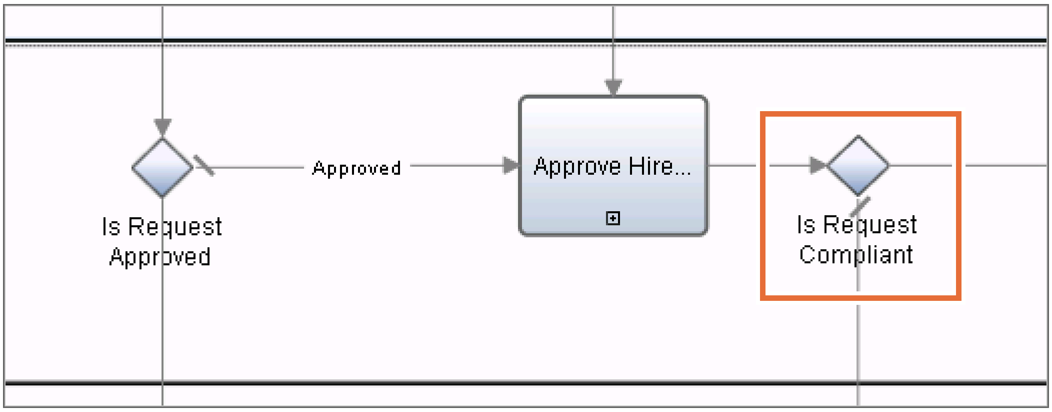

__ a. On the **Variables** tab, create another Private variable: `isApproved (String)`

__ b. In the **Documentation** field, insert the following text:

```
0 = Not approved
1 = Approved
```



__ 3. Create a variable to control the **Is Request Compliant** exclusive gateway.

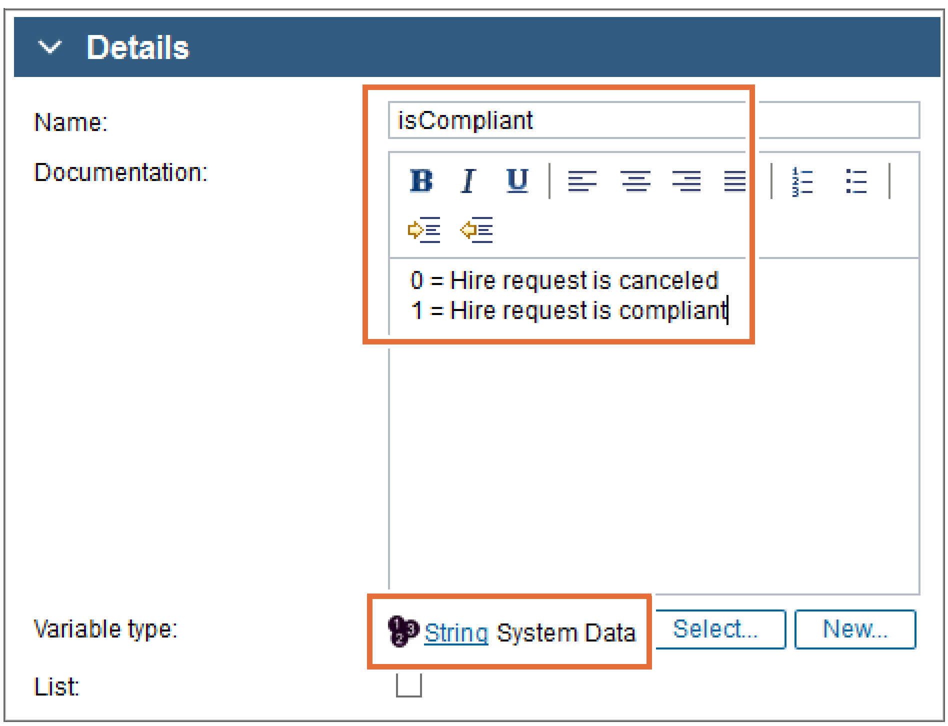



__ a. On the **Variables** tab, create another Private variable: `isCompliant (String)`

The `isCompliant` process flow variable maps to a variable in the Approve Hire Request linked process.

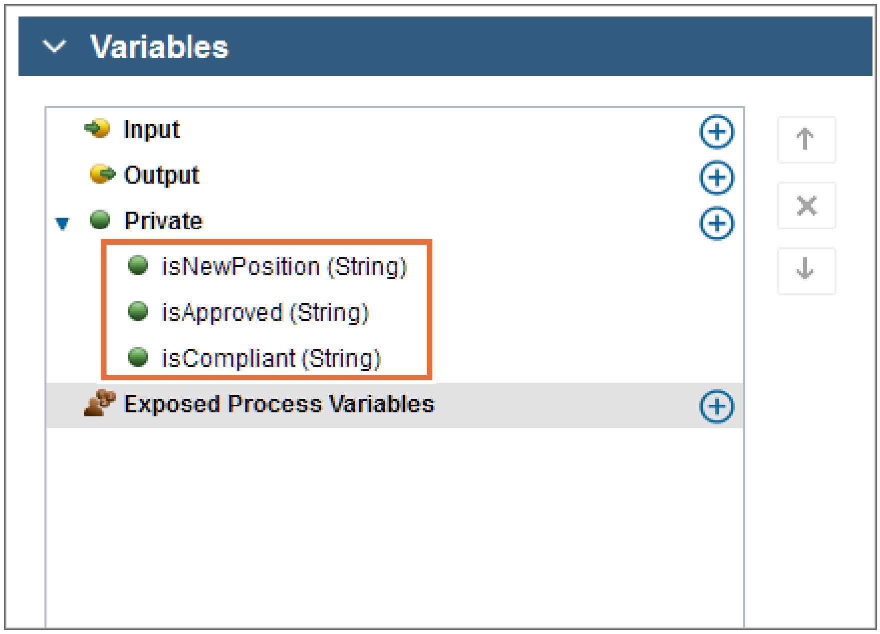
__ b. In the **Documentation** field, insert the following text:

```
0 = Hire request is canceled
1 = Hire request is compliant
```




__ c. Save your changes.

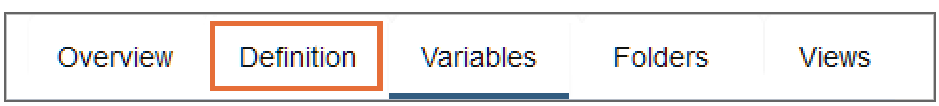
You created three private variables in this exercise.



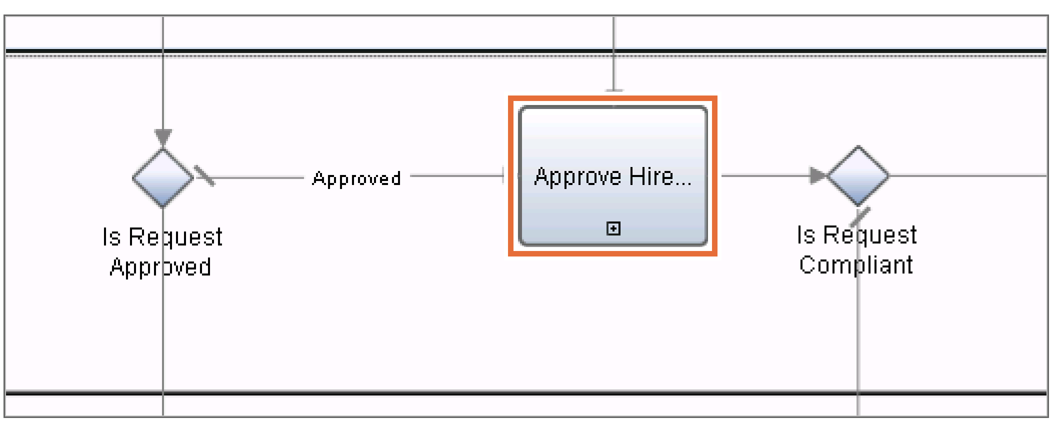

## *Part 3: Create process flow variables for the Approve Hire Request linked process*

__ 1. Create a variable for the **Is Salary Compliant** exclusive gateway in the **Approve Hire Request** process. Because the parent process uses this data, create a variable that passes data through the linked process as an input and output variable.

__ a. Return to the **Hiring Request Process** model by clicking the **Definition** tab.



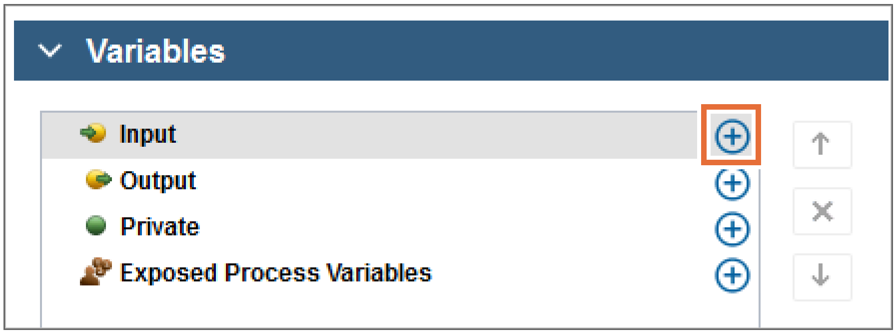

__ b. Open the **Approve Hire Request** linked process by double-clicking the linked process in the diagram.




Be sure that you select the **Approve Hire Request** linked process, not the **Approve New Hire Request** activity. Look for the linked process marker on the bottom center.

__ c. Click the **Variables** tab for the **Approve Hire Request** linked process.

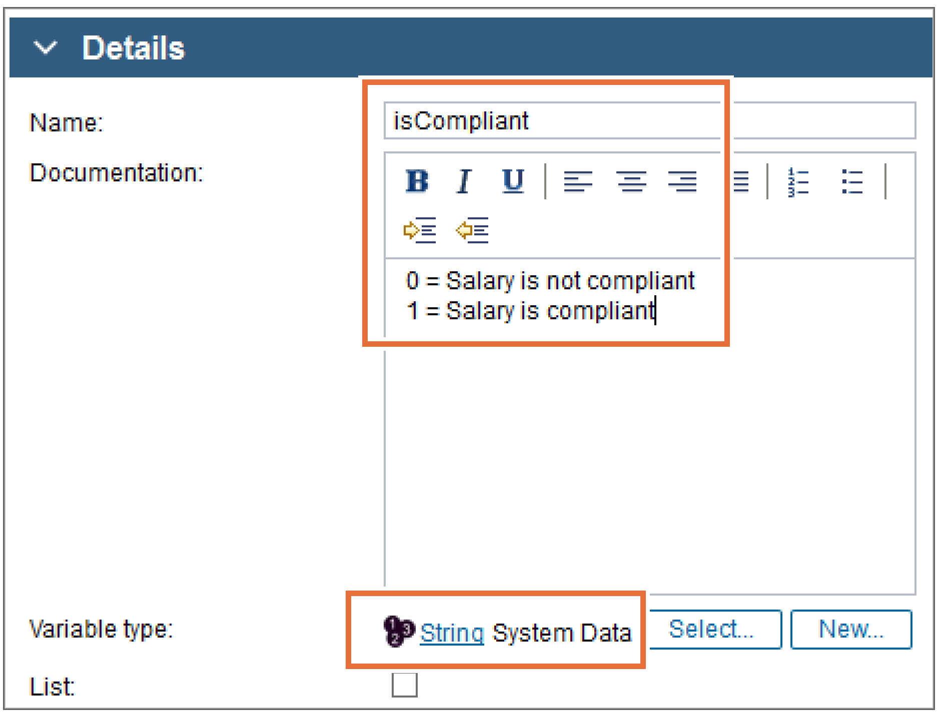
__ d. Click the **(+) plus sign** next to Input.




__ e. Name the input variable: `isCompliant`
Verify that the Variable Type is **String**.

__ f. Insert the following text in the **Documentation** field:

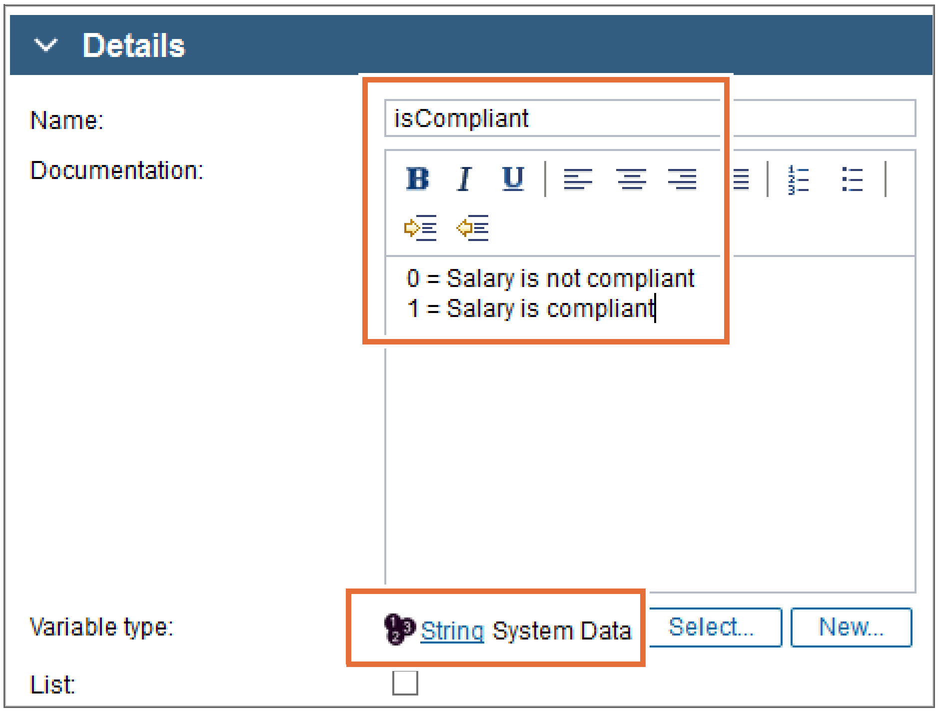
```
0 = Salary is not compliant
1 = Salary is compliant
```



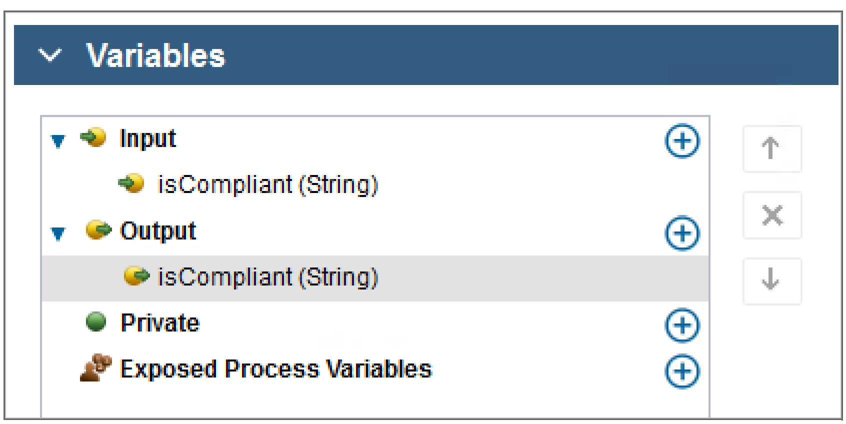

__ g. Create an output variable: `isCompliant (String)`

__ h. Insert the following text in the **Documentation** field:

```
0 = Salary is not compliant
1 = Salary is compliant
```

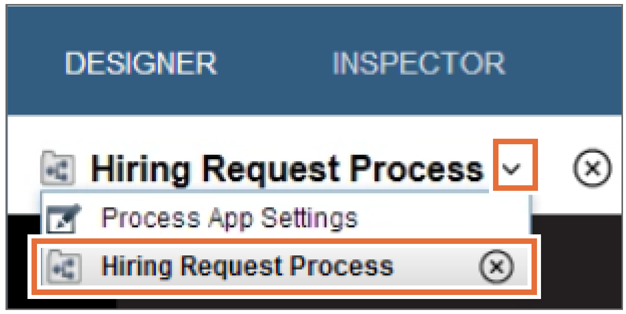



__ i. Save your work.

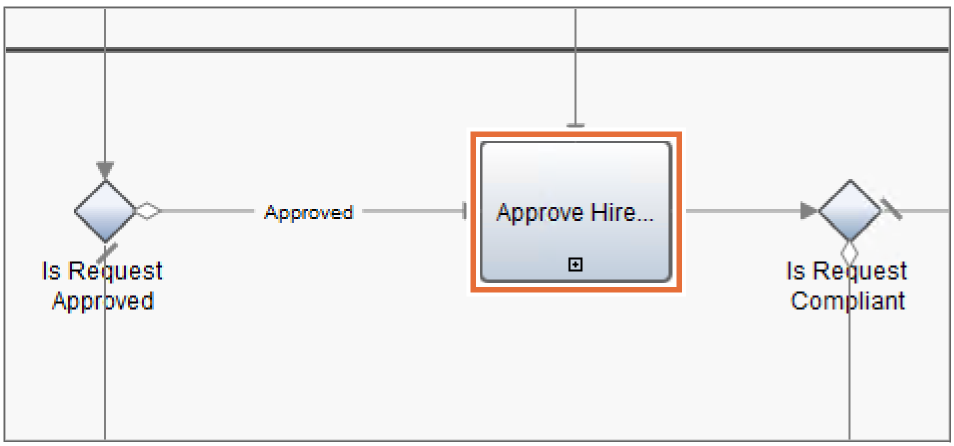


__ 2. Map the process flow variable to the Approve Hire Request linked process.

__ a. Return to the Hiring Request Process by opening the **History** menu at the top and selecting **Hiring Request Process**.

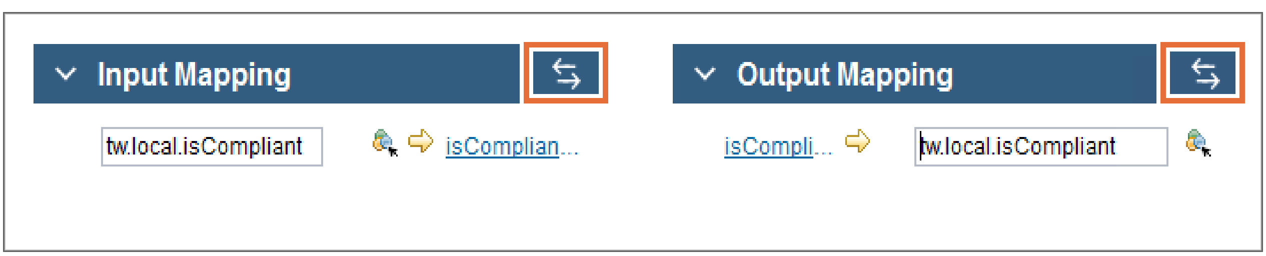



__ a. On the **Hiring Request Process**, select the **Approve Hire Request** linked process.




__ b. Click the **Data Mapping** menu.

__ c. Click the **auto-map icon** on both the Input Mapping and Output Mapping sections.




Because the variables that are mapped use the same name and variable type, the auto-mapping feature is appropriate to be used here. This feature is especially helpful when there are many variables that are mapped with the same names and types.

__ d. Save your work.

When you run the process, the value of the isCompliant variable is passed in and out of the Approve Hire Request linked process. Any changes to the variable while the Approve Hire Request process runs are reflected when the `isCompliant` variable is passed back to the parent process.

## Part 4: Implement gateways

When you modeled the process during Playback 0, you added several gateways to the process. Now you must implement these gateways. In this section, you enhance the process by implementing the exclusive gateways in both the Hiring Request Process and the linked process. You use the simple variables that were created in the previous sections to implement the gateways.

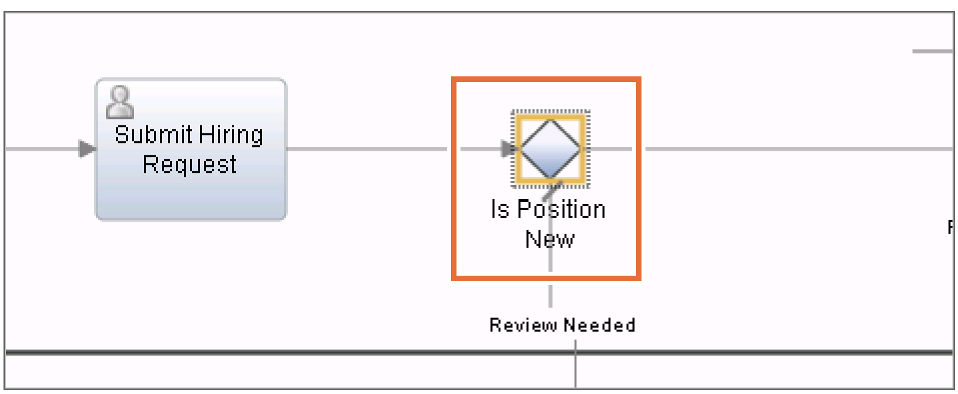
You implement the exclusive gateways in the Hiring Request Process process and the Approve Hire Request linked process to meet the following requirements:

- If the request is for a new position, the request is forwarded to the General Manager. After the General Manager receives the request, the General Manager approves or rejects the request.
- If the request is rejected, the system processes the request and the Hiring Request process is complete. If the request is approved, the process flows to the Approve Hire Request linked process.
- In the linked process, after the requisition is submitted, an automated system-level check for salary compliance starts. If the request meets salary compliance, the hiring request is forwarded to Legal for review. If the request is canceled, it is automatically sent to the HR Positions database and processed.

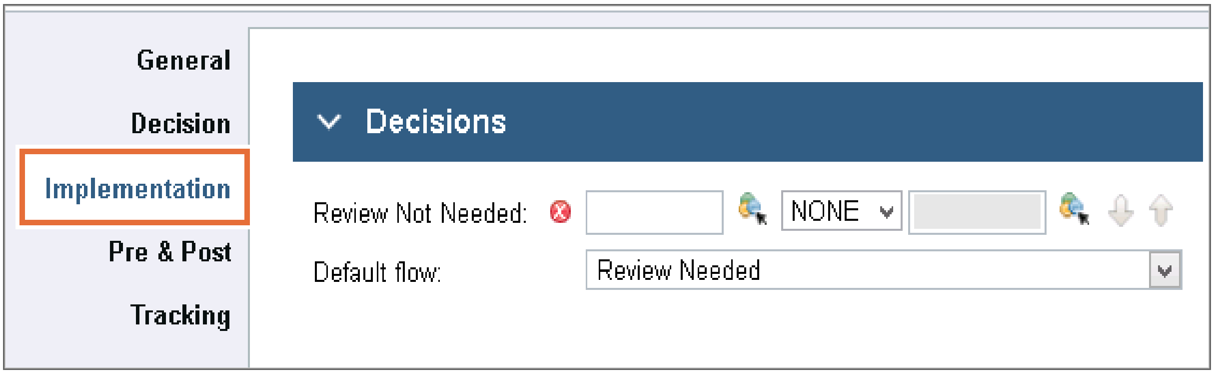
__ 1. Implement the **Is Position New** exclusive gateway.

When the position is new, the process must flow to the Approve New Hire Request activity down the Review Needed flow.

__ a. In the Hiring Request Process diagram, click the **Is Position New** exclusive gateway. Note where the default flow is. This changes when you implement the gateway later in the exercise.



__ b. Open the **Properties > Implementation** menu.



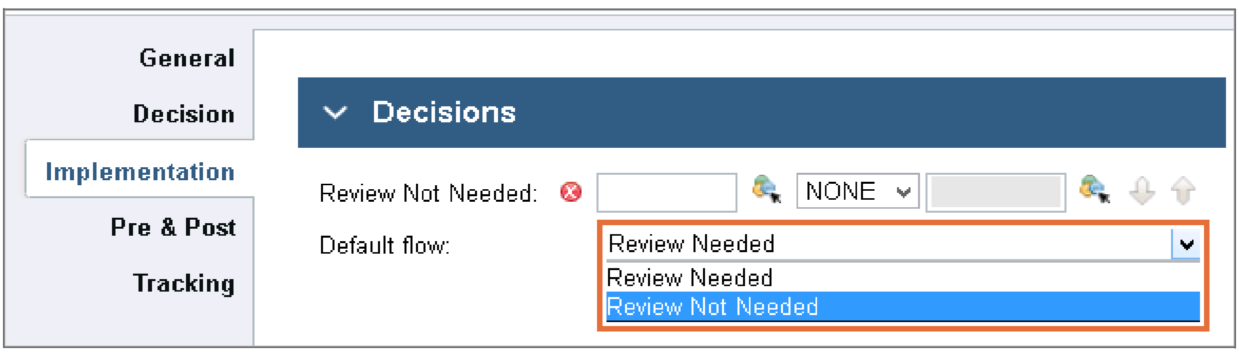

---

## Information

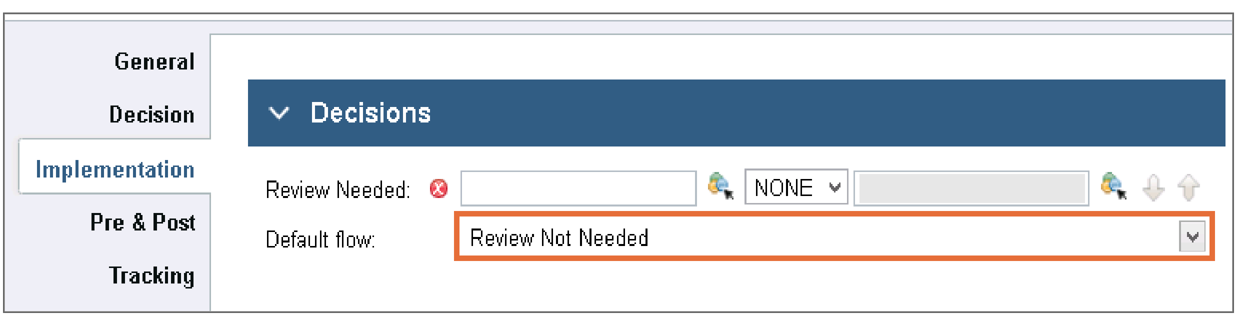
You see a red “x” next to the field while the decision is being configured. It clears when the decision is defined.

---

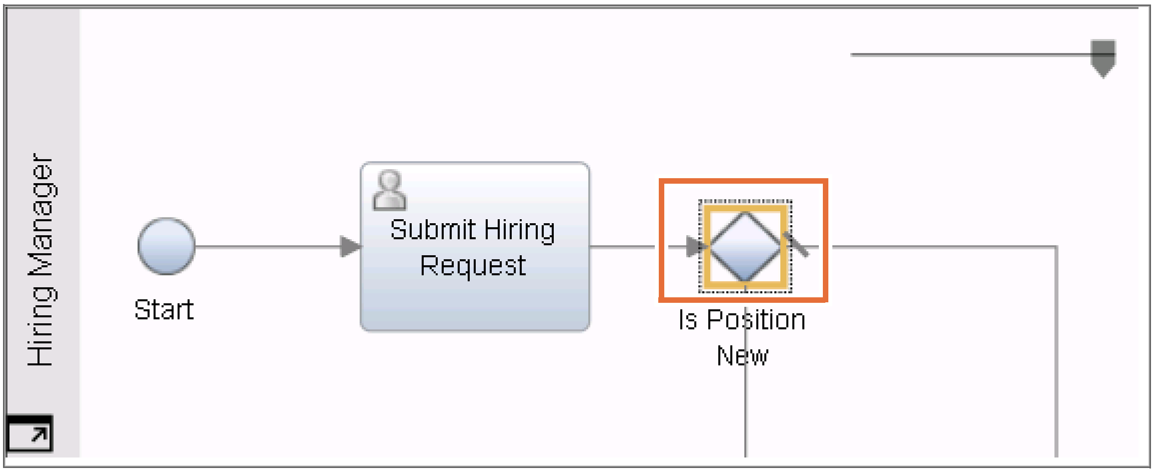
__ c. In the **Decisions** section, click the arrow to expand the **Default flow** selection box.



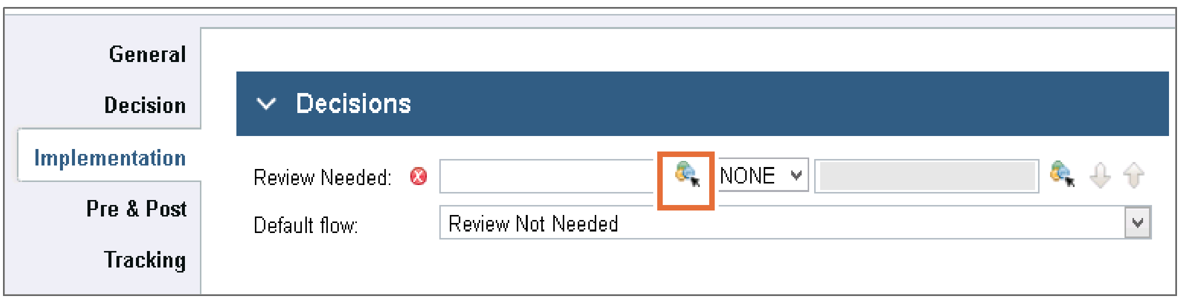

__ d. Select **Review Not Needed**.



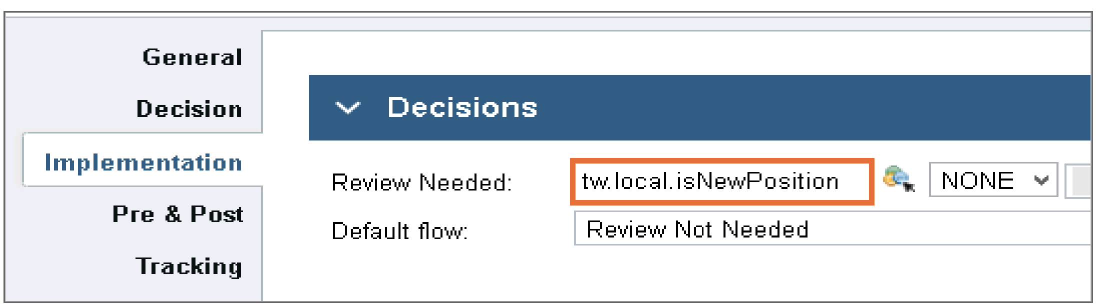
__ e. Verify that the default flow is now changed for the **Is Position New** gateway.




__ f. In the first field to the right of the **Review Needed** label, click the **Variable Picker** icon.

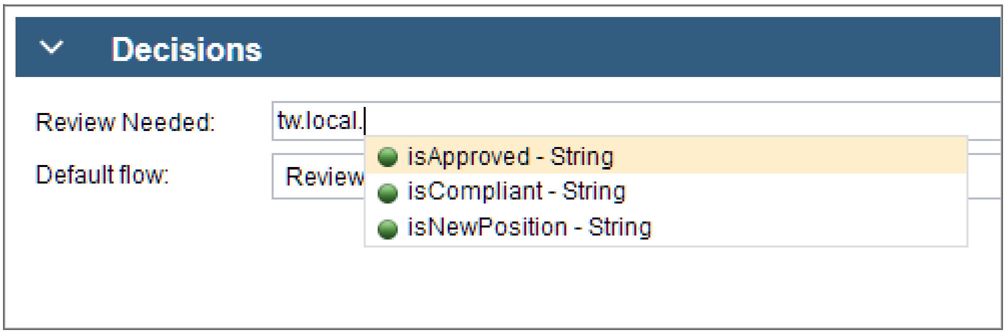



__ g. Select **isNewPosition (String)**. The field is populated with the variable: `tw.local.isNewPosition`



**Hint**

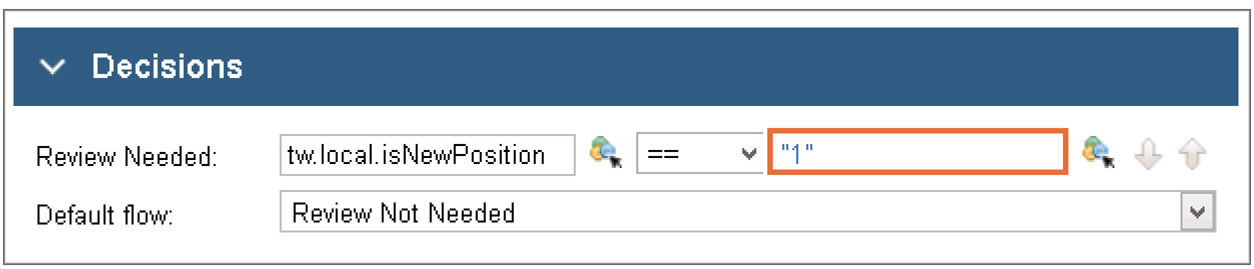
You can also type this variable and as you type and press **Ctrl+Spacebar**, the auto-complete feature suggests different options to you that match what you already typed. The options are filtered as you complete your entry.




The auto-complete window provides other contextual information on the object or object type. Use the mouse or the up and down arrows on the keyboard to select the correct option, and press Enter to select the option.

__ h. Next, change the evaluation value to: ==

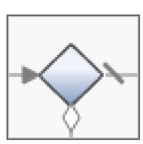
__ i. In the last field to the right of **Review Needed**, enter the value: "1"




The condition is set for the Is Position New gateway. If the first condition is not met, the default condition is Review Not Needed.

**Important**

Once the gateway is implemented, the conditional flow is flagged with a diamond at the point at which it exits the gateway.

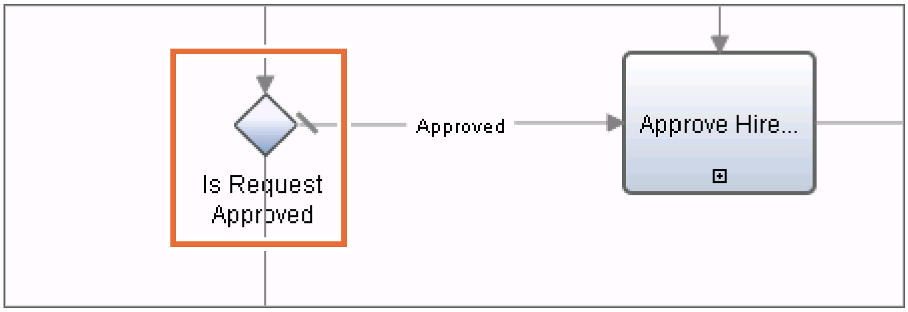


__ j. Save your work. The red “x” clears when the decision is saved

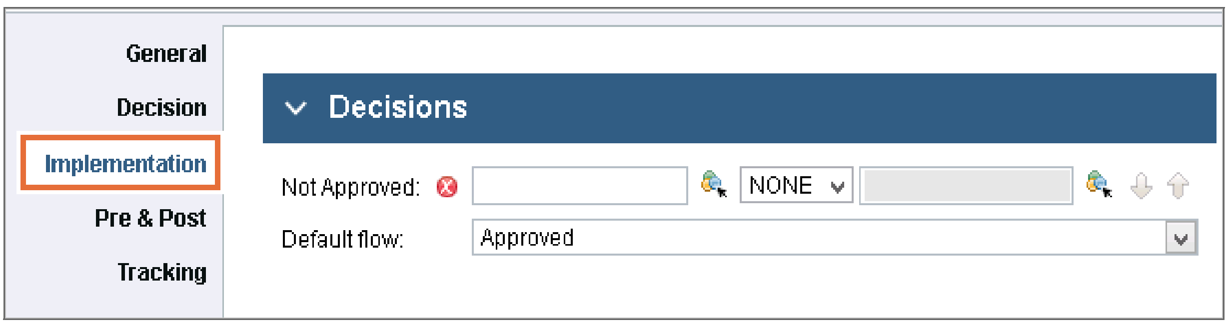
__ 2. Implement the **Is Request Approved** exclusive gateway.

When the request is approved, the process must flow to the Approve Hire Request linked process down the Approved flow.

__ a. Select the **Is Request Approved** exclusive gateway on the process.

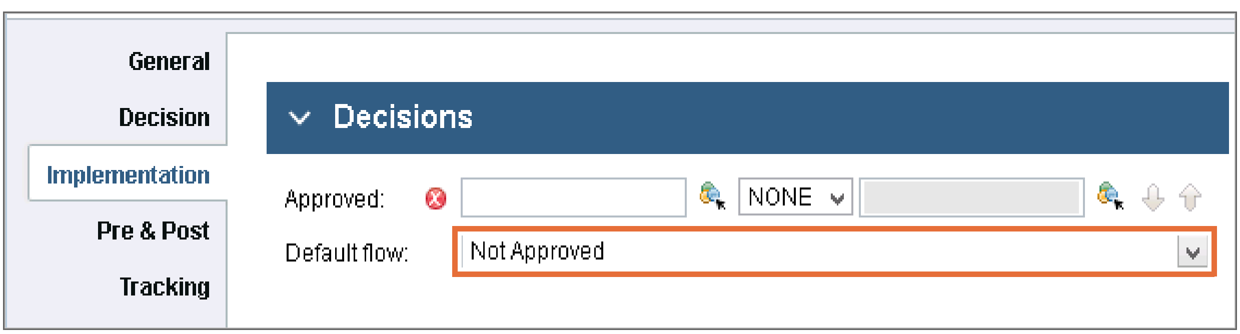



__ b. In the Properties pane, open the **Implementation** menu.




__ c. In the **Decisions** section, expand the Default flow selector.

__ d. Select **Not Approved**. This selection makes **Not Approved** the default sequence flow. The order of sequence flow starts with **Approved** at the top and **Not Approved** on the bottom.

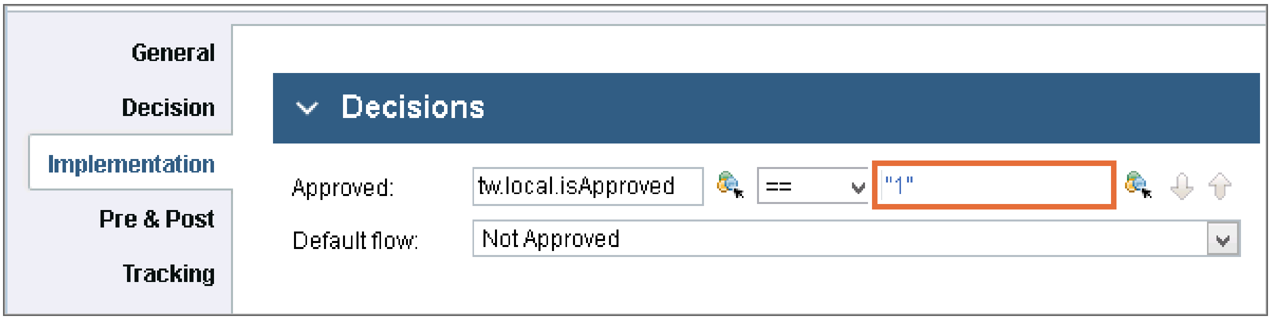



__ e. In the first field to the right of **Approved**, select the following using the variable picker:

`tw.local.isApproved`

__ f. Next, change the evaluation value to: ==

__ g. In the last field, enter the value: "1"

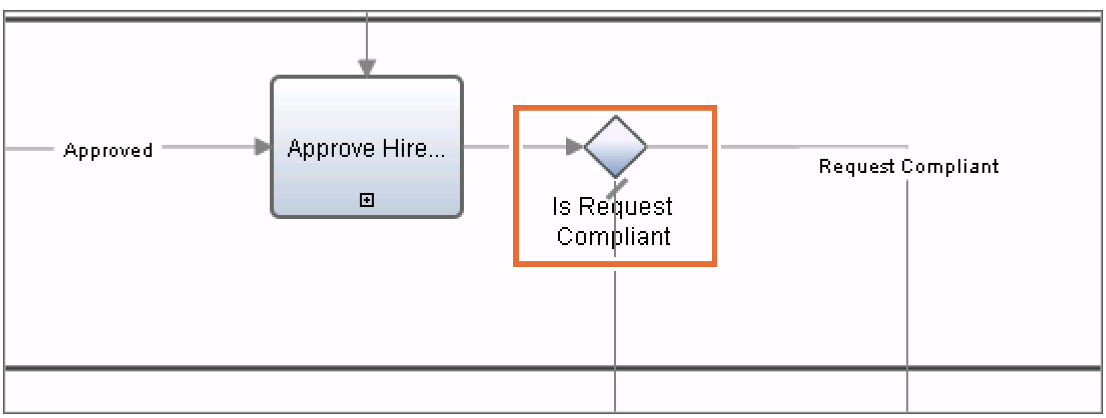



__ 3. Implement the **Is Request Compliant** exclusive gateway.

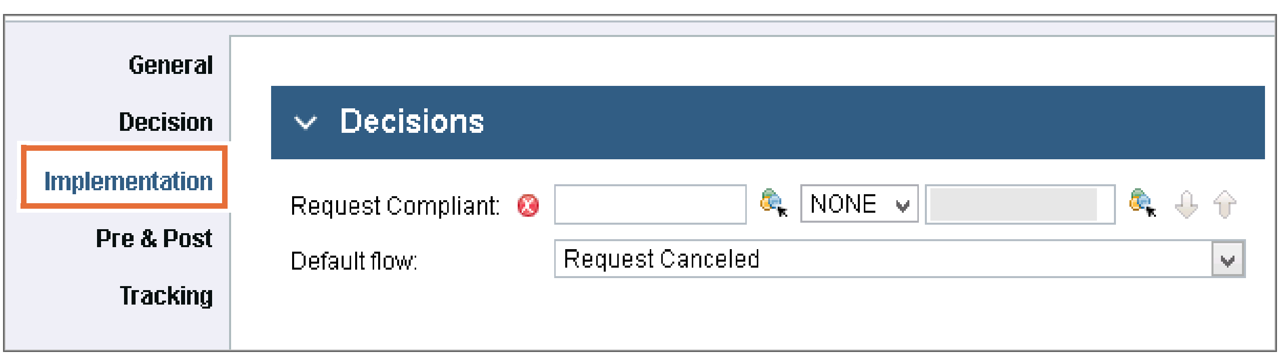
When the request is compliant, the process must flow to the Review Posting activity down the Request Compliant flow.

The conditions for this gateway are set in a decision service in Playback 2. All the conditions that must be met are established as a business rule. The intent for this playback is merely to control the flow. For now, focus on implementing the gateway, not how the data is generated.

__ a. Click the **Is Request Compliant** gateway on the process.

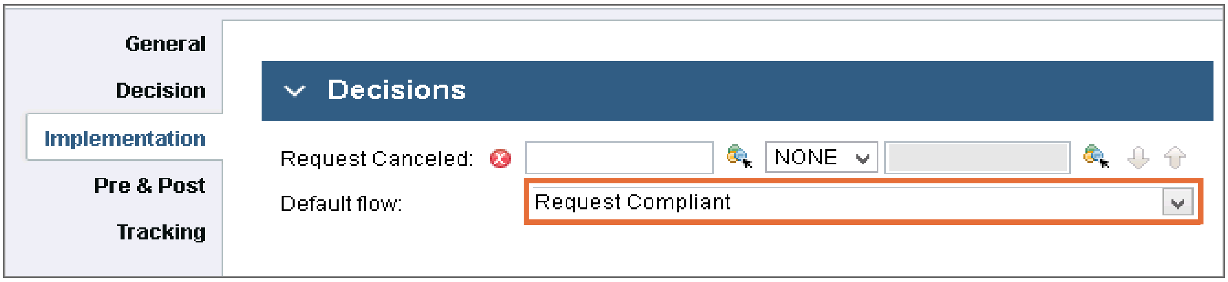



__ b. Click the **Implementation** menu.




__ c. In the **Implementation** section, expand the **Default flow** selector.

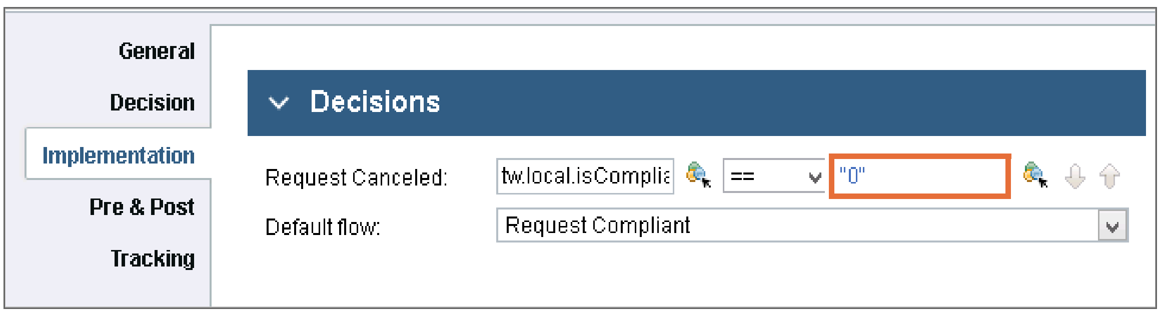
__ d. Select **Request Compliant**. This arrangement makes **Request Compliant** the default sequence flow. The order of sequence flow expresses that the **Request Canceled** option is first and that the **Request Compliant** option is second.




__ e. In the first field to the right of **Request Canceled**, enter: `tw.local.isCompliant`

__ f. Change the evaluation value to: ==

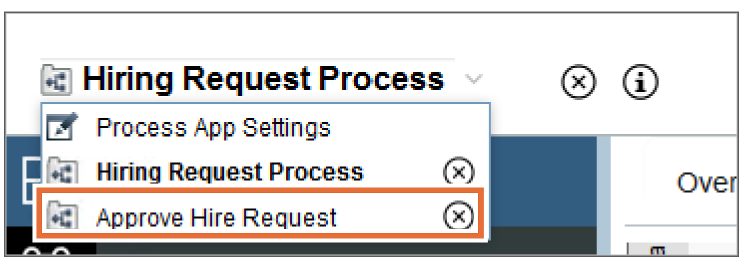
__ g. In the last field to the right, enter the value: `"0"`




__ h. Save your work.

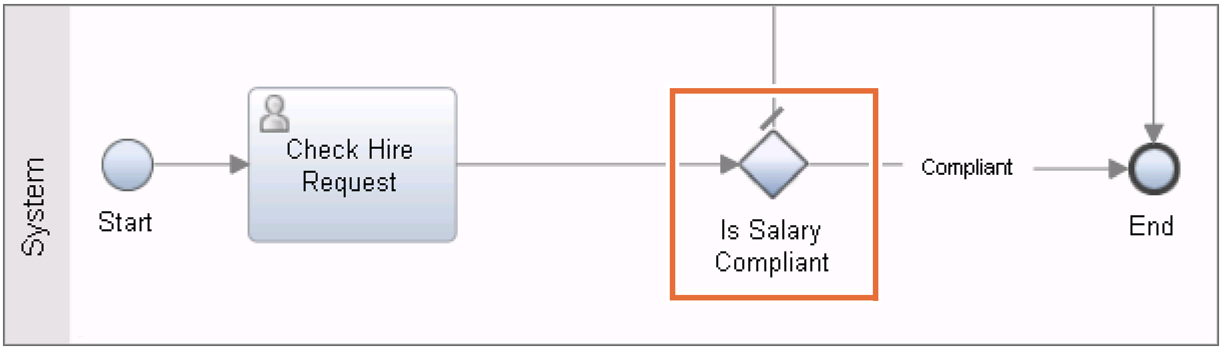
__ 4. Implement the **Is Salary Compliant** exclusive gateway in the **Approve Hire Request** linked process. When the salary is compliant, the flow must move to the end event.

__ a. Open the **Approve Hire Request** linked process from the History menu.




If you don’t see the process in your history menu, open the Approve Hire Request from the library on the left.

__ b. On the **Definition** tab, select the **Is Salary Compliant** gateway.




Make sure that you select the **Is Salary Compliant** gateway, not the **Is Salary Acceptable** gateway.

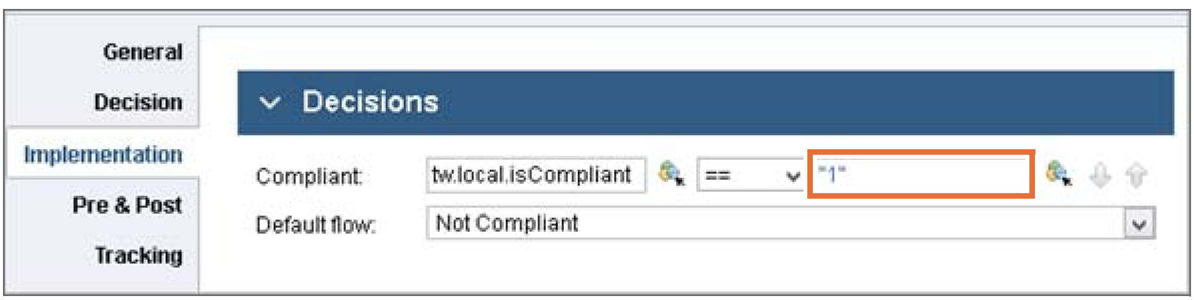
__ c. In the **Decisions** section of the **Implementation** panel (not the Decisions panel) below, select **Not Compliant**, if not selected as the default flow.

This arrangement makes **Not Compliant** the default sequence flow. The order of sequence flow in the Implementation section ensures that the **Compliant** option is assessed first, and if the first assessment is false, the **Not Compliant** option is taken.

__ d. In the first field to the right of **Compliant**, enter: `tw.local.isCompliant`

__ e. Change the evaluation drop-down value to: ==

__ f. In the last field to the right of **Compliant**, type: `"1"`




__ g. Save your work.

## *Part 5: Implement routing for an activity*

Effective business process management is about routing the right task to the right teams at the right time. Using IBM Business Automation Workflow, authors use lanes (sometimes known as swimlanes for a pool) to identify the process teams. Process developers implement routing in the process for each activity to the right team in the lane. The goal of this section is to implement the teams for the lanes and implement assignment routing for all activities in the process.

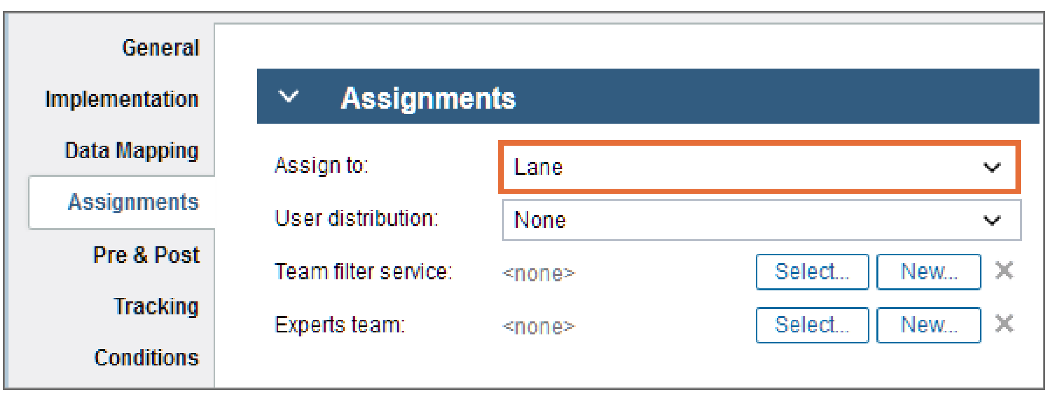
__ 1. Route the Submit Hiring Request activity in the Hiring Request Process to **Lane** and distribution to **Last User**.

__ a. Open the **Hiring Request Process**.

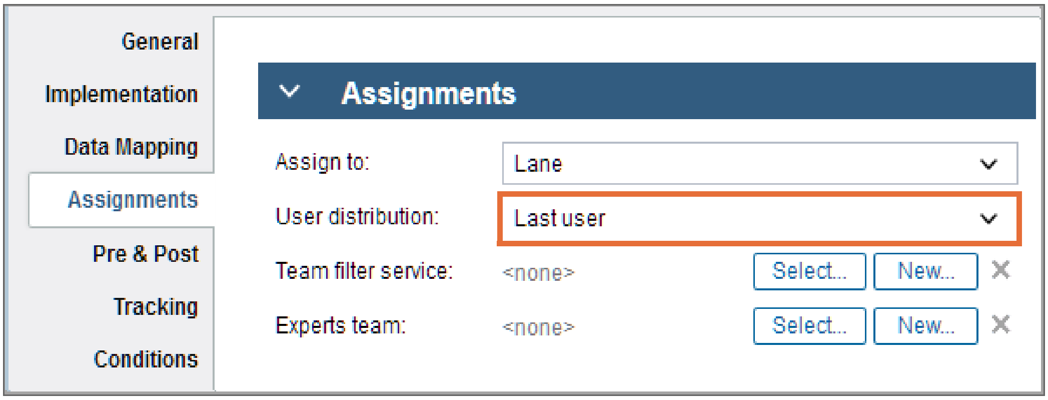
__ b. On the **Definition** tab, select the **Submit Hiring Request** activity.


__ c. Open the **Assignments > Assignments** section.

__ d. Verify that the default assignment for **Assign to** is: Lane



__ e. Change the **User distribution** to: `Last user`




### Information

The Last user option assigns the runtime task to a user who completed an earlier task.

- If the assignment option is set to Lane, the task is assigned to the user who completed the activity that immediately precedes the selected activity in the lane.
- If the assignment option is set to Team, the task is assigned to the user who completed the last task that was assigned to the same team.

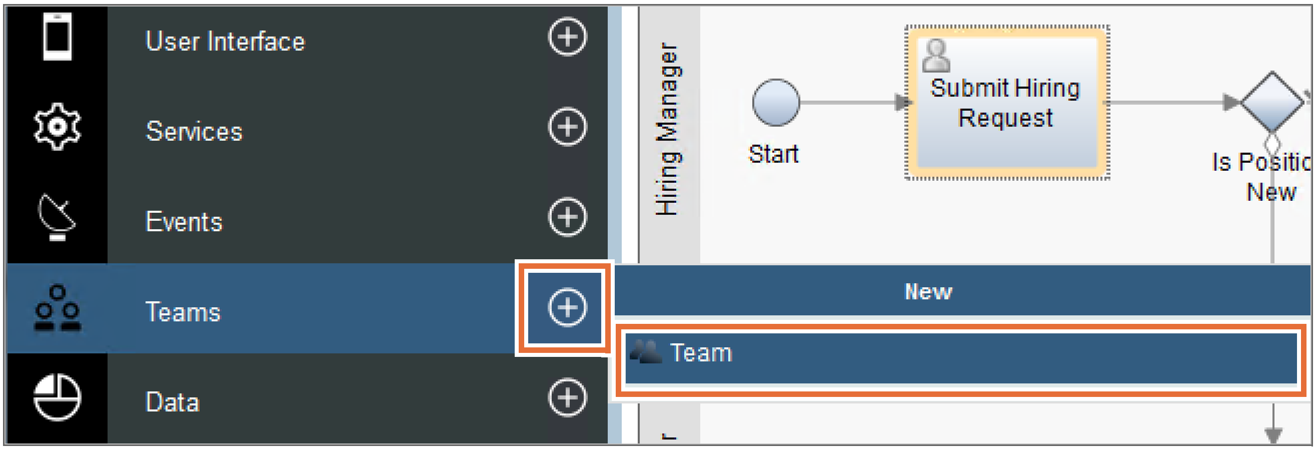
In both cases, the task is assigned only to an established last user, if the user is a member of the group that is associated with the task. An established last user is determined by using the policy valid for the Lane or Team assignment option. If the user is not a member of the group that is associated with the task, then the task is assigned to the task group.

Because this activity is the first in this lane for this process, the system automatically assigns the first task to the user that creates the instance. When a user creates an instance of this process, the server displays any screens that are configured for the task. You create those screens in the next playback phase.

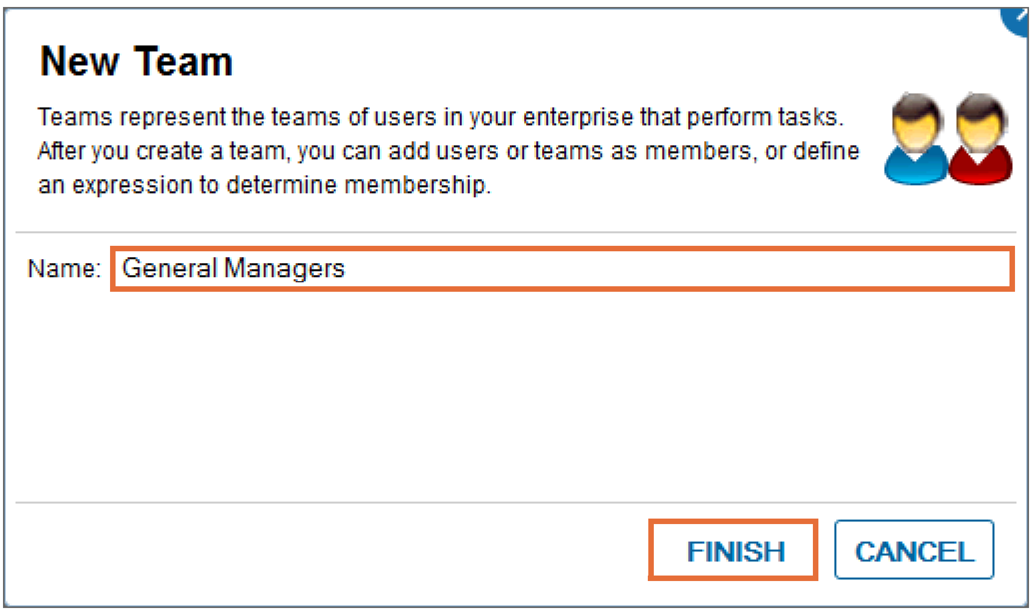
__ f. Save your changes.

__ 2. Create a team and add a member.

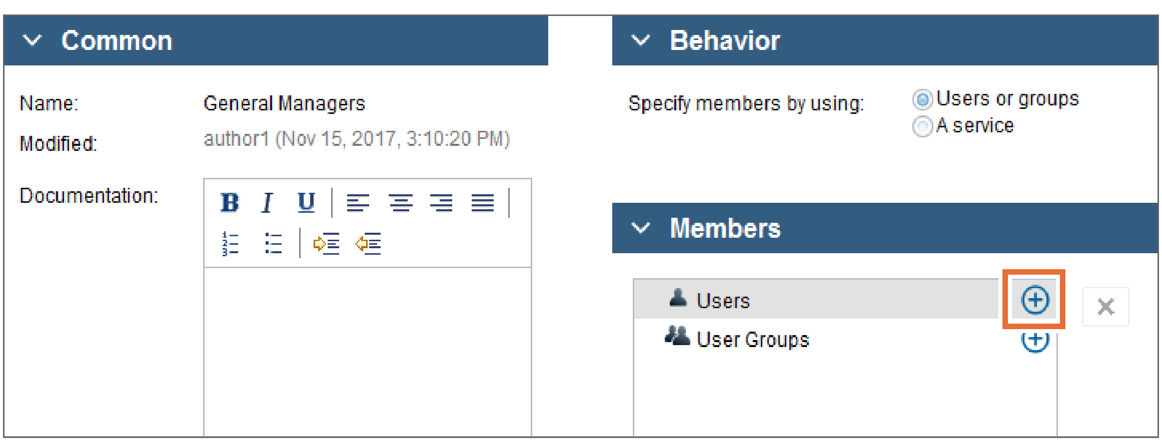
__ a. In the process library, click the (**+**) plus sign next to **Teams**. Click **Team**.




__ b. Name the team `General Managers` and click **FINISH**.

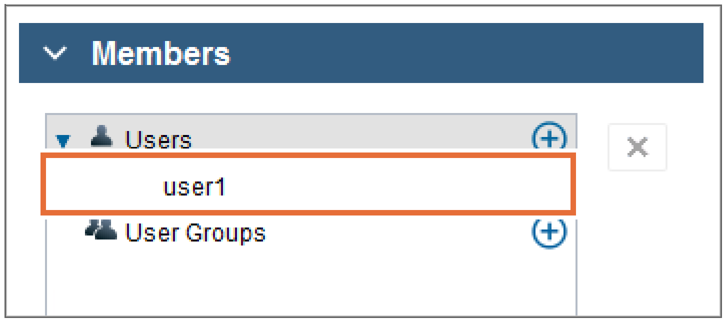



__ c. In the **Members** section, click the (**+**) plus sign next to **Users**.



__ d. Add `user1` to the team from the list.

You now have a populated list in the **Members** section with `user1` for the General Managers team.



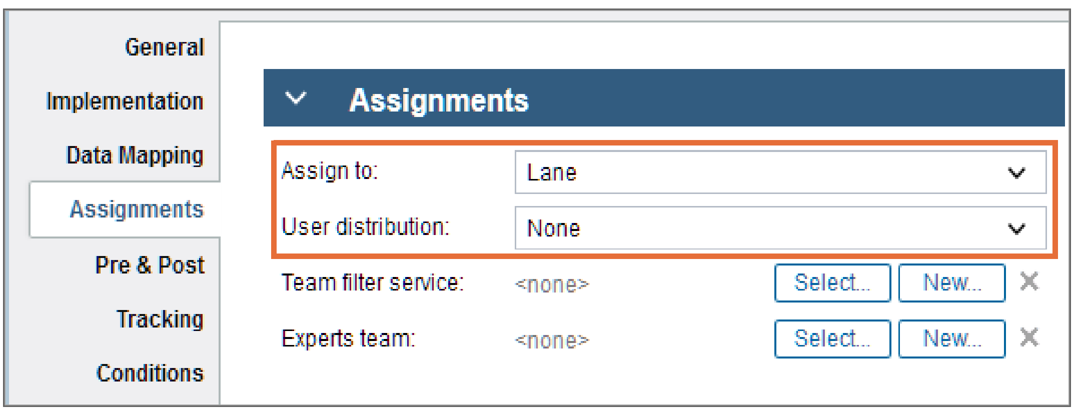

__ e. Save your work.

__ 3. Implement the routing for **Approve New Hire Request**.

__ a. Open the **Hiring Request Process**.

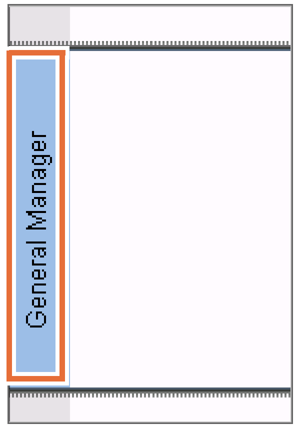
__ b. On the **Definition** tab, select the **Approve New Hire Request** activity.

__ c. In the panel below, in the **Assignments > Assignments** section, leave the value of **Assign to:** as `Lane` and the value of **User distribution** as `None`.

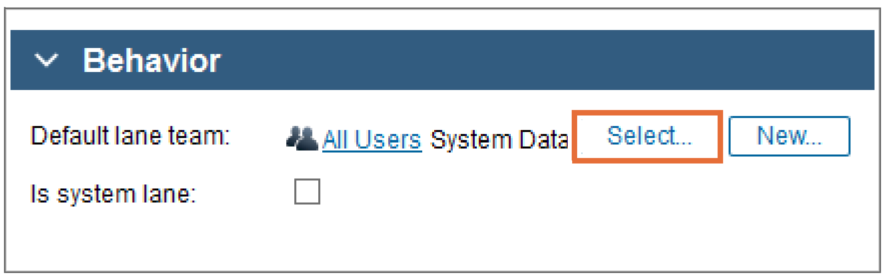



The default **User distribution** option is `None`. IBM Business Automation Workflow does not assign the task to any user when using this option. All the other options in this selection assign the task to a user of the system.

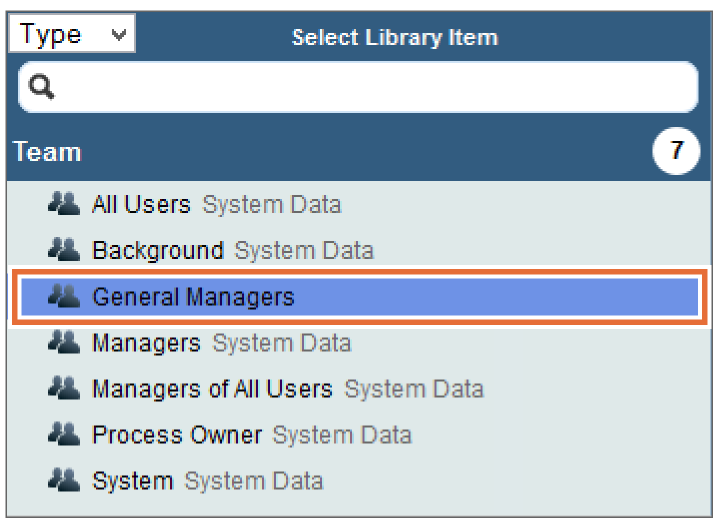
__ d. On the General Manager lane, click the **label** on the left.



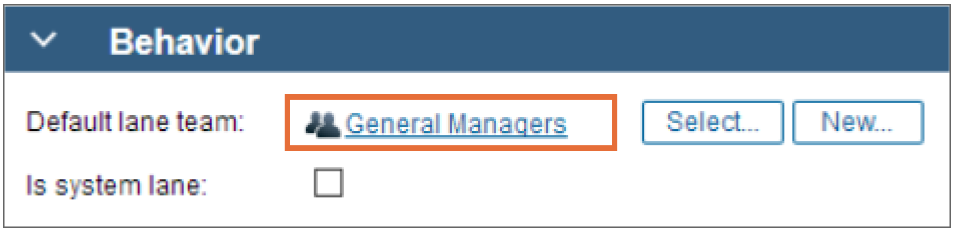

__ e. In the panel below, go to the **General > Behavior** section, click **Select** next to Default lane team.




__ f. Click the **General Managers** team to select it.



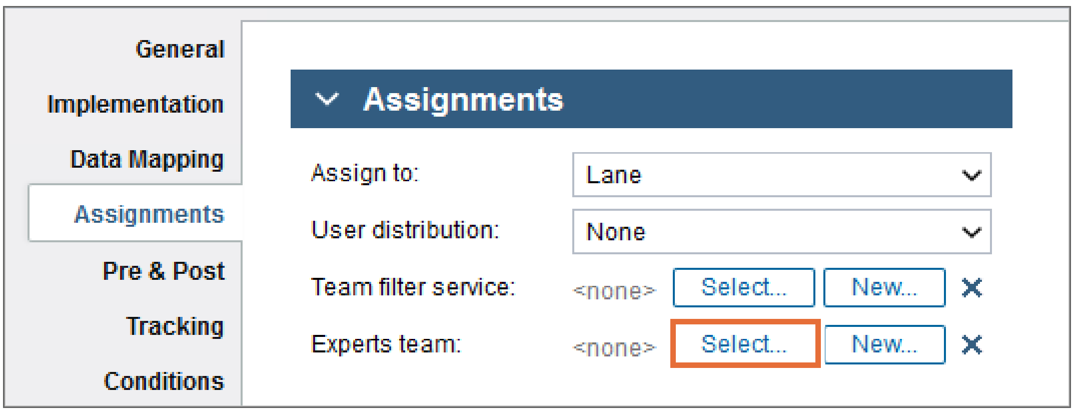
The **General Managers** team is assigned as the default team for the **General Managers** lane.



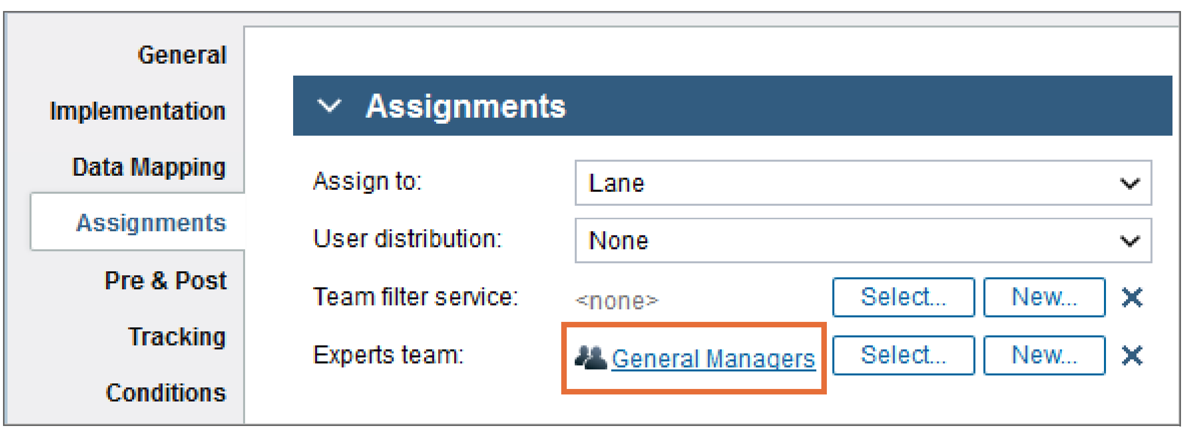

__ 4. Assign the **General Managers** team as the **Experts Group** for the Approve New Hire Request activity.

__ a. On the **Hiring Request Process**, select the **Approve New Hire Request** activity.

__ b. Open the **Assignments > Assignments** section. Click **Select** next to the Experts team.




__ c. Click **General Managers** to assign the General Managers team as the Experts team.



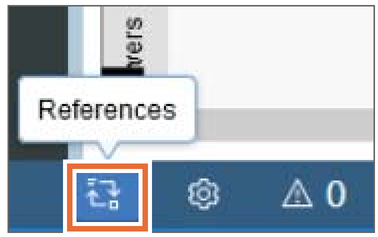

__ d. Save your work.

## Part 6: Review artifact references

As you add more artifacts to your solution, it is a good time to review artifact references. When you work with an artifact in the Process Designer, it is useful to know where the artifact is used so that you can see the impact of potential changes that you want to make.

__ 1. Review the local scope references for the Hiring Request Process artifact.

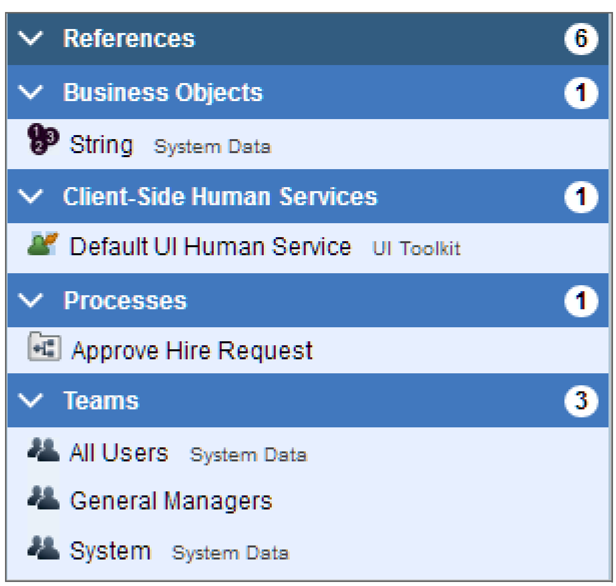
__ a. With the Hiring Request process open, click the **references icon** in the status bar at the bottom.




__ b. In the References pane, review the **local scope** references.


Local scope shows where artifacts are used only in the current process application and its dependencies. This is the default view. Currently, you can see that there are no artifacts in the process model with a local scope.

__ c. In the References pane, review the remaining **references**.

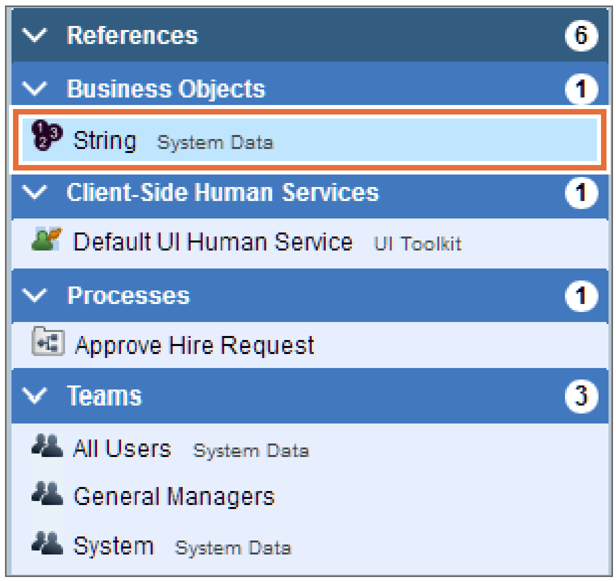



Depending on timing, it might take a minute or two for the General Managers team to display.

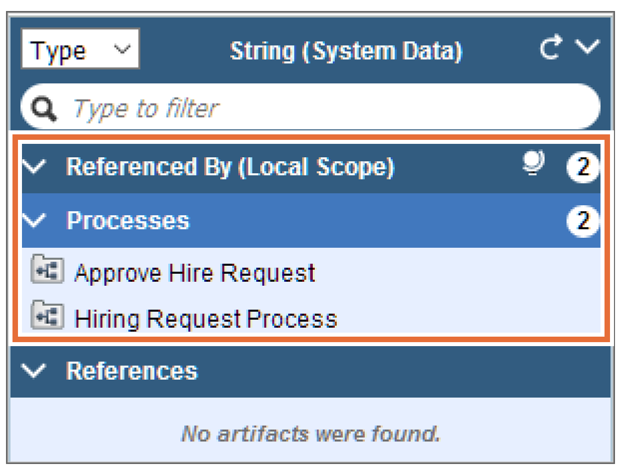
Specifically, the list shows the references to the system toolkit artifacts. For example, since you defined a variable to pass between the parent and nested processes, the String object is referenced.

The numbers on the right are the instances in the project. You should see the same number having followed the lab instructions. If not, then your are missing something in the lab.

__ d. Review local references of the String artifact by clicking the **String section**. You might need to refresh the References section to see the new team.

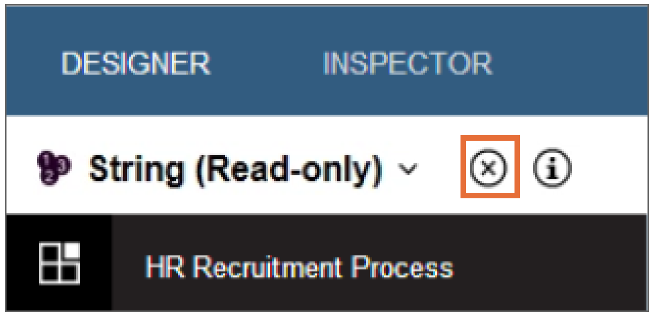



__ e. For the String references, there are two local references that point to the two process models that use the string variable.




__ f. Close the references panel.

__ g. Close the String (Read-only) window.




The String object is read-only since it is a reference to the System Toolkit.

You have completed this exercise.

This exercise looked at setting variables in business processes, implementing an intermediate timer event on a process, implementing exclusive decision gateways, and establishing routing for an activity. You also evaluated artifact dependencies by reviewing the references for some of the artifacts.

The assets for Playback 1: Controlling process flow, are complete. You now can control gateways by assigning values to variables inside the process. The next exercise looks at building business objects and creating a reusable client-side human service.

## End of exercise

# Exercise 5. Playback 1: Business data, services, and coaches

## Estimated time

01:30

## Overview

In this exercise, by using the core requirements, you determine and create all of the necessary assets to support a coach in the Hiring Request Process. You use complex business objects (variable types) to organize your data, and pass data into and out of a linked process. You build a service and define guided user interactions with a coach. You also implement a service for an activity, and map variables between a nested service and an activity. You model the coach by using the concept of grids.

## Objectives

After completing this exercise, you should be able to:

- Determine and organize data when provided with a written process
- Add business objects and object types
- Create a client-side human service
- Add variables and business objects to a process application
- Create and configure a coach to obtain process participant input
- Model a coach by using the concept of grids
- Add coach controls to control process flow
- Create a client-side human service and coach for the General Manager review activity
- Implement an activity by attaching a service and mapping data

## Introduction

In Playback 1, you create human services. Users can use human services to access task assignments in a web-based user interface. At this stage of development, it is important that the functions of the human service are implemented. Users interact with web-based interfaces called coaches to complete their assigned tasks. Enhancements can be added later. This exercise is about making sure that users have what they need in terms of business data and task assignment information to complete the process activity.

The human services with coaches are implemented on two activities in the Hiring Request Process process. The coaches are used during playbacks to provide business data to the process and control the process flow.

This exercise and the next provide practice in building and moving controls on a coach. Repetitive steps are purposely provided to gain experience in this skill.

This exercise is long, so a break is included in the middle of the instructions.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

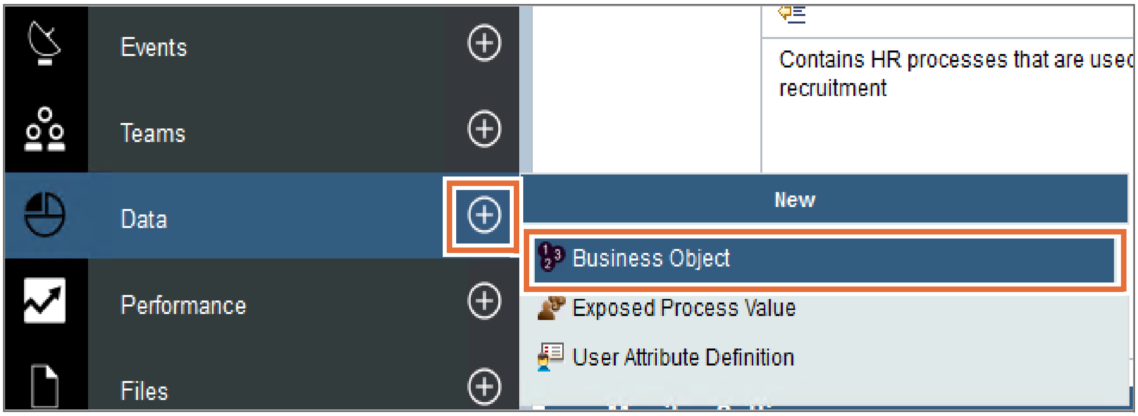
# Exercise instructions

## *Part 1: Build business objects and variables*

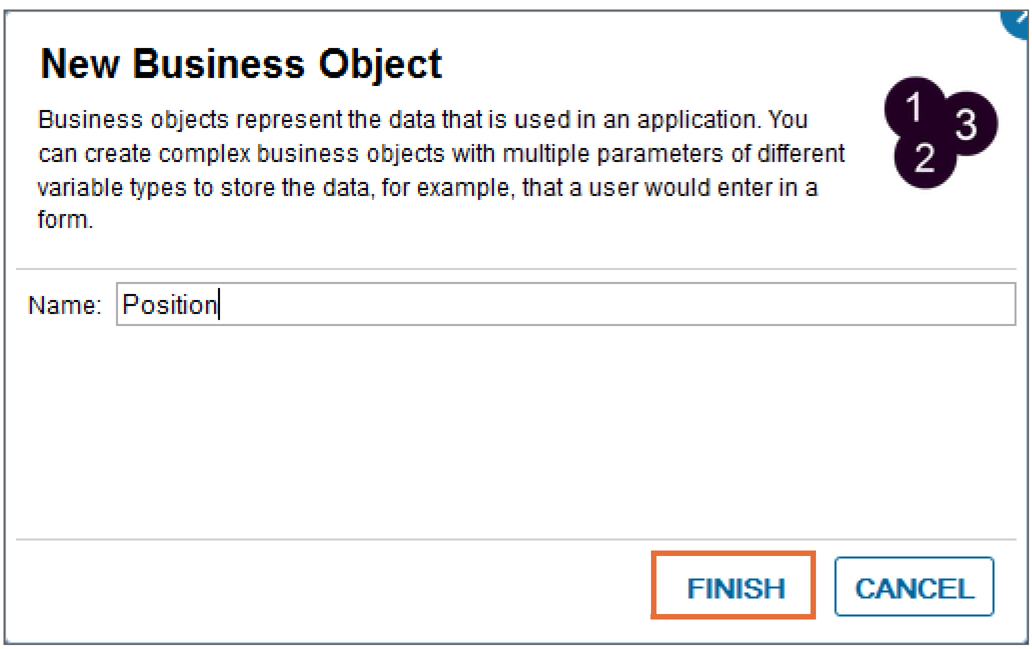
__ 1. In this section, you create a complex variable (business object) to hold the position data. Then, you use this variable on the coach to allow users to input the position data. Later in the lab, you map these variables to move the data through the process.

__ a. Open the **HR Recruitment Process** process application.

__ b. In the library, click the **(+) plus sign** next to the Data category, and select **Business Object**.

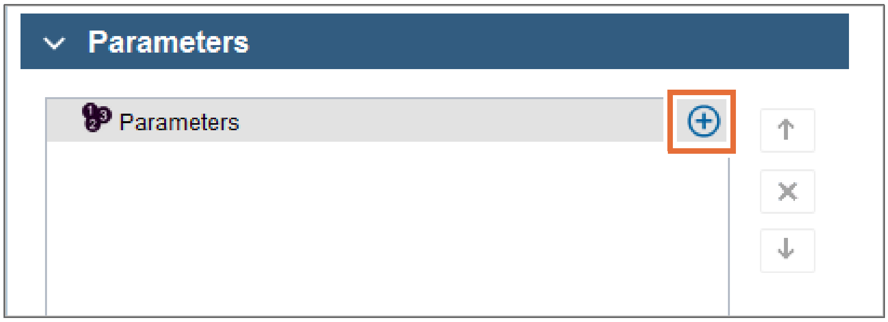



__ c. Name the business object `Position` and click **FINISH**.



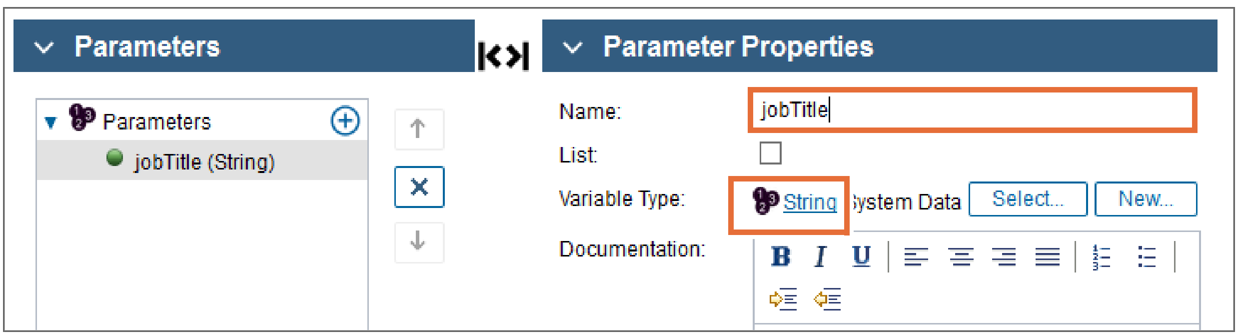

Always follow the naming convention for the variable names. Variable names are case-sensitive. Capitalize the first letter when creating business objects (for example, `Employee`), but use camel case for the parameters or instantiation of the variable (for example, `employeeId`). Camel case refers to a word or string of letters that has no space and has an uppercase letter in a position other than the first letter.

__ d. You now see the settings page for the business object. Add a parameter to the business object by clicking the **(+) plus sign** next to Parameters.




__ e. Change the name of the parameter to: `jobTitle`

__ f. Keep the variable type as **String**.

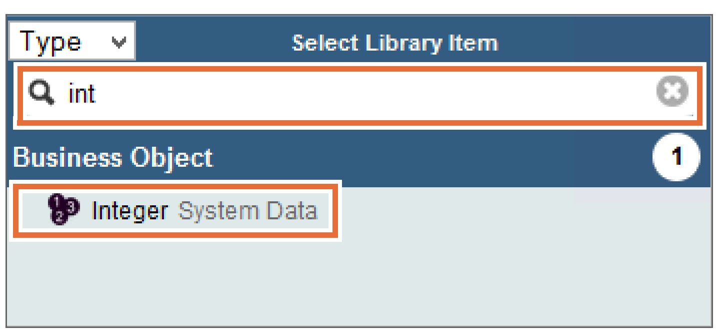



__ g. Repeat steps 1d – 1f to add the following parameters:

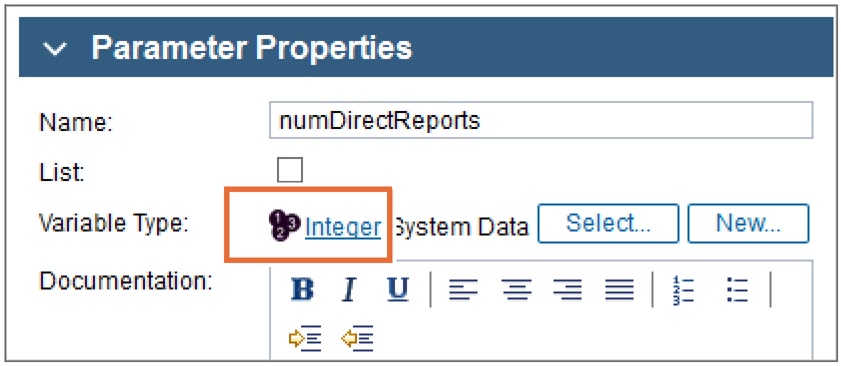
- `jobDescription` (String)
- `jobLevel` (String)

__ h. Add another parameter that is named: `numDirectReports`

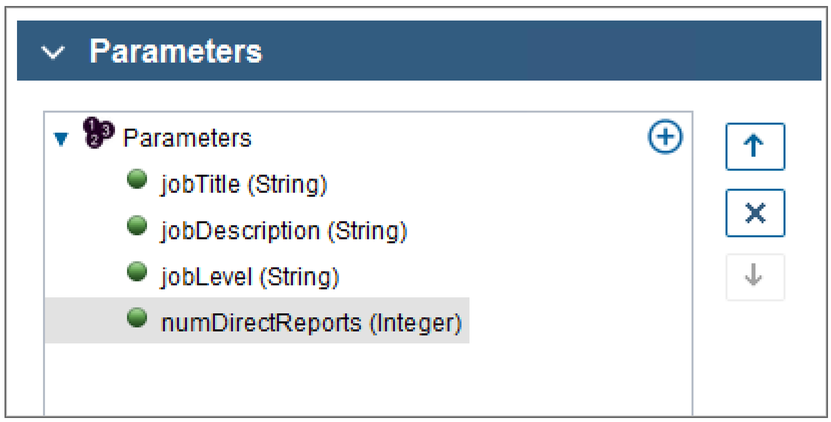
__ i. Change the type of the `numDirectReports` attribute to an Integer. Click **Select** to the right of Variable Type. In the filter, enter `int`. You see the Integer variable type. You can filter the variable types by typing the letters of the variable type you are looking for. Select the Integer variable type.



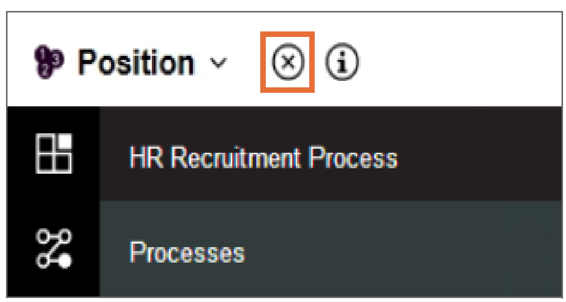
j. Verify that the Integer variable type is selected.




__ k. Save your work. Your business object has four parameters.



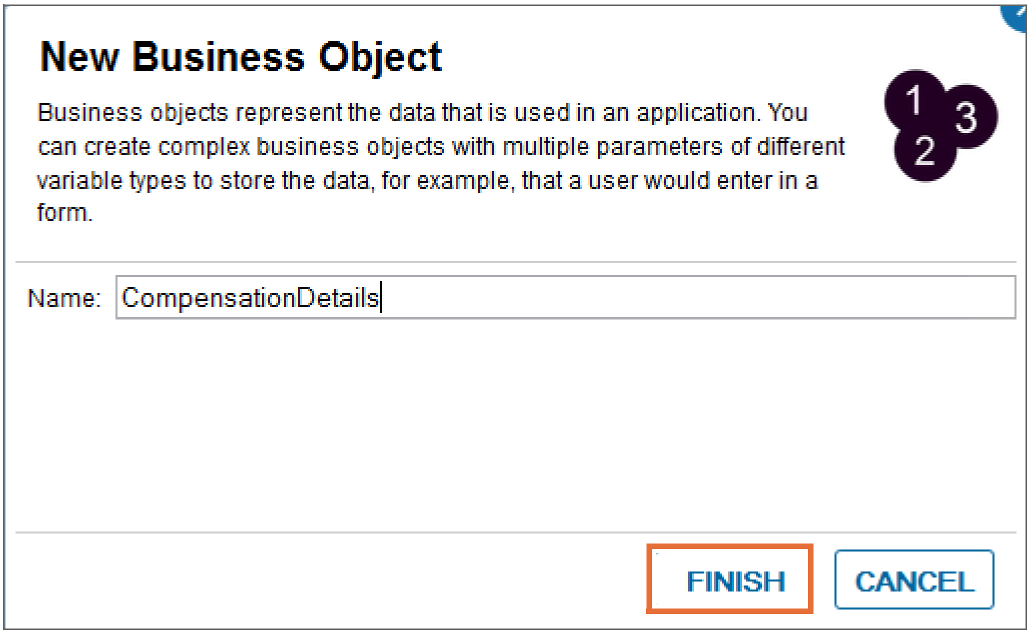

__ l. Save your work and close the business object.




__ 2. Create a second business object that is named `CompensationDetails` to allow users to input compensation data on a coach.

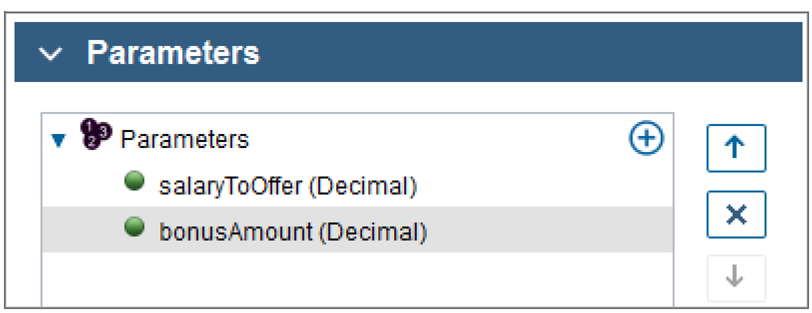
__ a. Click the **(+) plus sign** next to the Data category in the library, and select **Business Object**.

__ b. Name the business object `CompensationDetails` and click **FINISH**.




__ c. Add the following parameters for the complex variable `CompensationDetails`. Remember to change the variable type by clicking **Select** to the right of **Variable Type** to change it to Decimal rather than the default (String):

-`salaryToOffer` (Decimal)
-`bonusAmount` (Decimal)




__ d. Save your work and close the business object.

__ 3. Create a business object that is named `RecruitingDetails` to allow users to input recruiting information on a coach.

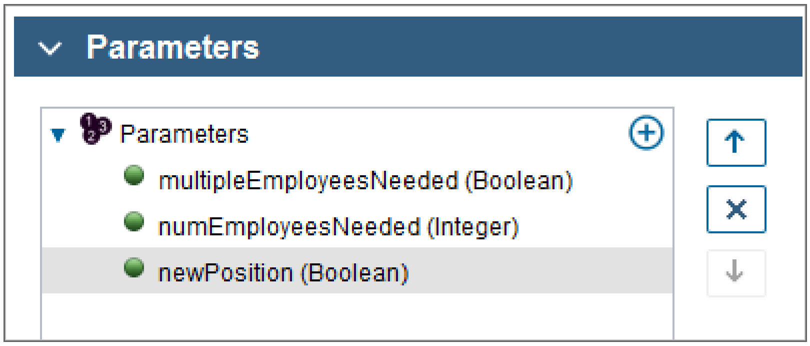
__ a. Create a business object: `RecruitingDetails`

__ b. Include the following parameters in the complex variable:

-`multipleEmployeesNeeded` (Boolean)

-`numEmployeesNeeded` (Integer)

- `newPosition` (Boolean)



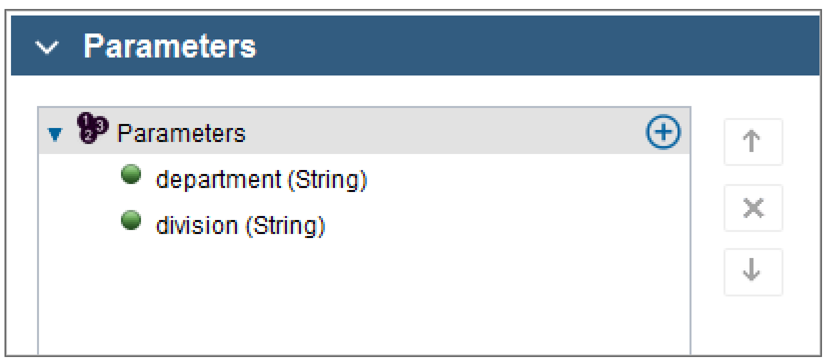

__ c. Save your work and close the business object.

__ 4. Create a fourth business object that is named `DepartmentDetails` to allow users to input department data on a coach.

__ a. Create a business object: `DepartmentDetails`

__ b. Include the following parameters in the complex variable:

-`department (String)`
-`division (String)`




__ c. Save your work and close the business object.

__ 5. Create a business object that is named `HiringRequisition` to save data about the requisition. Model this variable to hold the other objects you created. Use this variable as the main parent business object to hold all the data for the process.

__ a. Create a business object: `HiringRequisition`

__ b. Add the following parameters:

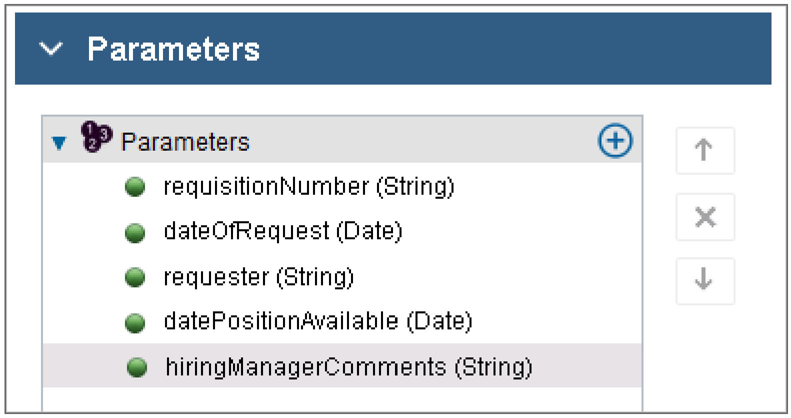
-`requisitionNumber (String)`

-`dateOfRequest (Date)`

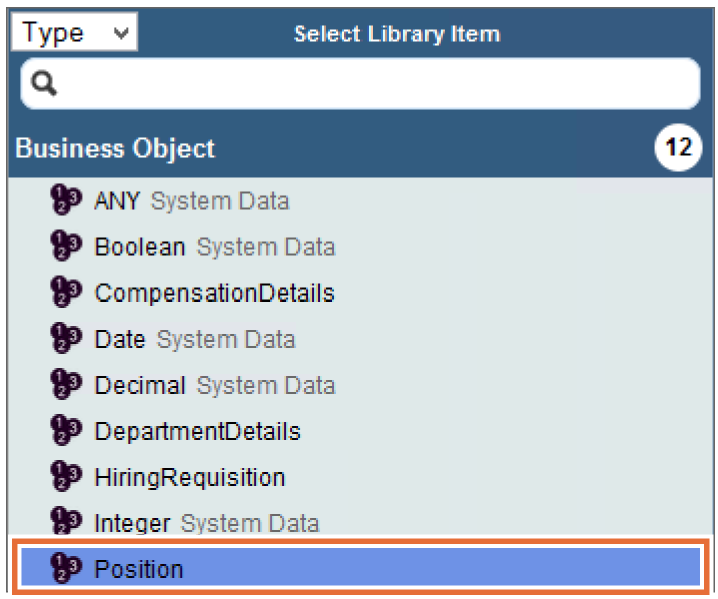
-`requester (String)`

-`datePositionAvailable (Date)`

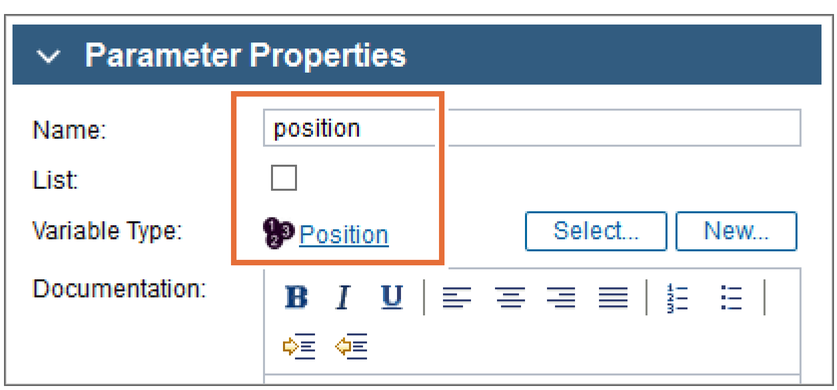
-`hiringManagerComments (String)`




__ c. Add a parameter that is called `position`. This time, click **Select** to the right of **Variable Type** and use the **Position** business object that you created.

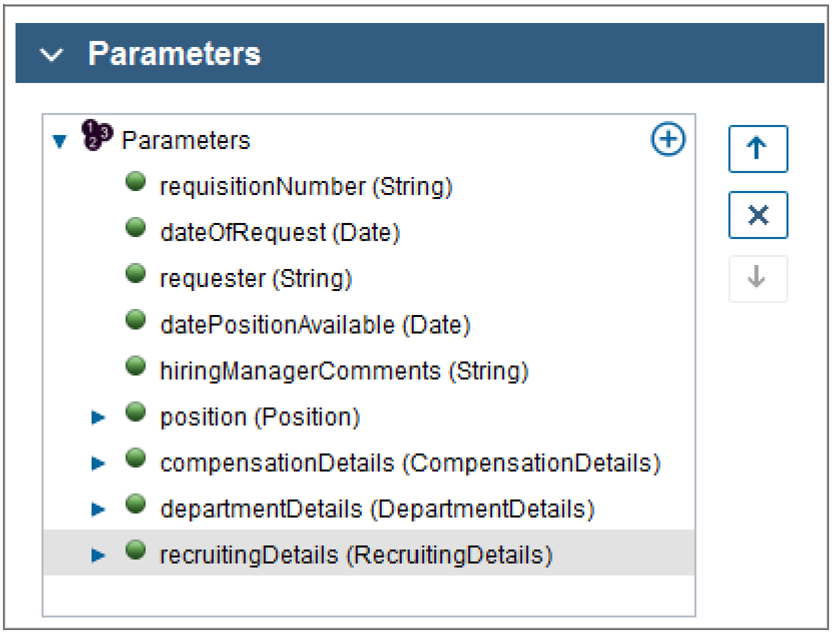



d. Verify that the variable type is capitalized and the variable name starts with a lowercase letter. Recall that business objects are capitalized and variables are not.



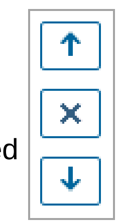
__ e. Continue to add the other business objects as parameters:

```
-compensationDetails (CompensationDetails)
-departmentDetails (DepartmentDetails)
-recruitingDetails (RecruitingDetails)
```




**Information**

Use the up and down arrows to move parameters. The order of the parameters is important. As you add the business objects to your coach, the parameters are sequenced according to where they are in the list.



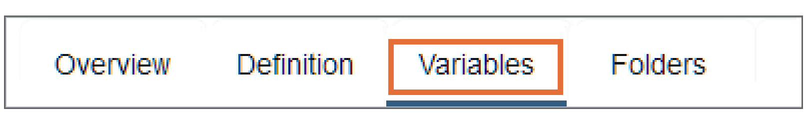
__ f. Save your work and close the business object.

__ 6. Add the `requisitionDetails (HiringRequisition)` variable to the Approve Hire Request linked process as an input and output variable.

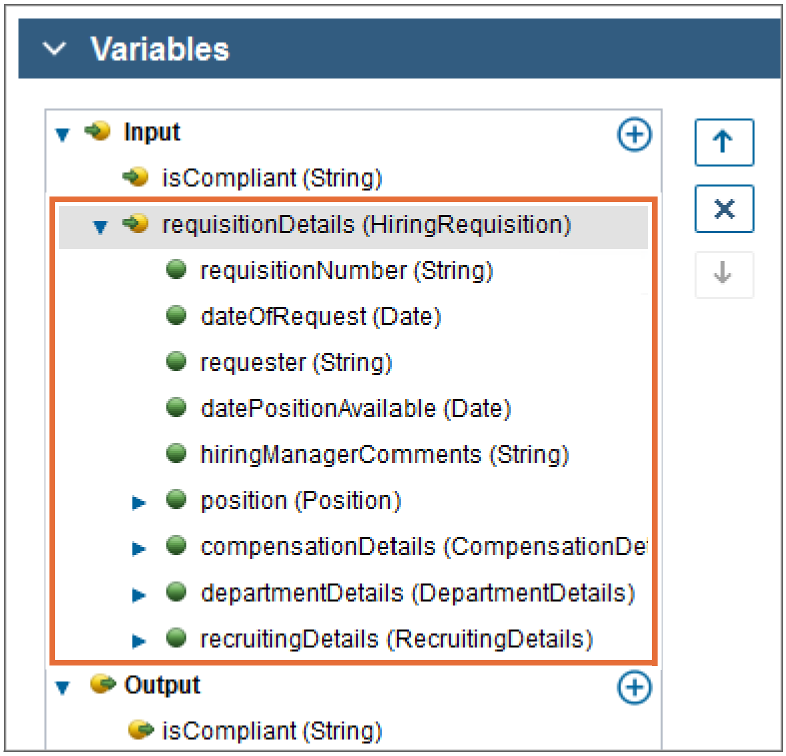
To pass data from the parent process (`Hiring Request Process`) to your linked process (`Approve Hire Request`), you must create and map variables. Because the process is a linked process, you must pass the data object to the linked process by way of the input and output variable settings in the linked process. Variable mapping is required for linked processes because they are a separate process from the parent process.

__ a. Open the **Approve Hire Request** linked process.

__ b. Click the **Variables** tab.



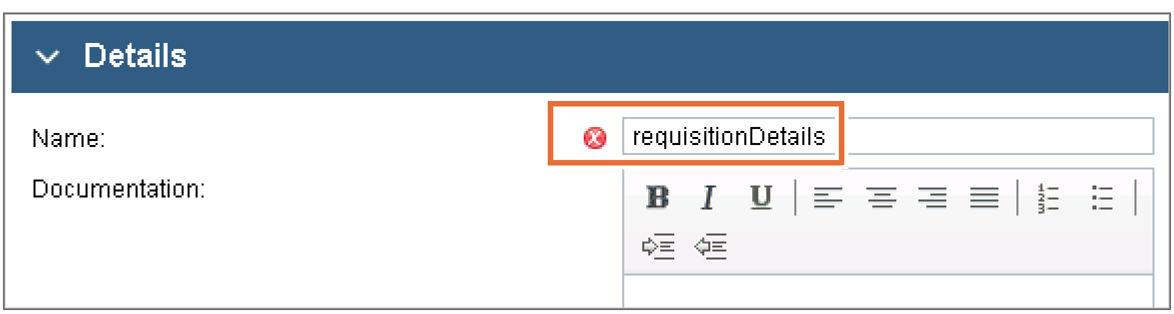
__ c. Add an Input variable `requisitionDetails (HiringRequisition)` to the process. After adding the variable, you can expand the requisitionDetails to view its contents.



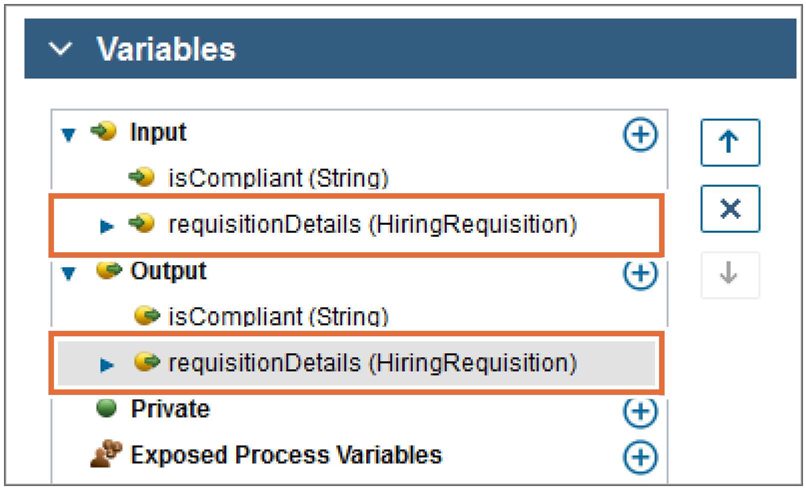

__ d. Add an output variable `requisitionDetails (HiringRequisition)` to the process.

**Important**

When you type in the output name, the variable is immediately flagged because you have an input and an output with the same name, but different variable types. The system defaults the variable to a `String`. When you set the variable type to match the type of the input variable, the error disappears. This can occur anytime an incomplete definition is made including data mapping.



__ e. You added the variables to the subprocess. Save your work.

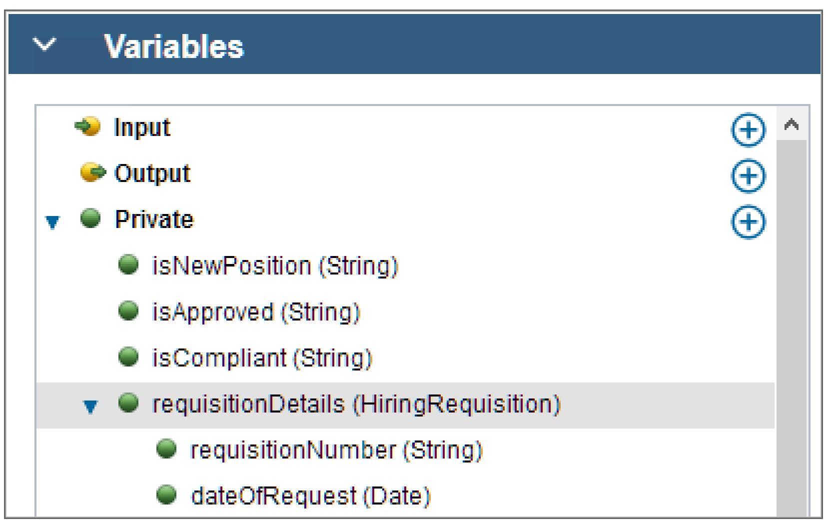



Make sure that the input and output variable names exactly match. If the names do not match, then you have two different variables in the linked process. The input data to the process is not sent as output to the parent process unless you copy the contents of one variable into the other. Copying and pasting the variable name is a good practice.

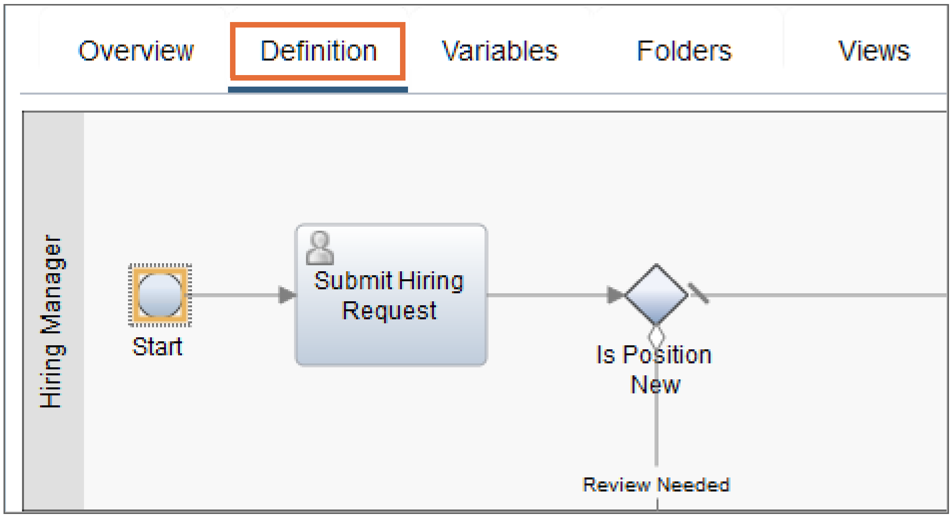
__ 7. Create and map the requisitionDetails variable from the higher-level process to the linked process.

__ a. Open the **Hiring Request Process** and click the **Variables** tab.

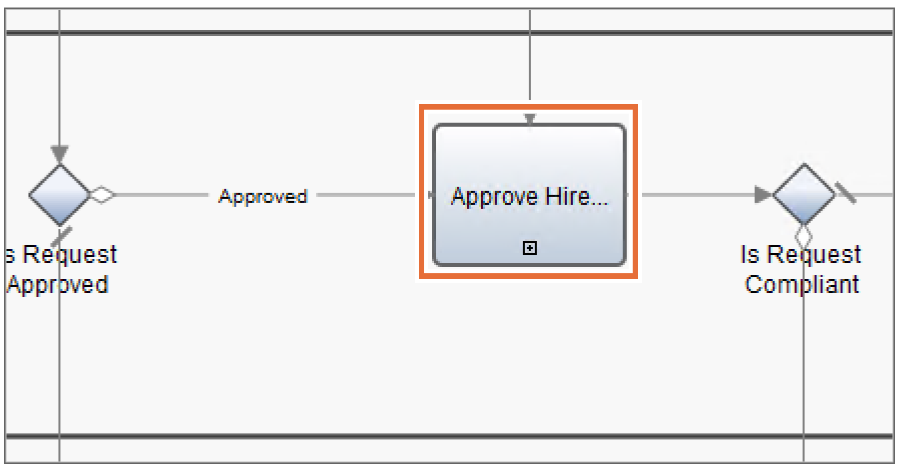
__ b. Create a private variable: `requisitionDetails (HiringRequisition)`



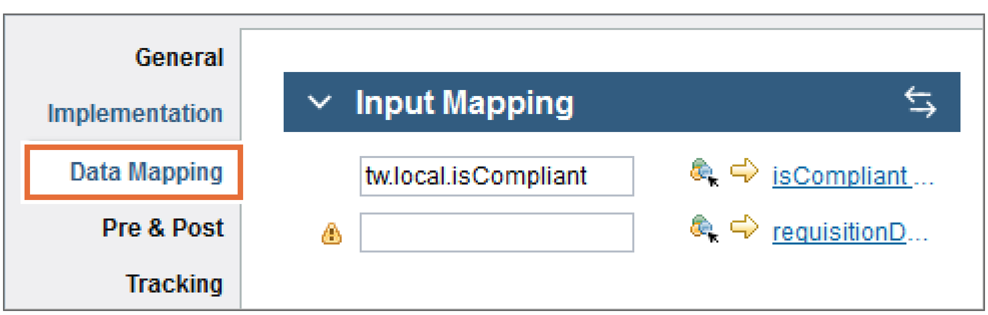
__ c. Click the **Definition** tab.




__ d. Click the **Approve Hire Request** linked process.

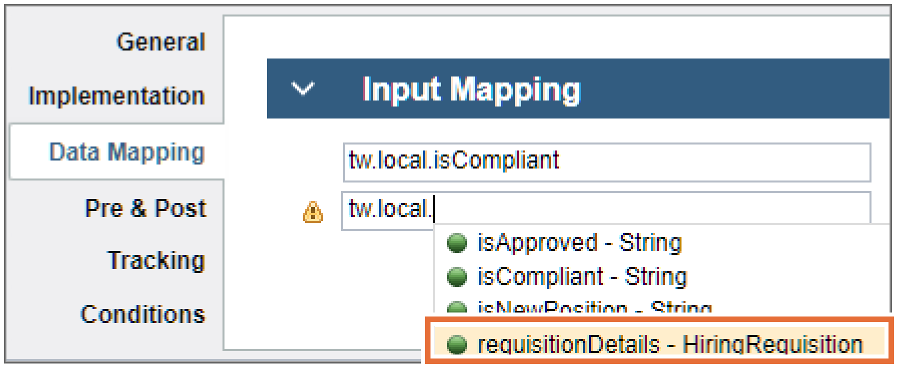



__ e. Click the **Data Mapping** menu in the Properties panel below.



__ f. Map `tw.local.requisitionDetails` to both the **Input Mapping** and **Output Mapping** sections. Place the cursor in the box next to requisitionDetails (HiringRequest) and enter `tw.local.` and press the **Ctrl+Space** to view the auto complete options. Select **requisitionDetails - HiringRequisition**.




__ g. Verify your data mapping and save your work.


**Optional**

You have completed part 1 of 3, so you are now approximately a third done with this exercise. The next section is large, so now would be a good time to take a 10-minute break.

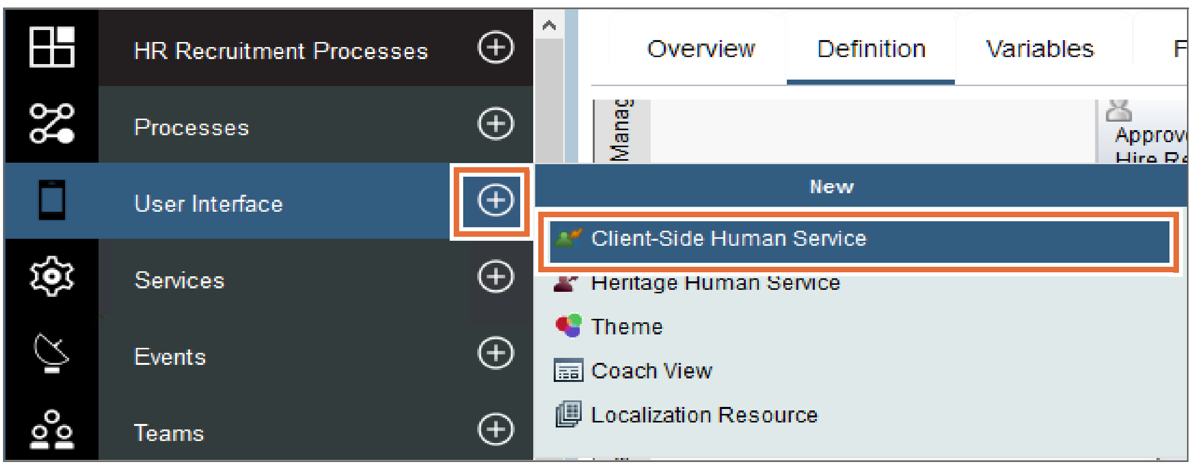
## *Part 2: Create a reusable client-side human service and define input controls*

Up to this point, each activity is implemented by using the Default UI Human Service. In this section, you define and implement custom services and coaches for activities in the process. In the next section, you create a coach to collect information from process teams. The custom service passes the collected information in the coach to the process.

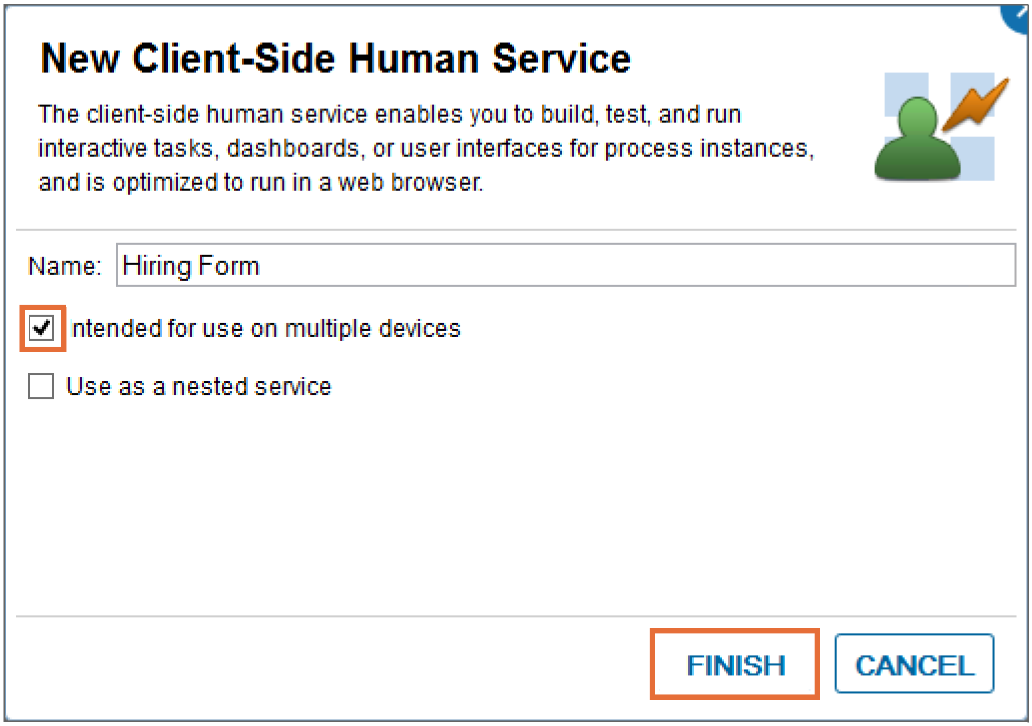
Two activities on the process need a coach that has the same fields: the Submit Hiring Request and the Approve Hiring Request. One requires the ability to input data, and the other needs the data to be read-only with an approval mechanism. In this part of the exercise, you create a single coach for both activities. Because the same data can be shown for multiple activities on the process, you create a coach that can be reused for data input and viewing data (read-only).

__ 1. Create a client-side human service.

__ a. In the Process Designer library, click the **(+) plus sign** next to User Interface, and select **Client-Side Human Service**.

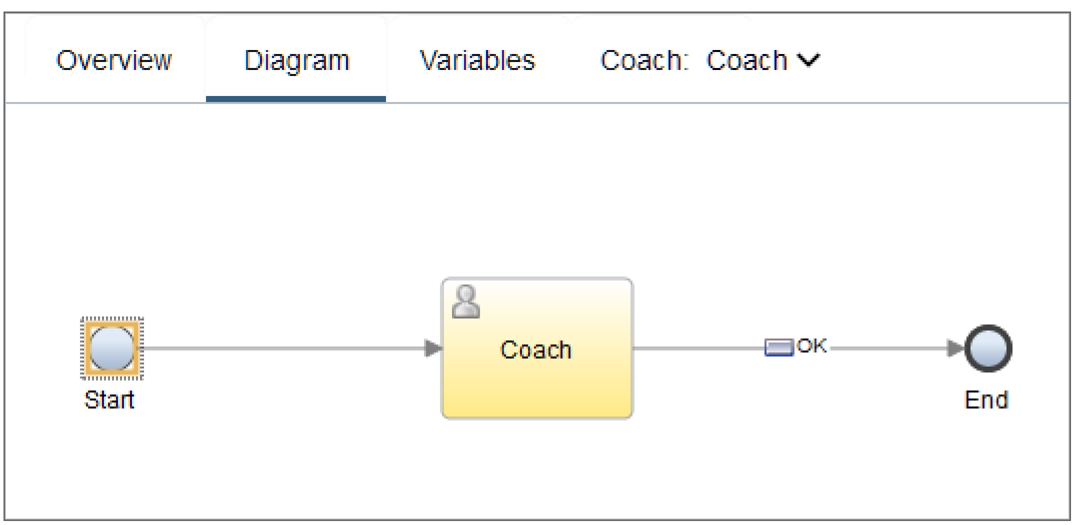


__ b. Name the client-side human service: `Hiring Form.` Select the **Intended for use on multiple devices** checkbox and click **FINISH**.




Intended for use on multiple devices enables the new service to be used on multiple device types. With this option, you can use responsive controls for all the new coaches and coach content that you add to the service.

__ c. The new human service opens in the Process Designer editor.

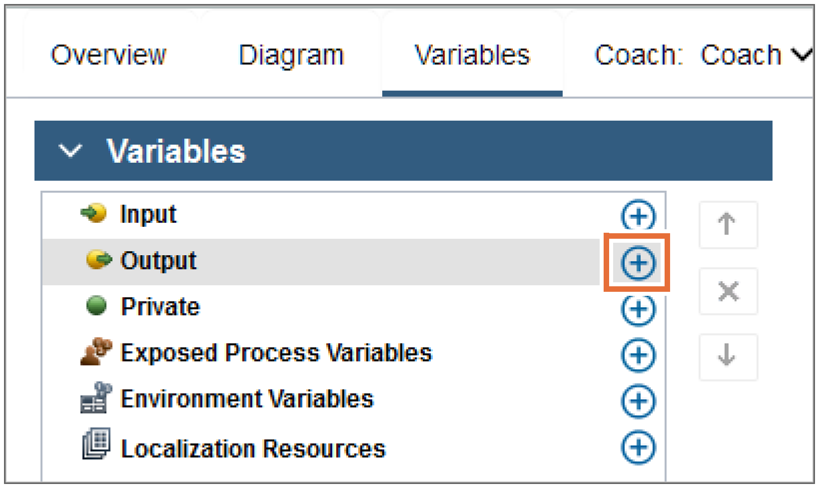



__ 2. Add the `requisitionDetails` output variable to the human service.

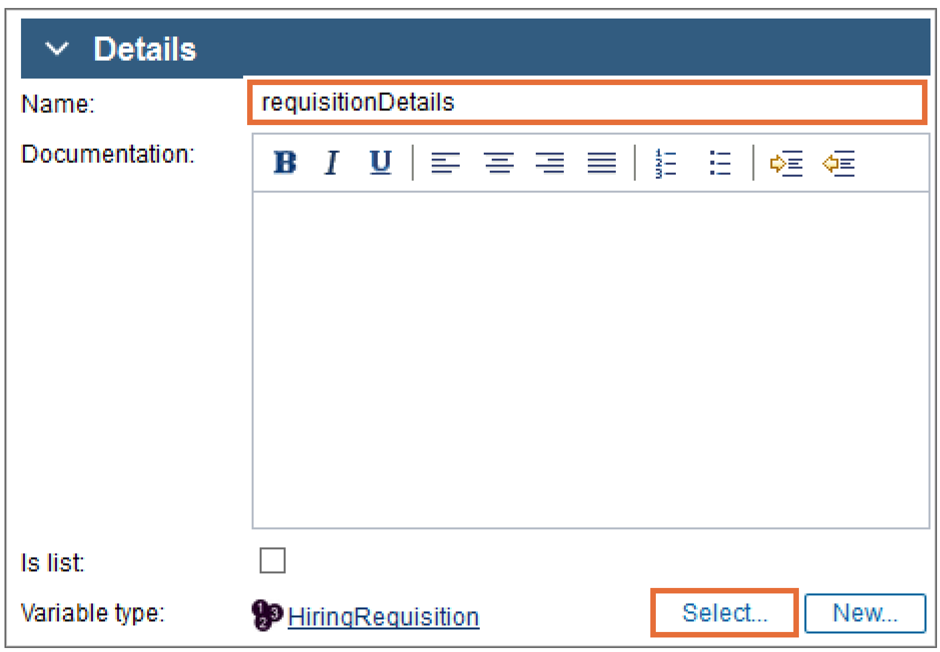
__ a. In the `Hiring Form` service, click the **Variables** tab.

__ b. Click the **(+) plus sign** next to Output.




__ c. Name the variable: `requisitionDetails`. Click **Select** to the right of the Variable type option to select **HiringRequisition**.

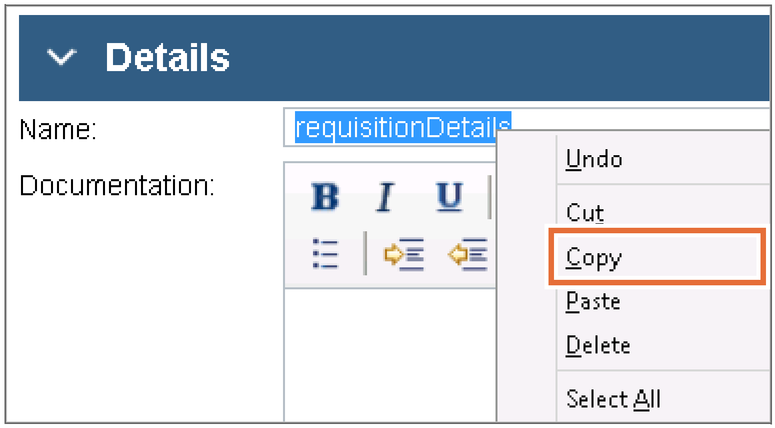



Because you need all of the parameters of the **HiringRequisiton** object, you use the full object in the coach. If you needed only part of the object, include that parameter and map the parameter from the parent object when mapping inputs and outputs.

__ d. Save your work.

__ 3. Because you reuse this coach for multiple activities, add matching input variables to the human service. If any data is entered as part of the process instance from the first activity, you can present it on the coach.

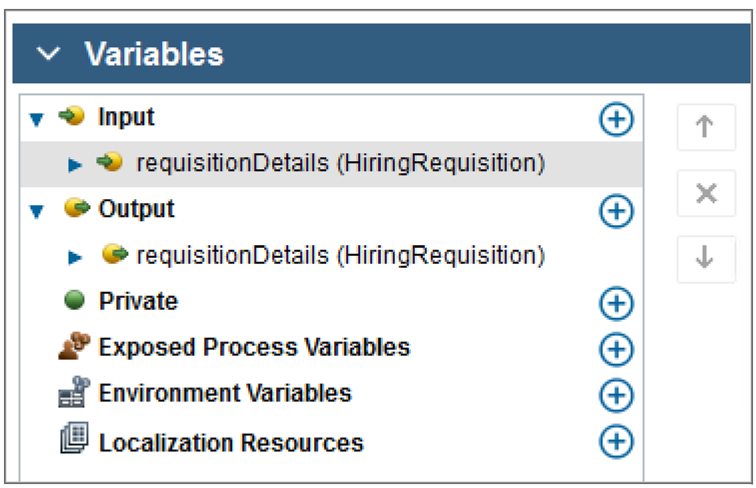
__ a. Click the output variable `requisitionDetails`, highlight the name, and copy it.




__ b. Click the **(+) plus sign** next to Input and paste `requisitionDetails` into the **Name** field.

__ c. Click **Select** to the right of the Variable type option.

__ d. Change the type to **HiringRequisition**.

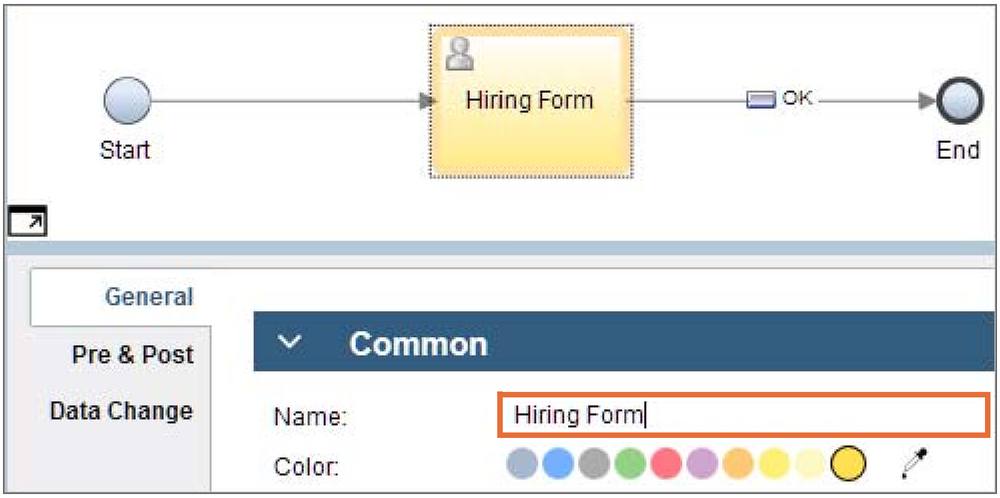



__ e. Save your work.

__ 4. Now that the variables are created, add the inputs that are bound to the variables in the coach.

__ a. In the Hiring Form service, click the **Diagram** tab.

__ b. Click the **Coach** node on the canvas, and in the **General > Common** section, change the name of the coach to: `Hiring Form`

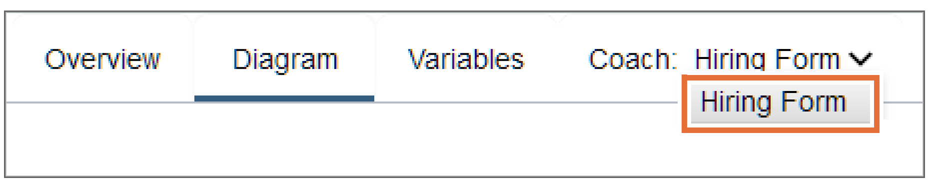



__ c. Save your work.

__ 5. Build the coach by using a two-column grid and add the Requisition Number and Requester variables.

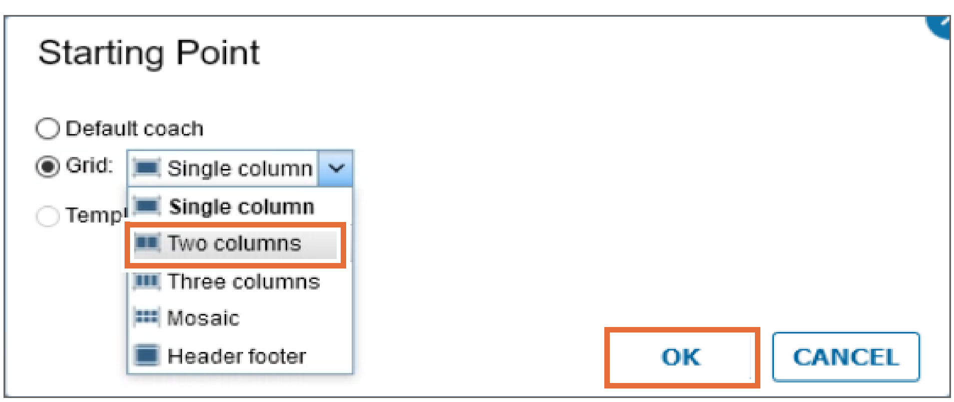
Build the first section that contains all the simple variables from the Requisition Details object. Consider how items are laid out. This consideration determines whether the coach is laid out vertically or horizontally.

__ a. For the Hiring Form service, click the down arrow next to **Coach: Hiring Form** and select **Hiring Form**.

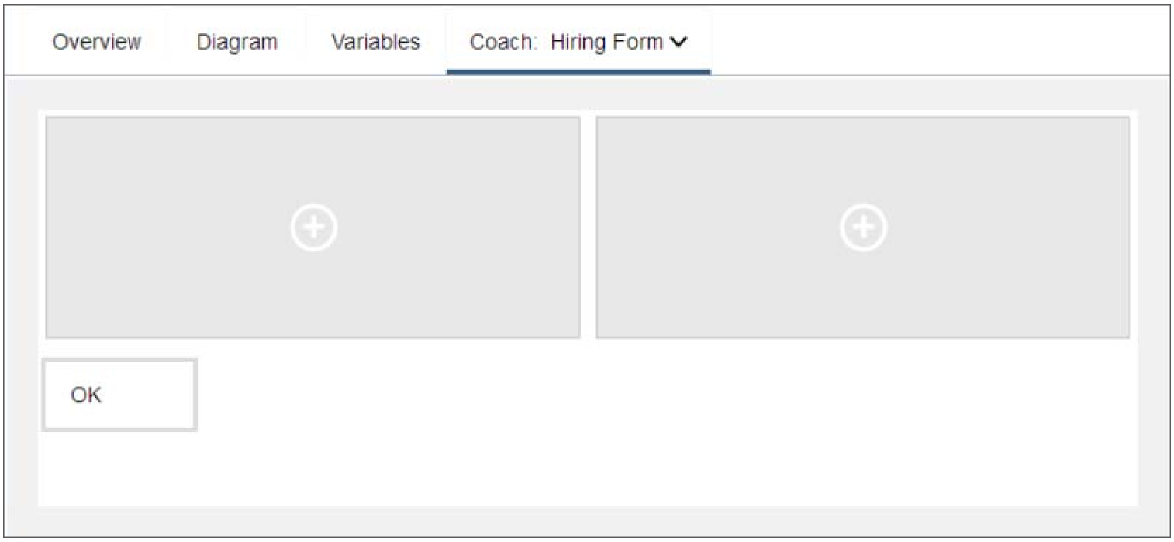



When you renamed the Coach in the diagram, the name changed at the top from Coach to Hiring Form

__ b. A message box for setting a Starting Point appears. Click **Grid** and in the drop-down, select **Two Columns** and click **OK**.

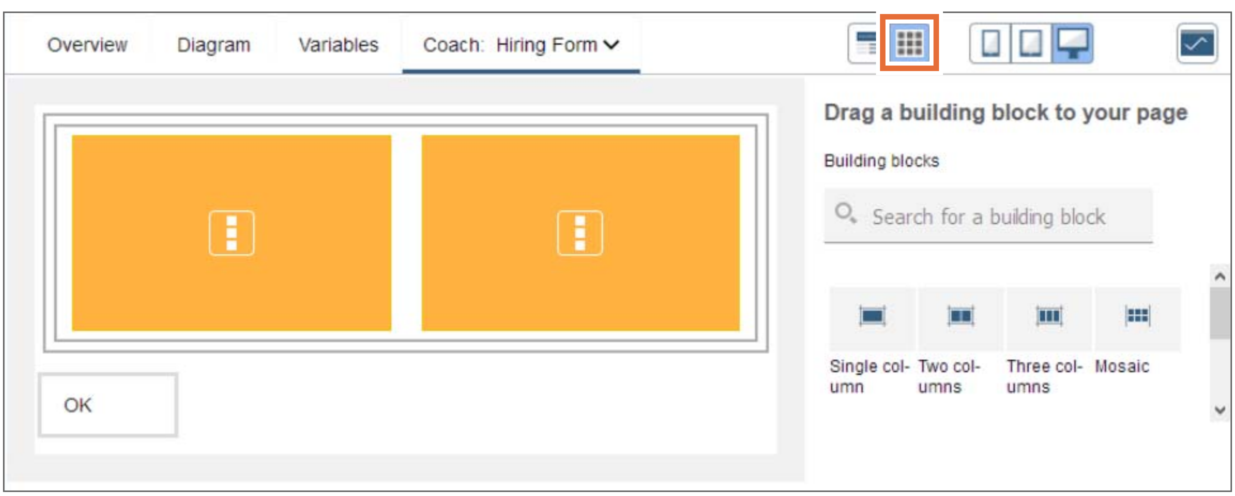


A two-column grid appears for the **Hiring Form** coach layout. This layout is the layout that you start from to build your coach.



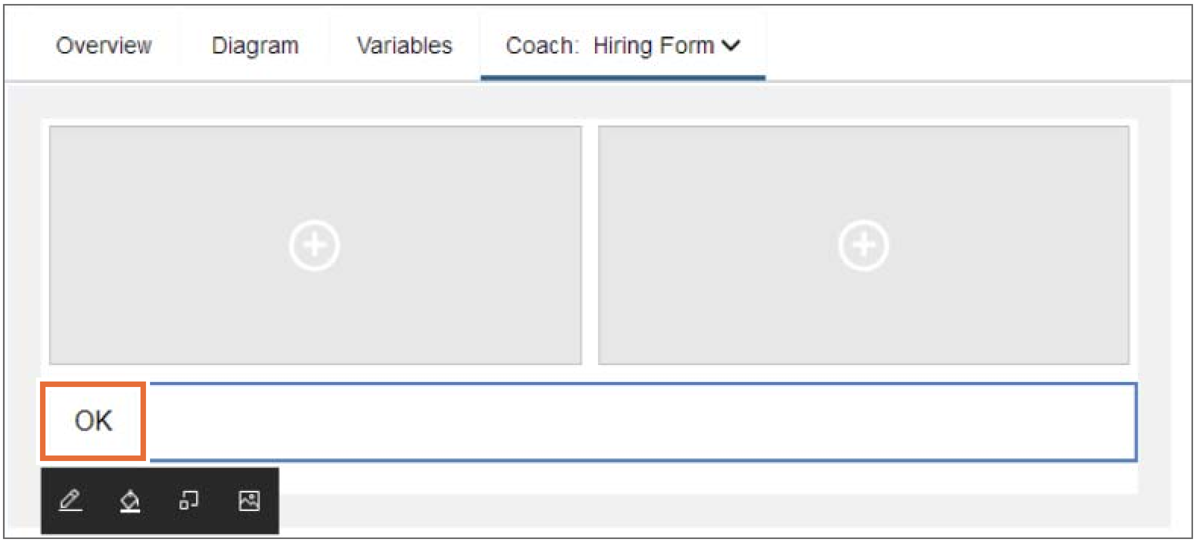

The grid layout and content layout are mutually exclusive. When in grid view, you can arrange grids. When in content view, you cannot arrange grids but you can add content to each grid cell. The Content / Grid toggle at the top highlights which is displayed. The color of the grids might vary slightly depending on your monitor settings.


__ c. Click the **Grid toggle** to view the grid layout.

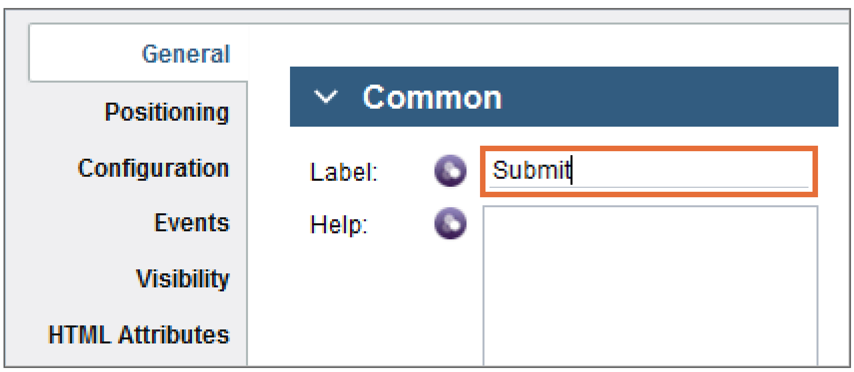



__ d. Click the **Content toggle** to view the content layout.

__ e. Click **OK** on the coach canvas to select the button.

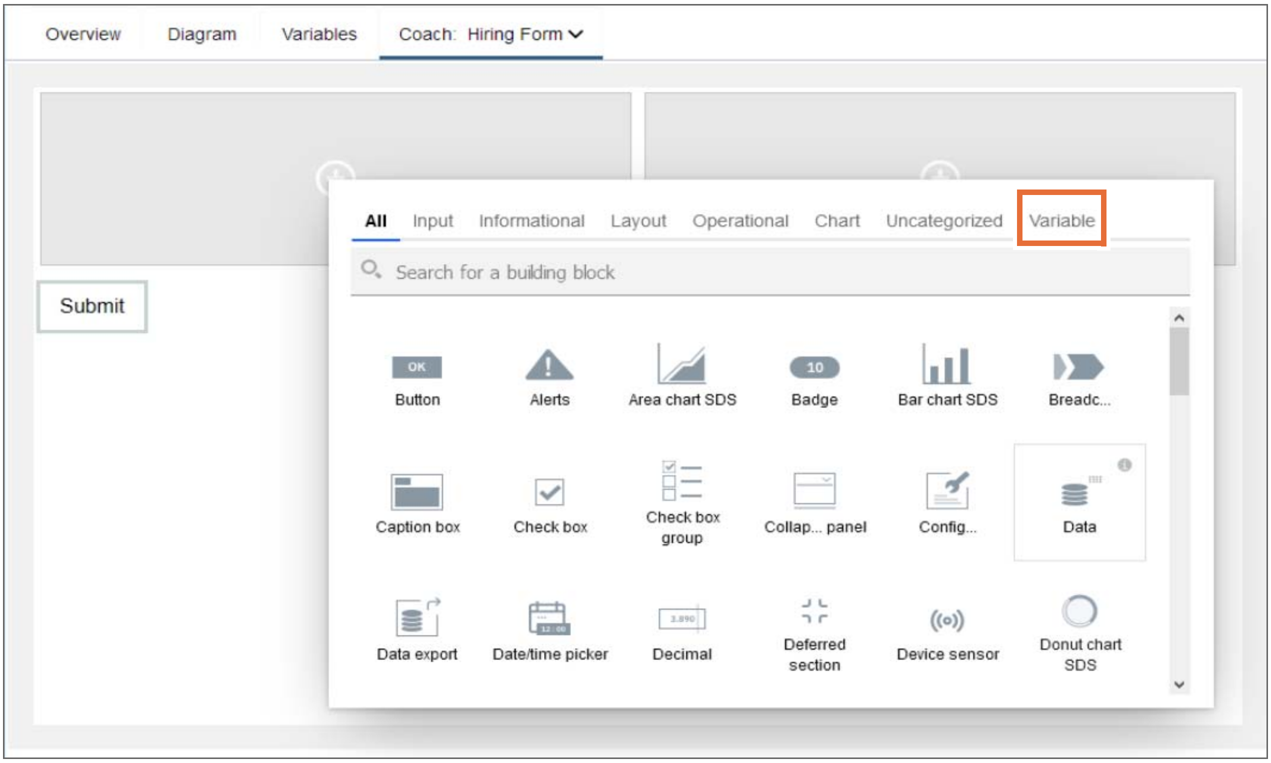



__ f. In the **General > Common** section, change the label of the control button to: `Submit`

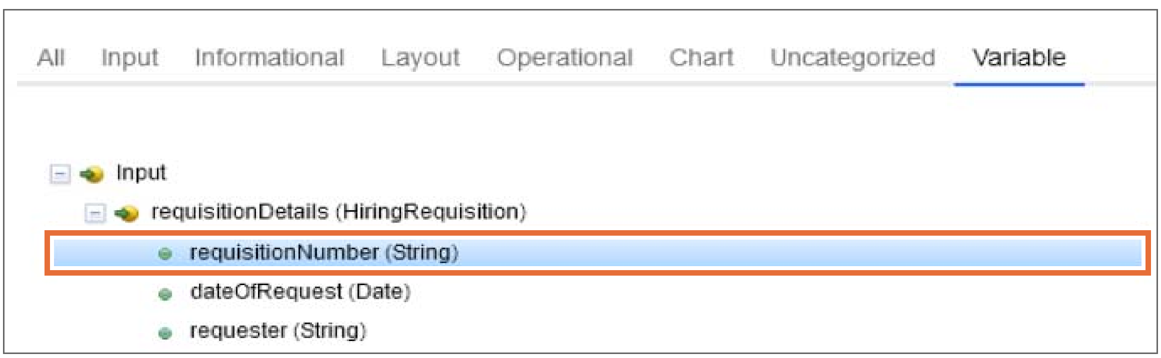



__ g. The first inputs that you add to the coach are the parameters of the `requisitionDetails` variable. In the left column of the grid, click the white **(+) plus sign** in the middle.

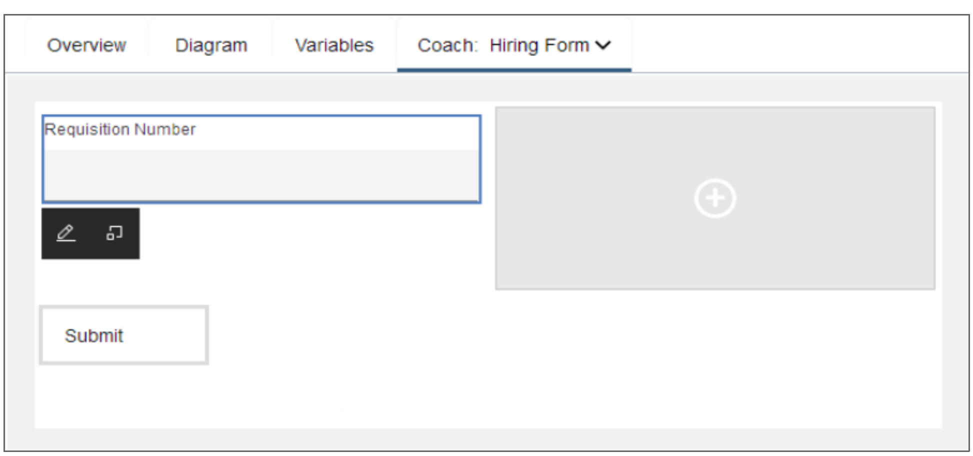
__ h. In the newly opened window, click the **Variable** tab.



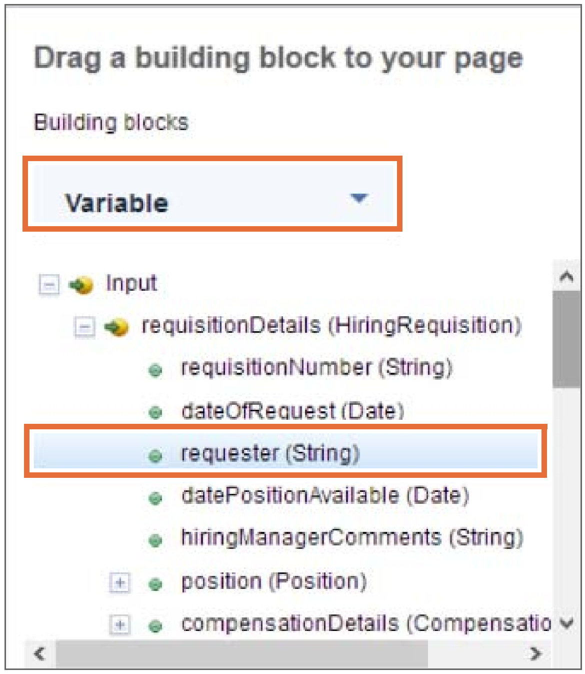

__ i. Select **Input > requisitionDetails > requisitionNumber**



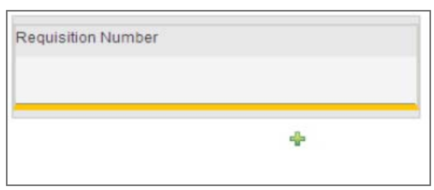

The result is that the requisition number is placed in the left column of the grid and is highlighted.



__ j. Select **Variable** in the **Building blocks** panel and select **Input > requisitionDetails > requester**

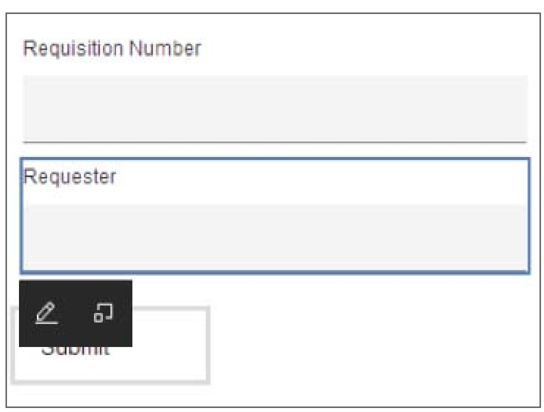



__ k. While holding down the left mouse button, drag the `requester` variable underneath the **Requisition Number** input control. Let go of the mouse button when the bottom border displays a gold highlight.



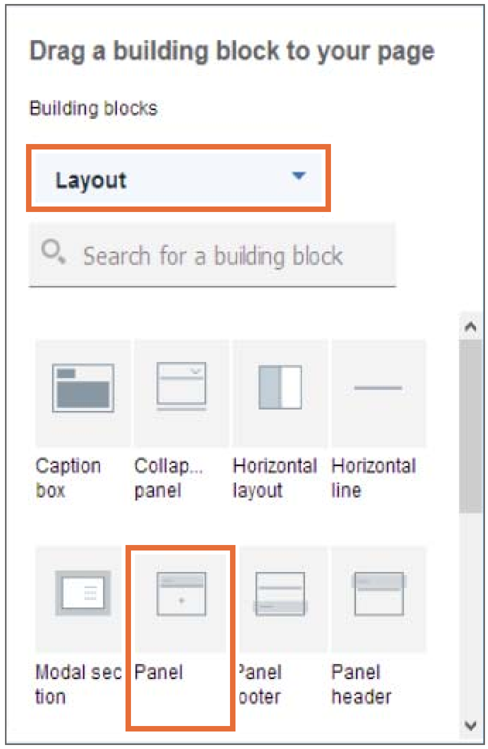

A **green plus (+)** is displayed when you are in an area that the item can be dropped. A **red circle with an (x)** appears if the mouse is hovering in an area where it cannot be dropped.

__ l. Verify the **Requester** input control is placed below **Requisition Number**.

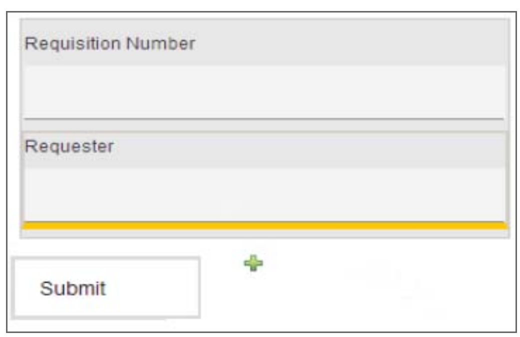


6. Create a panel and populate it with the date details

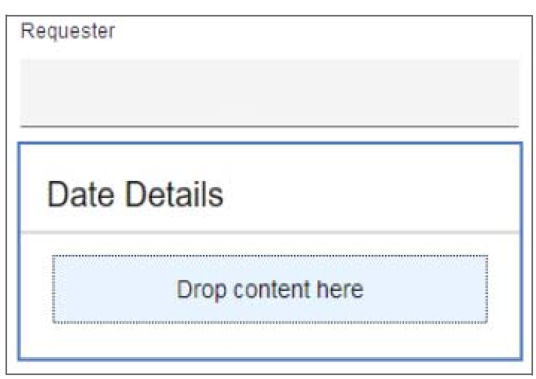
__ a. Select **Layout** in the **Building blocks** panel and select **Panel**




__ b. While holding down the left mouse button, drag the **Panel** underneath the **Requester** input control. Let go of the mouse button when the bottom border displays a gold highlight.



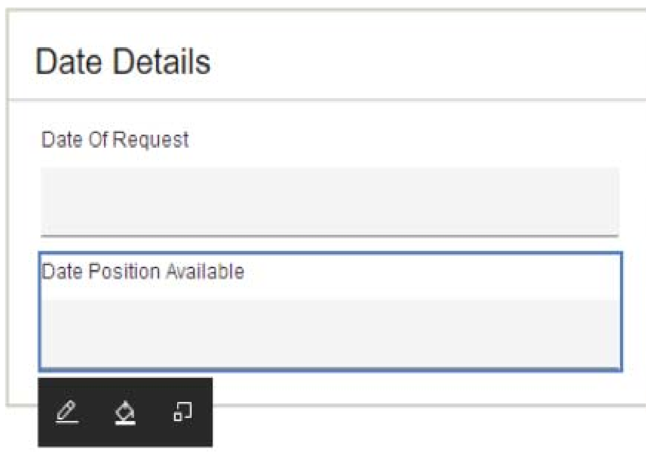

__ c. Since the panel is put in place with the name already highlighted, change the name to `Date Details` and press the **Enter** key.



__ d. Select **Variable** in the **Building blocks** panel and select **Input > requisitionDetails > dateOfRequest**


__ e. While holding down the left mouse button, drag the `dateOfRequest` to the section of the **Date Details** panel labeled **Drop content here**. Let go of the mouse button when the **Drop content here** section is highlighted and the cursor displays a green plus.

__ f. Repeat the last two steps to drag `datePositionAvailable` underneath the **Date Of Request** input control.

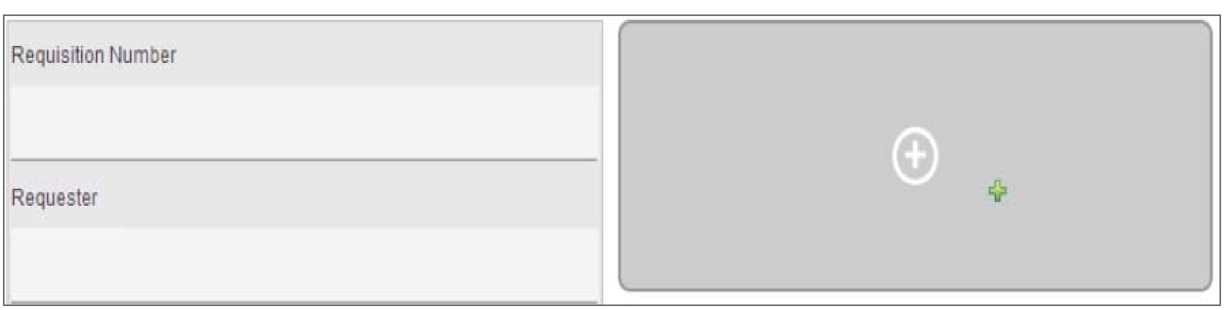



__ g. Save your work.

__ 7. Create a panel and populate it with the Position business object.

__ a. Select **Layout** in the **Building blocks** panel and select **Panel**

__ b. While holding down the left mouse button, drag the **Panel** to the grid on the right. Let go of the mouse button when the section is highlighted and the cursor displays a green plus.

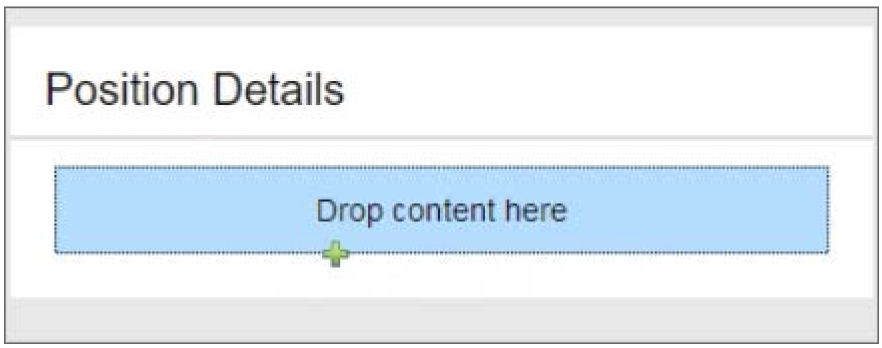


__ c. With the name highlighted, label the panel `Position Details`.

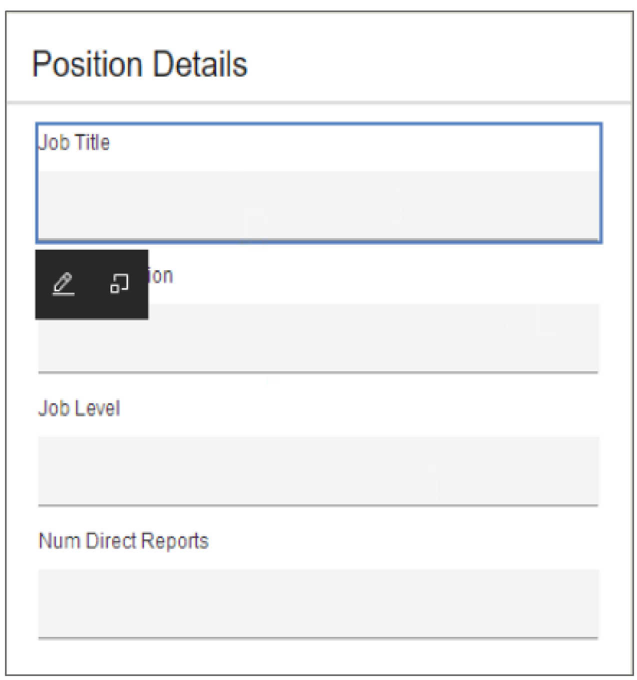

__ d. Click the empty space below the panel on the canvas to refocus the mouse.

__ e. Select **Variable** in the **Building blocks** panel and select **Input > requisitionDetails > position**

__ f. While holding down the left mouse button, drag `position` to the section of the **Position Details** panel labeled **Drop content here**. Let go of the mouse button when the **Drop content here** section is highlighted and the cursor displays a green plus.



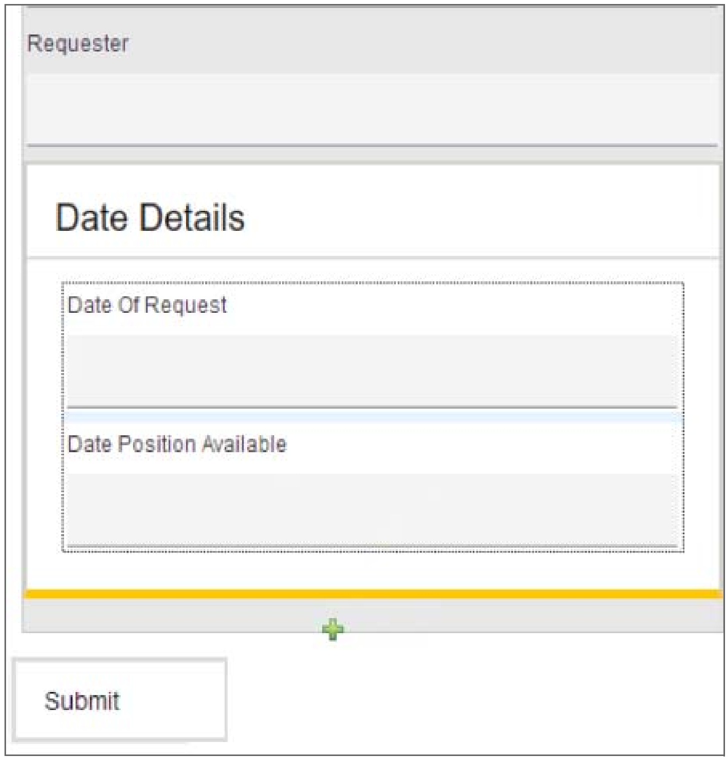

__ g. The section populates with the parameters of the `position` variable.



__ 8. Create a panel and populate it with the Recruiting Details business object.

__ a. Select **Layout** in the **Building blocks** panel and select **Panel**

__ b. While holding down the left mouse button, drag a **Panel** underneath the **Date Details** panel. Let go of the mouse button when the bottom border displays a gold highlight.

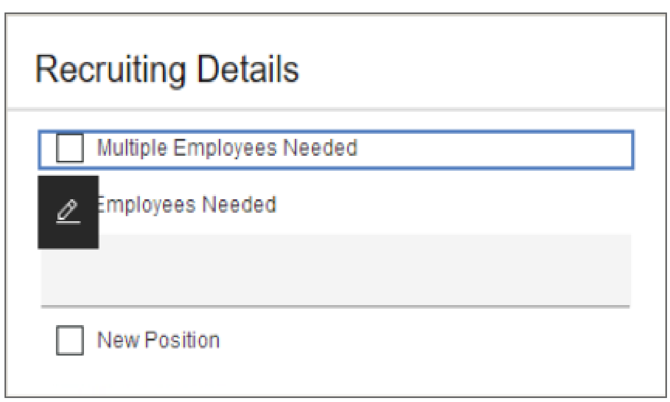



__ c. With the name highlighted, label the panel `Recruiting Details` and press Enter

__ d. Select **Variable** in the **Building blocks** panel and select **Input > requisitionDetails > recruitingDetails**

__ e. While holding down the left mouse button, drag `recruitingDetails` to the section of the **Recruiting Details** panel labeled **Drop content here**. Let go of the mouse button when the **Drop content here** section is highlighted and the cursor displays a green plus.

__ f. The section populates with the parameters of the `recruitingDetails` variable.

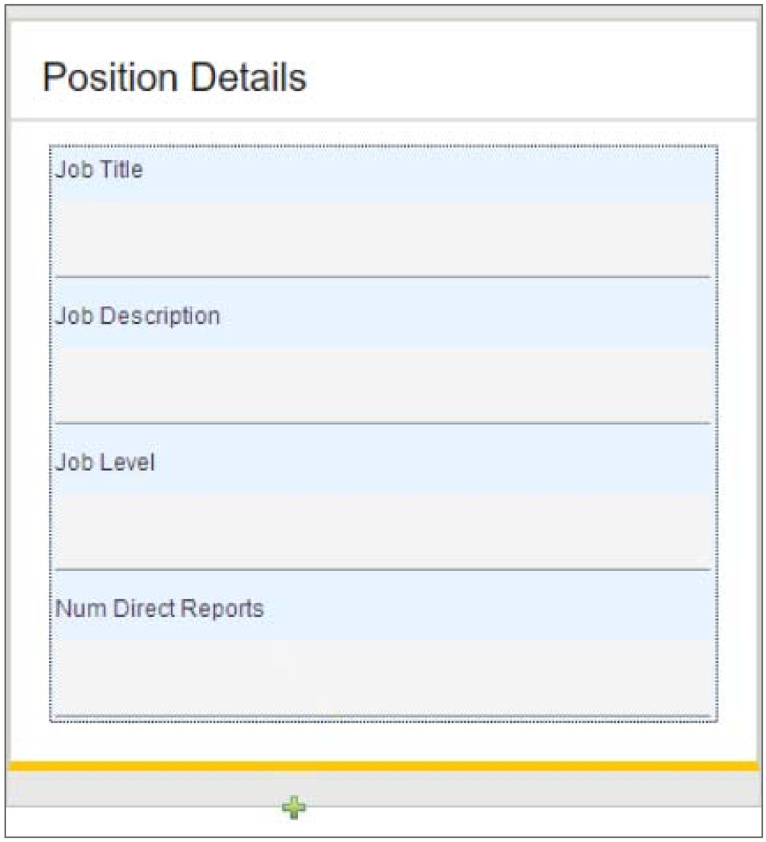



__ g. Save your work.

__ 9. Create a panel underneath Position Details and to the right of Recruiting Details and populate it with the Compensation Details business object.

__ a. Select **Layout** in the **Building blocks** panel and select **Panel**

__ b. While holding down the left mouse button, drag the **Panel** underneath the **Position Details** panel. Let go of the mouse button when the bottom border displays a gold highlight.



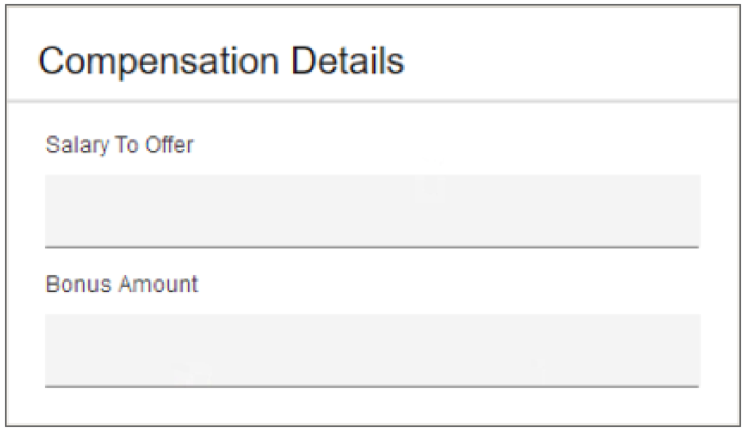

__ c. With the name highlighted, label the panel `Compensation Details` and press Enter

__ d. Select **Variable** in the **Building blocks** panel and select **Input > requisitionDetails > compensationDetails**

__ e. While holding down the left mouse button, drag `compensationDetails` to the section of the **Compensation Details** panel labeled **Drop content here**. Let go of the mouse button when the **Drop content here** section is highlighted and the cursor displays a green plus.

__ f. Click the empty space below the panel on the canvas to refocus the mouse.

__ g. The section populates with the parameters of the `compensationDetails` variable.




__ 10. Create a panel and populate it with the Department Details business object.

__ a. Select **Layout** in the **Building blocks** panel and select **Panel**

__ b. While holding down the left mouse button, drag the **Panel** underneath the **Recruiting Details** panel. Let go of the mouse button when the bottom border displays a gold highlight.

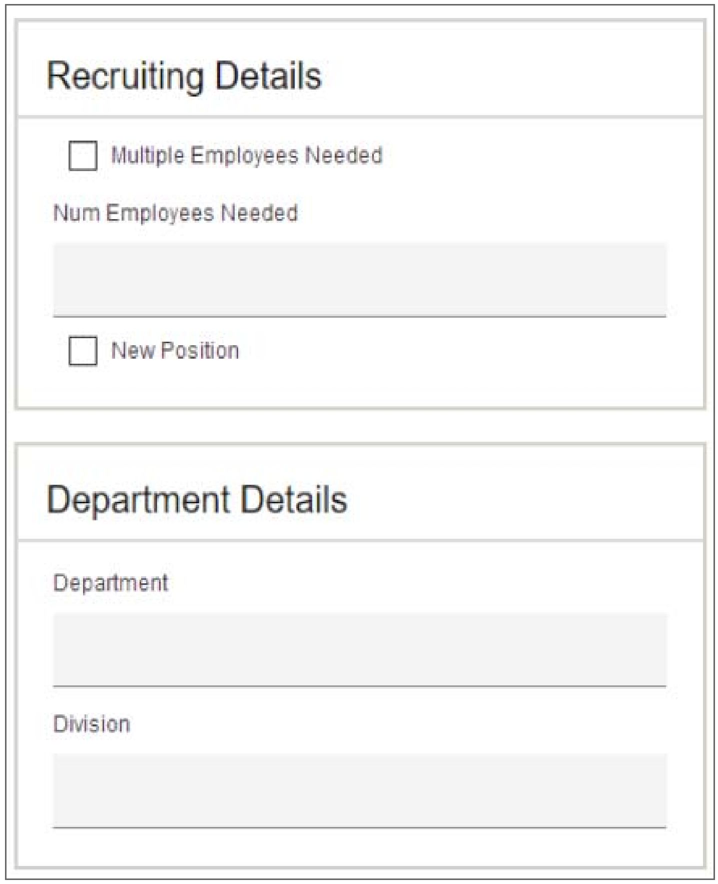
__ c. With the name highlighted, label the panel `Department Details` and press Enter

__ d. Select **Variable** in the **Building blocks** panel and select **Input > requisitionDetails > departmentDetails**

__ e. While holding down the left mouse button, drag `departmentDetails` to the section of the **Department Details** panel labeled **Drop content here**. Let go of the mouse button when the **Drop content here** section is highlighted and the cursor displays a green plus.

__ f. Click the empty space below the panel on the canvas to refocus the mouse.

__ g. The section populates with the parameters of the `departmentDetails` variable and is located underneath the Recruiting Details panel.

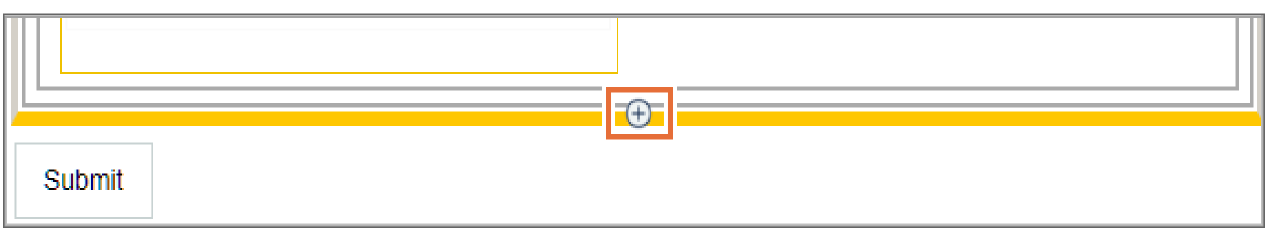



__ h. Save your work.

__ 11. Place a grid above the Submit button and add the Hiring Manager Comments input control.

__ a. Switch to the grid view by clicking the **Grid toggle**. This setting shows the grids on the canvas.

__ b. Hover your mouse and click the **(+) plus sign** on the bottom of the outer-most grid to add a grid above the Submit button.




__ c. Click the empty space below the panel on the canvas to refocus the mouse.

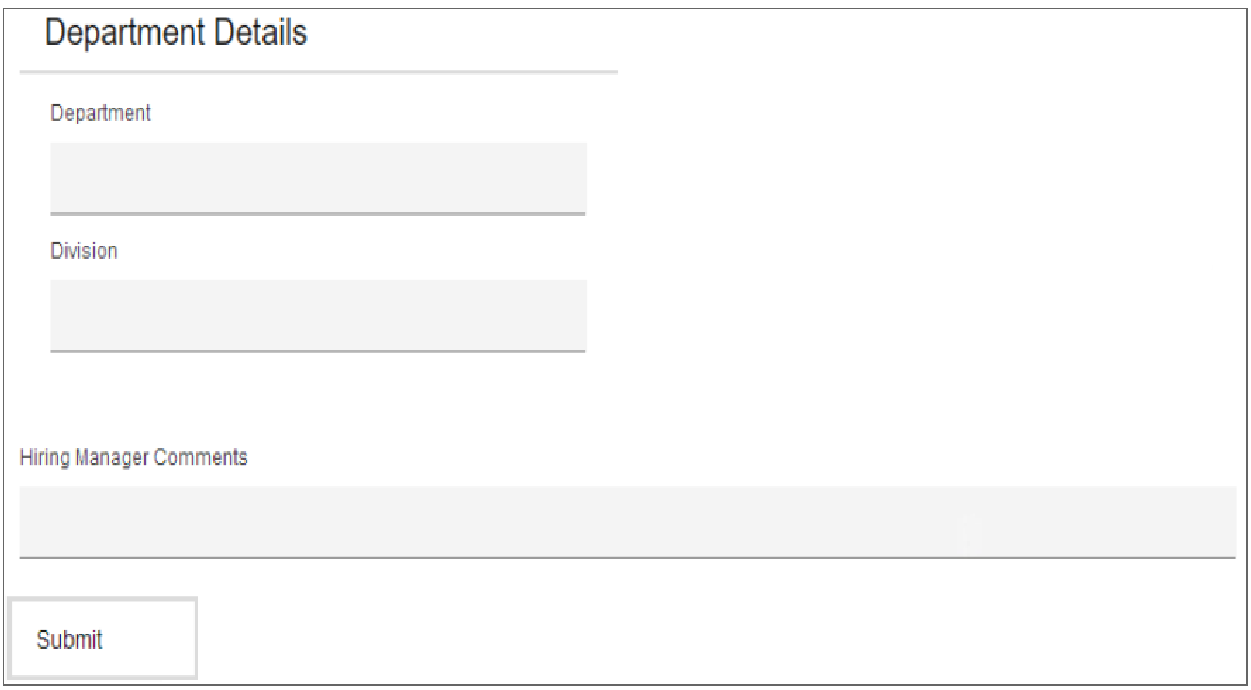
Your grid should have only a single line between the orange space and the submit button.


__ d. Switch to the Content view by clicking the **Content toggle**.

__ e. Under the Building Blocks panel, select **Input > requisitionDetails > hiringManagerComments**

__ f. While holding down the left mouse button, drag `hiringManagerComments` to the grid. Let go of the mouse button when the grid is highlighted and the cursor displays a green plus.

__ g. Click the empty space below the panel on the canvas to refocus the mouse.



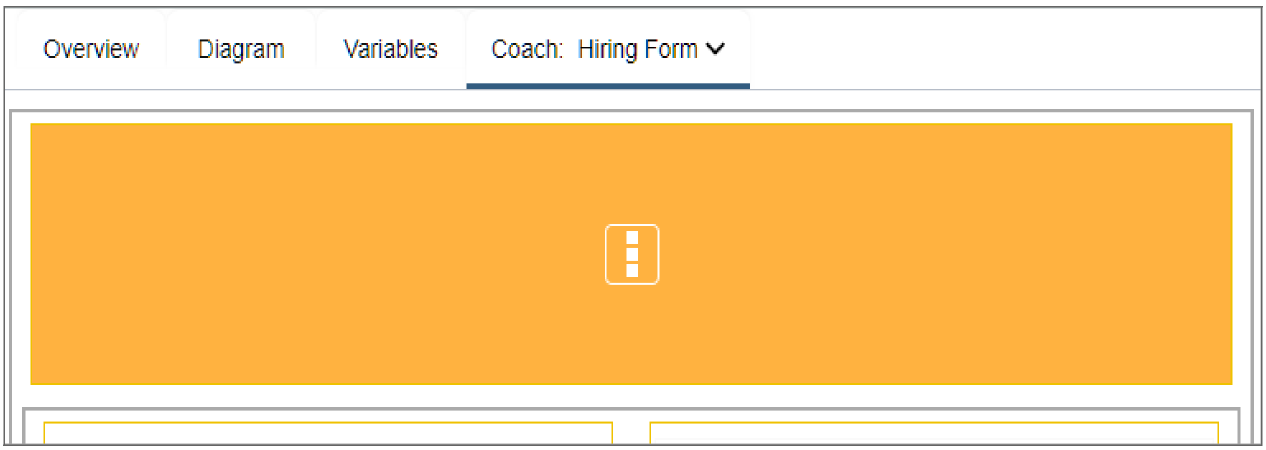

__ h. Save your work.

__ 12. Place a grid at the top of the coach and group the Requisition Number and Requester variables together, setting them apart from the other data on the screen in a section.

__ a. Switch to the grid view by clicking the **Grid toggle**.

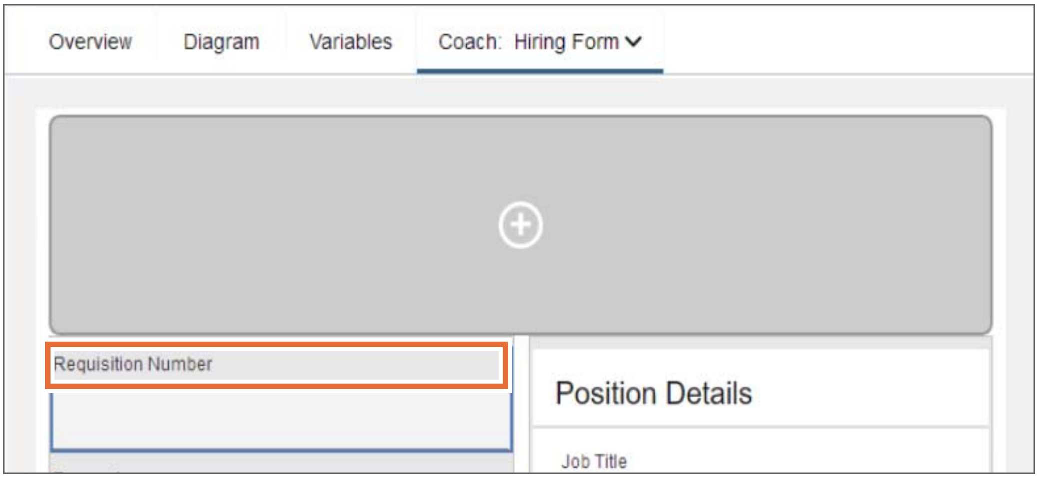
__ b. Hover your mouse and click the **(+) plus sign** on the top of the outer-most grid to add a grid above the two columns.


__ c. Verify your grid is above the grid containing the two columns.

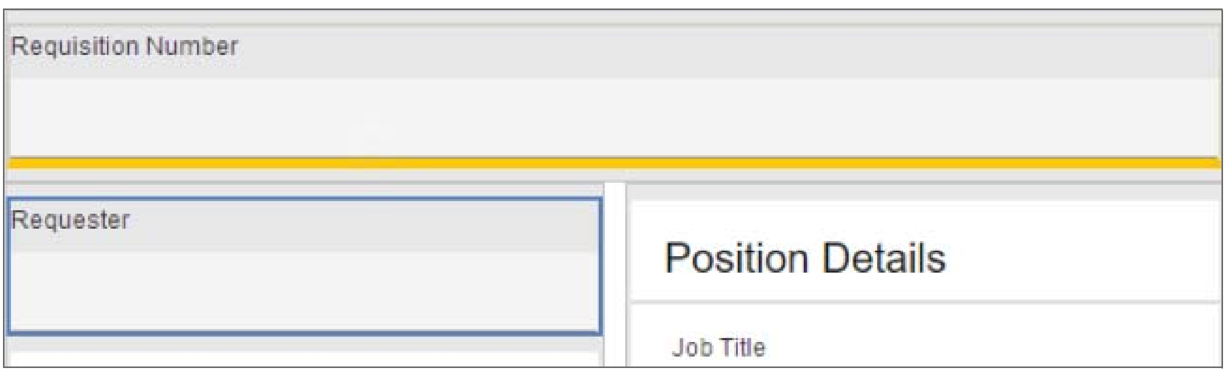



__ d. Switch to the Content view by clicking the **Content toggle**.

__ e. Place the cursor over the **Requisition Number** label, hold down the left mouse button, and drag the **Requisition Number** input control to the new grid at the top. When the grid is highlighted, let go of the mouse button. The mouse turns to the four-arrows icon when the control can be dropped.




__ f. Perform the same action for the **Requester** input control, placing it under the Requisition Number control. Use the yellow highlight as a guide to the placement of the control.

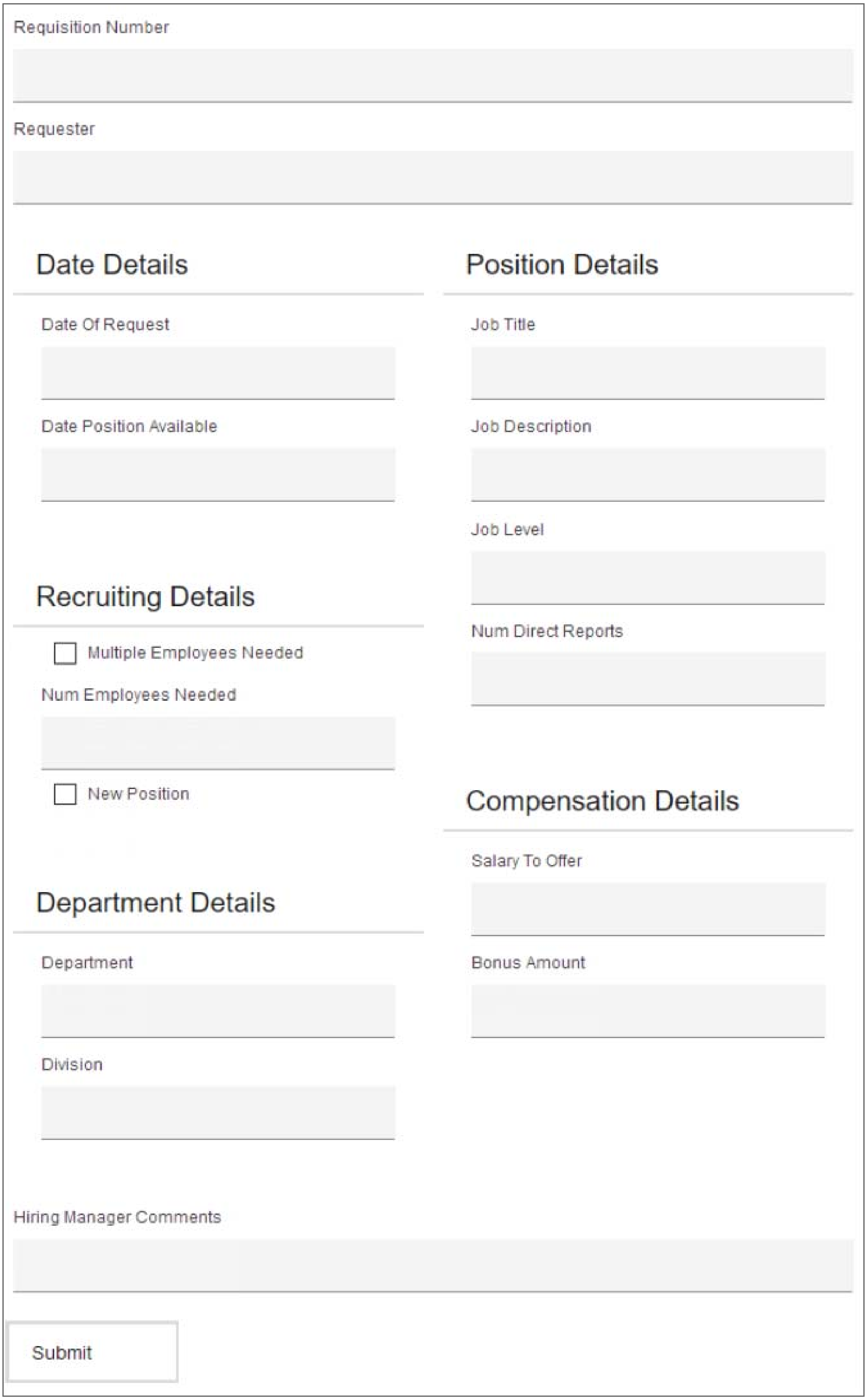



__ g. Switch to the Grid view.

__ h. Verify that your grid was placed above the two columns.


__ i. Switch back to the Content view and save your work..

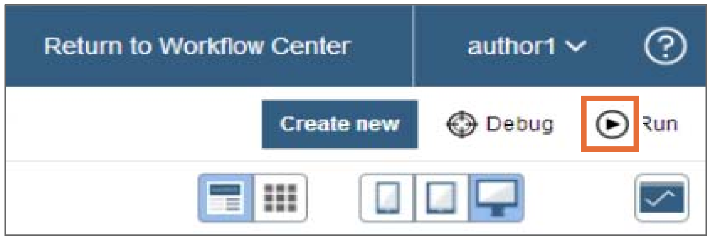
__ j. Verify your coach



__ k. Drag any panels out of place to their correct location. Save your work.

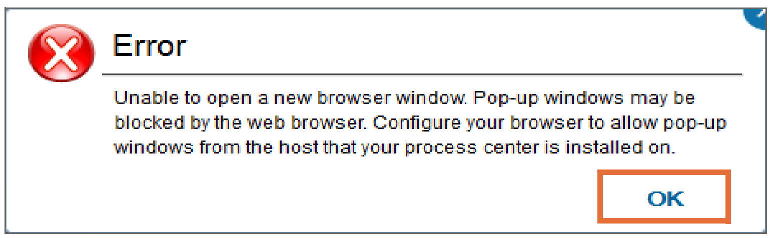
__ 13. Run the coach.

__ a. Now that a working coach has variables that are bound to the fields, run the coach and view it in a browser. Click **Run** at the upper-right corner of the window.

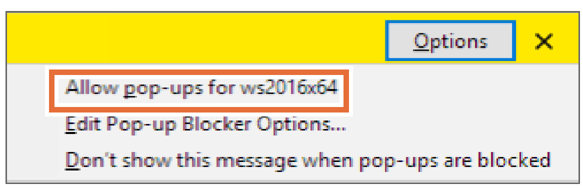



**Troubleshooting**

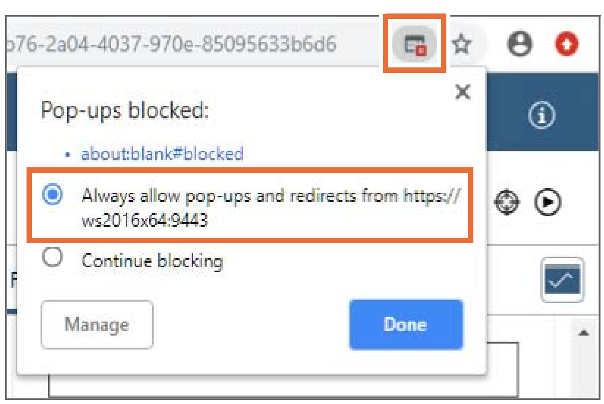
If pop-up windows are disabled in the browser, the browser prevents the site from opening the pop-up window. Click **OK**.




To correct the problem in Firefox, in the yellow bar at the top of the browser, click **Options > Allow pop-ups for ws2016x64**.

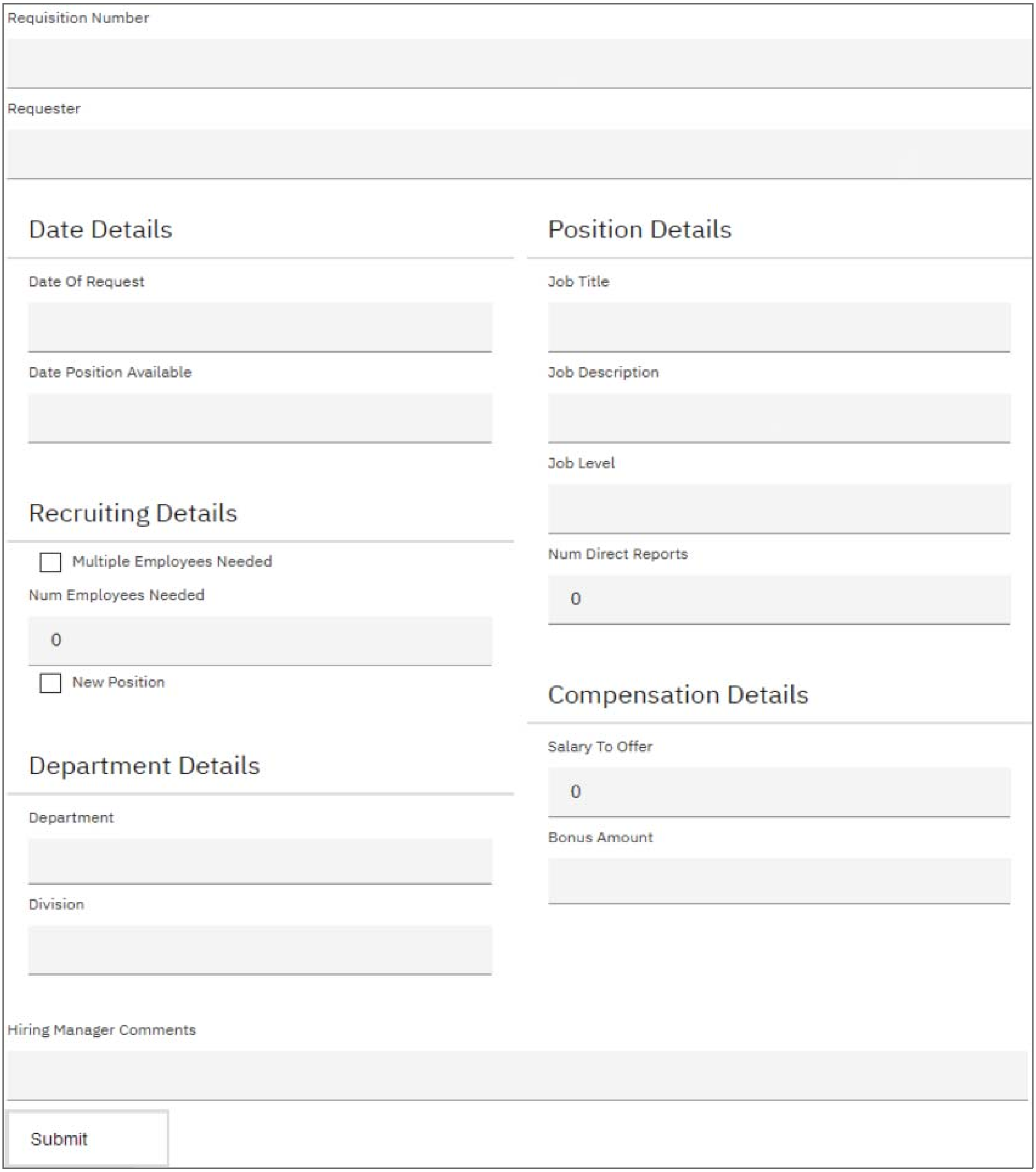



To correct the problem in Chrome, click pop-up blocker icon in the address bar and select **Always allow pop-ups and redirects from https://ws2016x64:9443**.



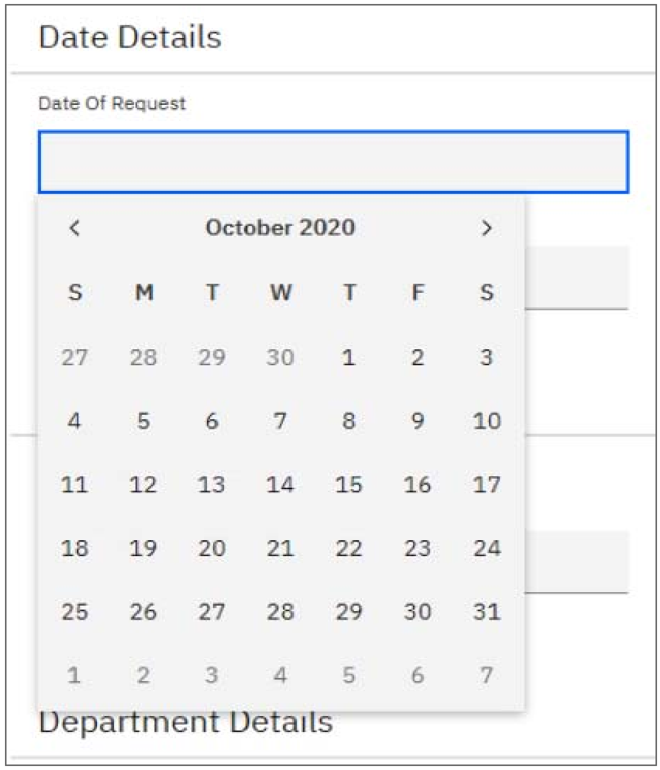

The pop-up window that appears shows the coach designer. Close this window and rerun the coach.

__ b. Another browser window opens and displays the coach.




The coach looks exactly like the coach that you created in the designer window. Most of the design and functional elements of a coach work the same in the designer as they do after you run the coach. The test run gives you an idea of the final page for the business users, but it usually looks the same as the coach in the designer view. Keep in mind, testing for a customer production situation would entail thoroughly testing the solution on the company supported browser. This coach is not meant to be an attractive, fully functioning front end; you add more functional and visual features later on in this playback. The intent of this playback is to start data flow.

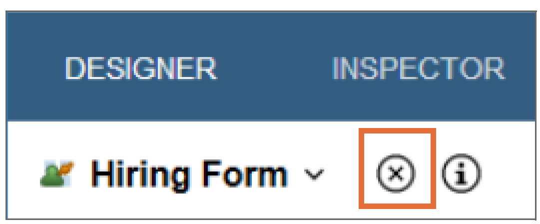
__ c. Click inside the **Date Of Request** input control to select a date, insert some data in other fields, and when you are ready, scroll down to click **Submit**.




__ d. The pop-up window displays **Service completed**

__ e. Close the pop-up browser window that you ran the coach in.

__ f. Close the **Hiring Form** client-side human service window.




## *Part 3: Reuse and debug the client-side human service*

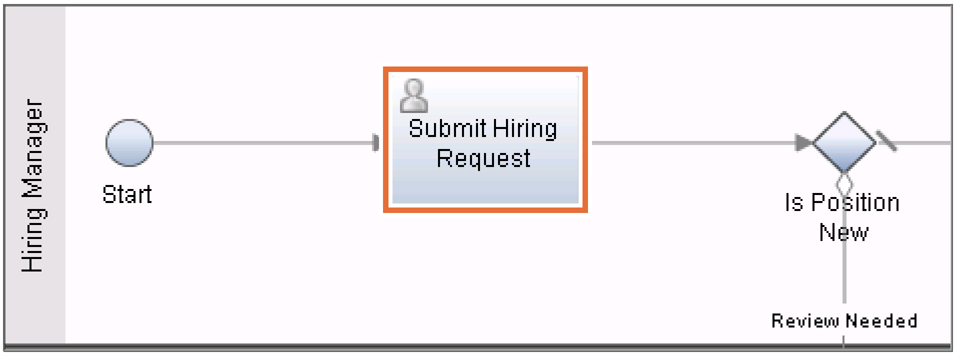
Because the **Approve New Hire Request** activity in the **Hiring Request Process** reviews the same data that is entered in the last human service, you can reuse the Hiring Form human service to add a service to an activity.

In the **Hiring Request Process** modeling effort, many of the activities are placed on the diagram canvas with default human services. It is time to implement the services that were created to the appropriate activities in the process. In this section, you replace the human services in the **Submit Hiring Request** and **Approve New Hire Request** activities in the Hiring Request Process business process with the appropriate human services.

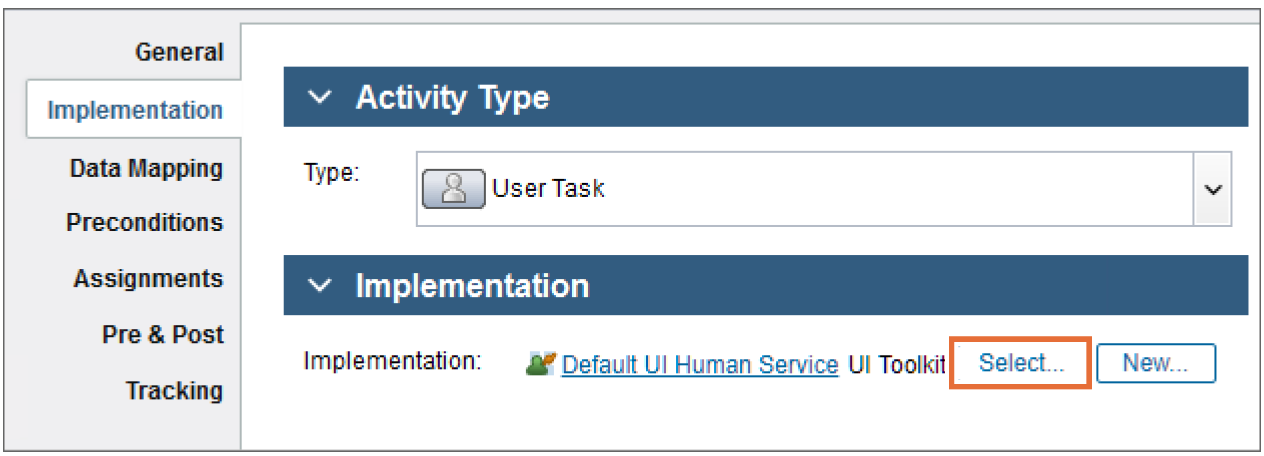
__ 1. Implement the **Submit Hiring Request** activity service.

__ a. In the Process Designer, open the **Hiring Request Process** process.

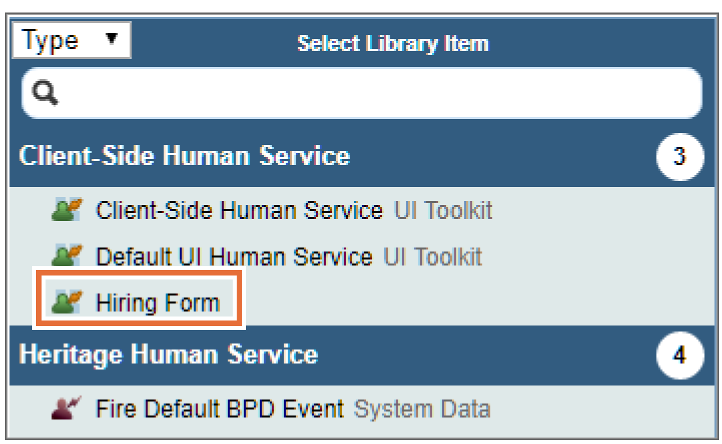
__ b. Click the **Submit Hiring Request** activity.




__ c. Click Select in the **Implementation > Implementation** properties section and select the service that implements the user task.



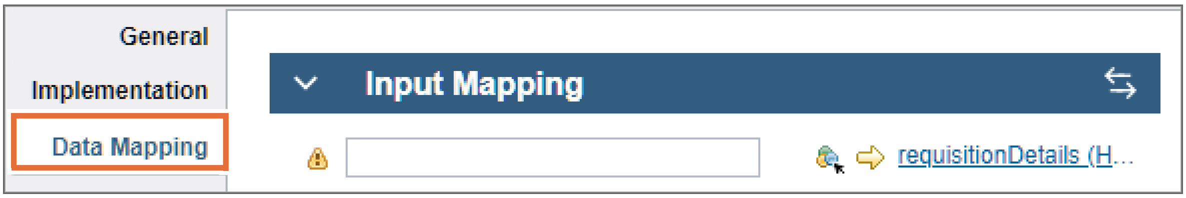

__ d. Select **Hiring Form** from the list.



__ e. The Hiring Form is now the implementation for the activity.


__ 2. Map the Submit Hiring Request input and output.

__ a. Open the **Data Mapping** menu for the **Submit Hiring Request** activity.




You will see a warning until the input mapping is complete.

__ b. Click the **auto-map icon** on both the Input Mapping and Output Mapping sections.


__ c. Save your work.

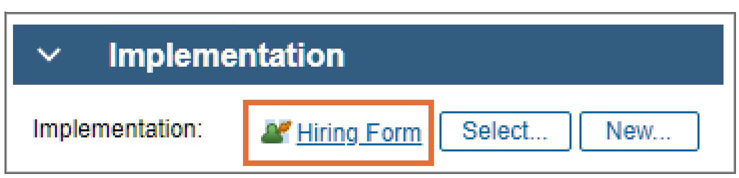
__ 3. Implement the **Approve New Hire Request** activity.

__ a. Click the **Approve New Hire Request** activity in the **Hiring Request Process** process.

__ b. Click **Select** in the **Implementation > Implementation** properties section.

__ c. Select the **Hiring Form** client-side human service from the list.

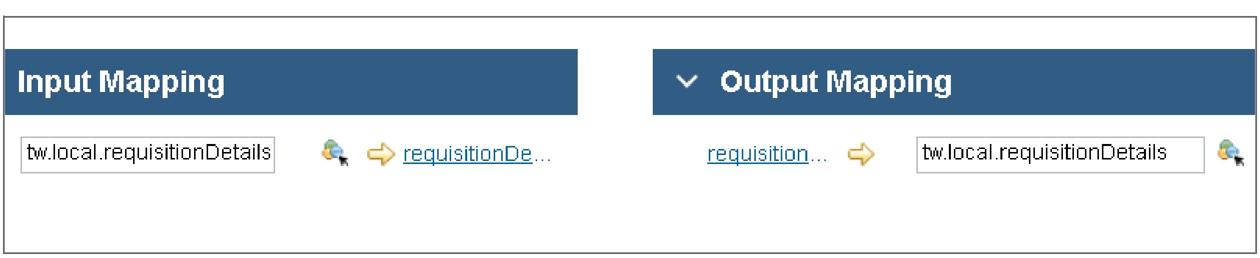
__ d. The **Hiring Form** is now the implementation for the activity.




__ 4. Map the **Approve New Hire Request** inputs and outputs.

__ a. Open the **Data Mapping** menu for the **Approve New Hire Request** activity.

__ b. Map `tw.local.requisitionDetails` to both **Input Mapping** and **Output Mapping**.

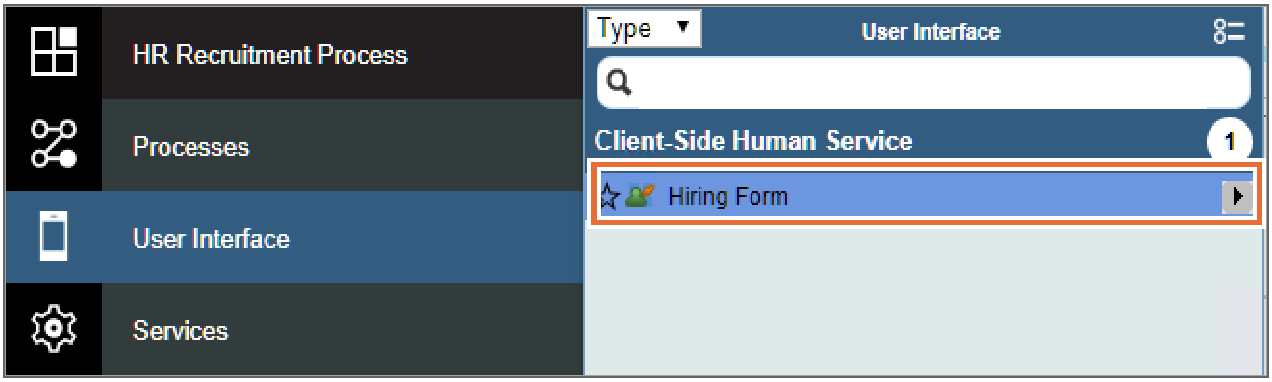


__ c. Save your work.

__ 5. Add a Script activity to the Hiring Form to evaluate whether the request is for a new position or not.

This script activity determines whether parameter **newPosition** is set to true. If it is, it updates the value of the **isNewPosition** variable to 1, otherwise it sets the value to 0.

__ a. In the process library, click **User Interface > Hiring Form**.



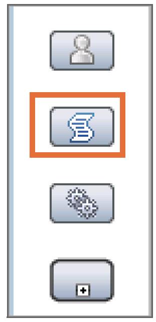

__ b. Click the **Variables** tab.

__ c. Click the (**+**) plus sign next to Output.

__ d. Add the variable: `isNewPosition (String)`

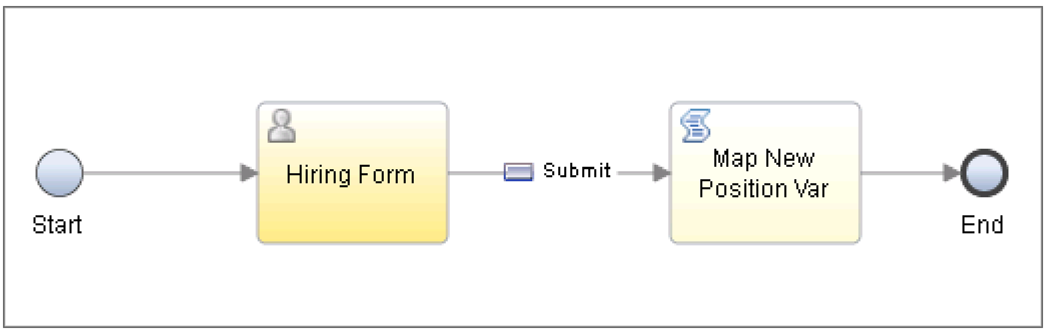

__ e. Click the **Diagram** tab.

__ f. Drag a **Client-Side script** between the **Hiring Form** coach and the **End** event. Place the script directly on the sequence flow line.



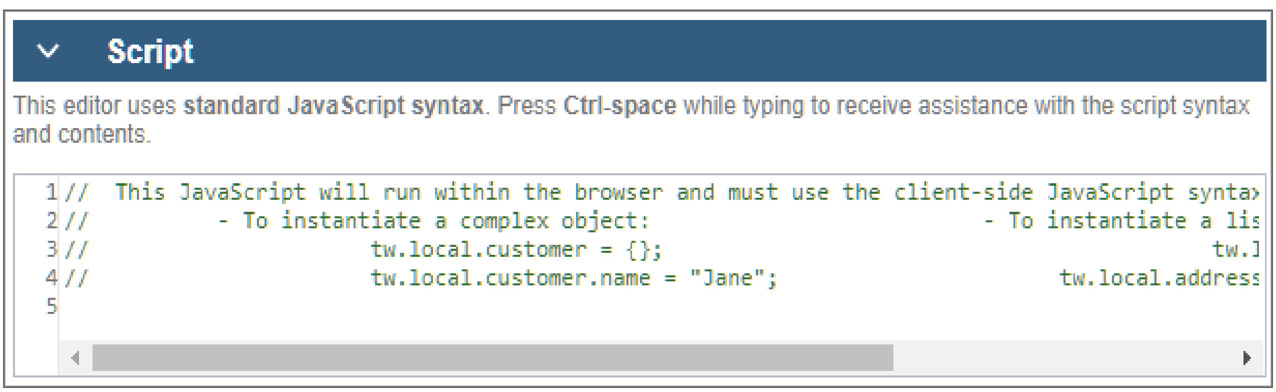
The script is inserted into the sequence flow.

__ g. Name the new client-side-script: `Map New Position Var`




__ h. Click the **Map New Position Var** script activity that you placed on the canvas.

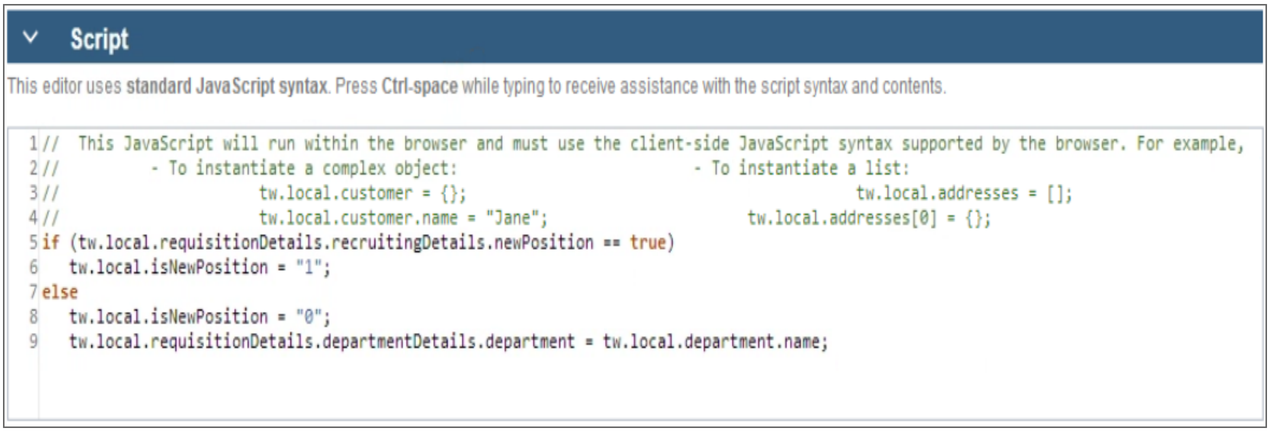
__ i. Open the **Script** section in the properties below.




The client-side script box has some example code commented out. You can erase the code, or you can add the code below the commented code. This code provides examples of how to instantiate different types of variables.

__ j. Copy and paste the following code in the **Script** field. You can copy the code from the file `Script1.txt` at the location: `C:\labfiles\Support files\Ex5`

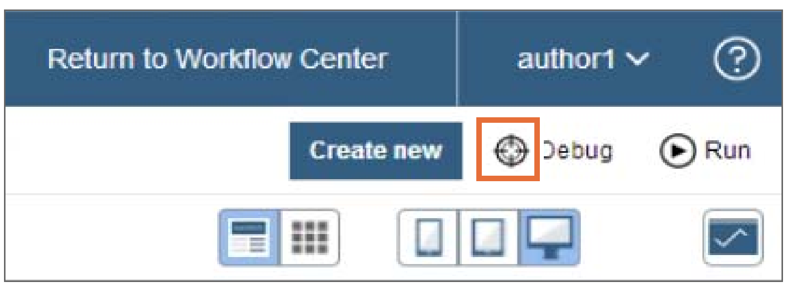
```
if (tw.local.requisitionDetails.recruitingDetails.newPosition == true)
   tw.local.isNewPosition = "1";
else
   tw.local.isNewPosition = "0";
```



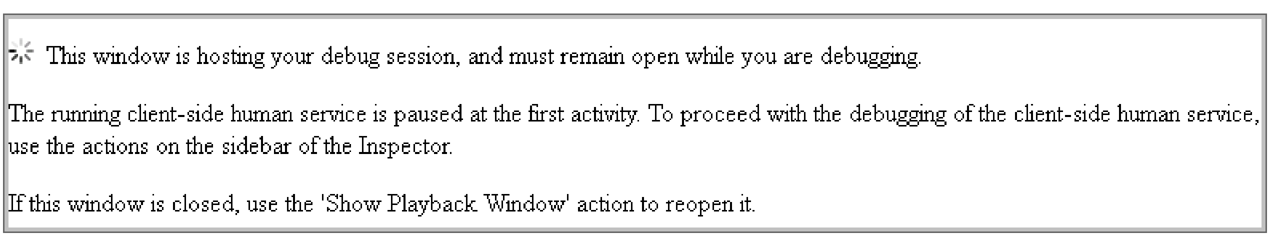

__ k. Save your work.

__ 6. Debug the **Hiring Form** client-side human service and verify that data is flowing.

__ a. Click the **Debug** icon.



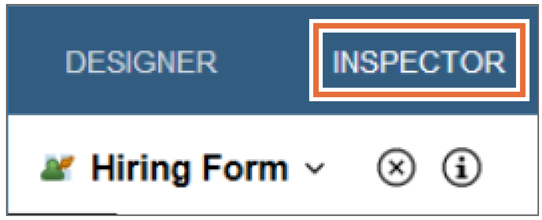

__ b. A new browser window is displayed, but this time it is in debug mode.



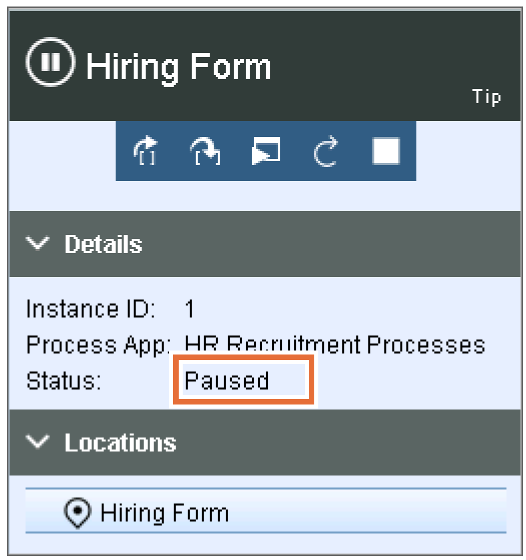

__ c. Minimize this new browser window to view the client-side human service designer window.

**Information**

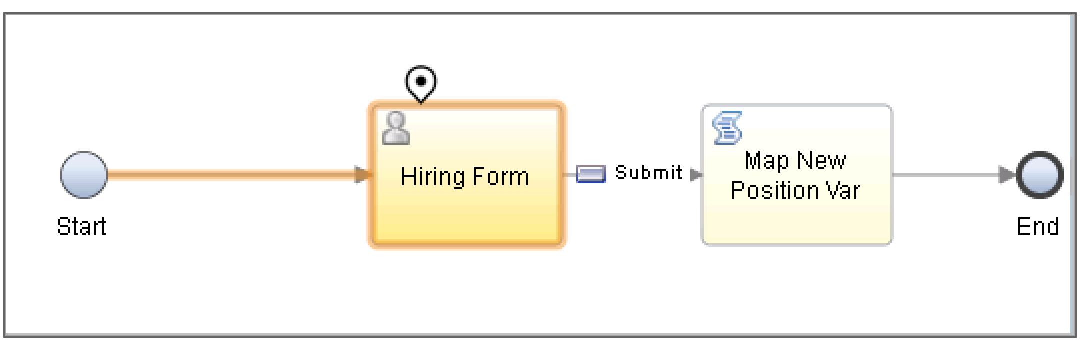
The coach designer switches to the **Inspector** tab.



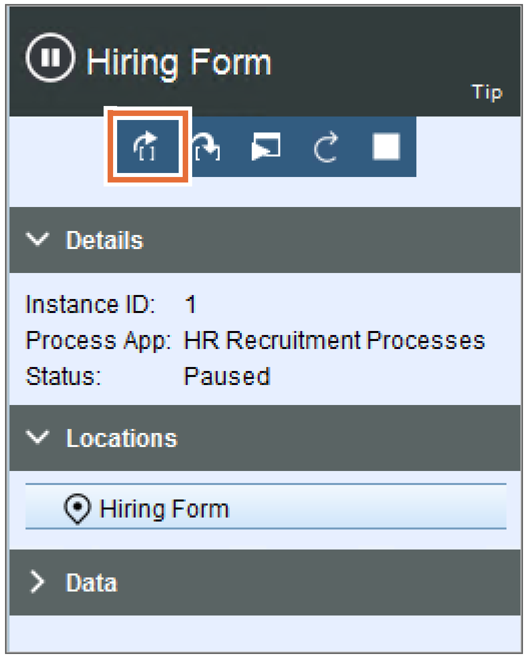

__ d. The inspector section is displayed on the right side. The status shows that the execution is **Paused**.



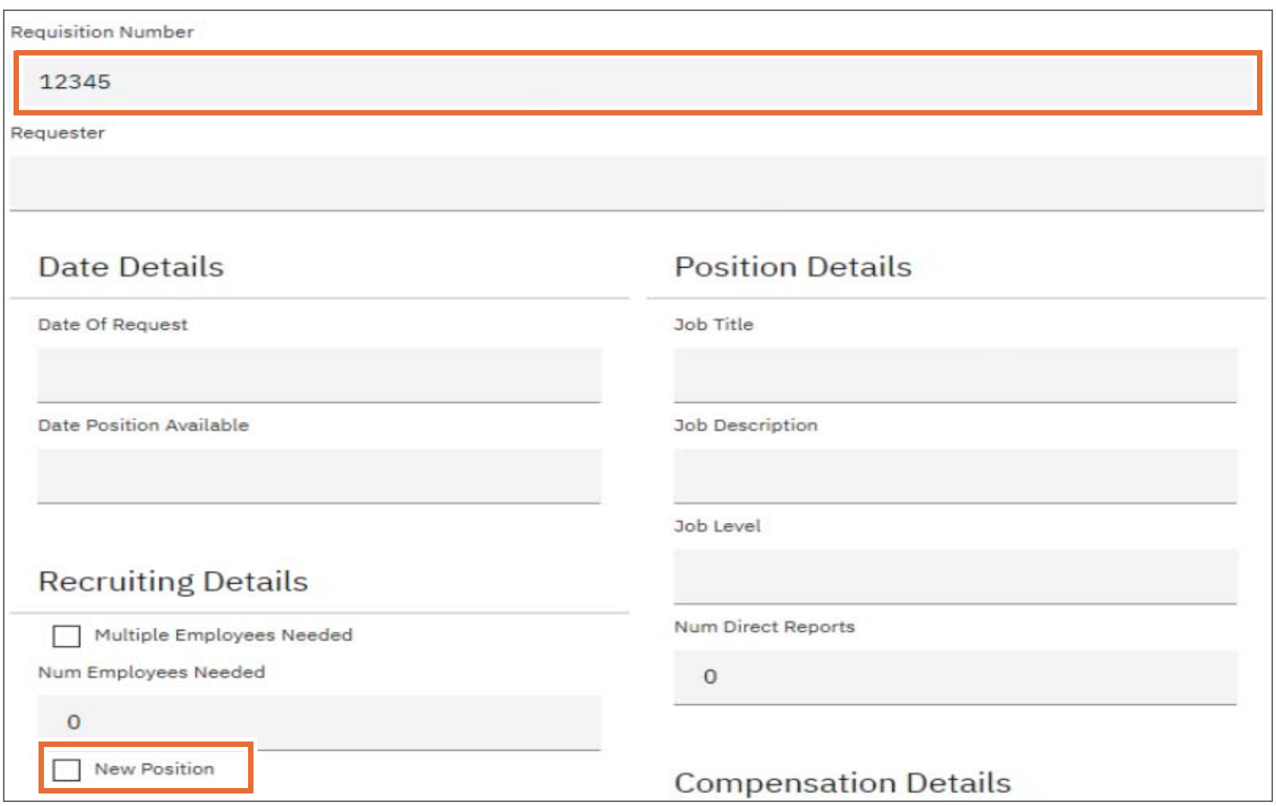

__ e. If not already selected, click the **Diagram** tab. The inspector shows a token that is waiting to execute this step in the process.



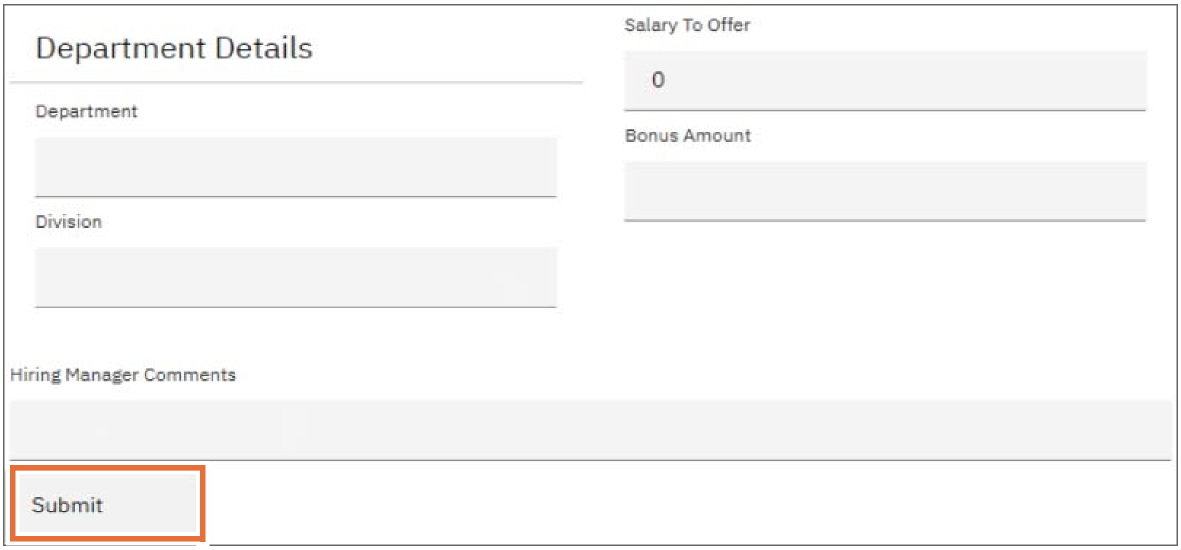
__ f. Click the **Step over** icon for the Hiring Form coach.




__ g. The second browser window renders the coach. Enter `12345` as the Requisition Number. Ensure that the **NewPosition** checkbox is not selected.

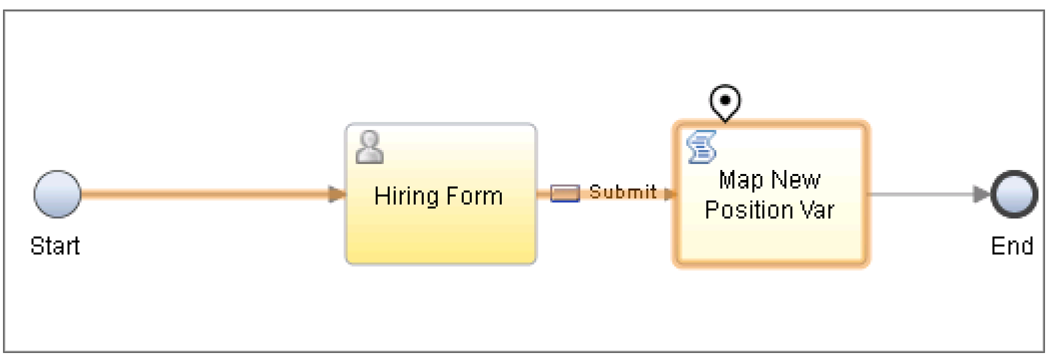


__ h. Click **Submit**.

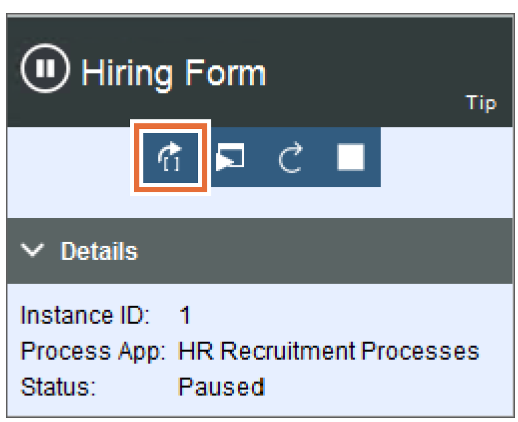



__ i. Minimize the coach browser window and return to the designer browser window.

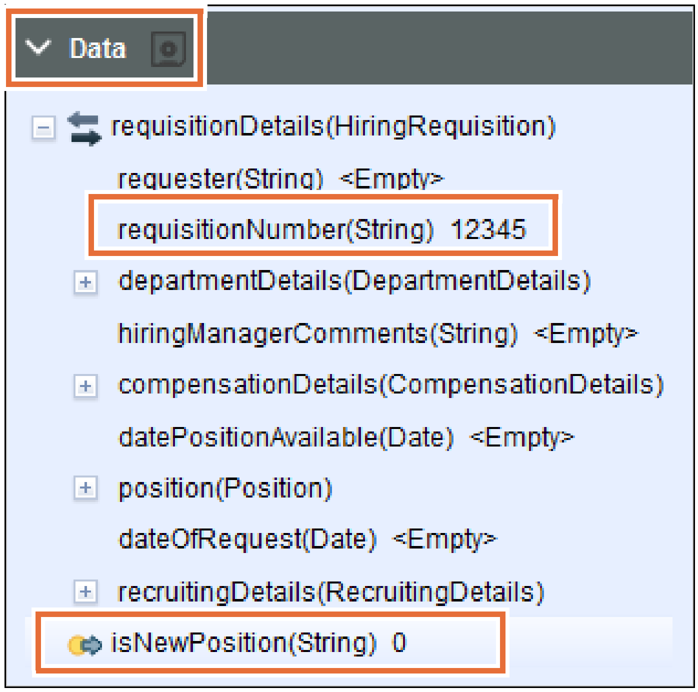
The Hiring Form coach is complete, and the token moves to **Map New Position Var**.




__ j. The status is **Paused**, waiting for the next step in the service. Click **Step over** to run the Map New Position Var server script.



__ k. The status changes to **Finished** and the token disappears because it reached the end event. On the right, expand the **Data** section and click the (**+**) plus sign next to **requisitionDetails** to see the updated value for the `requisitionDetails.requisitionNumber` variable. Verify that the updated value for the `requisitionDetails.requisitionNumber` variable is 12345. The `tw.local.isNewPosition` variable is set to the String "0".




__ l. Run the service again, and this time select the **New Position** checkbox on the coach. The `isNewPosition` value should be set to 1.


__ m. Return to the Debug Hiring Form and verify that the service is completed. Close this browser window.

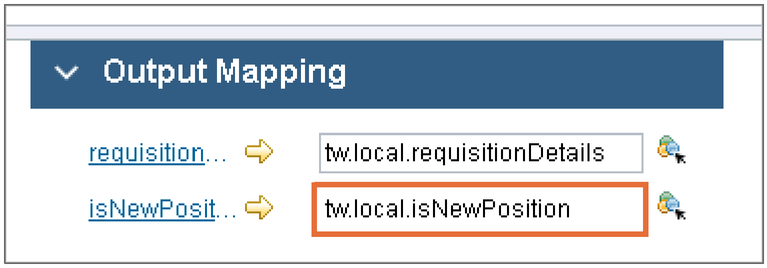
__ n. The Status is **Finished**, and you are currently on the **INSPECTOR** tab. Return to the **DESIGNER** tab. The activity and flow highlights disappear in the Designer mode.

__ 7. Map the new variable in the process.

__ a. Open the **Hiring Request Process** process.

__ b. Click the **Submit Hiring Request** activity.

__ c. In the **Data Mapping** properties section, map the `isNewPosition` output variable to `tw.local.isNewPosition`.




__ d. Click the **Approve New Hire Request** activity.

__ e. In the **Data Mapping** properties section, map the `isNewPosition` output variable to `tw.local.isNewPosition`.

__ f. Save your work.

You have completed this exercise.

Playback 1: Business data and services is complete. In this exercise, by using the core requirements, you created all of the necessary assets to support a coach in the Hiring Request Process. You modeled the coach using grids and common controls. You used complex business objects (variable types) to organize your data, and pass data into and out of a linked process. You built a human service and defined guided user interactions with a coach. You also implemented a service for an activity, and mapped variables between a linked service and an activity.

## End of exercise

# Exercise 6. Playback 1: User interface design and implementation

## Estimated time

01:30

## Overview

In this exercise, you group controls into tabs on a coach and change the appearance of the coach by applying a theme.

## Objectives

After completing this exercise, you should be able to:

- Create tabs on a coach
- Change the appearance of a coach by applying a custom theme
- Change the coach layout for a mobile format
- Configure controls to respond to different screen sizes
- Debug the coach by using a responsive sensor

## Introduction

In Playback 1, the enhancements that the business wants for process application user interfaces are implemented. Often, the initial development efforts are marred with requests to begin with full fidelity user interfaces. It is not uncommon to have this request because it is driven from a desire to impress executive level stakeholders with prototypical user interfaces that can be manipulated. It is better to reserve this type of development for this playback stage. You ensure that the data model is in place, the functions of the user interface are approved, and all the remaining process application interactions and integrations are complete. Now the enhancements can be done without fear that rework would be needed later because of a change in requirements to the items in the prior playbacks. Changes can happen, but the likelihood is that those functional requirement changes are handled in Optimization and not within the development cycle because consensus is reached to move to this stage of development.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

# Exercise instructions

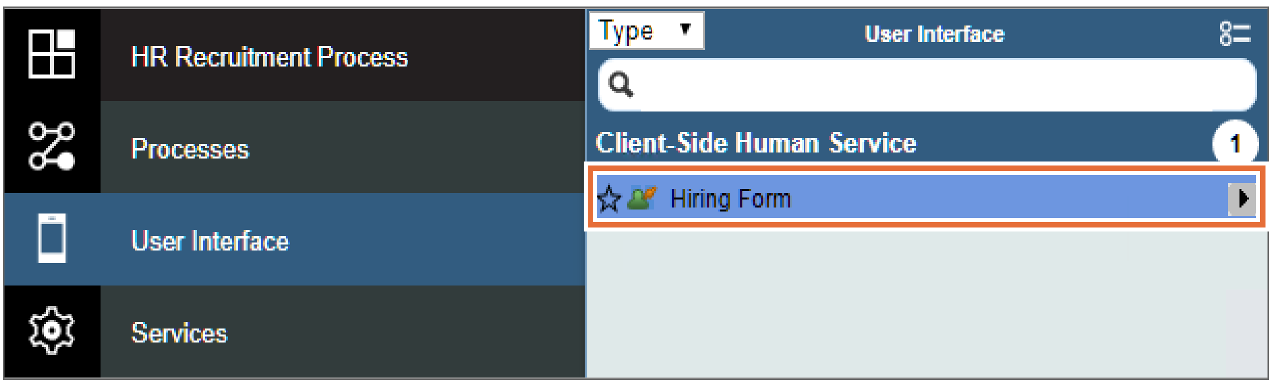
## *Part 1: Group controls into tabs on a coach*

In one of the previous playbacks, you provided a functional look at the coaches in the Hiring Request Process. Enhance the coaches with more features and functions. Start with creating a polished look to the coach by grouping similar data into tabs.

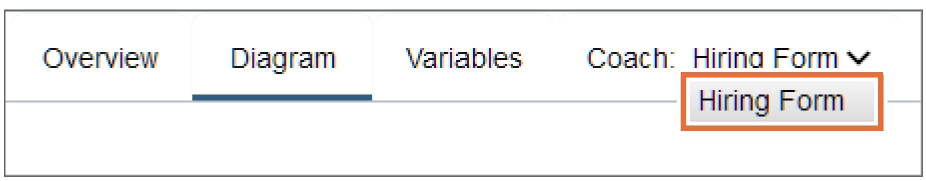
__ 1. Create a panel and populate it with two parameters from the Requisition Details business object.

__ a. Open the **HR Recruitment Process** process application.

__ b. In the Process Designer library, open the **Hiring Form**.

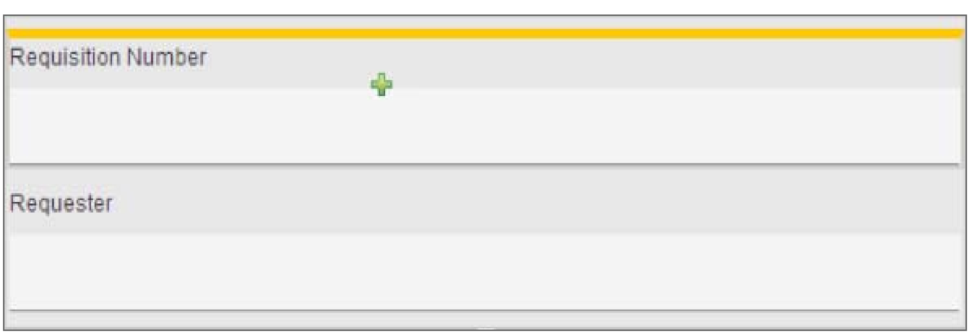



__ c. Click the down arrow next to **Coach: Hiring Form** and select **Hiring Form**.

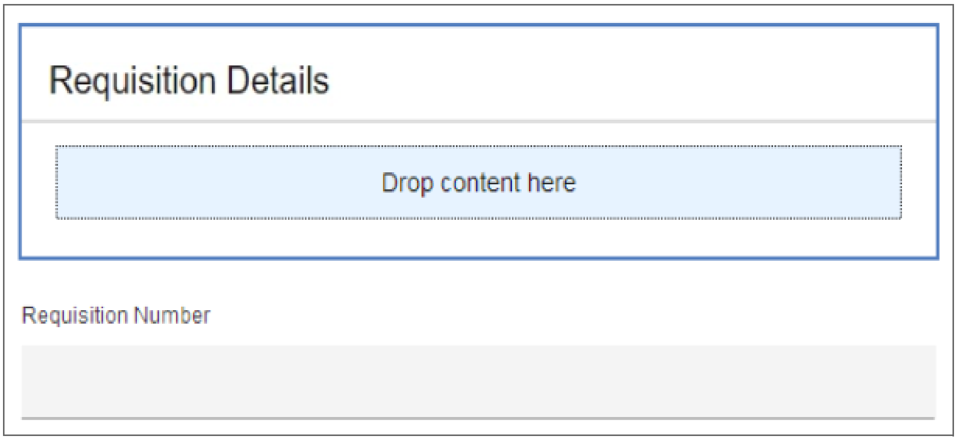



__ d. Select **Layout** in the **Building blocks** panel and select **Panel**

__ e. While holding down the left mouse button, drag the **Panel** above the **Requisition Number** input control. Let go of the mouse button when the top border displays a gold highlight.



__ f. With the name highlighted, label the panel `Requisition Details` and press Enter




__ g. Place the cursor over the **Requisition Number** label, hold down the left mouse button, and drag the **Requisition Number** input control to the section of the new panel labeled **Drop content here**. Let go of the mouse button when the panel is highlighted.

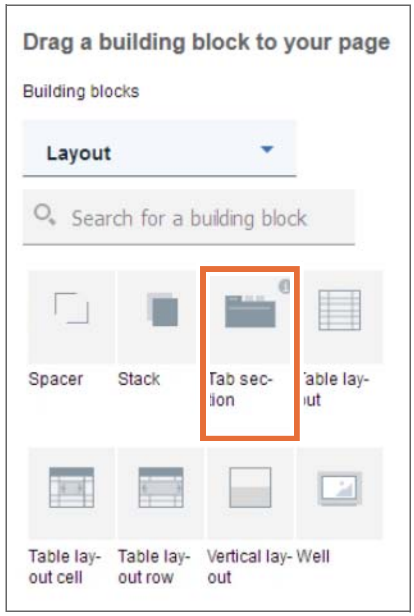
__ h. The result is the Requisition Number is in the new Panel.


__ i. Perform the same steps to place the **Requester** input control under the Requisition Number of the same panel. Use the yellow highlight as a guide to the placement of the control.

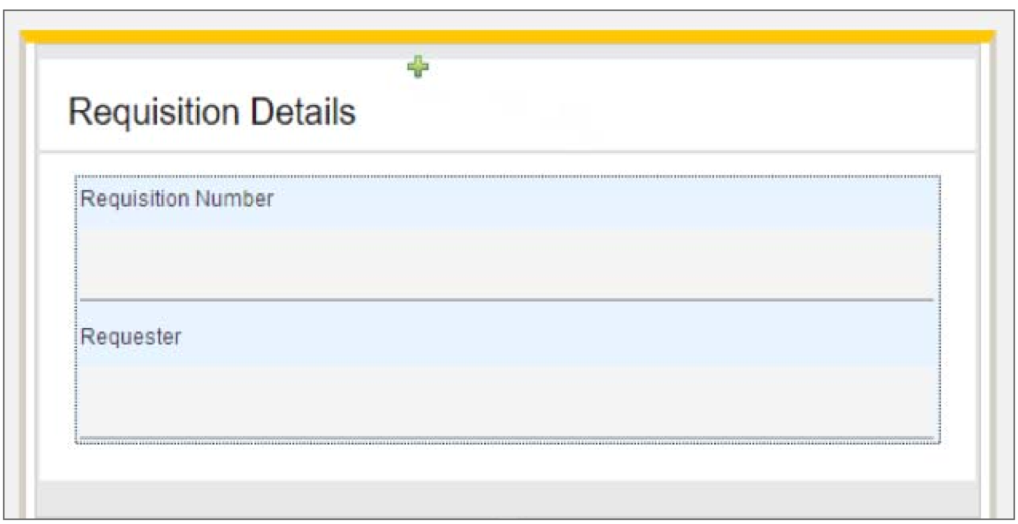
__ j. Save your work.

__ 2. Create a tab section to hold the grouped sections on the coach.

__ a. While holding down the left mouse button, drag a **Tab section** above the **Requisition Details** panel.

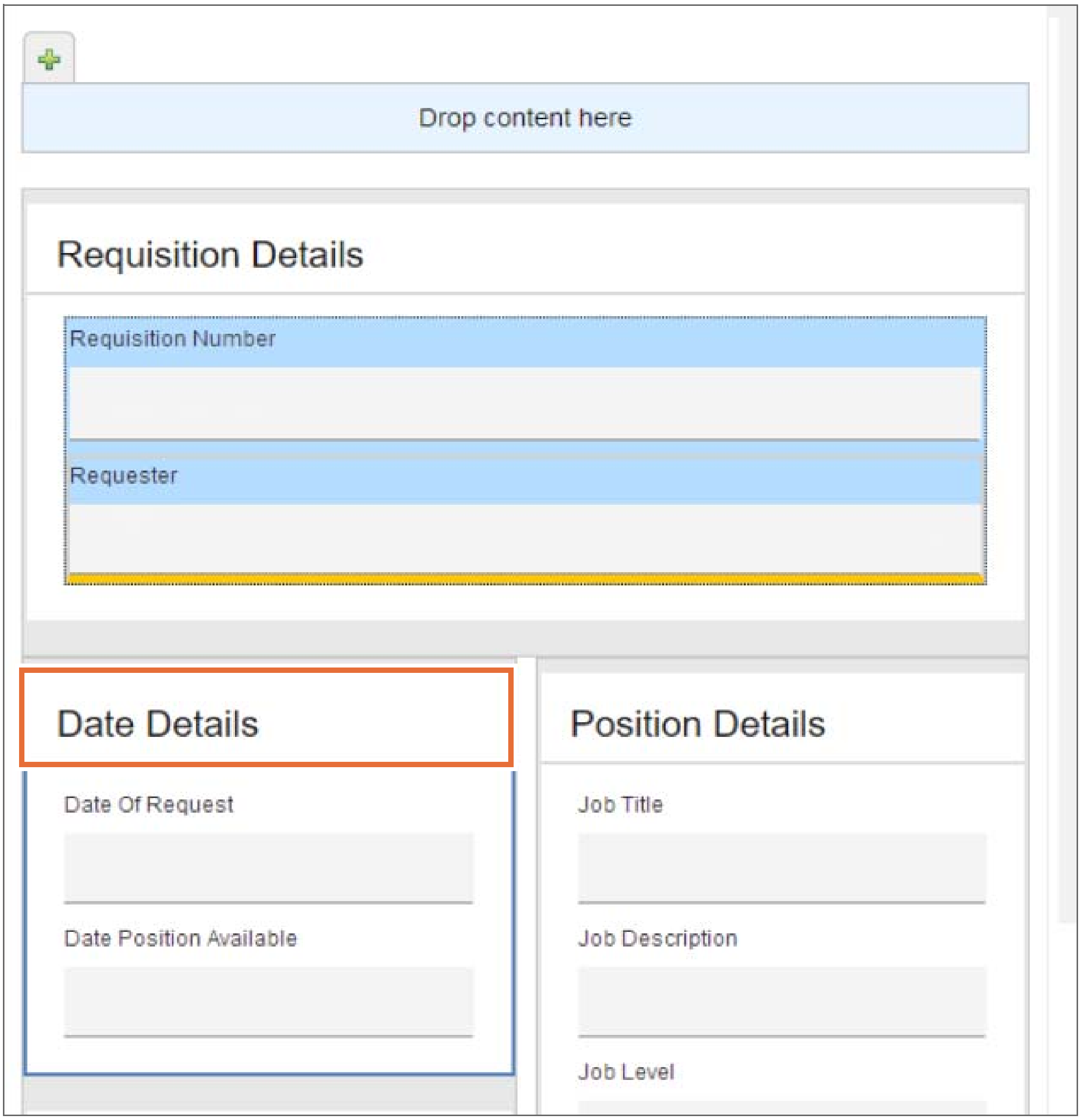



__ b. Let go of the mouse button when the top border displays a gold highlight.

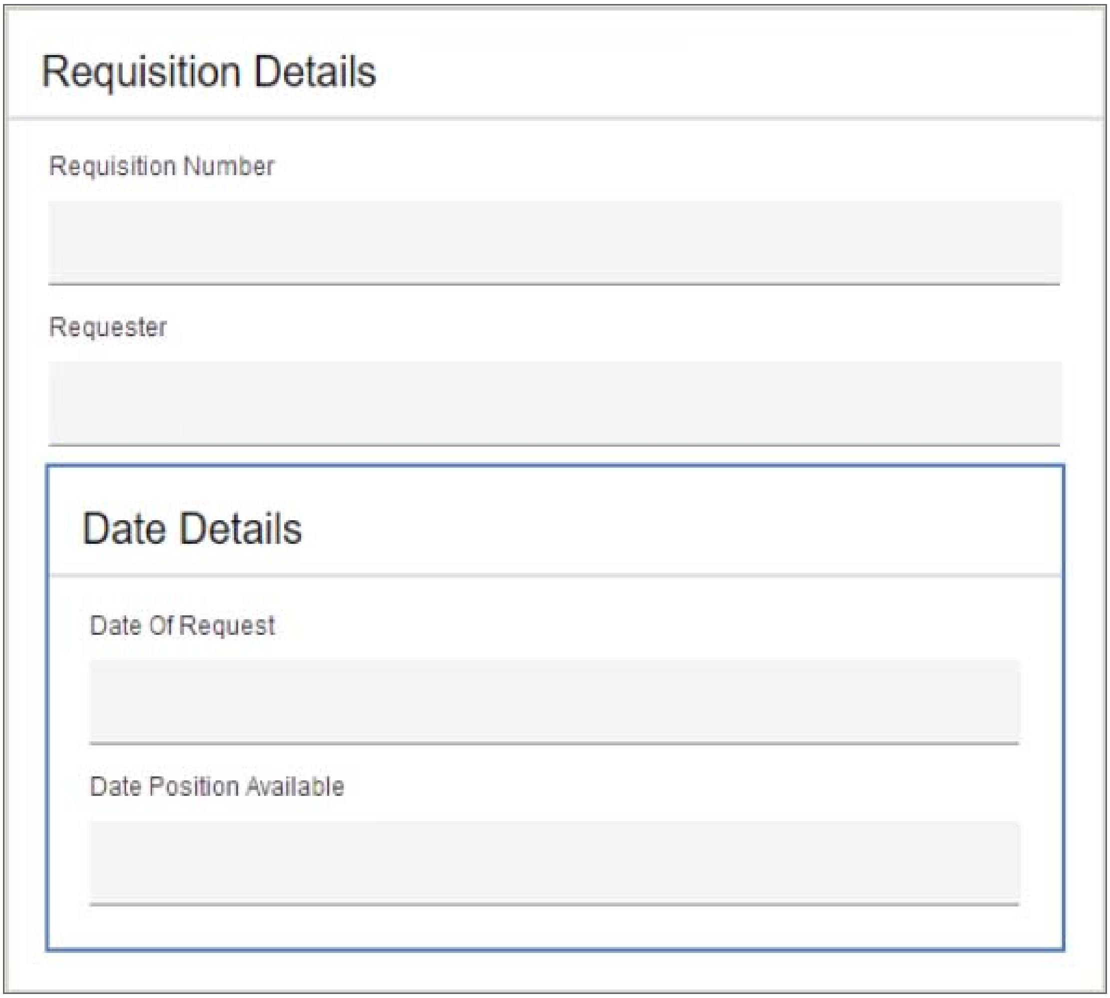


__ 3. Build the Requisition Details tab.

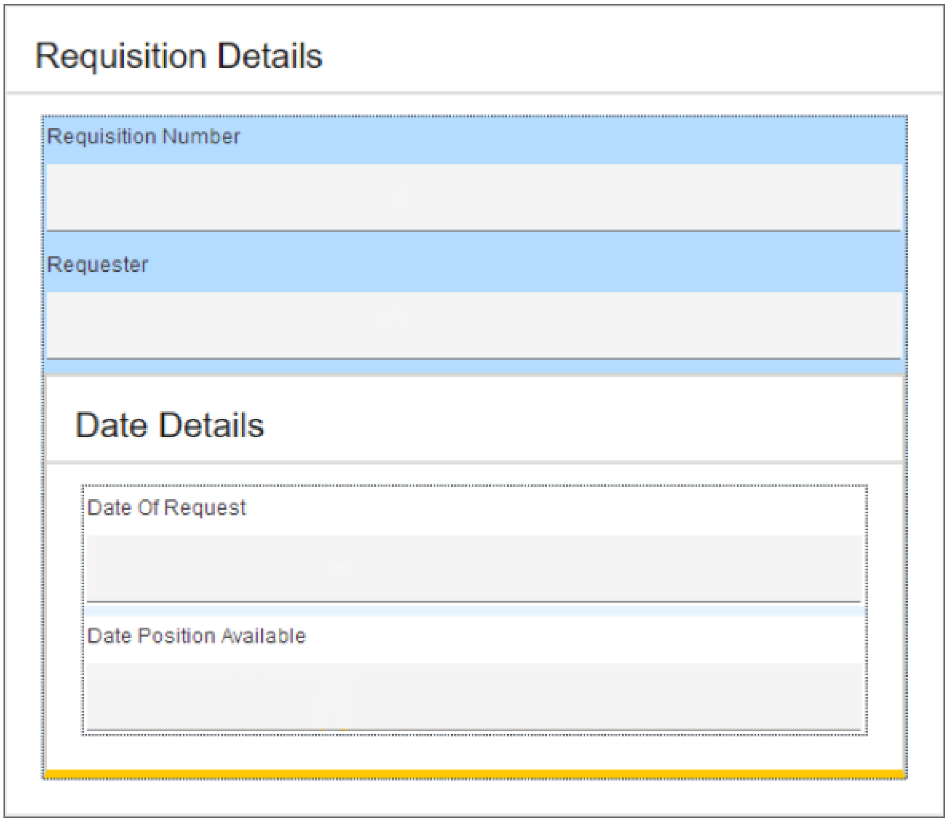
__ a. Select the **Date Details** label for the panel that contains the **Date of Request** and the **Date Position Available** input controls and drag it into the editable area of the **Requisition Details** panel under **Requester**.



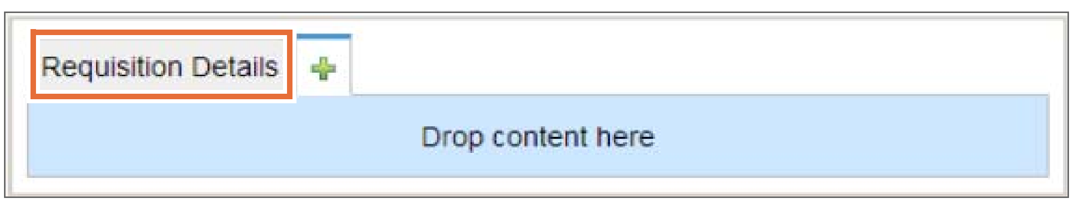
__ b. The **Requisition Details** panel should look like this screen capture after you add the **Date Details** panel.



__ c. Drag the **Hiring Manager Comments** control underneath the **Date Details** panel in the **Requisition Details** panel.

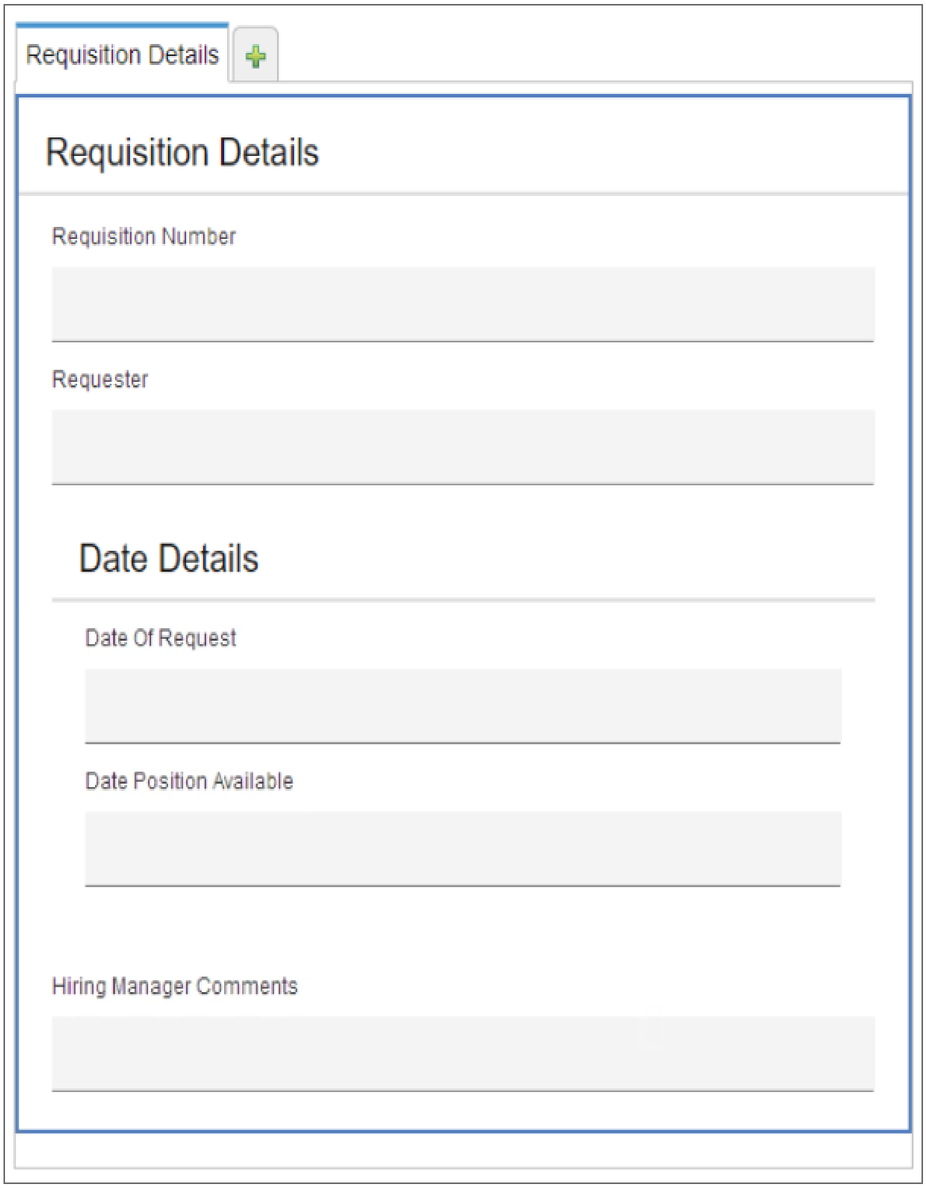



__ d. Drag the **Requisition Details** panel into the tabs area that is labeled **Drop content here**. The label on the tab changes to match the label that is applied to the panel, and a second tab is added.



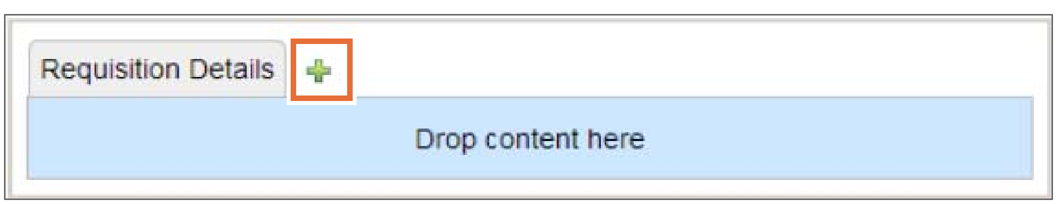

__ e. Click the tab to view the **Requisition Details**.

__ f. Your Requisition Details tab should look like the screen capture.



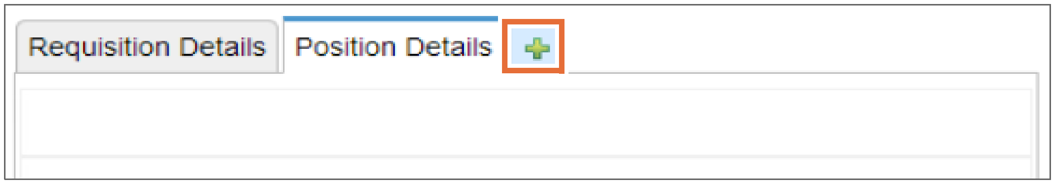

__ 4. Build the remaining four tabs.

__ a. Click the (**+**) plus sign on the tab next to the **Requisition Details** tab.

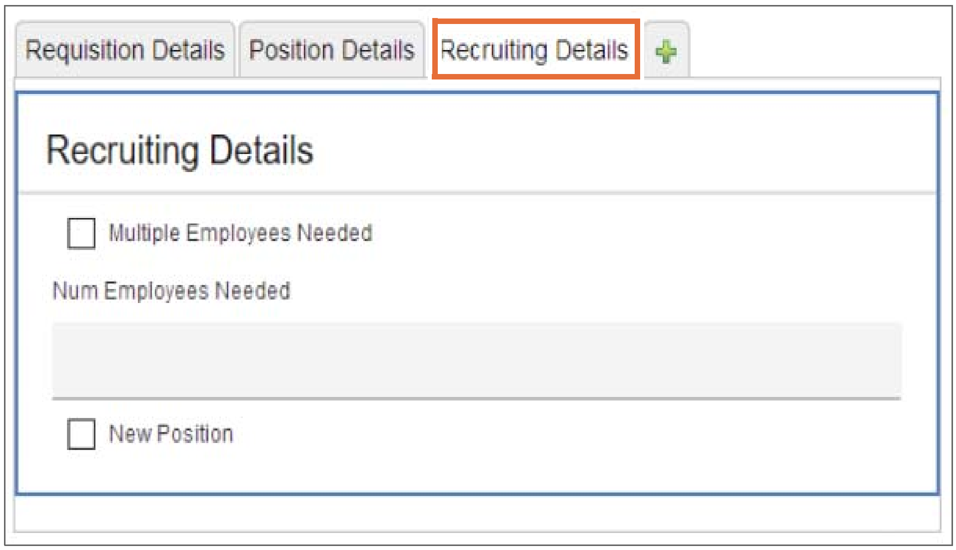



__ b. Drag the **Position Details** section onto the section titled “Drop content here”. This one is easier because all the controls are already in a single section.

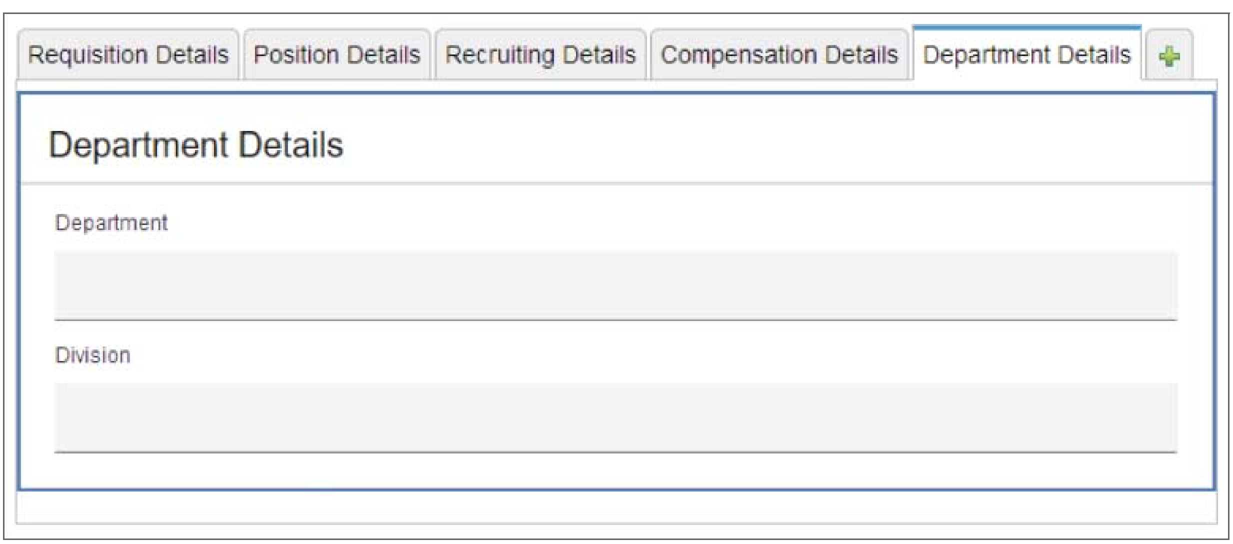
__ c. Create a tab for the **Recruiting Details**. This time, drag the **Recruiting Details** panel over the plus sign to the right of the **Position Details** tab.




__ d. Release the mouse button, and the section becomes the **Recruiting Details** tab next to the **Position Details** tab.



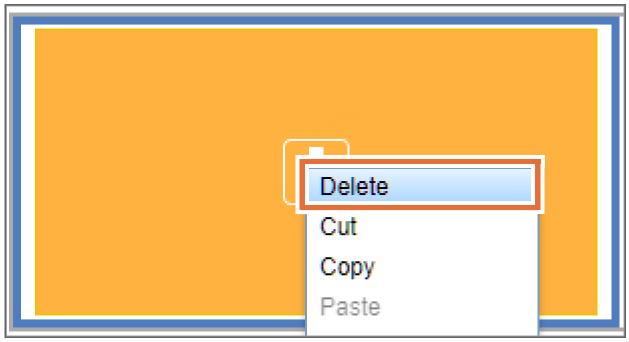

__ e. Repeat the last two steps for the **Compensation Details** and **Department Details**. Drag each of the respective sections onto their tabs.




__ 5. Delete the empty grids.

__ a. Return to the **Grid** view.

__ b. The empty grids are displayed. Right-click each of the four empty grids and click **Delete**.

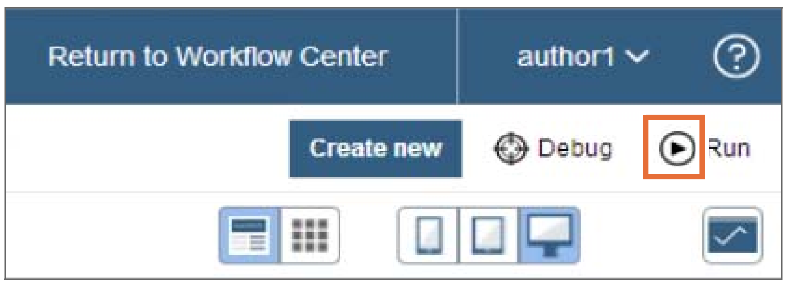



__ c. Return to **Content** view.

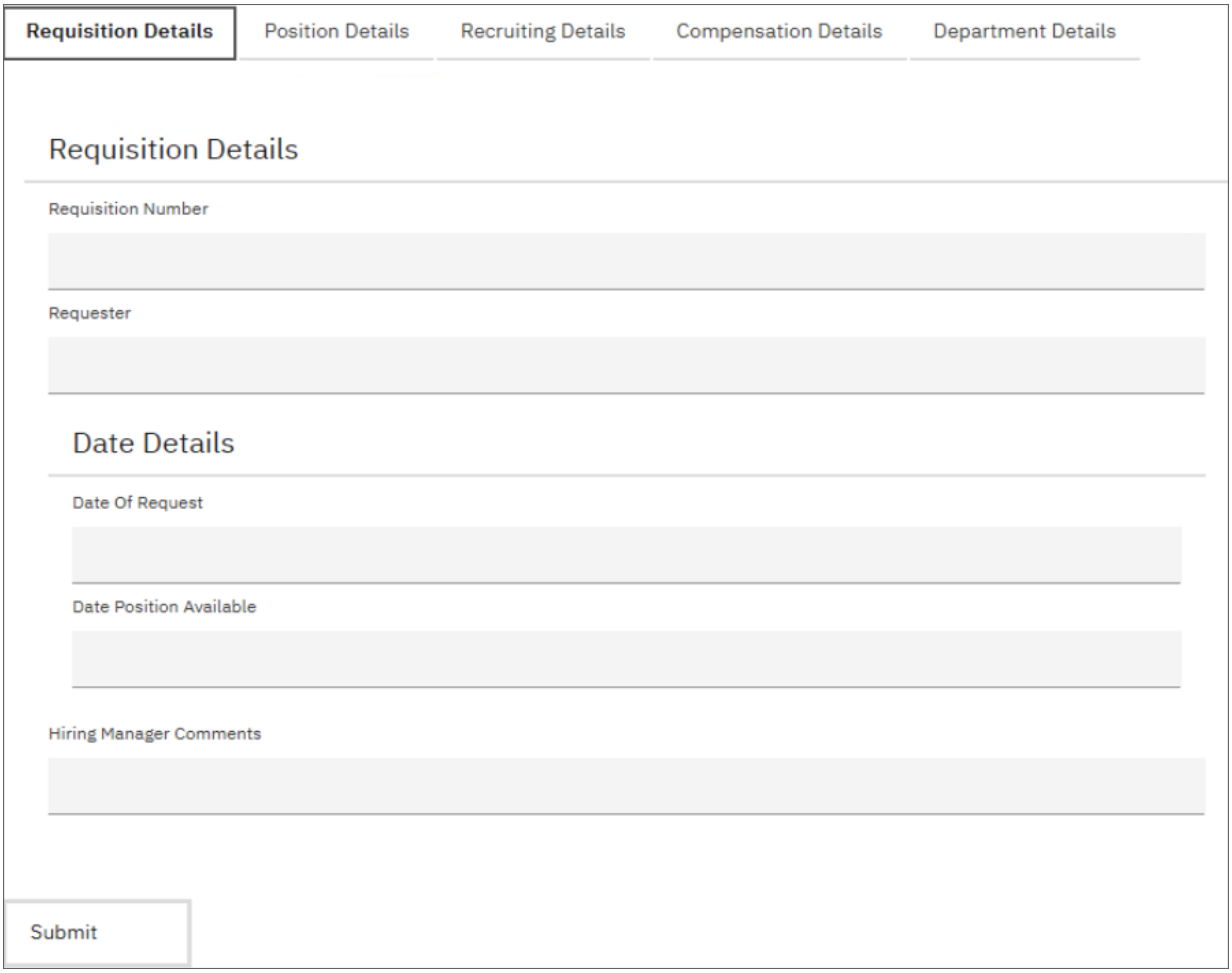
__ d. Save your work.

__ 6. View the coach.

__ a. Click **Run** at the top of the coach.



__ b. The coach now has multiple tabs. Click the tabs to see what is contained on each tab.




__ c. Close the browser that displayed the form.

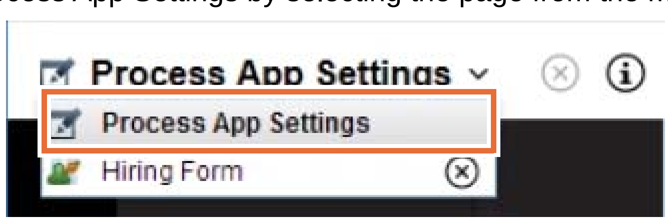
## *Part 2: Change the appearance of the coach by applying a custom theme*

Creating a theme consists of assigning values to a set of theme variables. By default, a new process application uses the Carbon Theme because it contains all the variables that the controls in the UI Toolkit use. To create a custom theme and use any of these settings in your process application, you use the Classic Theme as your starting point. You can then extend your theme by adding custom variables and modifying the variables.

__ 1. View the solution using the Spark Theme.

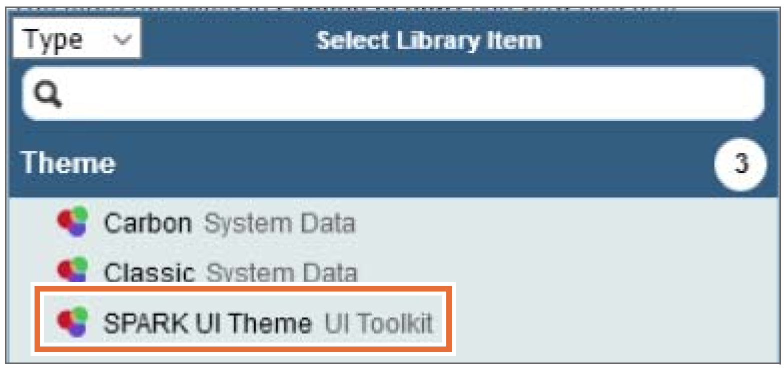
Three themes are provided out-of-the-box. You can use themes to quickly change the look and feel of a user interface.

__ a. View the Process App Settings by selecting the page from the menu.

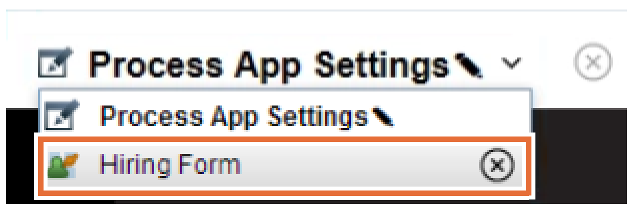


__ b. Under the Coach Designer Settings section, click Select next to the Theme.

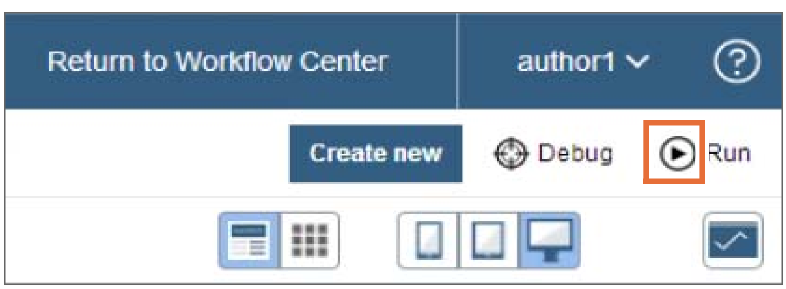

__ c. You are presented with the list of out-of-the-box themes. Select the SPARK UI Theme.



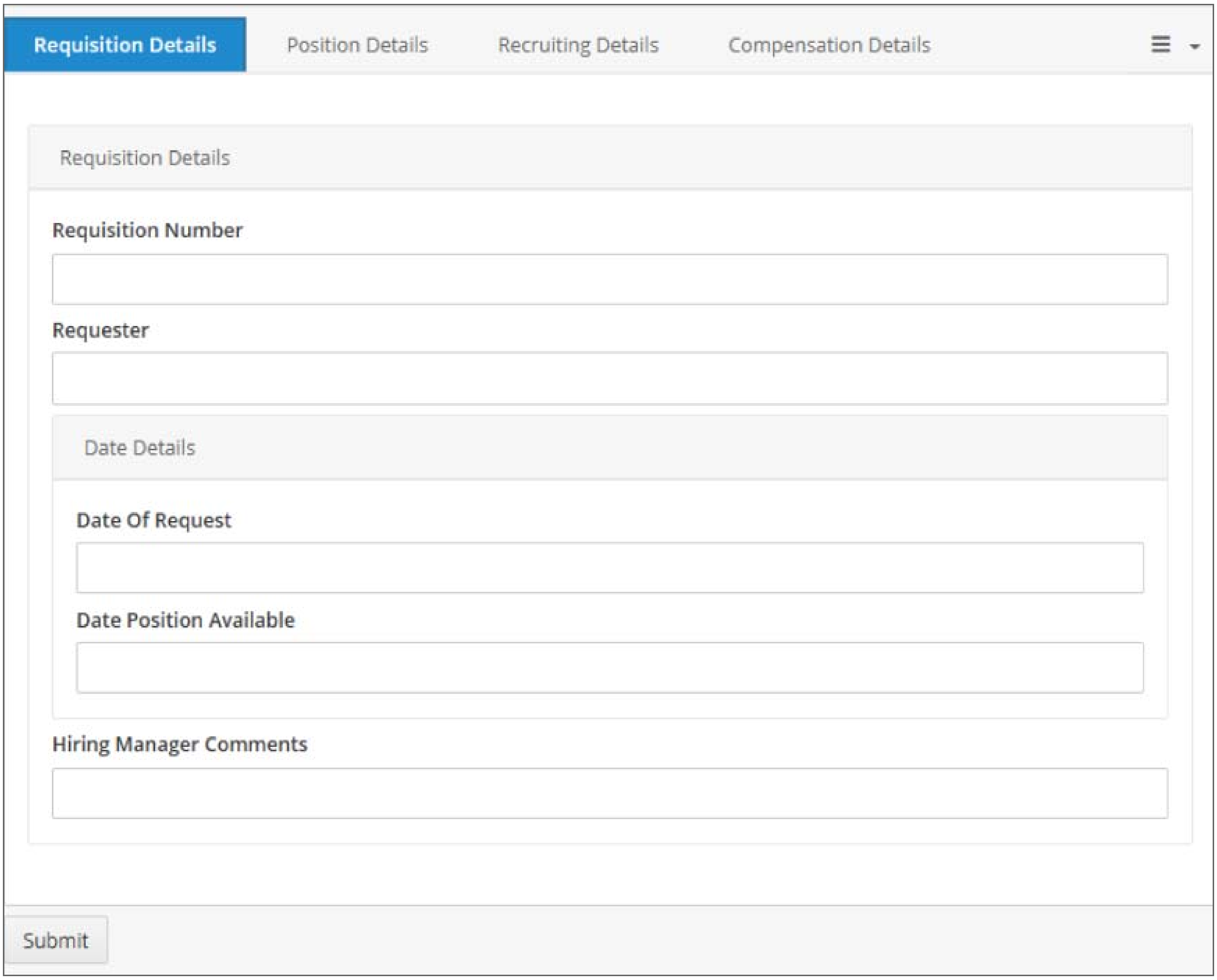

__ d. Return to the Hiring Form.




__ e. Click the **Run** icon at the upper-right side to display the coach with the new theme. Initially when you view the coach, the server must generate the CSS and render the new coach on the screen, so a delay might occur before you see the effect.



__ f. The coach is displayed using the Spark UI Theme.



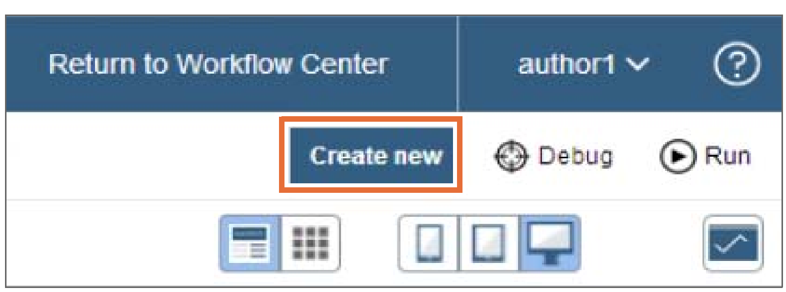

__ g. Close the browser that displayed the coach.

__ h. Optional: Repeat the process to display the Classic theme.

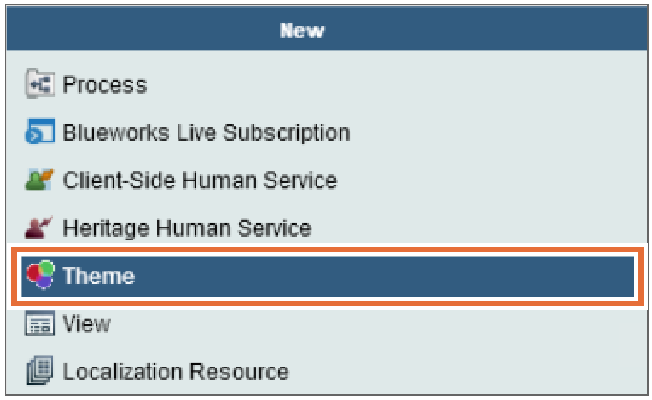
When you create a custom theme in the next step, you base it off of this theme.

__ 2. Create a custom theme for the HR Recruitment Processes process application.

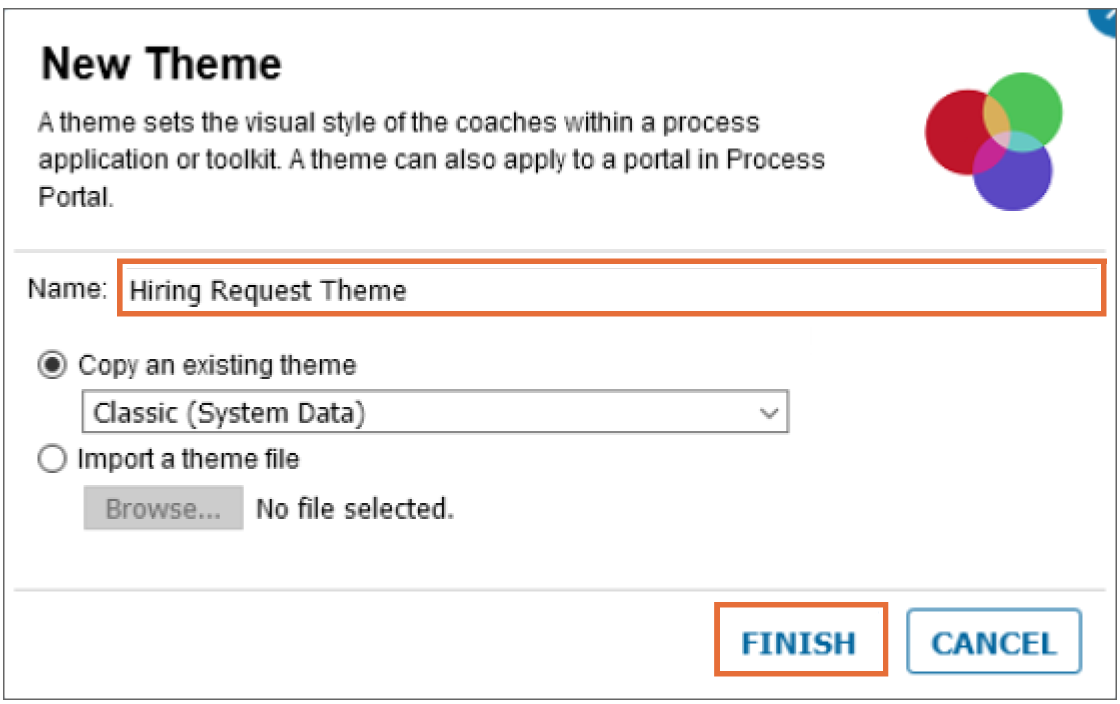
__ a. Click **Create New** on the upper-right corner of the Process Designer.



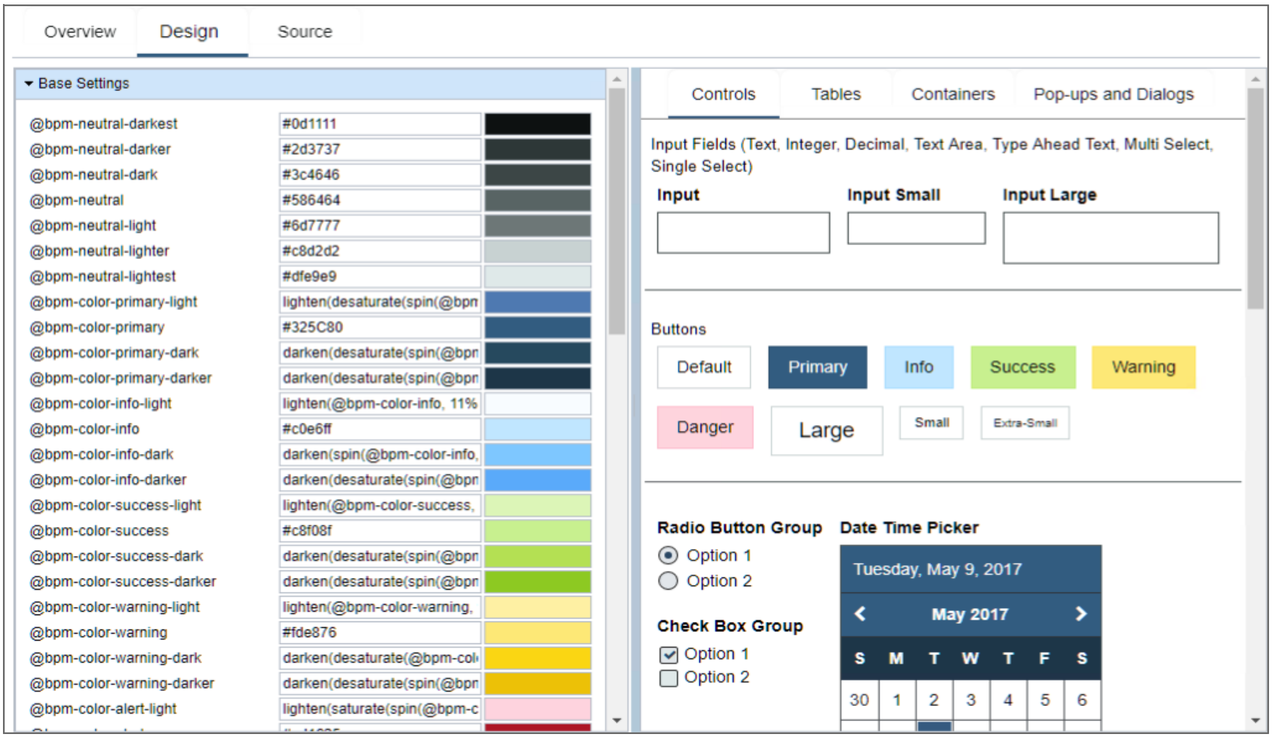
__ b. Click **Theme**.




__ c. Enter the name: `Hiring Request Theme`
Change the option to **Copy an existing theme: Classic (System Data)** and click **FINISH**.



__ d. The editor opens the new theme and defaults to the Design tab.




__ 3. Change the primary color of the theme

Management wants to change the color scheme to match the corporate color scheme. Change the settings of the default Classic Theme and apply the new theme to the process.

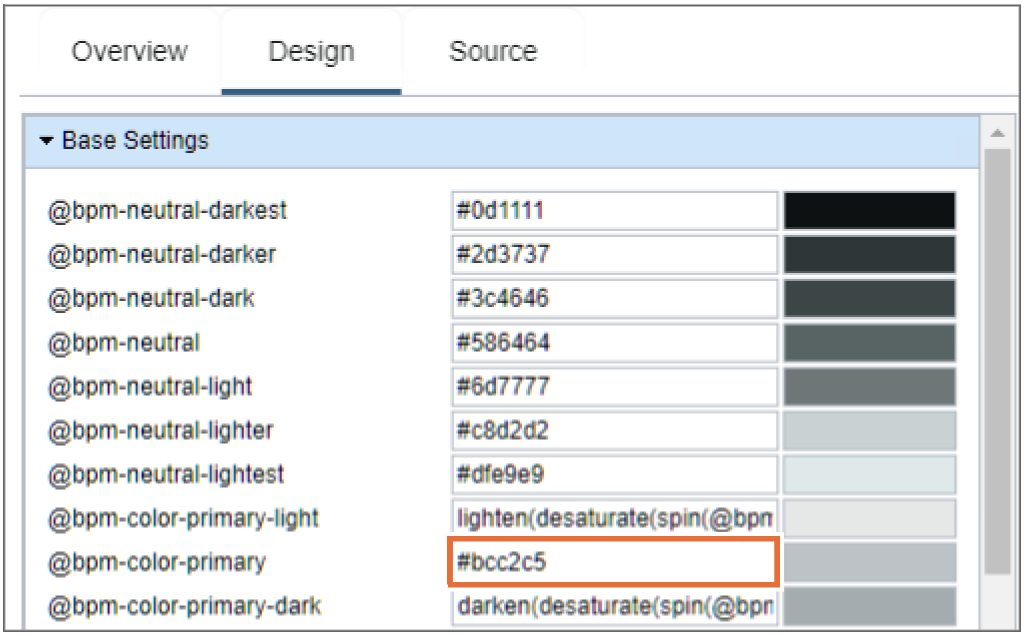
---

**Hint**

You can use the browser **Find in This Page** feature to locate text on the page.

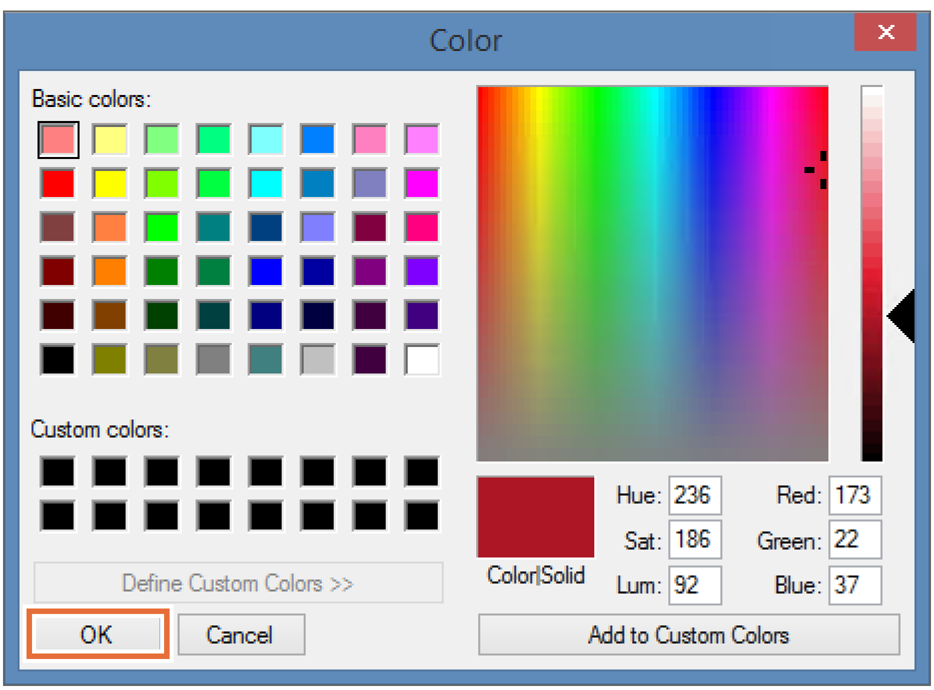
---

__ a. In the Design page of the theme editor, change the variable `@bpm-color-primary` in the **Base Settings** section from `#325C80` to: #bcc2c5

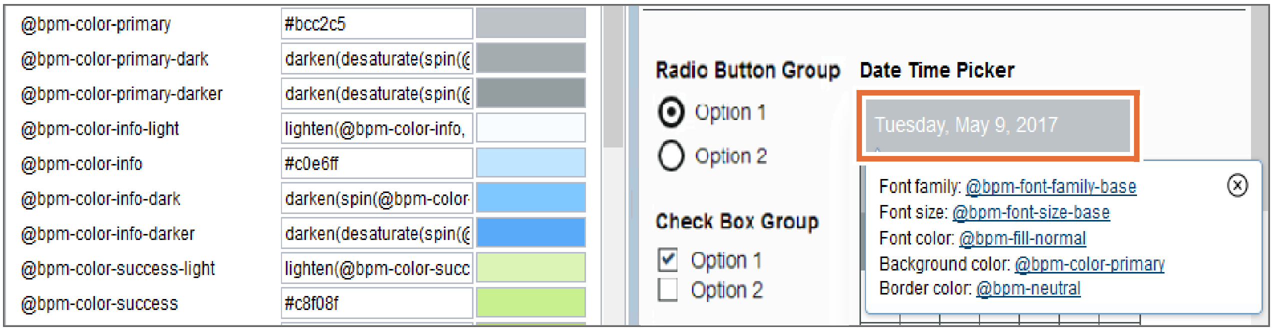



**Note**

Many variables also have a swatch that you can click, then choose a value directly from a picker, and click **OK**.

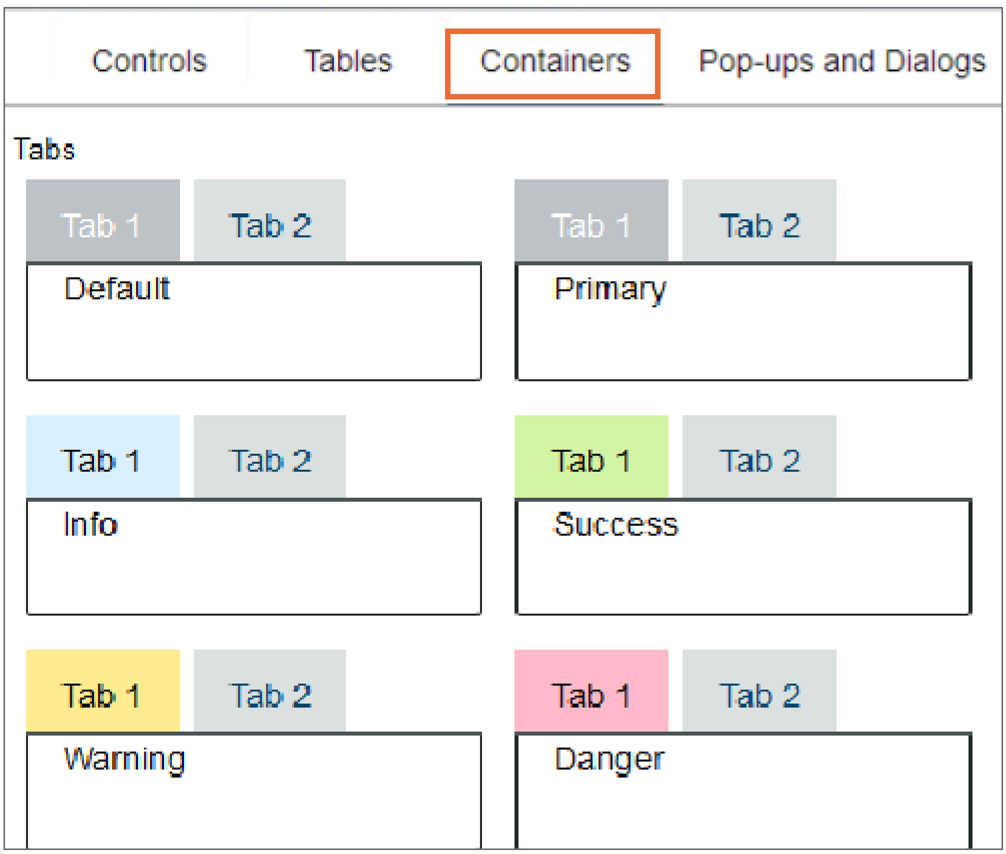


__ b. When you “blur” or click away from the color input field, you immediately see the color change for the affected controls on the right. Click any control on the right to see the CSS that affects the appearance of that object on the page.

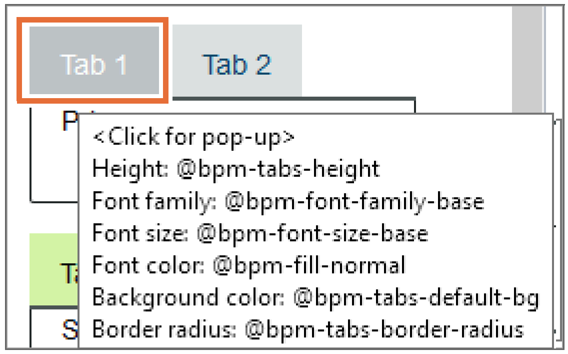



__ 4. Change the font color of the tabs.

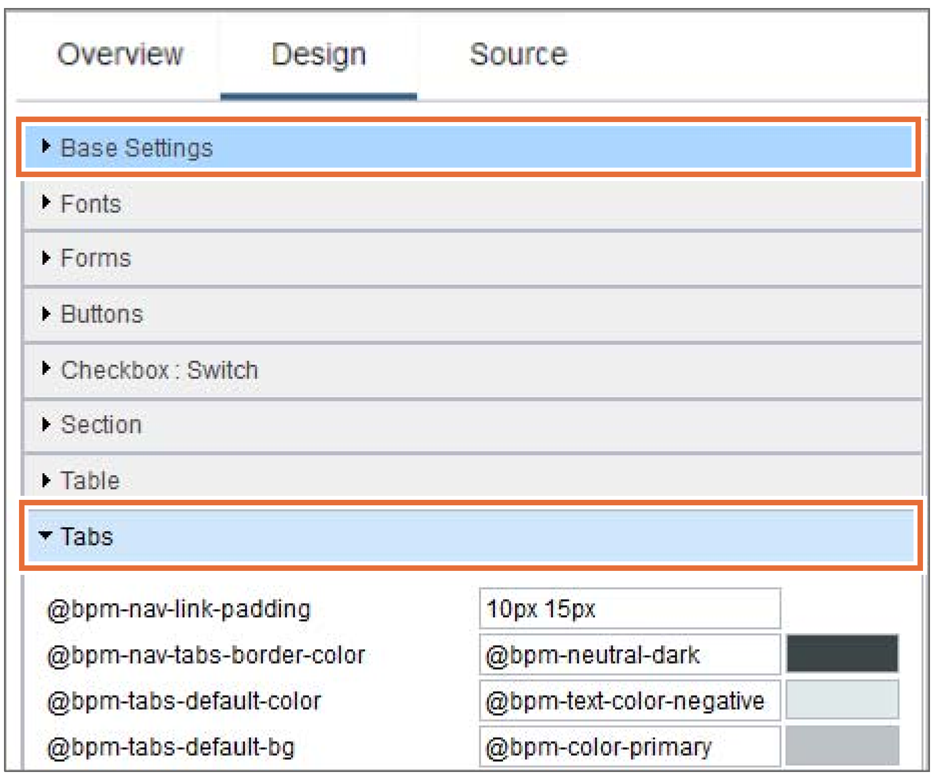
__ a. Click the **Containers** tab on the right.



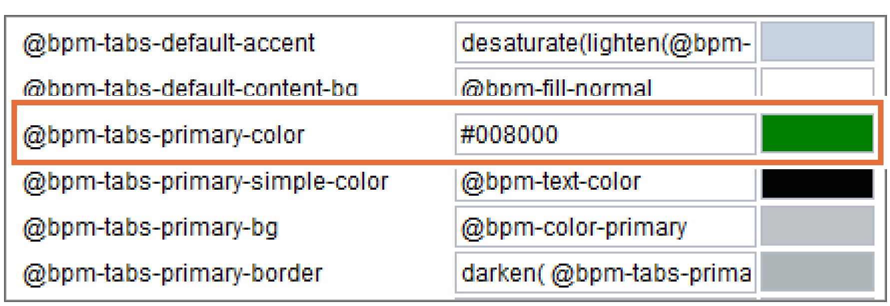
__ b. Hover over the Primary Tabs control in the first row in the right column. It shows all the variables that are associated with this control. For example, to change the font color of the tabs, you need to change the value of the `@bpm-tabs-primary-color` variable.




__ c. To change the font color, collapse the **Base Settings** section and expand the **Tabs** section.



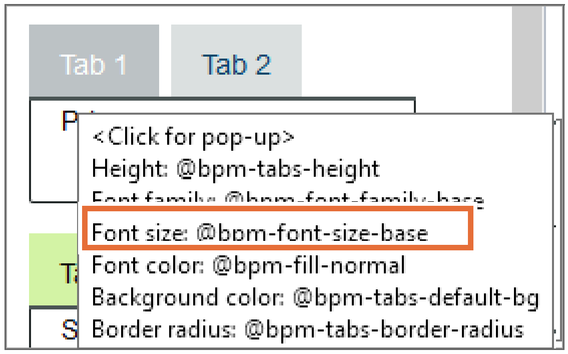

__ d. Look for the variable `@bpm-tabs-primary-color` and change the color to: `#008000`



You can choose to use any color of your choice. If you run into any problem, return to the step and follow that exactly, and then you can customize the color.

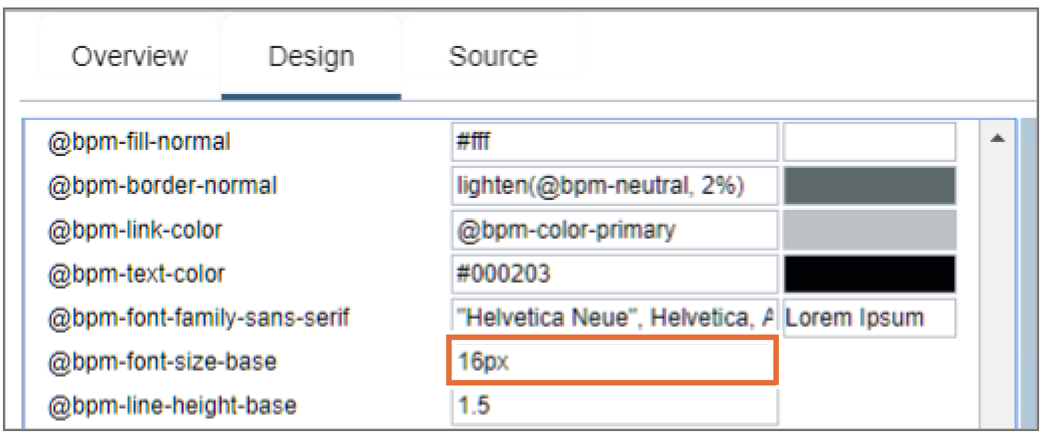
__ 5. Change the font size.

To change the font size, you need to change the value of the `@bpm-font-size-base` variable.



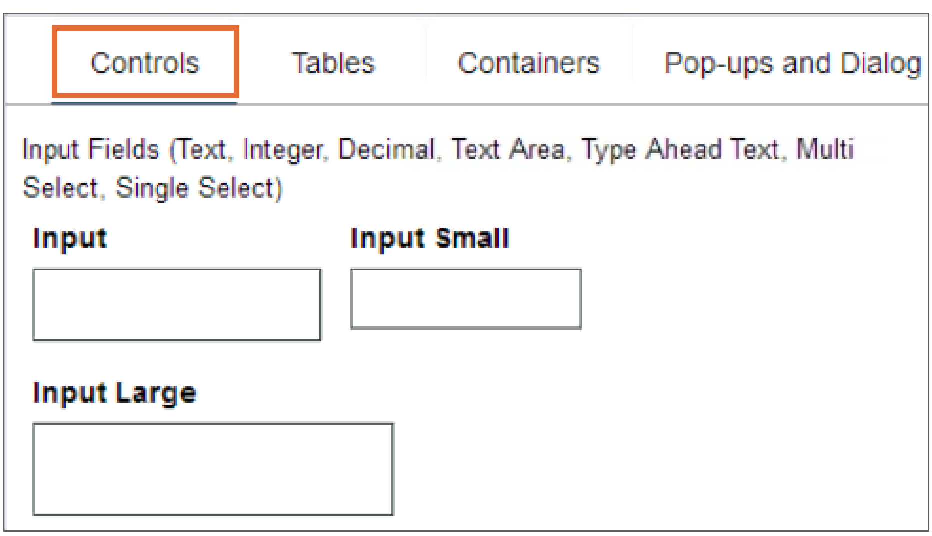

__ a. Collapse the Tabs section, and expand the **Base Settings** section.

__ b. Look for the variable `@bpm-font-size-base` and change the value to: `16px`



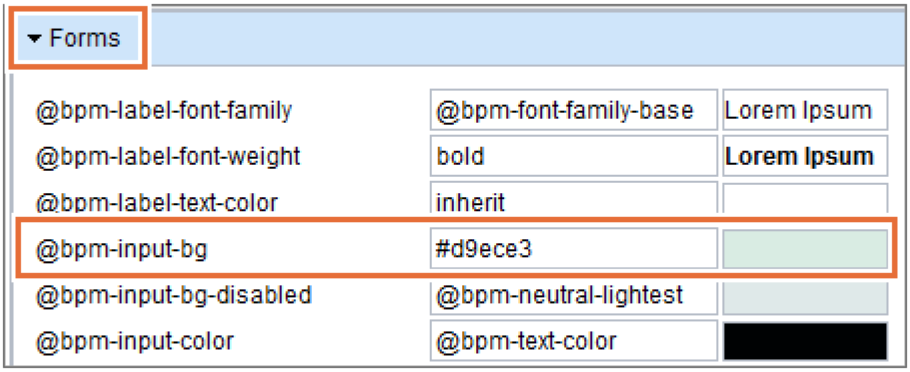

__ 6. Change the background color of the input controls.

__ a. Click the **Controls** tab on the right to view the input fields.



__ b. Expand the **Forms** section on the left.

__ c. Change the variable `@bpm-input-bg` to: `#d9ece3`

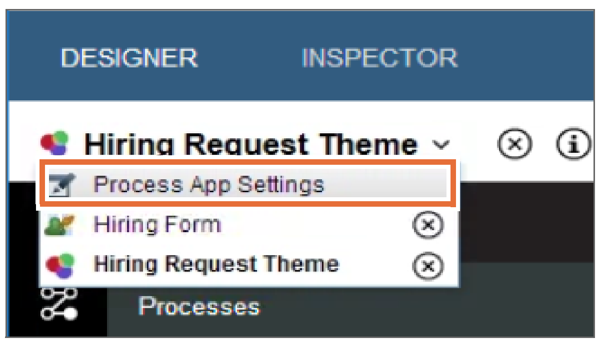



__ d. Click empty space outside the input control to refocus the mouse. The colors of the input controls are now highlighted with the chosen color.

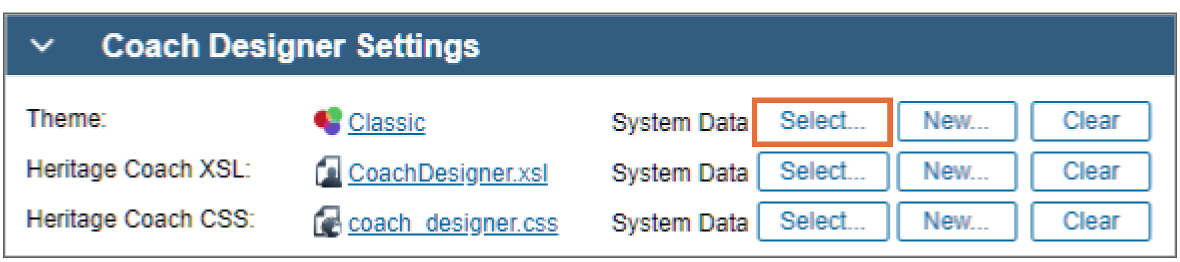
__ e. Save your work.

__ 7. Apply the new theme to the `HR Recruitment Processes` process application.

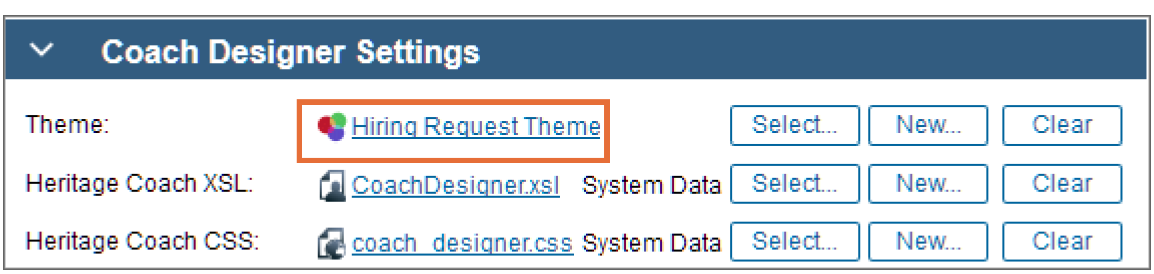
__ a. Open the **Process App Settings** from the History menu on the top.




__ b. In the Coach Designer Settings section, click **Select**.




__ c. Select **Hiring Request Theme**. The theme is now changed to the new theme.

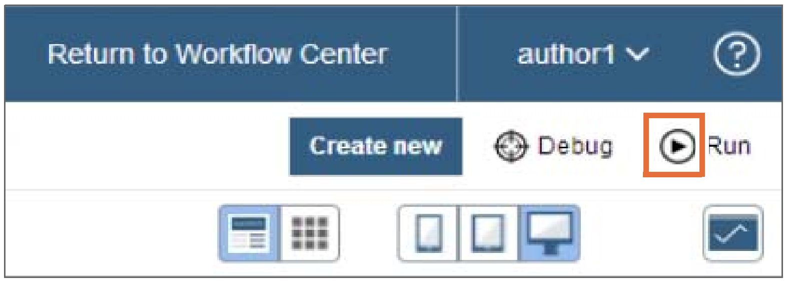


__ d. Save your changes. The save generates the CSS that the browser uses to display the controls in the layout.

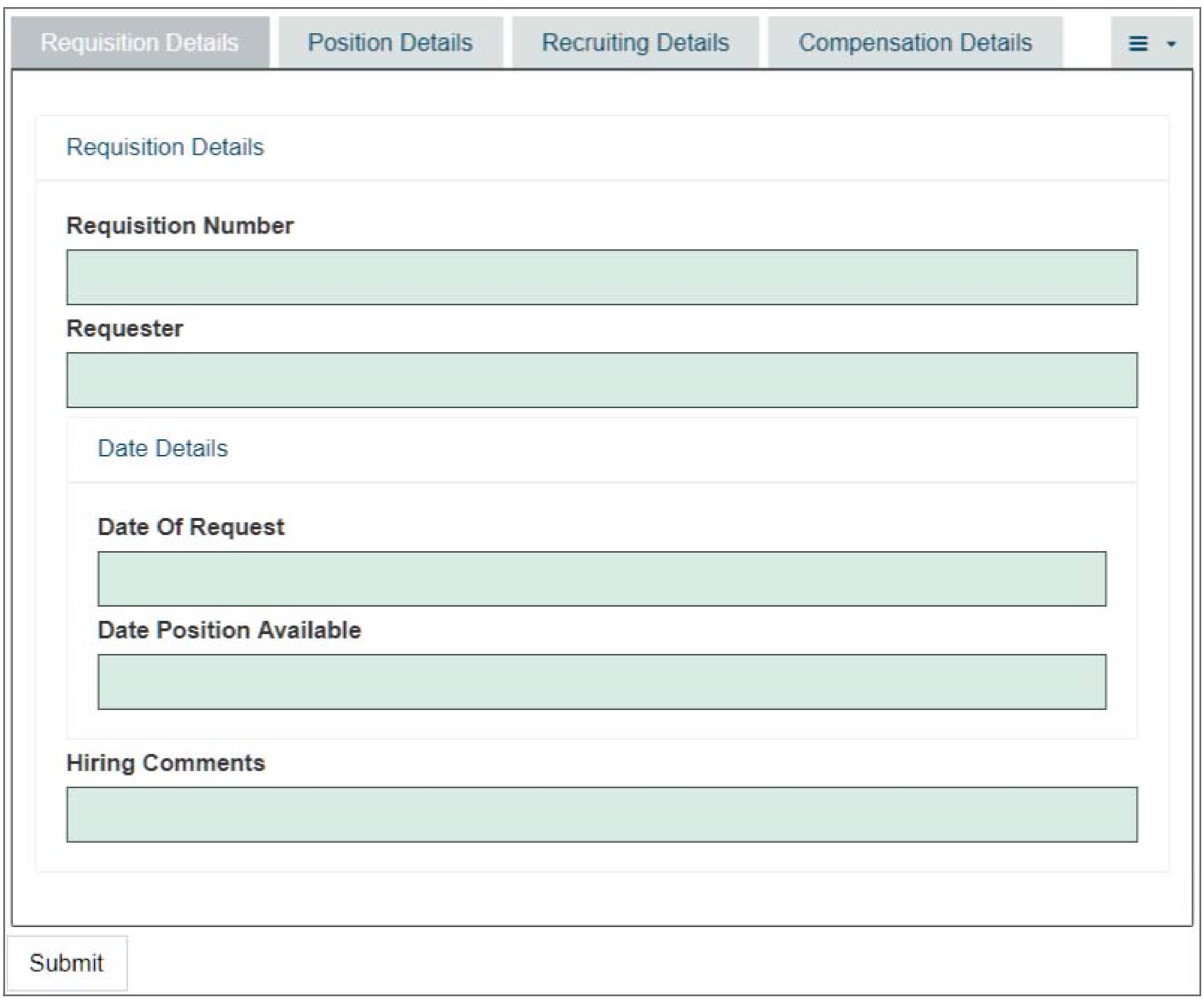
__ 8. Verify the theme changes

__ a. Open the **Hiring Form** Client-Side Human Service.

__ b. Click the **Run** icon at the upper-right side to display the coach with the custom theme. Initially when you view the coach, the server must generate the CSS and render the new coach on the screen, so a delay might occur before you see the effect.




The coach has the changes that you made in the Hiring Request Theme.




__ c. Close the browser window.

## *Part 3: Customize controls*

The coach framework offers many ways of customizing input controls and views. This part of your exercise shows some of the popular ways to customize your coach.

__ 1. Add an icon to the Date Details panel.

__ a. Select the **Requisition Details** tab on the coach.

__ b. Click the **Date Details** panel.


__ c. With the Date Details panel highlighted, click the **Select Icon** icon.


__ d. In the search field, enter `calendar`.

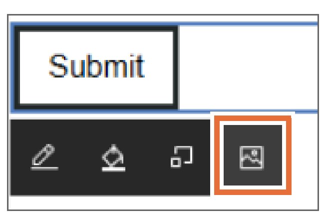
__ e. Select the second icon in the list labeled **calendar**.


__ f. The Date Details section now displays a calendar icon in the header of the panel.


__ 2. Change the appearance of the submit button.

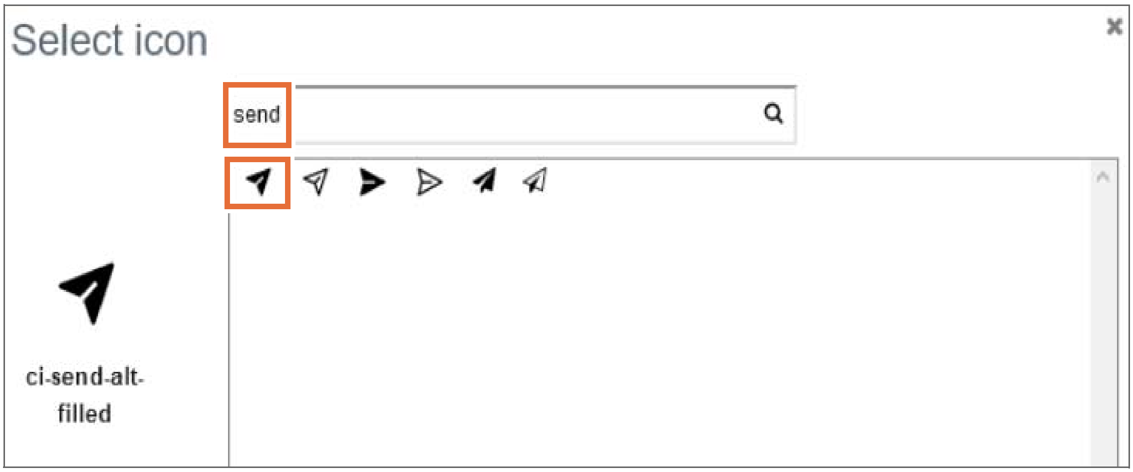
__ a. Select the **Submit** button on the canvas.

__ b. With the Submit button highlighted, click the **Select Icon** icon.

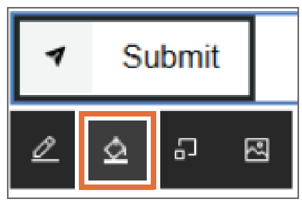



__ c. In the search field, enter send.

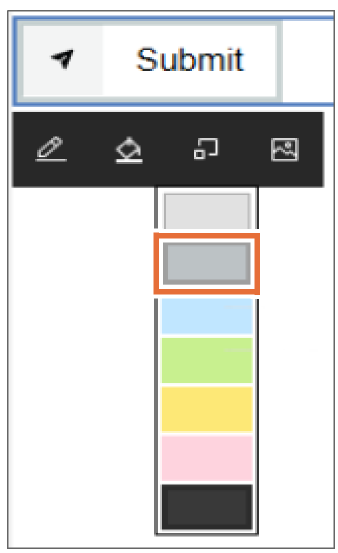
__ d. Select the first icon in the list.




__ e. With the Submit button highlighted, click the **Select Color** icon.




__ f. Select the **Primary** color.

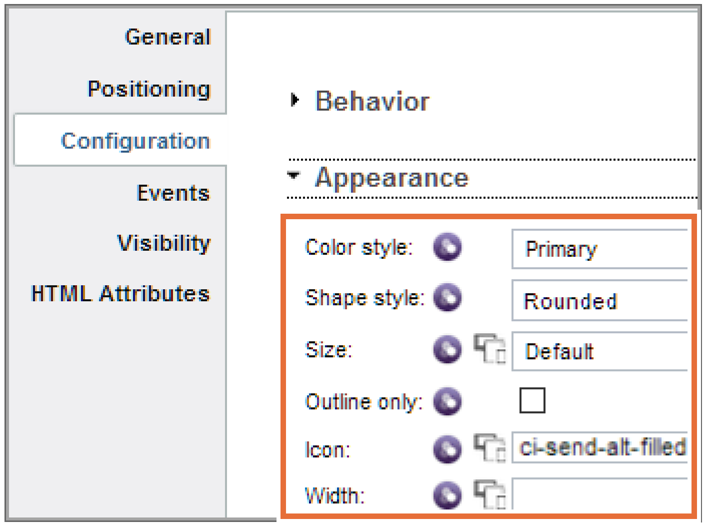



**Note**

The picker icons support the most popular configurations. For some configurations, you need to access the Configuration properties below.

__ g. With the **Submit** button highlighted, in the Properties section below, click the **Configuration** tab and expand the **Appearance** settings.

__ h. Notice the Color style and Icon which was configured in the prior steps. Set the shape style to **Rounded**. The button on the canvas reflects these settings as you make your configuration changes.




__ i. Click empty space outside the input controls to refocus the mouse.

__ j. The Submit button now displays the new configuration.


__ k. Save your work.

__ 3. Set the Num Employees Needed input control to visible only if the Multiple Employees Needed checkbox is selected.

It is common to hide controls and display them only when applicable. Since the Num Employees Needed input control is not needed unless there are multiple employees, this control is hidden until the Multiple Employees Needed checkbox is selected.

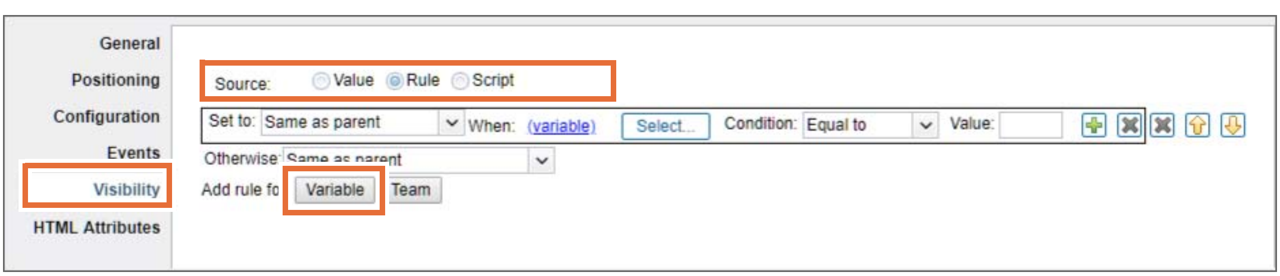
__ a. Click the **Recruiting Details** tab.

__ b. Select the **Num Employees Needed** input control

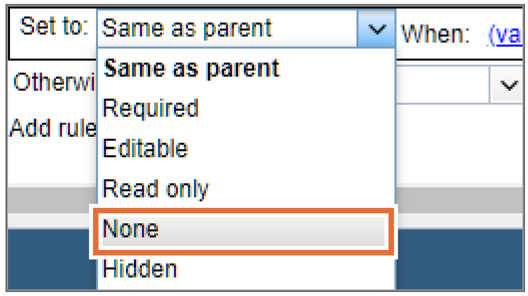
__ c. Select the **Visibility** properties section below

__ d. Click the **Rule** option for Source.

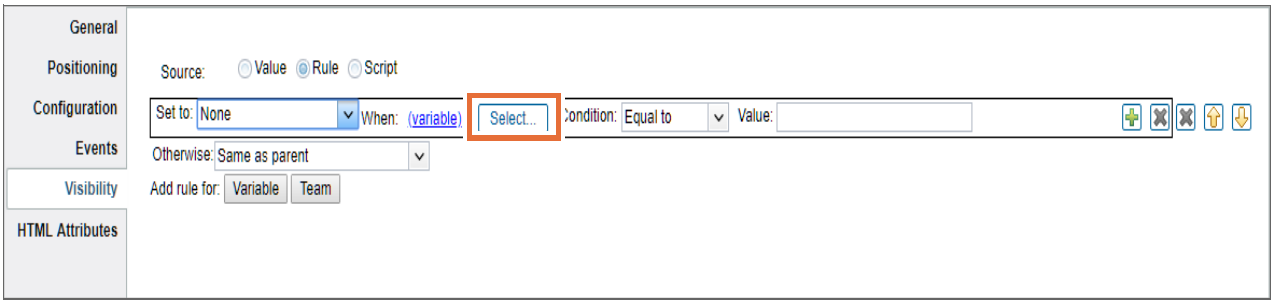
__ e. Click **Variable** for the Add rule for option.



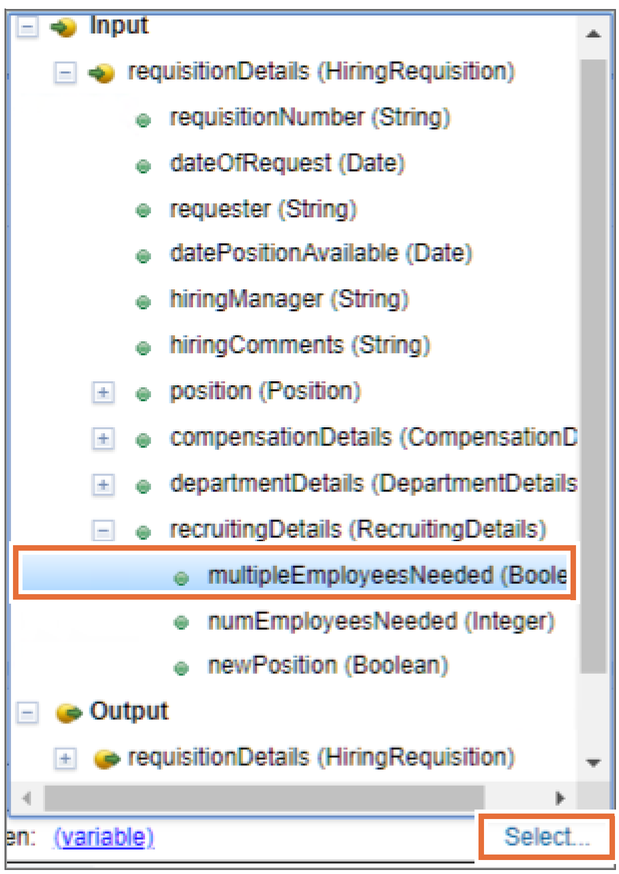
__ f. Select **None** for the Set to value.



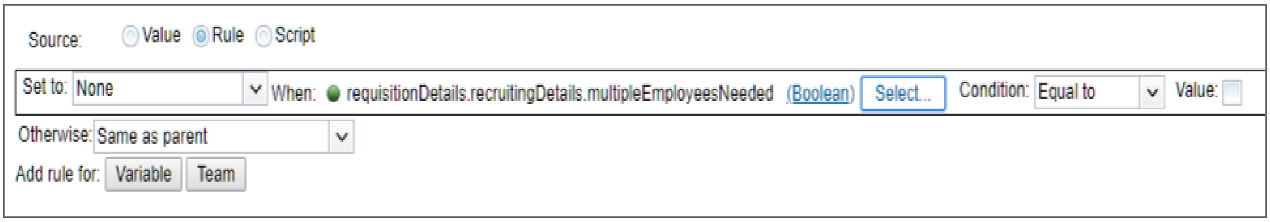

__ g. Click **Select**.




__ h. Navigate to **Input > requisitionDetails > recruitingDetails > multipleEmployeesNeeded (Boolean)**



__ i. Your rule is now configured.




The rule determines when to set the input control to "none." Since the input control is a Boolean, the value is depicted as true (Value checkbox marked) or false (Value checkbox not marked). This rule states:

Set visibility to "None" when `multiple.EmployeesNeeded` has a false value otherwise inherit the visibility setting from the parent.

### Information

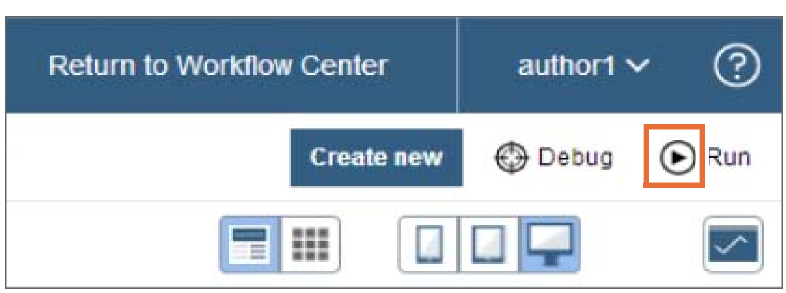
Visibility settings are described below:

| Visibility Setting | Description |
|---|---|
| **Same as parent** | Inherits visibility setting from the parent (in this case that would be the panel). |
| **Required** | Input control must have data before coach can complete |
| **Editable** | Input control is editable |
| **Read only** | Input control can only be viewed, not edited |
| **None** | Input control is not displayed including the space that it takes up on the coach |
| **Hidden** | Input control is not displayed, not including the space that it takes up. In this case, the space the control takes up does not change even though the control is hidden. This is useful if you do not want fields moving as a result of a field not displaying. |

__ j. Save your work.

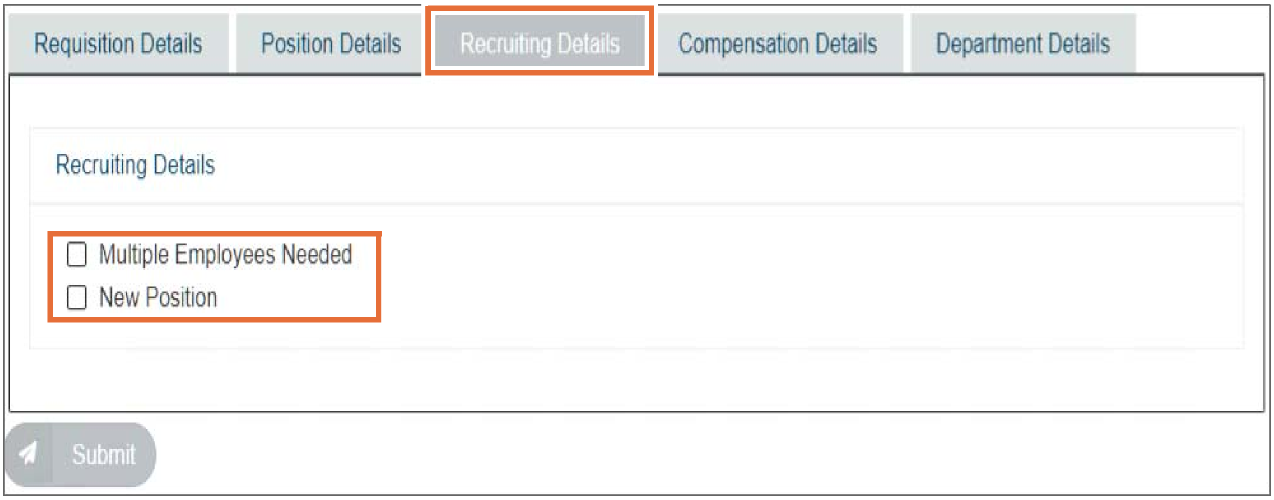
__ 4. Test Num Employees Needed input control visibility setting

__ a. Run the coach.




__ b. In the Coach window, click the **Recruiting Details** tab.

__ c. If you configured the visibility setting correctly, the **Num Employees Needed** input control is not visible since the **Multiple Employees Needed** value is `false`.




__ d. Test the setting by clicking the **Multiple Employees Needed** checkbox, setting the value to true. **The Num Employees Needed** input control displays below.

__ e. Close the browser window that you ran the coach in.

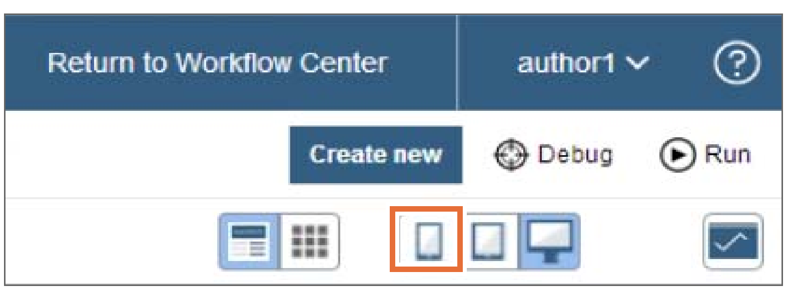
## *Part 4: Modify the mobile layout*

You are going to change the layout of the coach to the mobile layout so that the coach is displayed correctly on a mobile device. You make a few of the sections collapsible and change the width of a few input controls based on the width of the browser window.

__ 1. Configure the width of the Submit button to respond to the screen size.

__ a. Toggle the Properties section off by selecting the **Properties** icon in the bottom status bar.

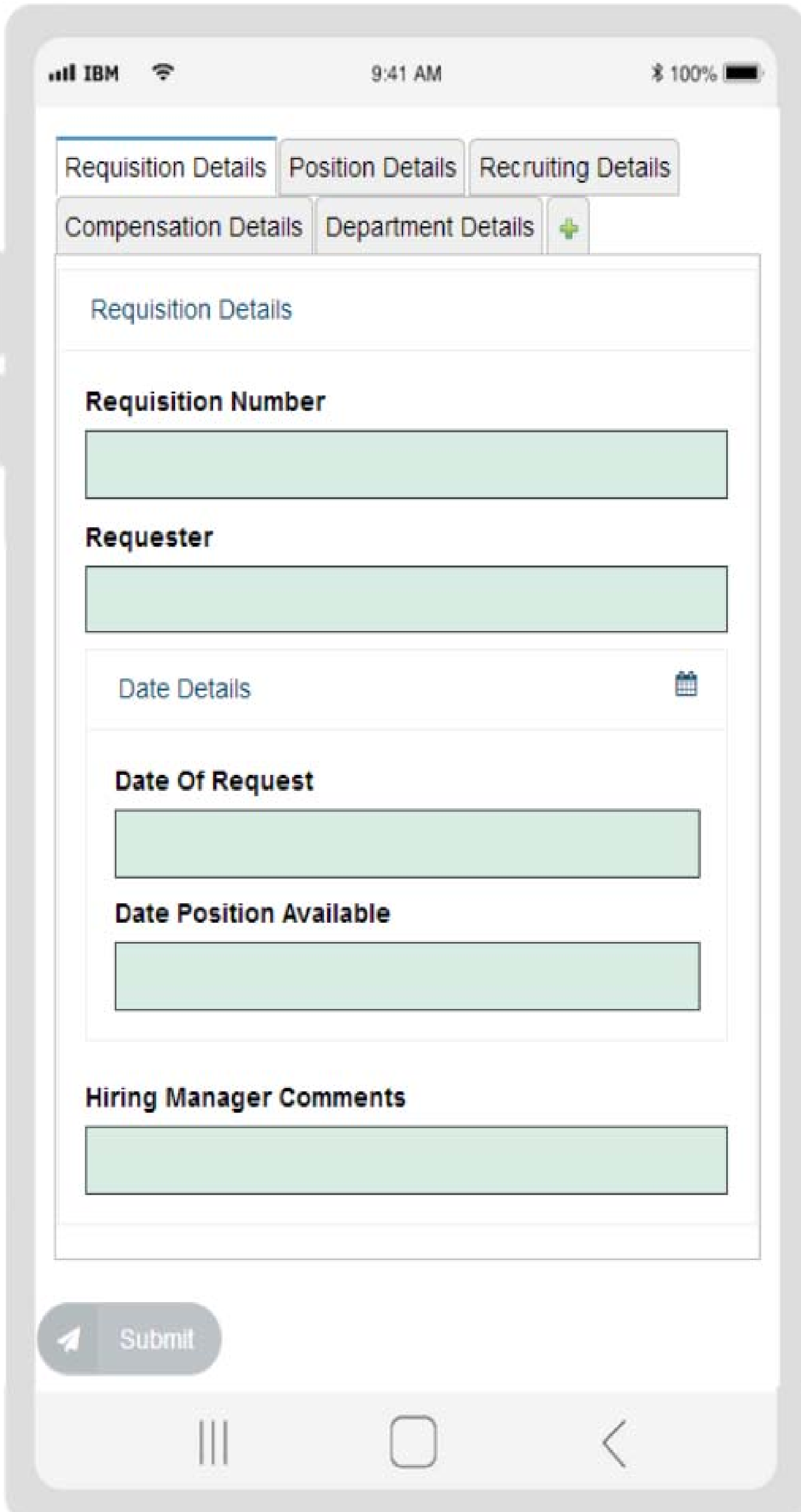

__ b. Click the **Small screen** icon above the palette.




**Hint**

Press **Ctrl** and **+** or **Ctrl** and **-** to increase and decrease the browser zoom setting.

__ c. The Hiring Form is presented in a small screen size that resembles a phone. Your screen might look different if you are focused on a different tab.

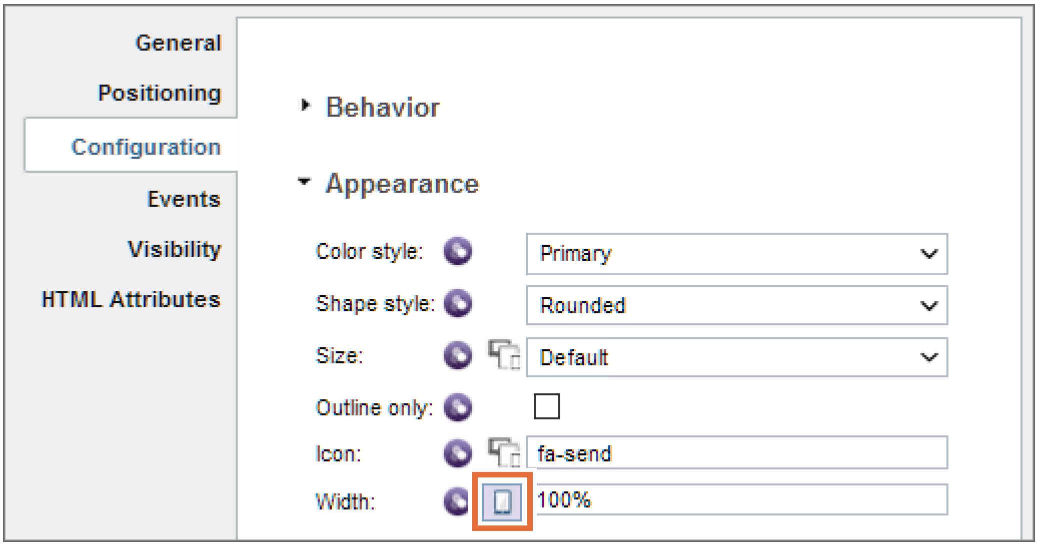



__ d. Toggle the **Properties** section on.

__ e. Select the **Submit** button.

__ f. Click the **Configuration** properties tab and expand the **Appearance** properties settings.

__ g. Enter `100%` in the **Width** configuration value. Refocus the mouse.




After you refocus the mouse, the icon changes to the small screen icon. This means that the setting is only made for the small screen size. You can configure different widths for the three different screens. The original setting for the large screen size has not changed. Make sure you refocus the mouse after changing the settings to verify the correct screen size setting.

__ h. The Submit button now expands to 100% of the screen.


__ 2. Configure the width of the Date Details panel to respond to the screen size.

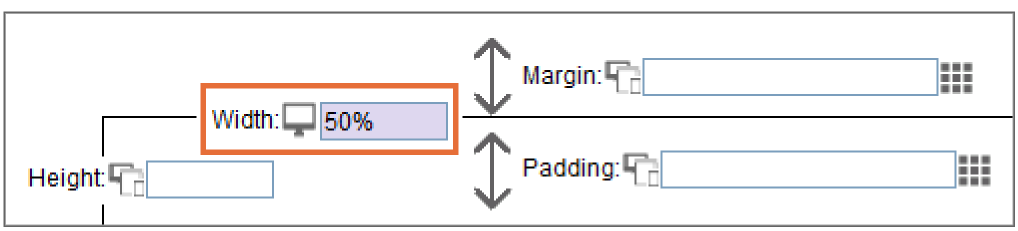
You can change the width settings in either the Configuration tab or the Positioning tab. When adding values for multiple screen sizes, it is recommended to use the Positioning tab.

__ a. On the **Requisition Details** tab, select the **Date Details** panel.

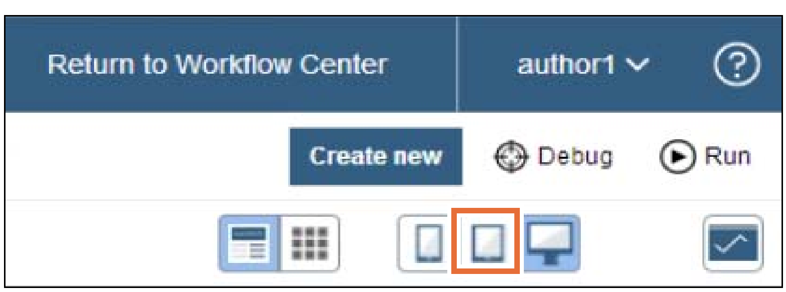

__ b. Click the **large screen** icon above the palette.


__ c. Set the **Positioning > Width** properties to `50%`. The entered width is applicable only for large screen devices. Click the space outside the input control to refresh the icon. The icon is set to large screen.

__ d. Refocus the mouse.



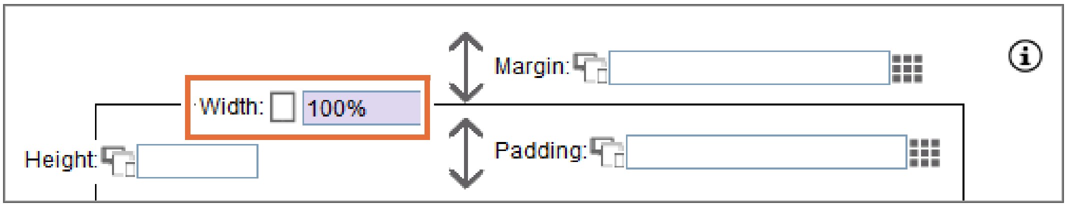

__ e. Click the **medium screen** icon above the palette.



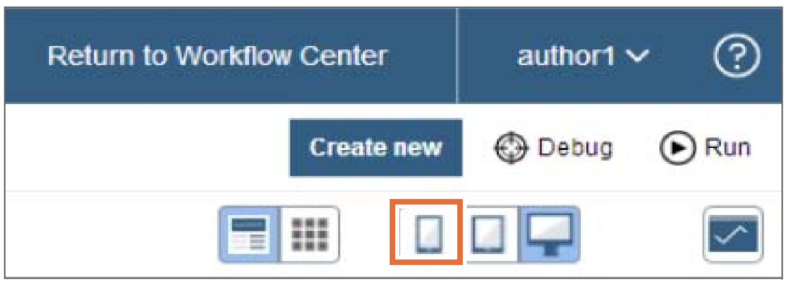

__ f. Set the **Positioning > Width** properties to `100%`. The entered width is applicable only for medium screen devices. Click the space outside the input control to refresh the icon. The icon is set to medium screen.

When you set the tablet setting (medium screen), the small screen automatically is set to `100%.` This setting is inherited from the tablet format setting, so you do not have to set it to `100%`.

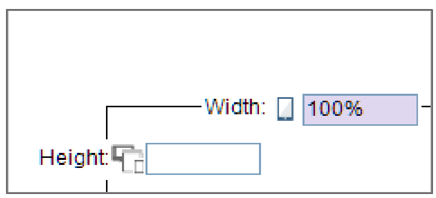
__ g. Refocus the mouse.




__ h. Click the **small screen** icon above the palette.




__ i. Verify that the width settings are set to `100%`.

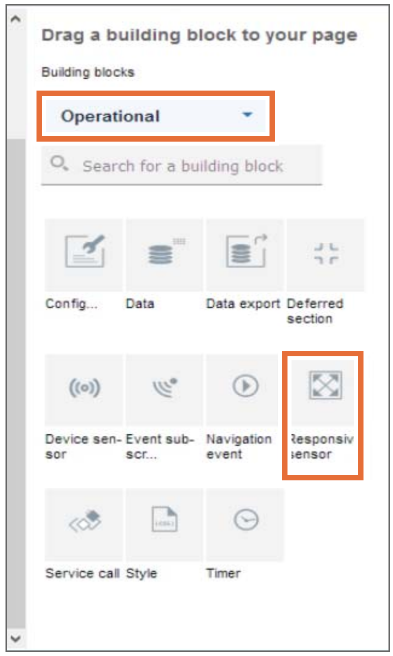



__ j. Save your work.

__ 3. Configure a responsive sensor.

__ a. Click the **large screen** icon above the palette to return to the web page layout.

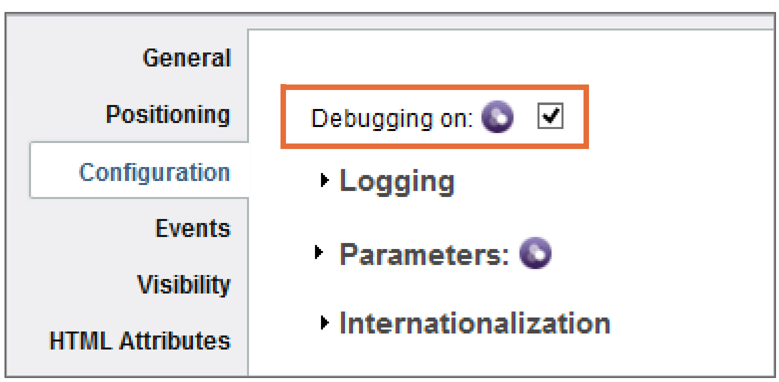
__ b. Select **Operational** in the **Building blocks** panel and select **Responsive sensor**




__ c. While holding down the left mouse button, drag the **Responsive sensor** below the **Submit** button. Let go of the mouse button when the bottom border displays a gold highlight.


__ d. While holding down the left mouse button, drag a **Configuration** below the Responsive sensor. Let go of the mouse button when the bottom border displays a gold highlight.

__ e. Under the **Configuration** properties, enable the option to turn **Debugging On**.




__ f. Access the Layout building blocks and while holding down the left mouse button, drag a vertical layout over the **Responsive Sensor**

__ g. When the **Drop content here** section is highlighted, let go of the mouse button

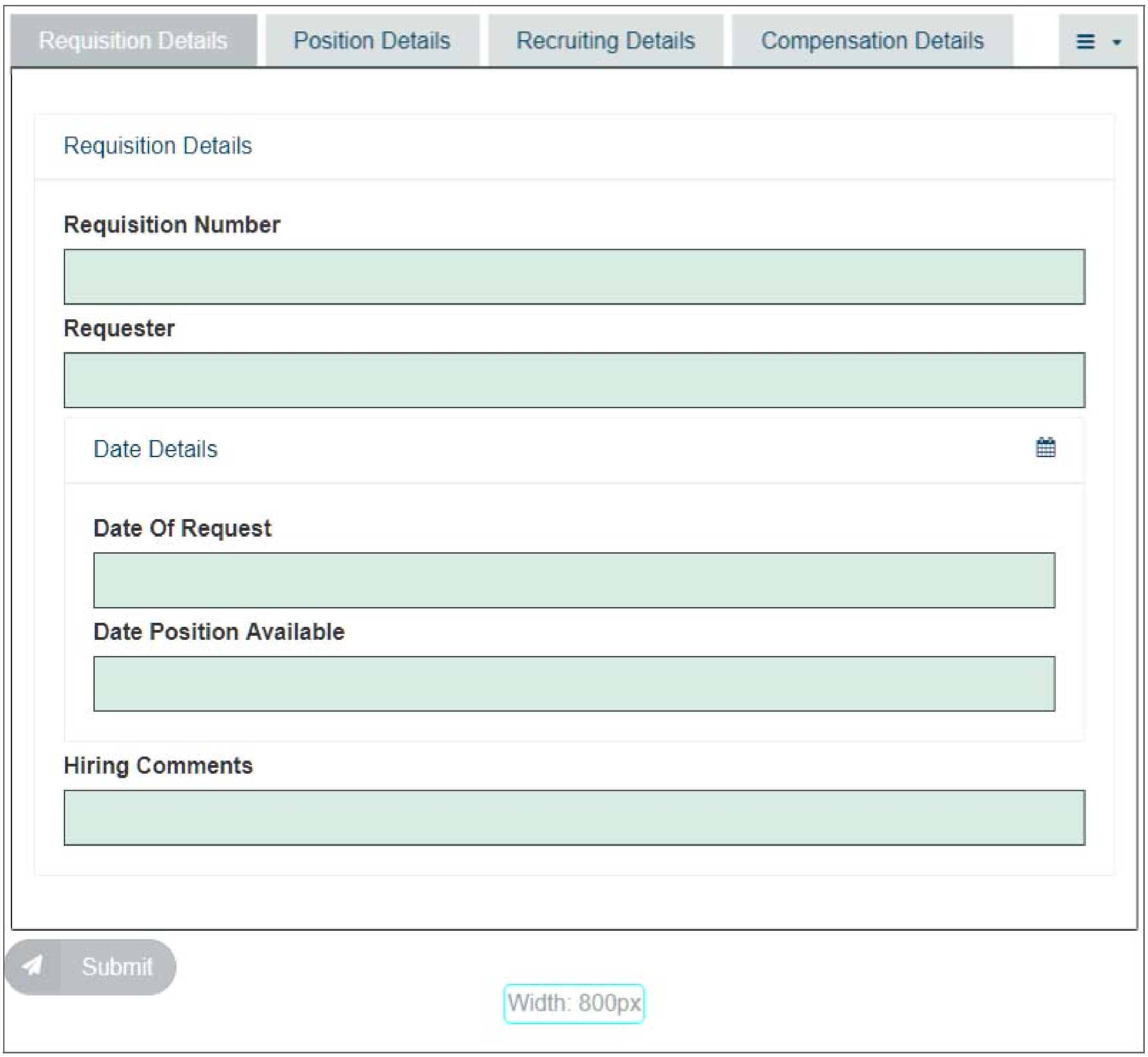
The Responsive Sensor requires a horizontal or vertical layout to correctly calculate the screen size. When the coach is run, the browser width is displayed in this Responsive Sensor as soon as the browser is resized to test the responsive settings.

__ h. Save your work.

__ 4. Verify the coach display for different screen sizes.

__ a. Click the **Run** button.

A new browser window displays the coach and the width is shown at the bottom. The Date Details panel in this image spans the entire browser window because the initial width is set to a medium screen size setting. Depending on the size of your screen, the layout width might vary.

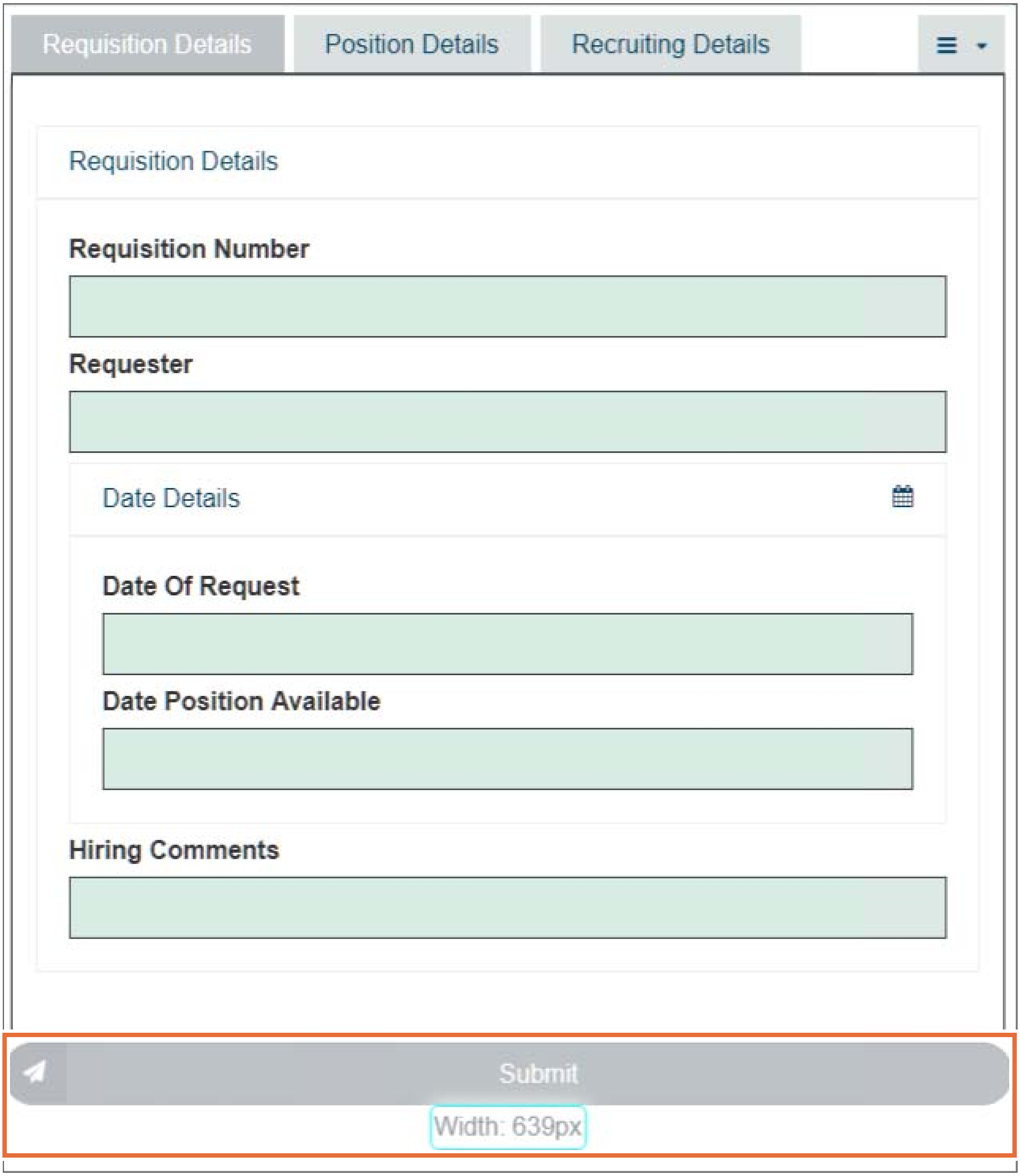


Recall the sizes for each screen.

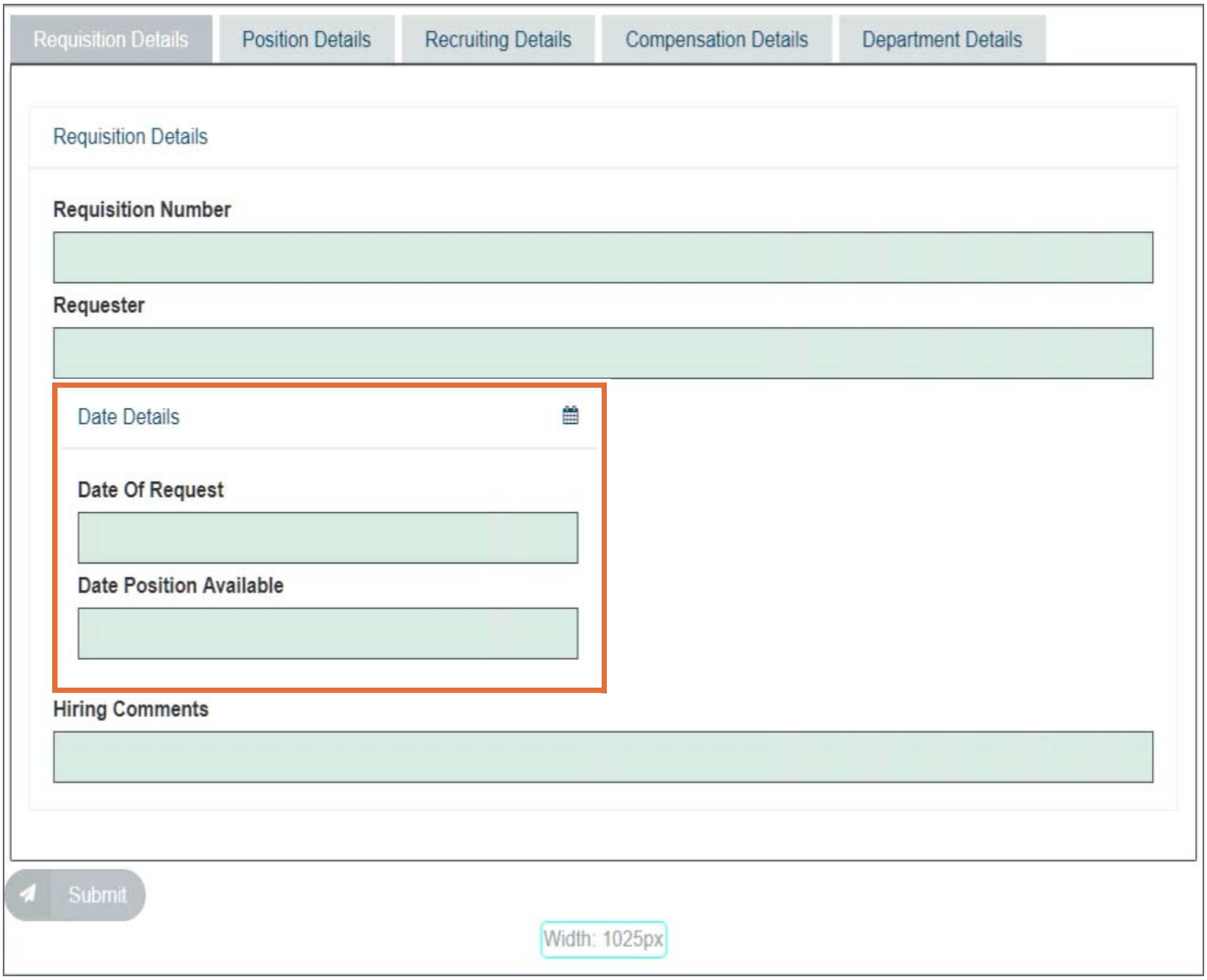
*Table 2. Screen sizes*

| Icon | Size | Resolution |
|---|---|---|
|  | Small | 640 pixels or less |
|  | Medium | 641 - 1024 pixels |
|  | Large | More than 1024 pixels |

__ b. Slowly reduce the width of the browser. When you pass the 640 pixels threshold, the Submit button expands to 100% of the screen width.

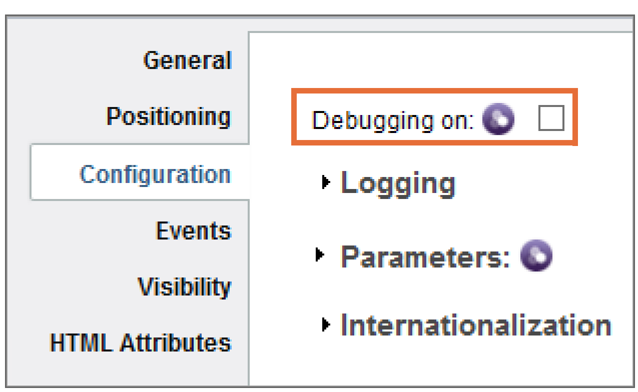


__ c. Now expand the browser window. When the width passes 1024 pixels, the Date Details panel contracts to 50% of the browser width because of the responsive control settings based on the large screen width setting.




__ d. Close the browser window.

__ e. Click the **Configuration** on the canvas. On the **Configuration** properties menu, clear the **Debugging on** checkbox to remove the width display on the coach.




__ f. Save your work.

You have completed this exercise.

Playback 1: User interface design and implementation is now complete. In this exercise, you completed the following:

- Created tabs on a coach
- Changed the appearance of a coach by applying a custom theme
- Changed the coach layout for a mobile format
- Configured responsive controls
- Debugged the coach by using a responsive sensor

As part of the development process, you now review the playback and examine its functions in the Process Portal. In the next exercise, you test the following functions by performing a playback:

- Does the process follow the correct paths?
- Does the application work as expected in the Process Portal?

As part of the playback, you demonstrate the process application in the Process Portal. You then create a toolkit in the Workflow Center and export your process application.

## End of exercise

# Exercise 7. Playback 1: Conducting the Playback session

## Estimated time

01:00

## Overview

This exercise covers how to conduct a Playback of your process. The exercise demonstrates the process, following various paths that flow from the exclusive gateways in the process and demonstrates tasks that are assigned. It describes the task that is created in the Process Portal inbox, depending on the swimlane and routing settings for an activity. You also create a toolkit to store and share these assets.

## Objectives

After completing this exercise, you should be able to:

- Log on to the Process Portal and create an instance of a process
- Demonstrate that the process follows the various paths modeled
- Use Process Portal to view the state of activities in a process
- Create a toolkit
- Create a snapshot in the Workflow Center
- Export the process application

## Introduction

The following list provides a quick view of what to accomplish in the playback portion:

1. Log on to the Process Portal and create a process instance.
2. Log on to the Process Portal as two different teams to show that the correct process team is being assigned the correct task.
3. Complete the human activities.
4. Demonstrate that the correct path is followed.
5. Change variable values and demonstrate following a different path.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

# Exercise instructions

## *Part 1: Prepare for the playback*

To enable a well-run playback in a customer engagement, you should practice running through the model beforehand. It is suggested that you run through the playback with a colleague several times to ensure a smooth presentation. This can also aid in finding unexpected issues.

It is common to prepare your environment for the playback to exemplify specific functionality explicitly demonstrated for the playback. In the case for this playback, to facilitate activities that time out in order to demonstrate different activity statuses in the Gantt chart, you change the timeout settings in two activities.

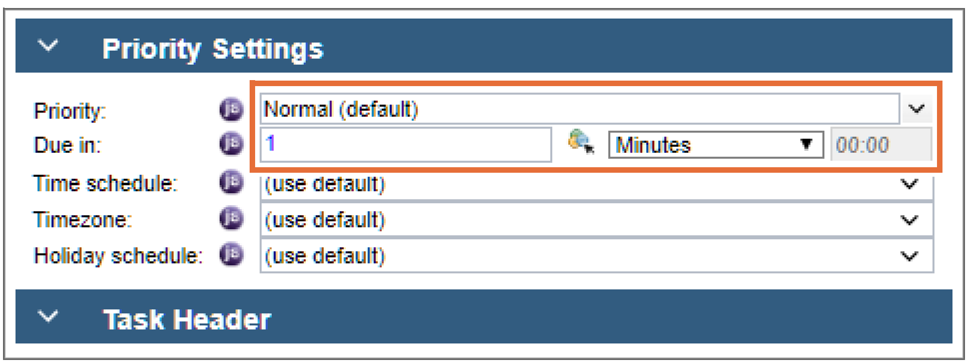
__ 1. Change the priority and timeout value for the **Complete Hire Request** activity.

__ a. Open the **HR Recruitment Process** process application.

__ b. Open the **Hiring Request Process**.

__ c. Click the **Complete Hire Request** activity.

__ d. In the **Priority Settings** section of the **Implementation** properties at the bottom, leave the **Priority** setting as the default and set the activity to be due in 1 minute.

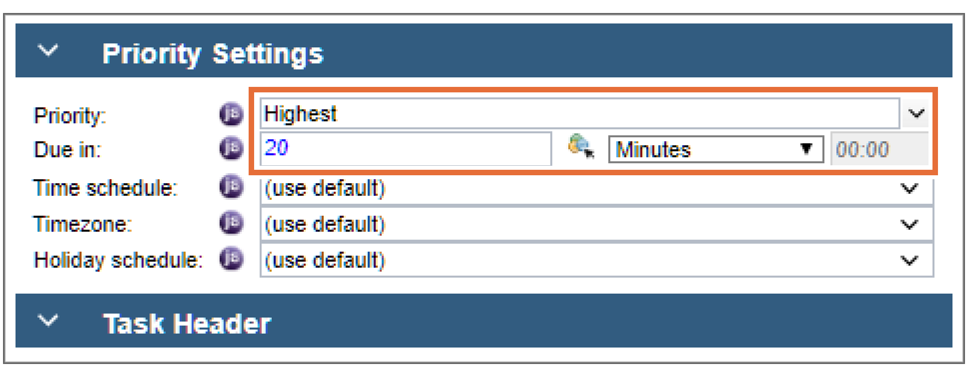



__ 2. Change the priority and timeout value for the **Review Posting** activity.

__ a. Click the **Review Posting** activity.

__ b. In the Priority Settings section of the **Implementation** properties at the bottom, select **Highest** for the Priority setting.

__ c. Set the activity to be due in 20 minutes.




__ d. Save your work.

When you demonstrate viewing work items later in this exercise, the different priority settings are displayed.

## *Part 2: Demonstrate the Review Needed path*

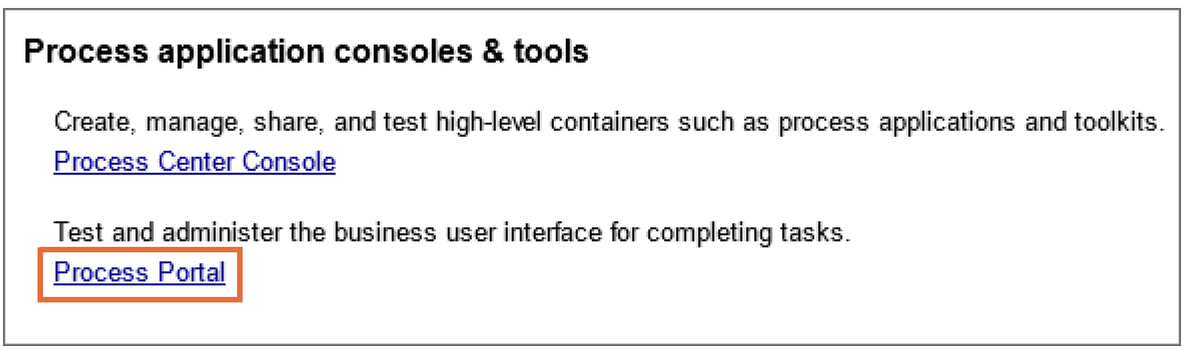
During this playback session, you demonstrate the process, following various paths that flow from the exclusive gateways in the processes, and demonstrate tasks that are assigned.

You accomplish a process playback session just as you would do when you seek consensus to move the project to the next playback phase. You log in as different users of the process to demonstrate tasks that different users accomplish in the process, demonstrating coaches that you created in the current playback phase.

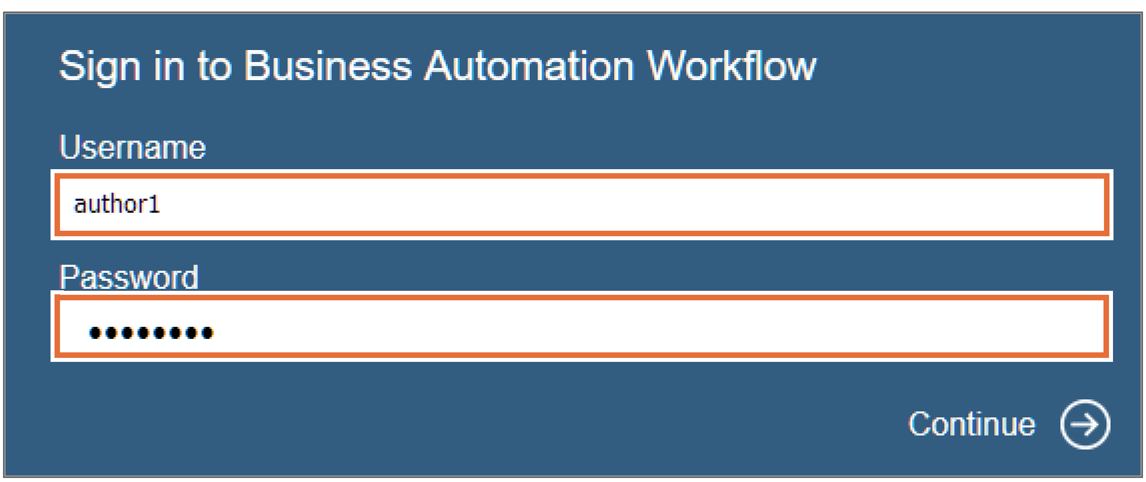
__ 1. Log on to the Process Portal.

__ a. Maximize the **IBM Process Center Quick Start** browser window from the Windows taskbar.

__ b. Click the **Process Portal** link.

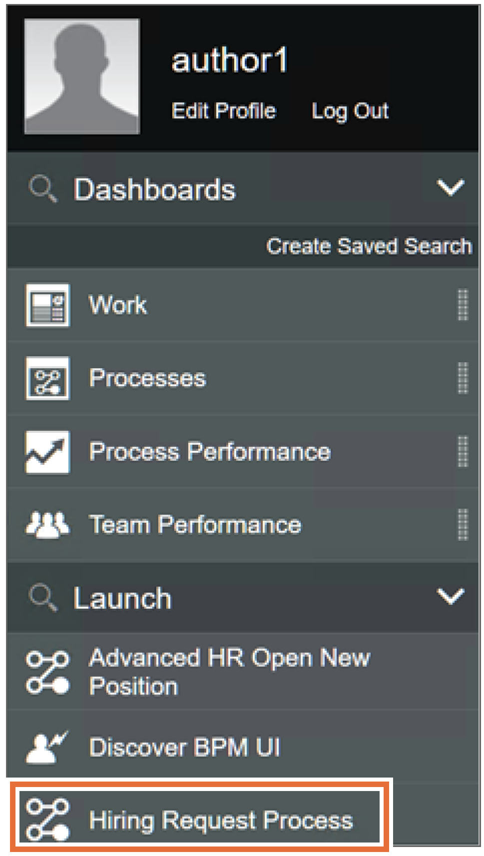



If prompted, log in with `author1` as the username and `passw0rd` as the password.

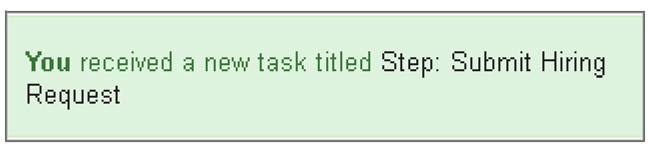


__ 2. Complete the first activity, Submit Hiring Request, in the Process Portal.

__ a. In the left frame, click **Hiring Request Process** to start an instance of the process.

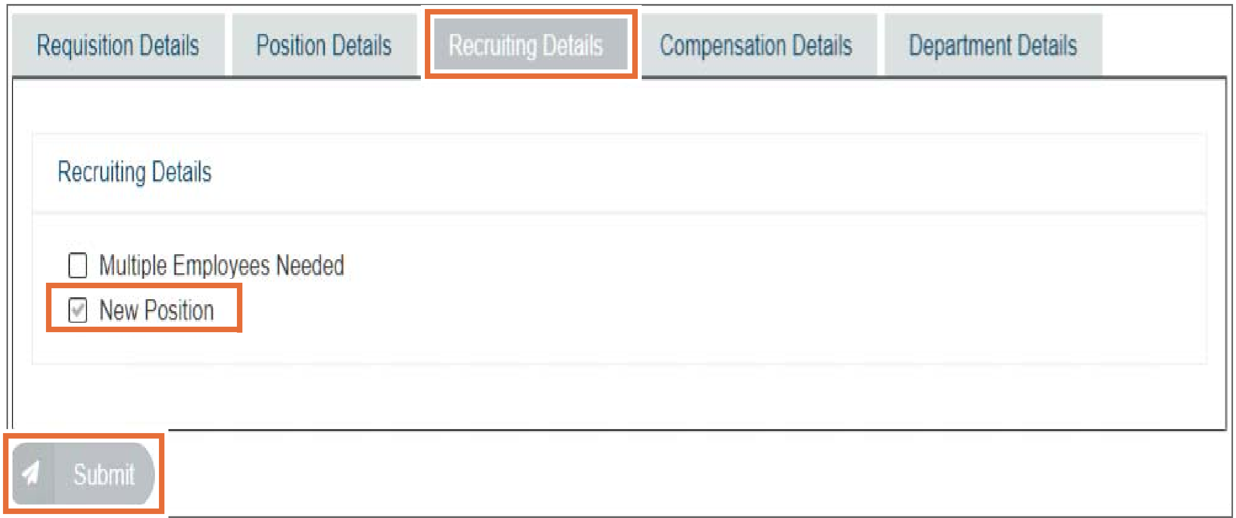



__ b. A notice is displayed to indicate that the process is started.



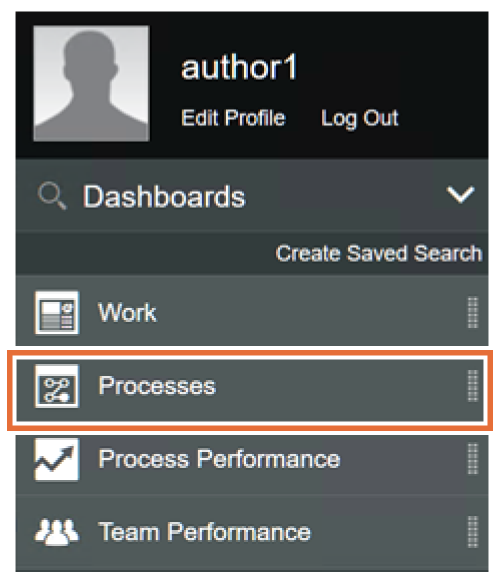

The first task starts automatically in the portal because you assigned the first task to Lane and Last User. A window is displayed with the human service coach.

__ c. Click the **Recruiting Details** tab, and select the **New Position** checkbox so that it follows the Review Needed path. Click **Submit**.




__ 3. View the task that is assigned to the next activity, `Approve New Hire Request`, in the process.

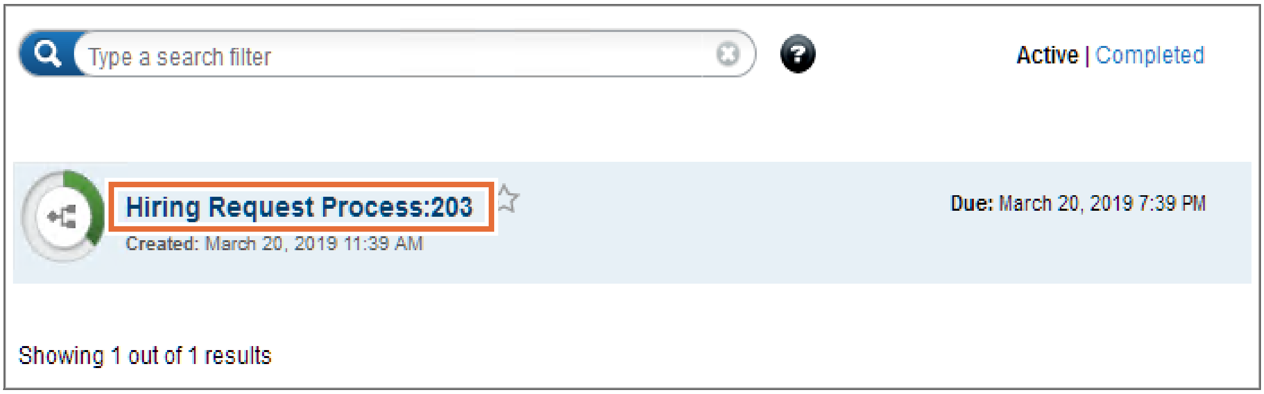
__ a. In the Library click **Processes**.




**Troubleshooting**

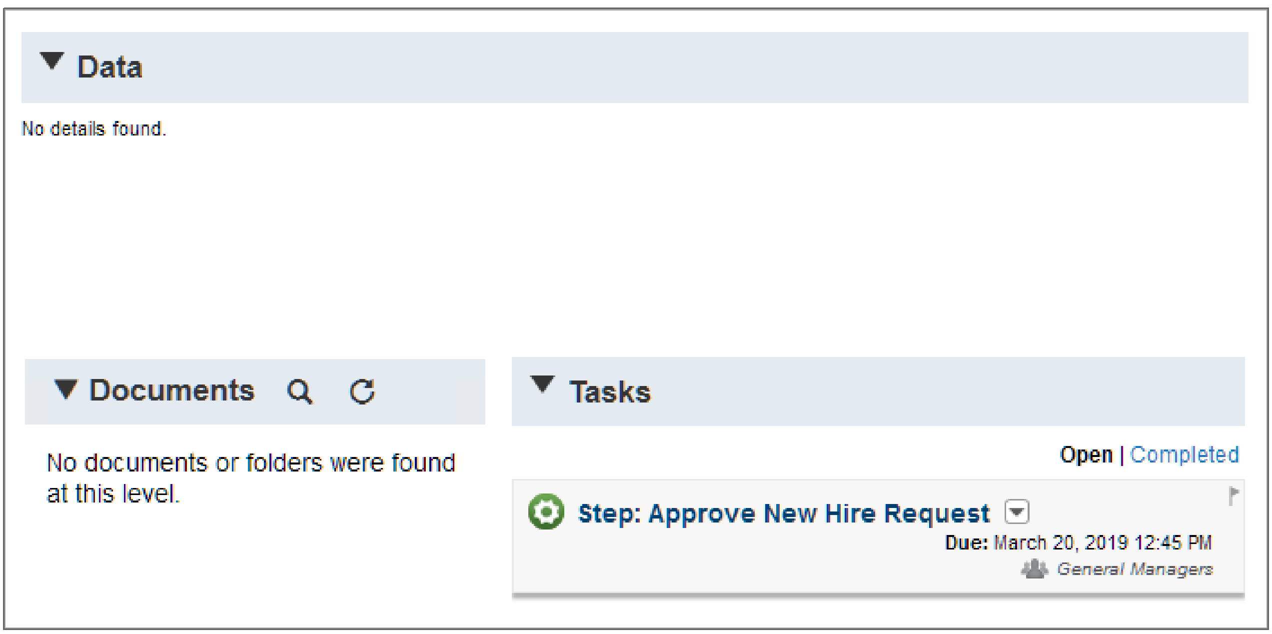
If the process is not displayed, stop the Process Center server, restart Windows, then restart the Process Center server.

__ b. The Processes page opens. Click **Hiring Request Process**.

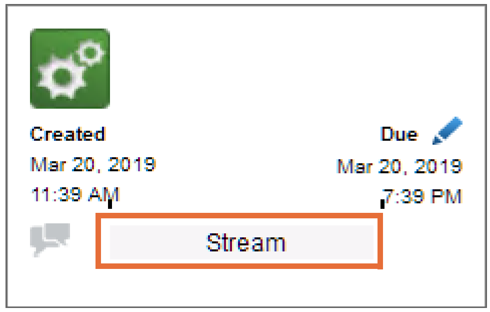



The process instance ID is the number next to the name of the process. The process instance ID can be different in your environment. If you have more than one instance, click the one with the highest instance number (the number after the colon).

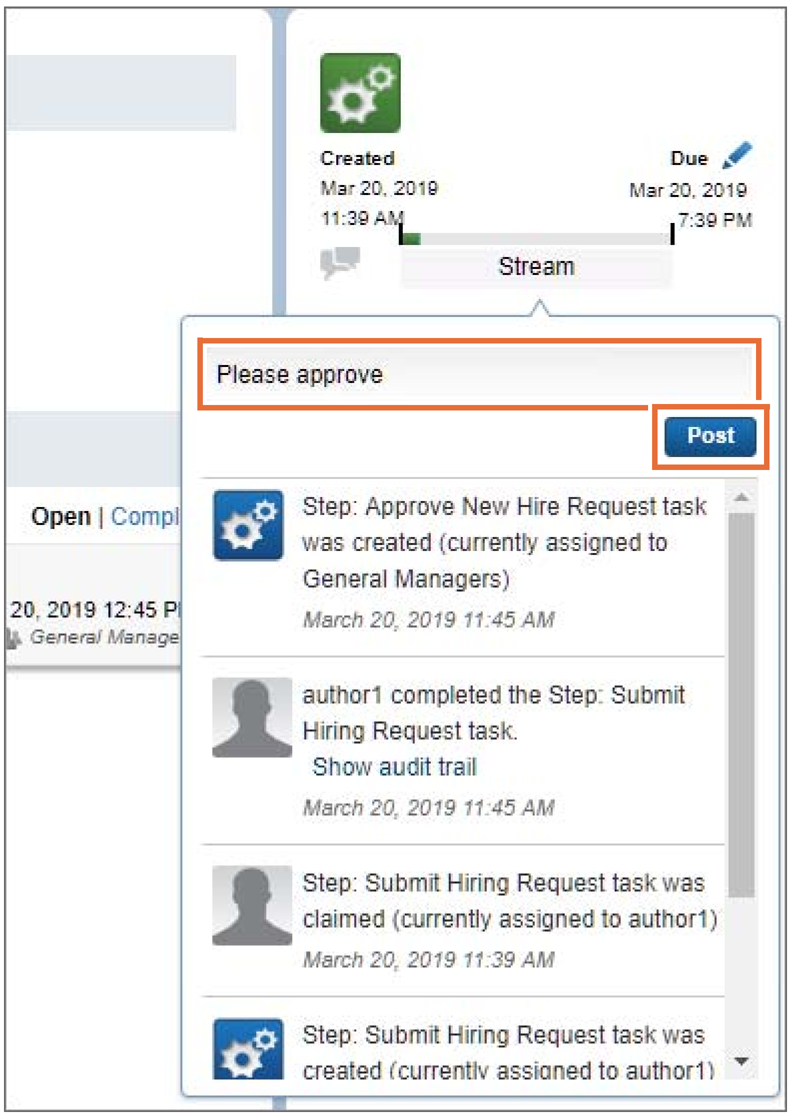
__ c. The process tab displays data about the instance. The bottom section that is labeled **Tasks** shows the next incomplete task in the process. Do **NOT** click the task. This task is assigned to the **General Managers** team.



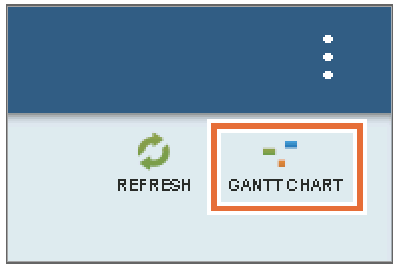
__ d. On the right, the creation date and the process due date are shown, along with a timeline that shows the current duration of the instance. Click the **Stream** area to view the actions that occur for this instance. Sometimes it takes a moment for the word Stream to appear.




__ e. To add to the stream, type `Please approve` into the stream field and click **Post**. Your comment appears as the most recent entry in the stream.

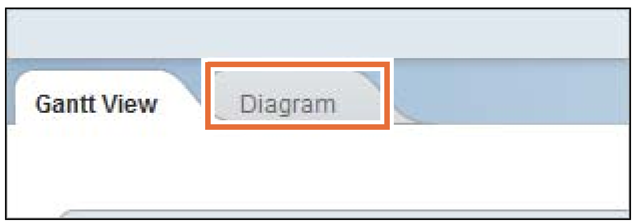


__ f. Click the **GANTT CHART** icon in the upper-right corner to go to the Gantt chart view.

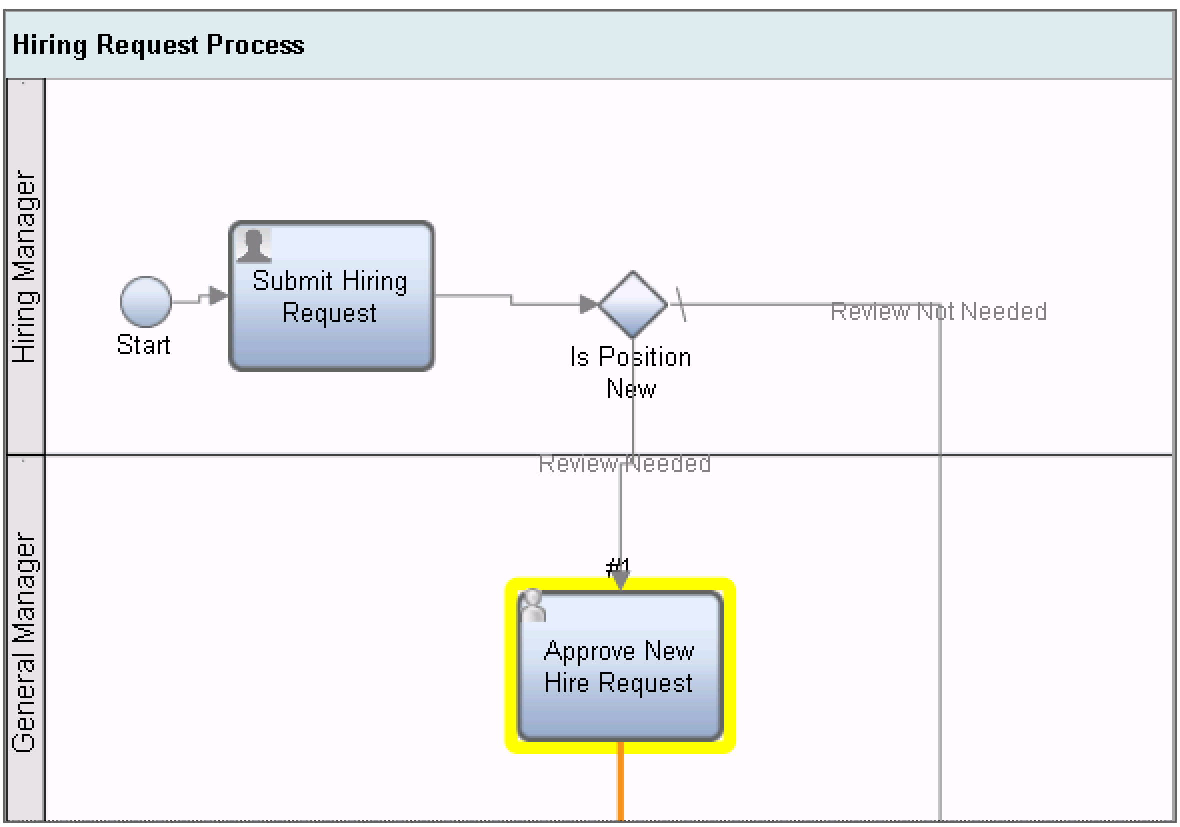



The Gantt chart is part of the Process Performance tab. When you click the link, you open the **Process Performance > Instance Details** page for the process instance. Wait for a few seconds as the loading of the page might take some time. You review the Gantt Chart later in this exercise after completing some activities. For now, you review the model and where you are at in the process.

__ g. Click the **Diagram** tab for a model view of the same process instance.




__ h. The model view of the process instance is displayed.

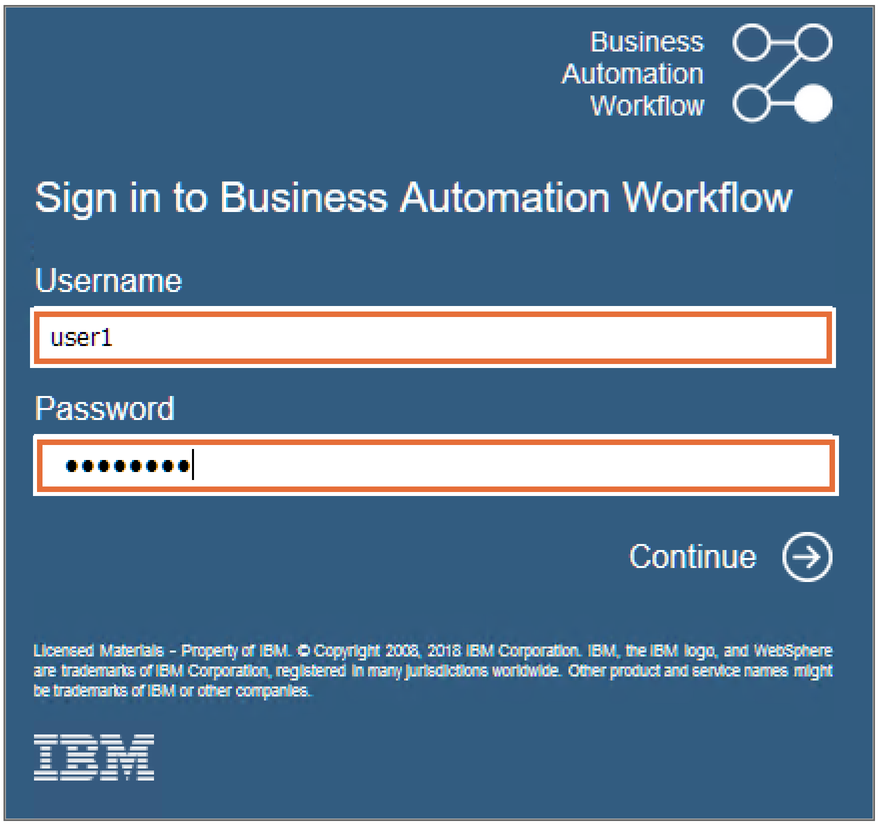


This view shows the process with the Approve New Hire Request task that is highlighted for the **Review needed** path, and in the next part of this exercise you work on the **Review not needed** path.

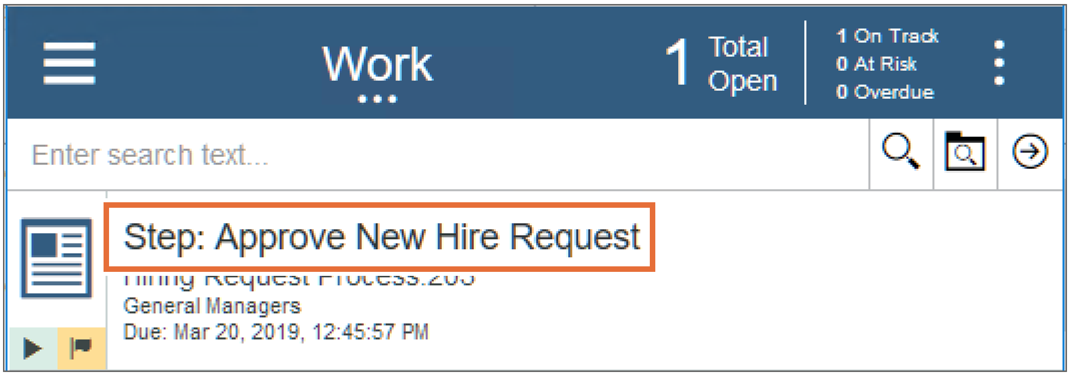
__ 4. Complete the Approve New Hire Request activity.

__ a. In the library, click **Log Out** to log out of the Process Portal.

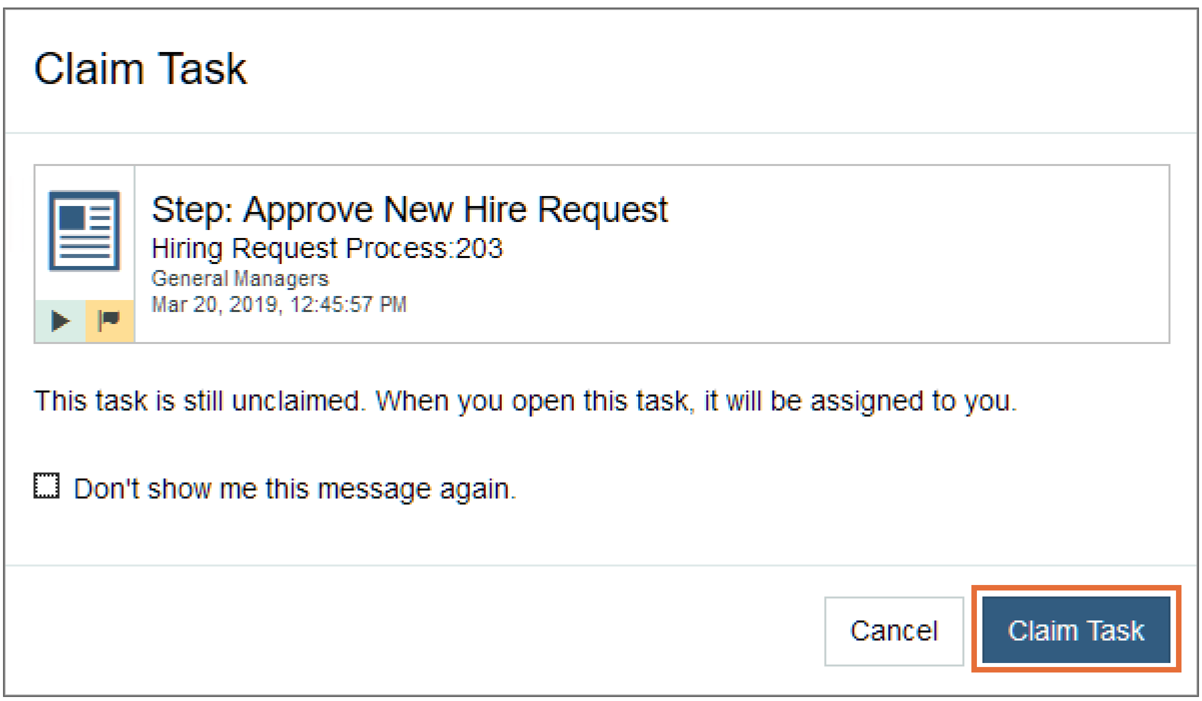

__ b. Log in to the Process Portal with the **Username** `user1`, **Password** `passw0rd`, and click **Continue**.



__ c. A task is in the inbox for the Hiring Request Process process. The task stopped at the Approve New Hire Request activity. Notice that the task is assigned to the General Managers team, of which `user1` is a member. Click the **Step: Approve New Hire Request** task. Depending on the amount of time spent, your task might be either On Track, At Risk, or Overdue.

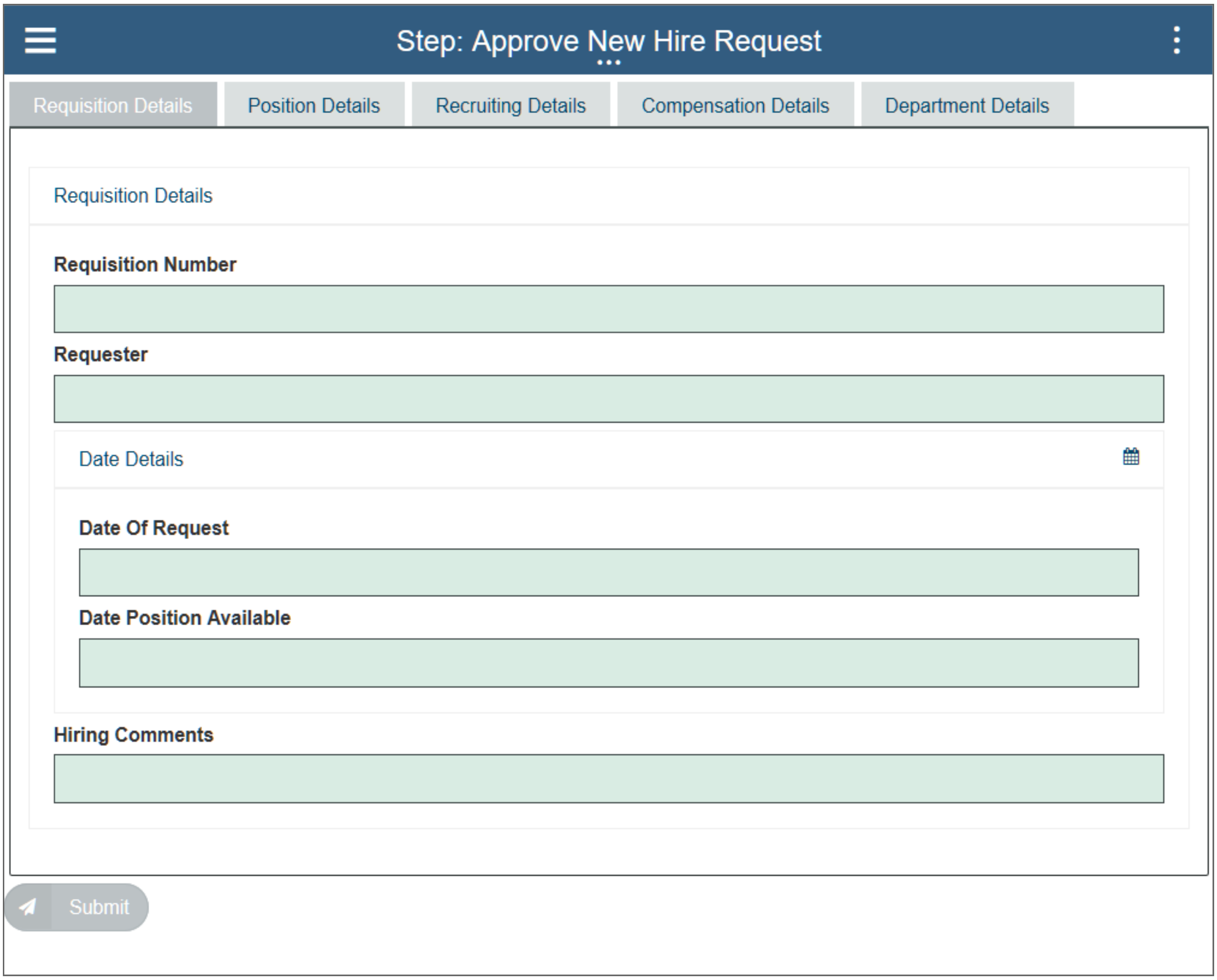



__ d. Click **Claim Task**, and the task is assigned to you.




__ e. A human service coach is displayed for the Approve New Hire Request activity.

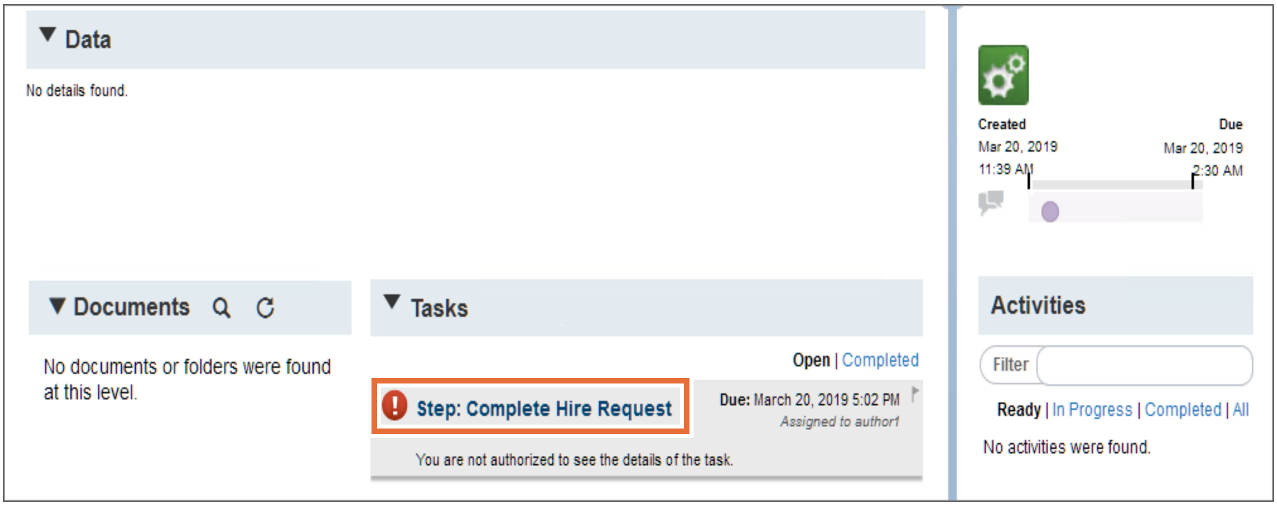
__ f. Drag the size of the browser window to display all tabs.




__ g. Enter any data in the fields and complete this task by clicking **Submit** in the human task service.

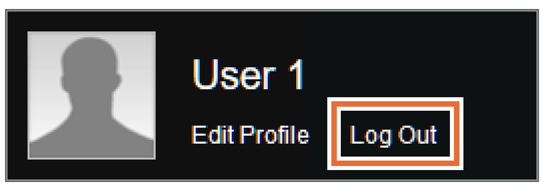
__ h. Click **Processes** in the library.

__ i. Click the **Hiring Request Process** instance with the same instance ID you worked on in the last step. The next task is assigned to `author1`. Click the task **Step: Complete Hire Request** and you get a message that states “You are not authorized to see the details of this task.”




This would be the place to start the discussion around task status tracking. This task shows as overdue due to the 1 minute time-out you configured earlier. If the task is not overdue, wait one minute and refresh the page (**F5**).

__ j. Log out of the Process Portal.




**Note**

If Log Out is not displayed, click the **Main Menu** icon on the left side.

## *Part 3: Demonstrate the Review Not Needed path*

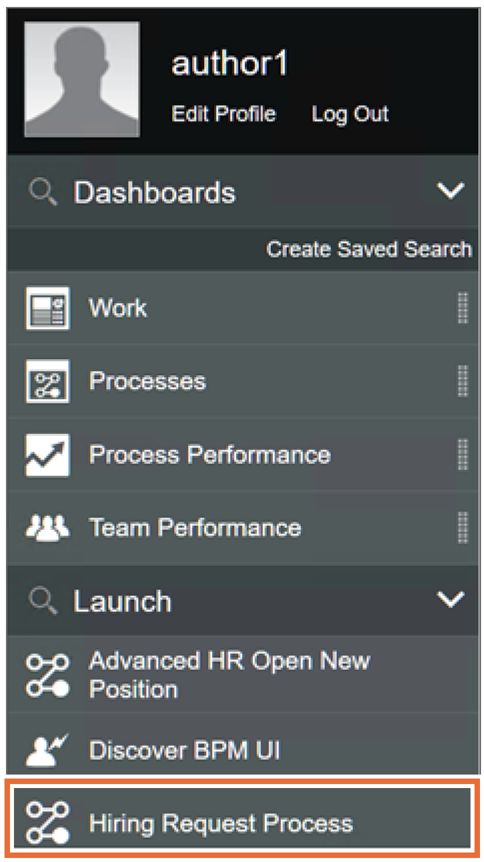
Now that you demonstrated the **Review Needed** path, this next part of the playback demonstrates the **Review Not Needed** path. It follows the same approach where you simulate the real-world experience of the different users inside the process as they log on to the portal and complete their tasks inside the process.

__ 1. Show the process as it follows the `Review Not Needed` path, and create an instance to test the “existing position” scenario.

__ a. Log in to the Process Portal with `author1` as the **username** and `passw0rd` as the **Password**.

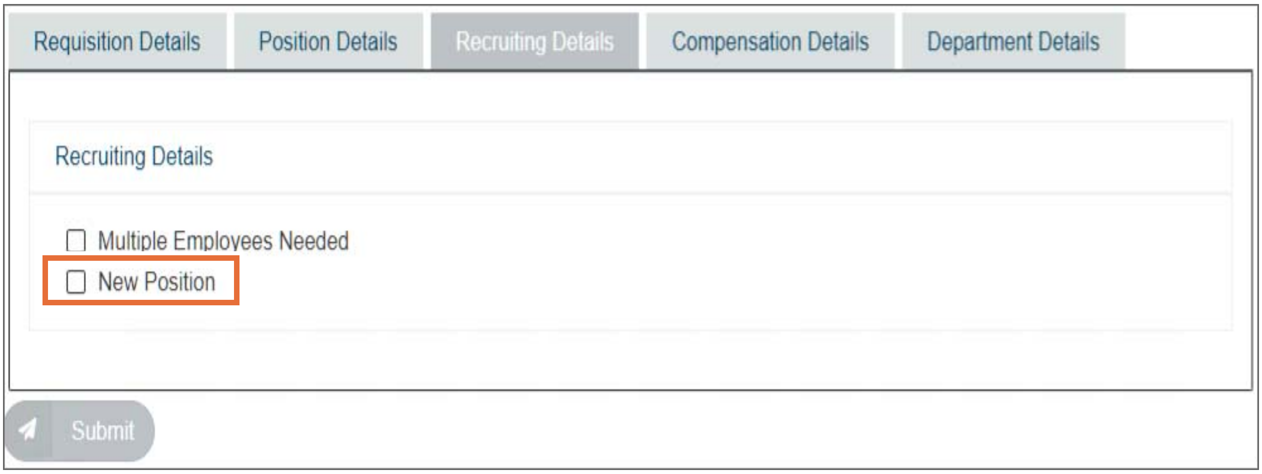
The incomplete step (Complete Hire Request) from the prior path shows up in the Work page.

__ b. Create an instance of the **Hiring Request Process** by clicking the link in the Launch section.




__ c. You immediately see in a dialog box that you received a new task. A window is displayed with the human service coach.

__ d. Click the **Recruiting Details** tab, and ensure that the **New Position** checkbox is cleared so that it follows the Review Not Needed path.



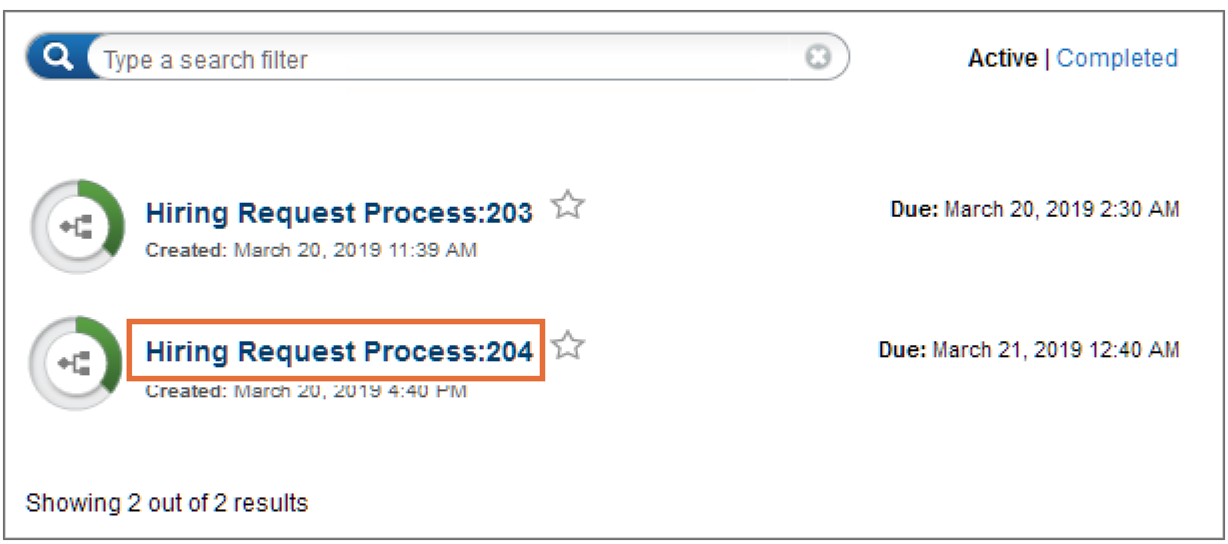

__ e. Enter any data for the other fields and click **Submit**.

You now have two items on your Work page.

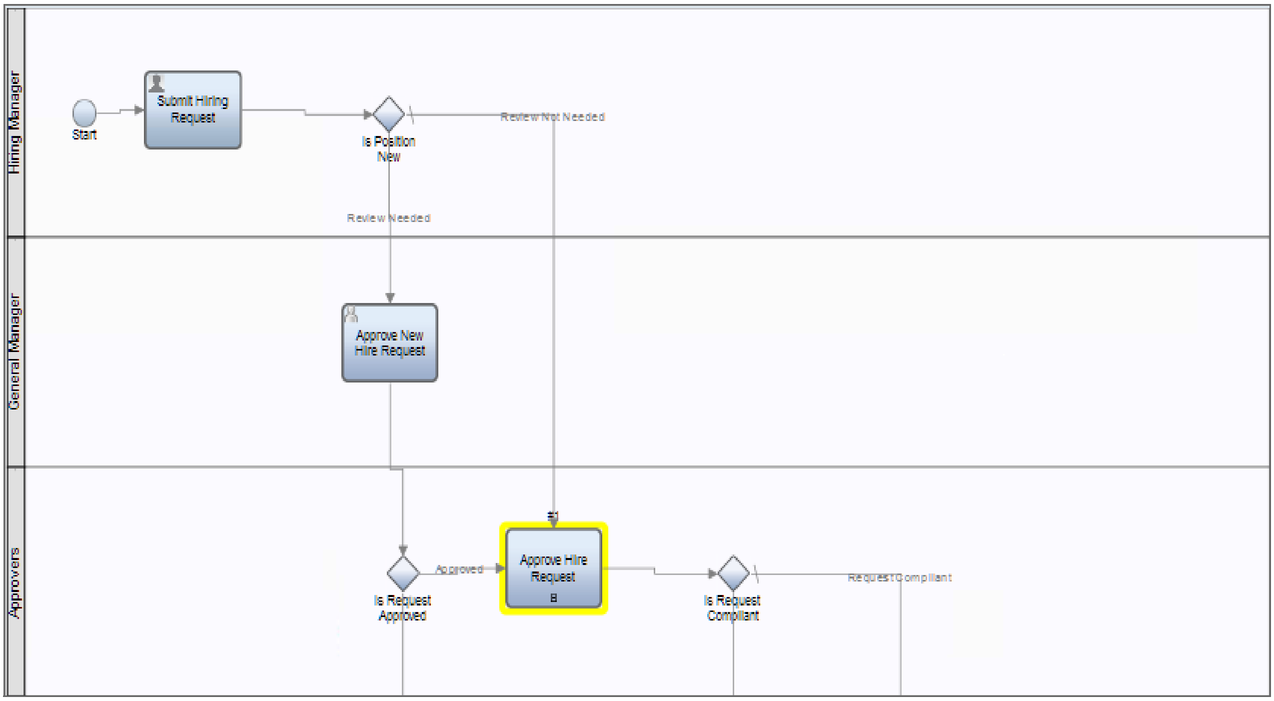
__ 2. Complete the process.

__ a. In the library, click **Processes**.

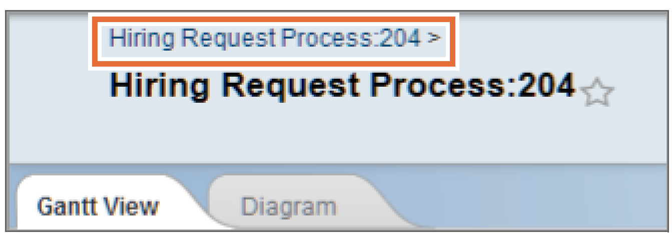
__ b. Click the process with the highest process instance ID. Your process instances are highlighted in green, yellow, or red to reflect their due date status (On Track, At Risk, Overdue).



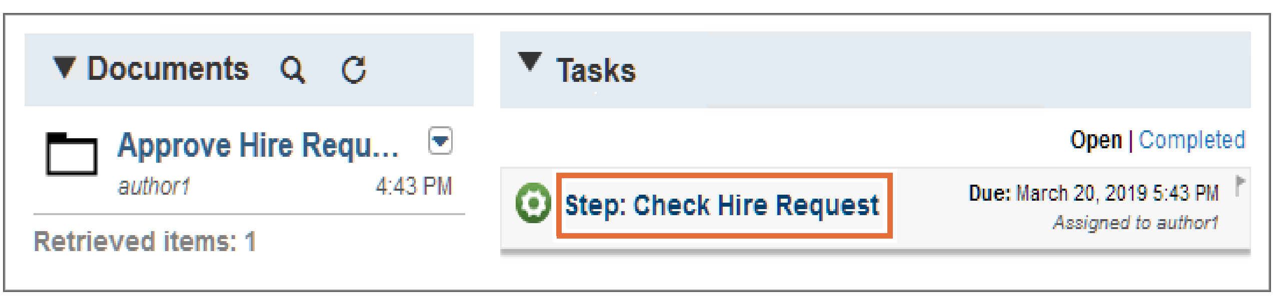
__ c. This time, the process follows the `Review Not Needed` path. Instead of passing to **Approve New Hire Request**, the process steps through the **Approve Hire Request** linked process. Click **GANTT CHART**, and then click **Diagram** to view the process diagram.



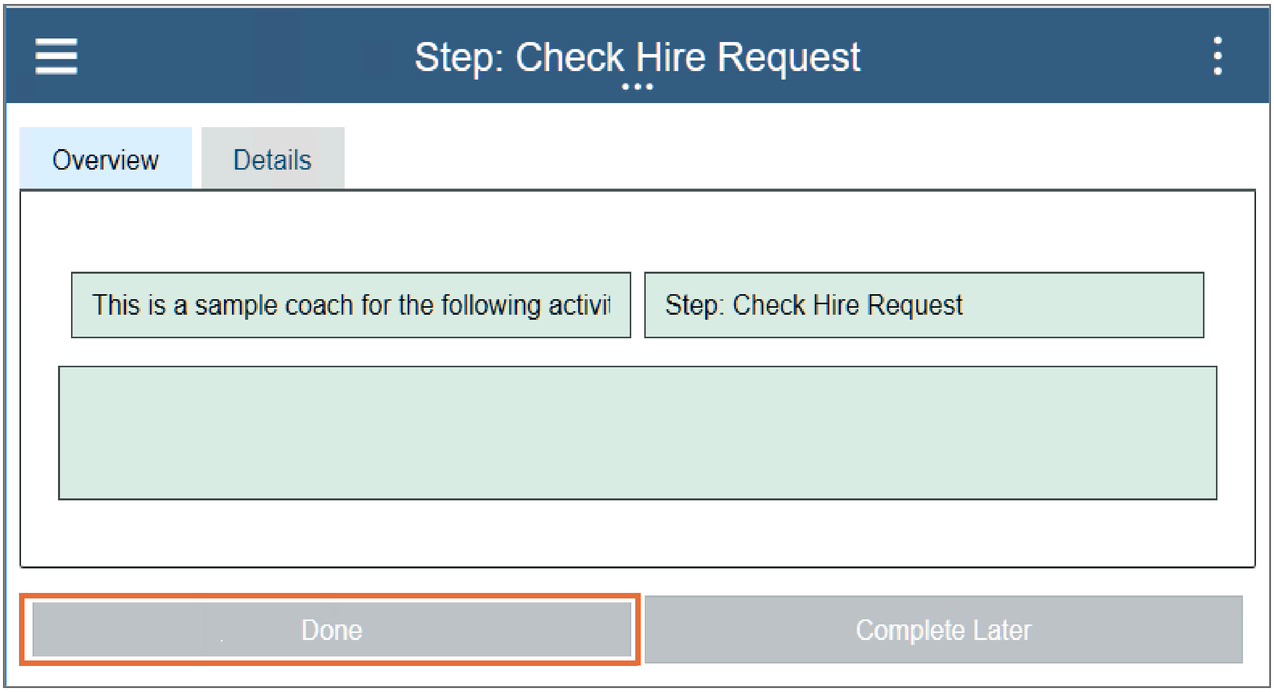

__ d. Click the link at the top to return to the Task.



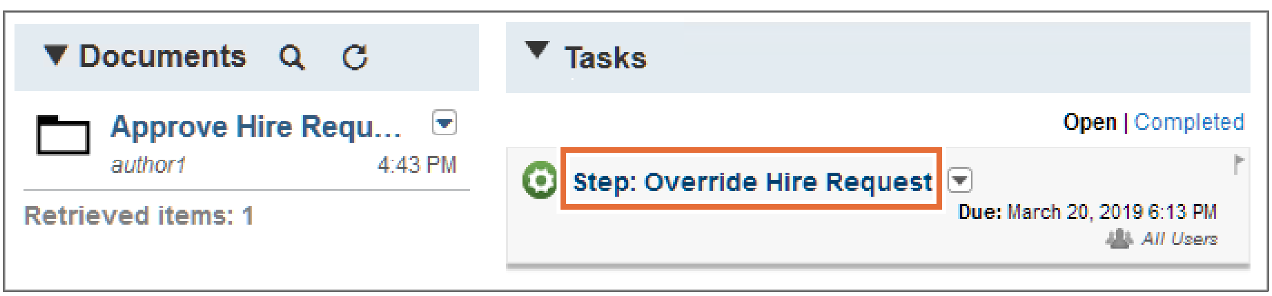

__ e. Click the **Step: Check Hire Request** task link. Recall this is the first task for the linked process.



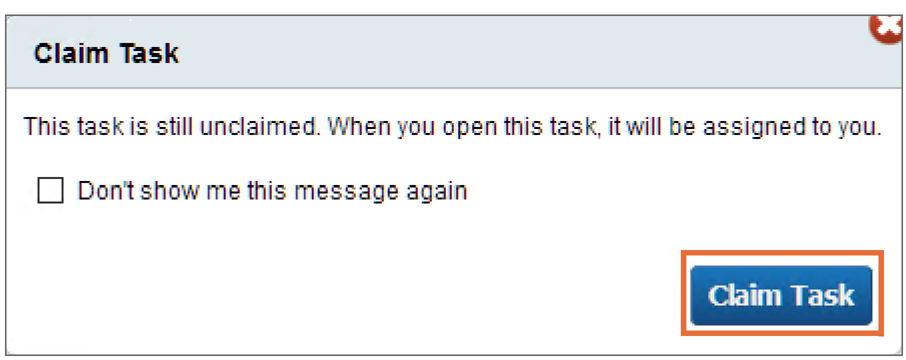
__ f. You are presented with the default user interface because you did not create a coach for this task. Click the Details tab to view details for the process instance. Click **Done** to complete the task.




__ g. Now the process moves to the next task. Click **Step: Override Hire Request**.



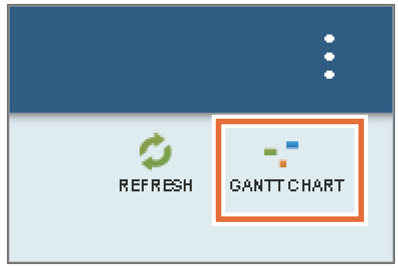

__ h. Click **Claim Task**.



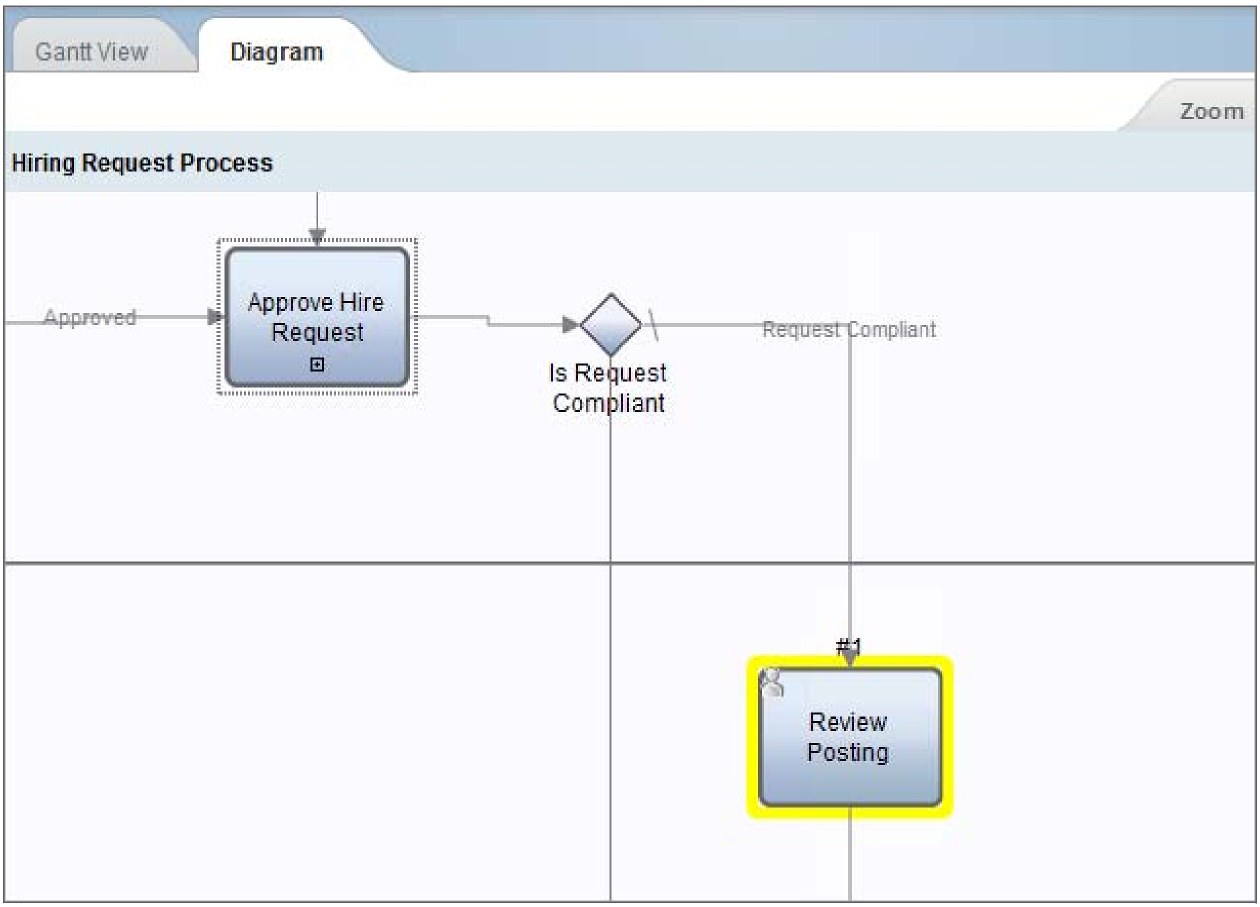

__ i. Click **Done** to complete the task.

__ j. The process now moves to **Step: Review Posting**.

__ k. Click the **GANTT CHART** icon on the upper-right corner of the window.




__ l. Click the **Diagram** tab for a model view of the same process instance.




This time, the process follows the **Review Not Needed** path. Instead of passing to **Approve New Hire Request**, the process stepped through the **Approve Hire Request** linked process, through the **Is Request Compliant** gateway, and then to the **Review Posting** activity. The process followed a different path from the first instance, which demonstrates that the **Is Position New** gateway and process flow data are working correctly when driven by business data on a coach. If the **Complete Hire Request** activity is active instead of the **Review Posting** activity, verify you configured the **Is Request Compliant** gateway correctly.

## *Part 4: Demonstrate activity tracking*

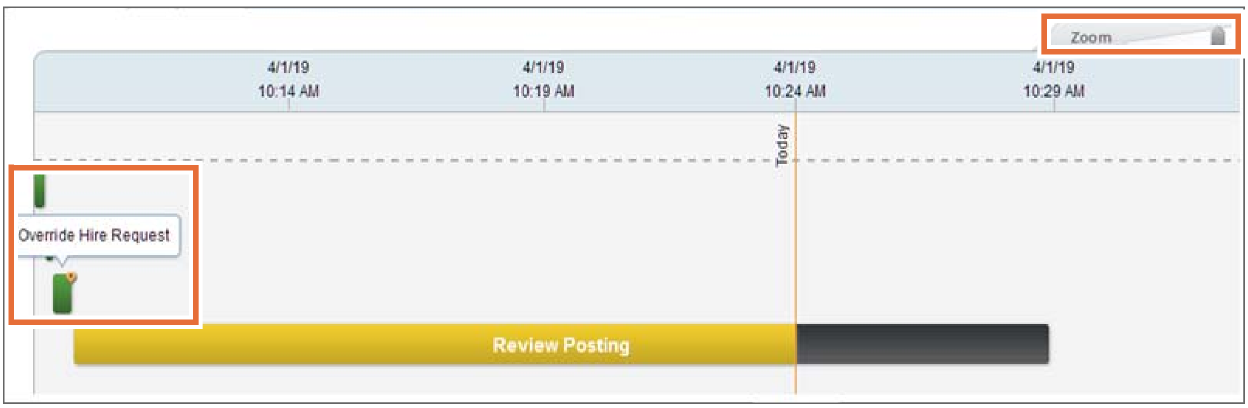
Now that the two processes have arrived at the activities that you changed settings for, you can continue the conversation around priority tracking and activity timeouts. It is a common practice to create mock scenarios when you perform playbacks to demonstrate specific functionality. In this case, you demonstrate how Process Portal flags activities that are highest priority and those that have passed their due date.

__ 1. Display the status of the Review Posting activity in the Gantt chart.

__ a. Click the **Gantt View** tab.

__ b. Use the Zoom control to zoom out so you can see the prior activities.

__ c. Mouse over the prior activity to display the name of the task.

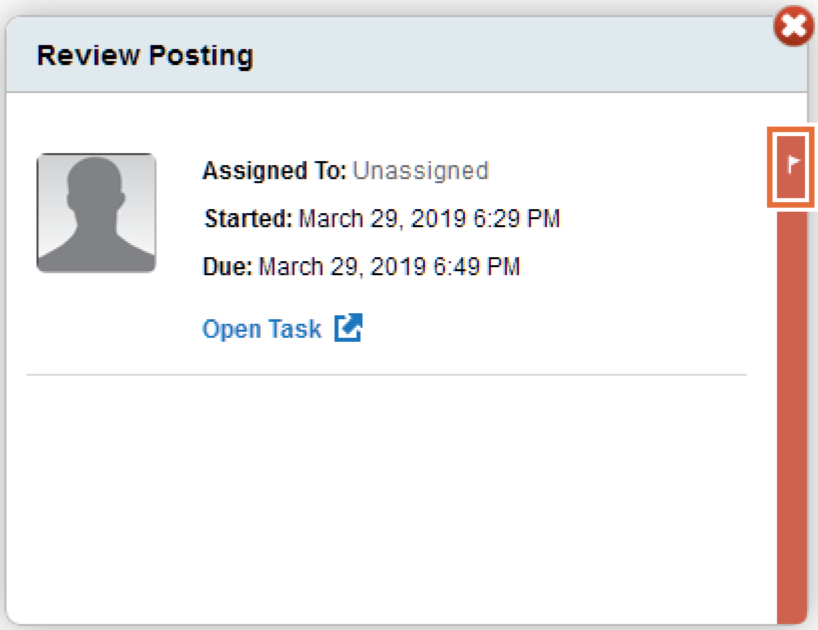



__ d. This chart displays the time that it took to complete each task. Your chart might look different depending on the amount of time that is spent on each activity. In the Gantt chart, four activities are displayed. Roll the mouse over the other three completed activities to display their names.

__ e. The first three activities were completed on time. The Review Posting activity is not complete. Depending on the amount of time spent, the Review Posting activity might also be past due. In the screen capture, the activity is displayed in yellow because it is at risk of becoming overdue. Click the **Review Posting** activity to view more details.

__ f. A pop-up window displays more details for the Review Posting activity.

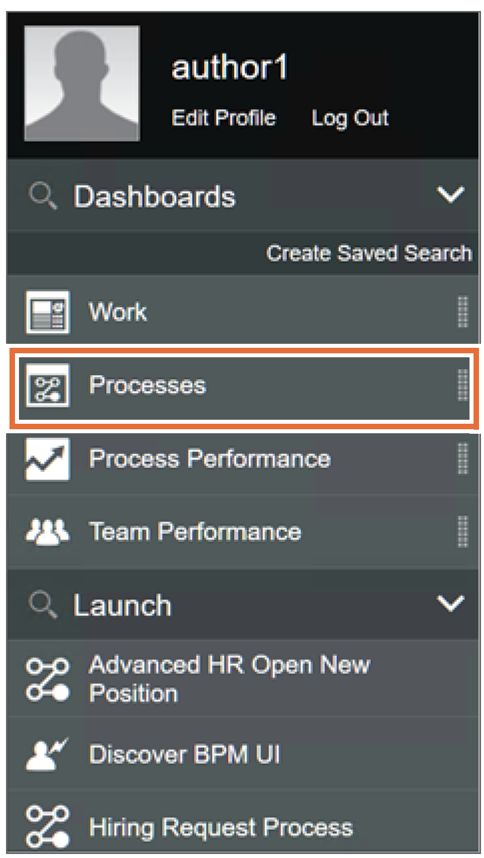
__ g. Roll the mouse over the flag. The setting that you configured earlier is displayed: **Very high**.




__ h. Click the red "x" to exit the window.

__ 2. Display the status of the Complete Hire Request activity in the Gantt chart.

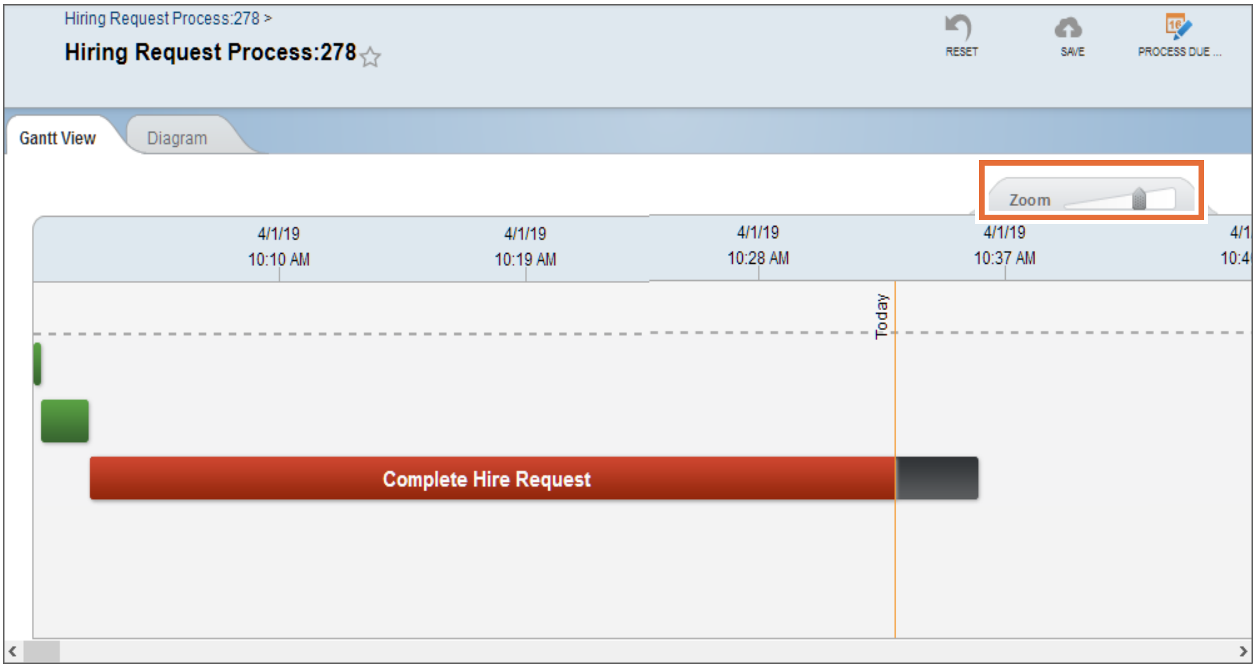
__ a. In the Library click **Processes**.



__ b. Click the earlier process (the process with the smaller process instance ID).

__ c. Click **Gantt Chart**.

__ d. Use the Zoom control to zoom out as necessary to view all activities.



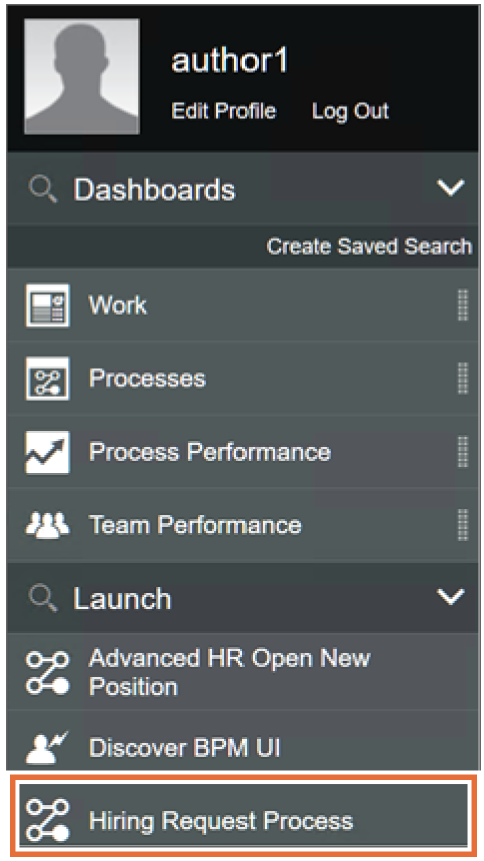

__ e. The Complete Hire Request activity is highlighted in red, depicting it is past due. Click the **Complete Hire Request** activity to view more details.

__ f. Roll the mouse over the flag. The default setting you configured earlier is displayed: **Medium**.

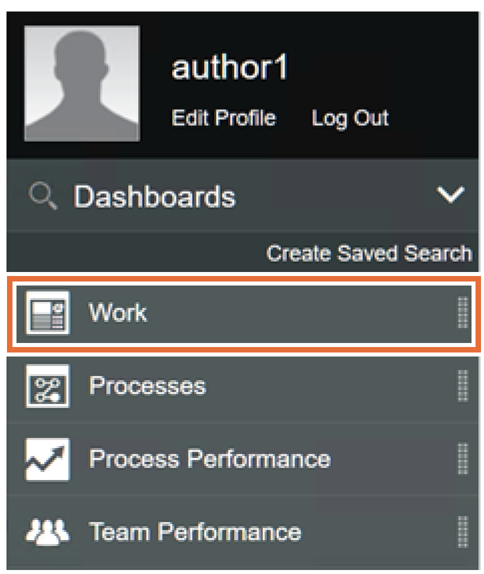
__ g. Click the red “**x**” to exit the window.

__ 3. Demonstrate the status of the activities on the Work page.

__ h. Click the **Main Menu** icon to return to the Library. To display an activity on track, create an instance of the **Hiring Request Process** by clicking the link in the Launch section.

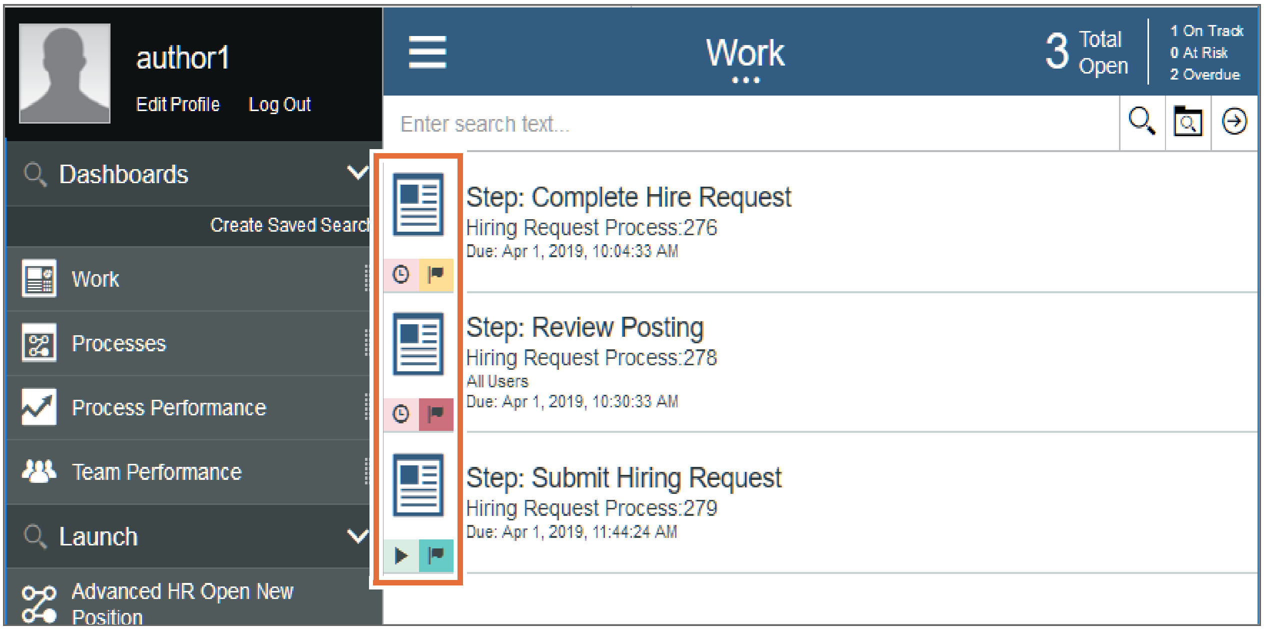



__ i. Do not complete the task. Click the **Main Menu** icon to return to the Library. Click **Work** to return to the Work page.




__ j. You now have three work items displayed.

__ k. Mouse over the priority and status icons to display their textual representation. For the purposes of the demonstration, there is no need to complete the activities. The project team agrees the priority and status functionality meets their needs. The next step is to create a toolkit, create a snapshot, and export the process application.




The status of your work items might vary depending on the timing.

**Reminder**

In this course, you did not implement all the coaches for the sake of brevity. In a real-world situation, one of the requirements to move to Playback 3 is that all the coaches are implemented in the process and all of them have the final presentation elements in place (CSS and JavaScript, for example).

## *Part 5: Reset the environment*

When demonstrating mock scenarios, it is important to reset the environment back to the defaults before moving on.

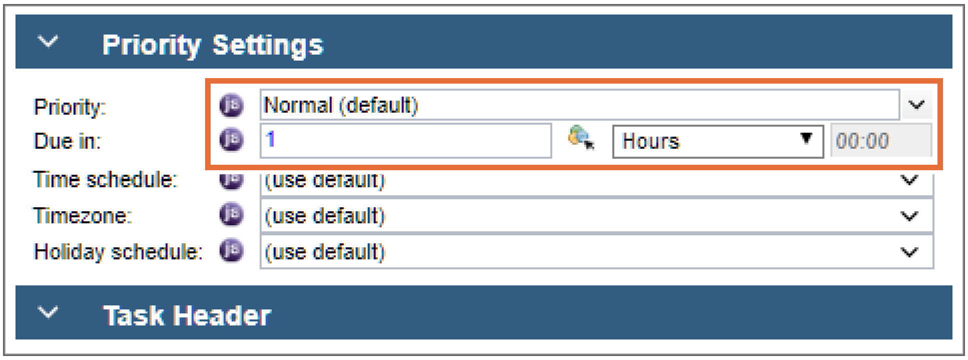
__ 1. Reset the timeout value for the Complete Hire Request activity.

__ a. Return to Process Designer and log in as `author1`.

__ b. Open the **Hiring Request Process** unless it is already open.

__ c. Click the **Complete Hire Request** activity.

__ d. In the **Priority Settings** section of the **Implementation** properties at the bottom, set the activity to be due in 1 hour.




__ 2. Reset the priority and timeout values for the Review Posting activity.

__ a. Click the **Review Posting** activity.

__ b. In the Priority Settings section of the **Implementation** properties at the bottom, select **Normal** for the Priority setting.

__ c. Set the activity to be due in 1 hour.

__ d. Save your work.

## *Part 6: Create the Hiring Requisition Toolkit and take a snapshot*

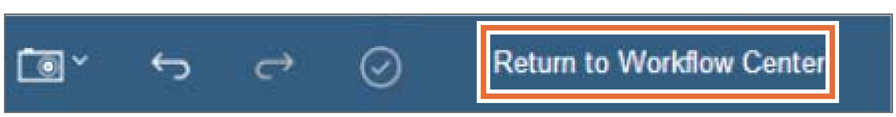
Now you have numerous assets, some of which might be useful to other developers. Create a toolkit from the assets you created as part of the process application HR Recruitment Processes.

A toolkit is a collection of assets that are shared between process apps or other toolkits during development. Attentive application of toolkits is an efficient method for managing the reusability and compartmentalization of project assets.

Starting in version 19.0.0.1, instead of duplicating effort creating and deploying two separate projects, you can use the enhanced Workflow Center user interface to put the process and case implementation in a workflow project that you can deploy as a single solution. You can also create, manage, and deploy your projects (such as process apps, case solutions, and toolkits) here.

__ 1. Create the Hiring Requisition Toolkit

__ a. Click **Return to Workflow Center**.

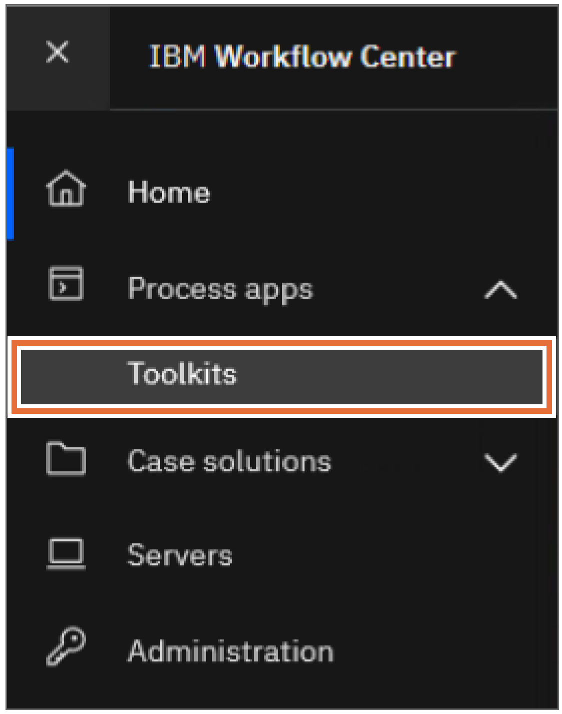



**Hint**

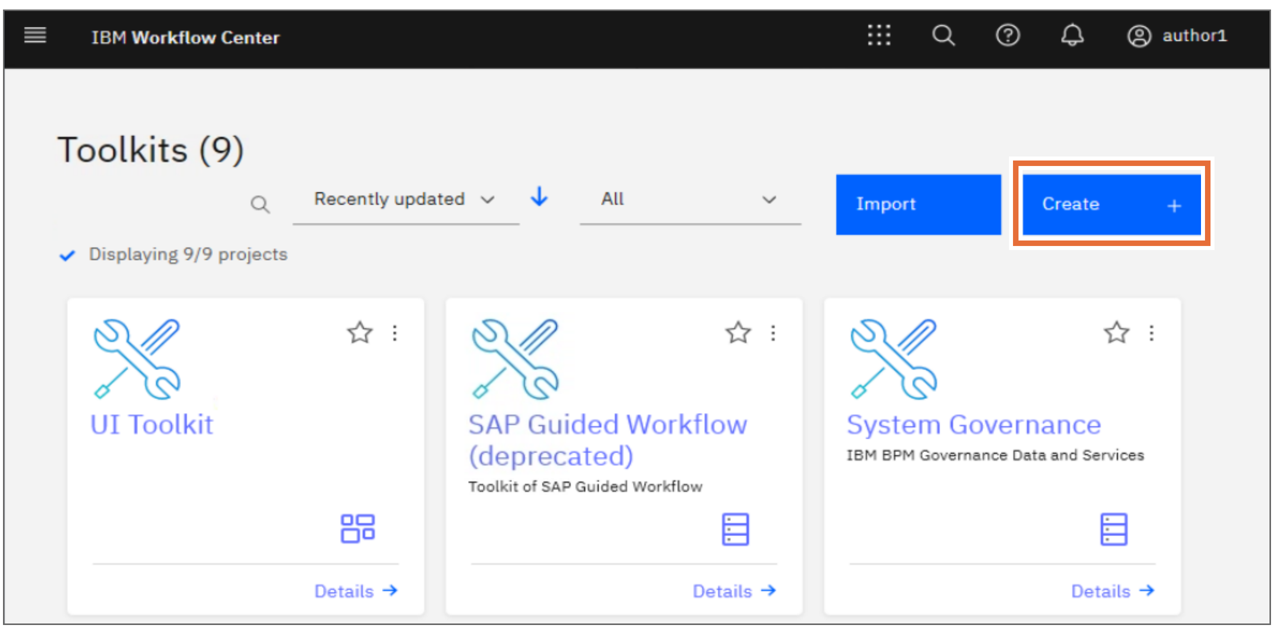
The **Return to Workflow Center** link returns you to the Workflow Center you originated from. If you opened the process application from the Classic Workflow Center, this link returns you there.

__ b. On the Workflow Center main page, click the navigation icon in the upper left and expand the **Process apps** section.

__ c. Click **Toolkits**




__ d. On the Toolkits page, click **Create**.




__ e. Enter the name for the toolkit: `Hiring Requisition Toolkit`

__ f. Verify that the Acronym is automatically created: `HRT`

**Troubleshooting**

If the Acronym control is not displayed, verify you are using the Advanced view mode.

__ g. Clear the **Open in Process Designer** checkbox if it is checked.

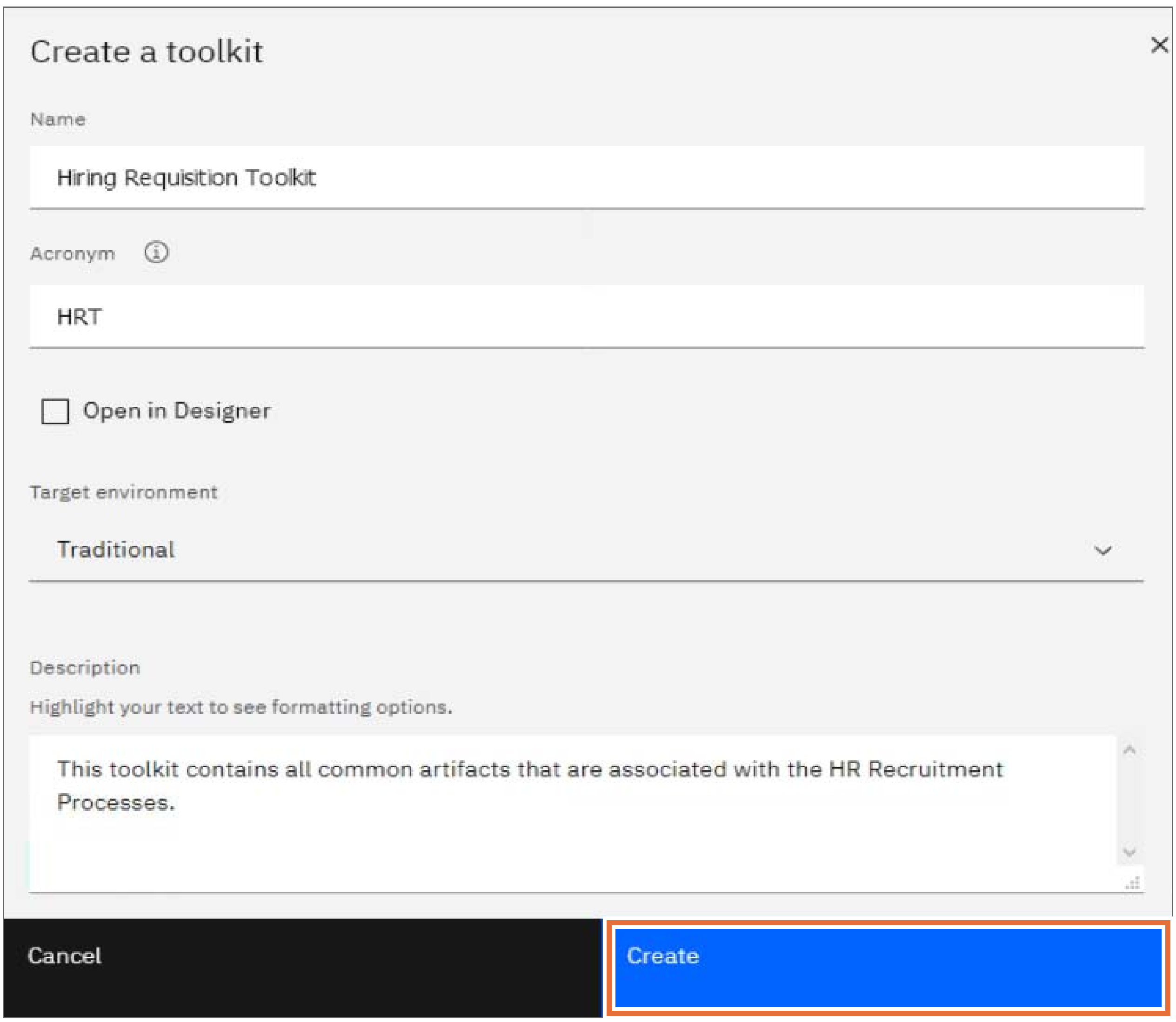
__ h. Select **Traditional** for the Target environment.

__ i. Enter a brief description: `This toolkit contains all common artifacts that are associated with the HR Recruitment Processes`.

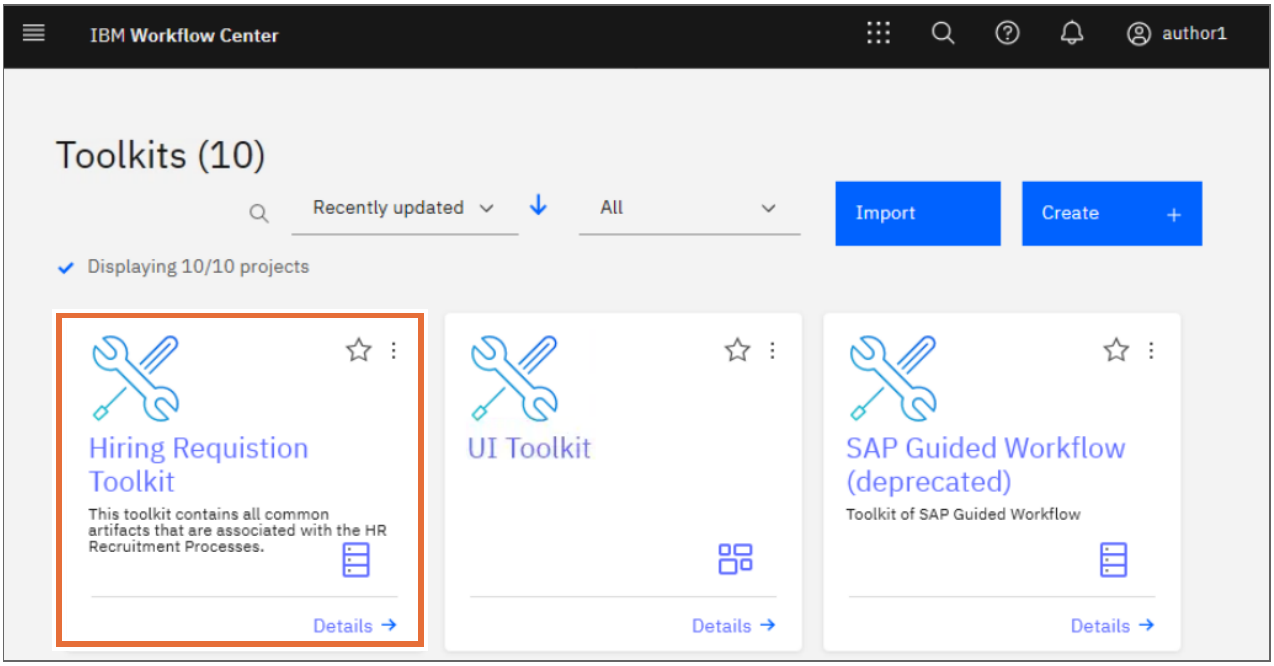
**Hint**

While completing the form, be sure not to click outside of it. Once the focus is taken away, the form is withdrawn. If this happens, click **Create** again to initiate the dialog.

__ j. Click **Create**.



__ k. Verify that the Hiring Requisition Toolkit shows up as a tile in the **Toolkits** list.



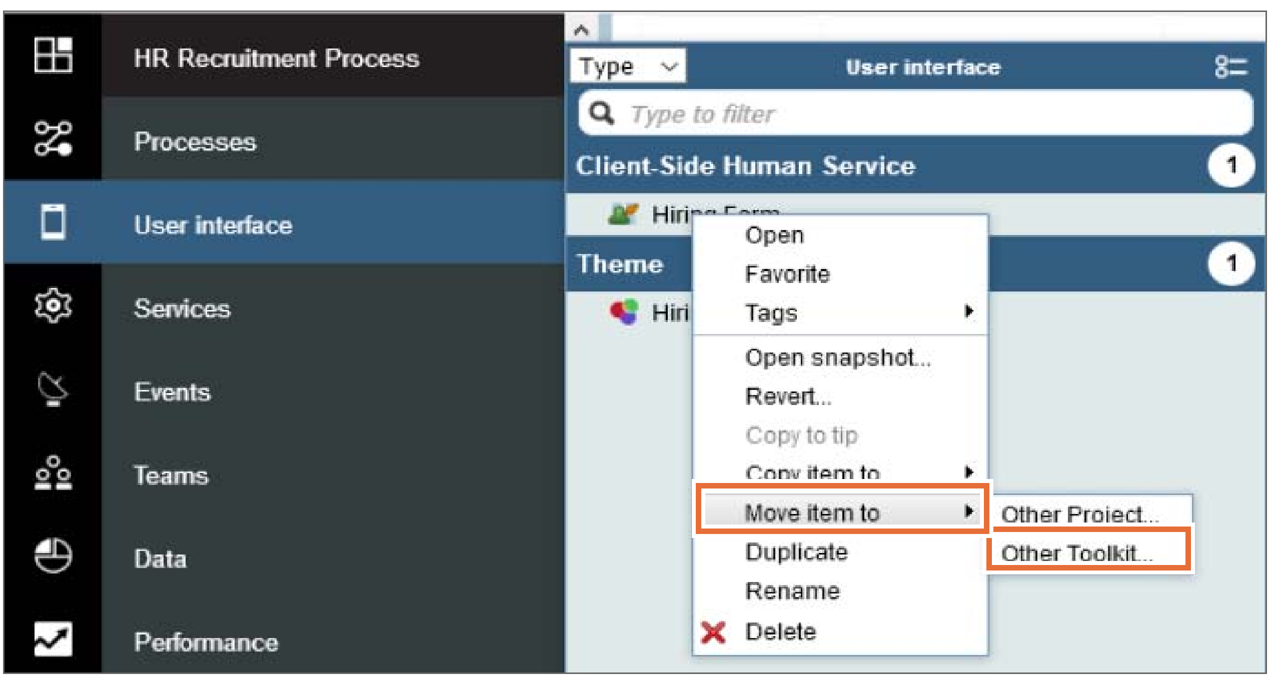

__ 2. Move the client-side human service into the toolkit.

__ a. On the Workflow Center main page, click the navigation icon in the upper left and click **Process apps**.

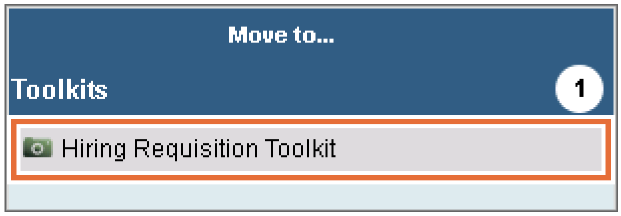
__ b. Click **HR Recruitment Process**.

__ c. In the library, right-click the **User Interface > Hiring Form** client-side human service.

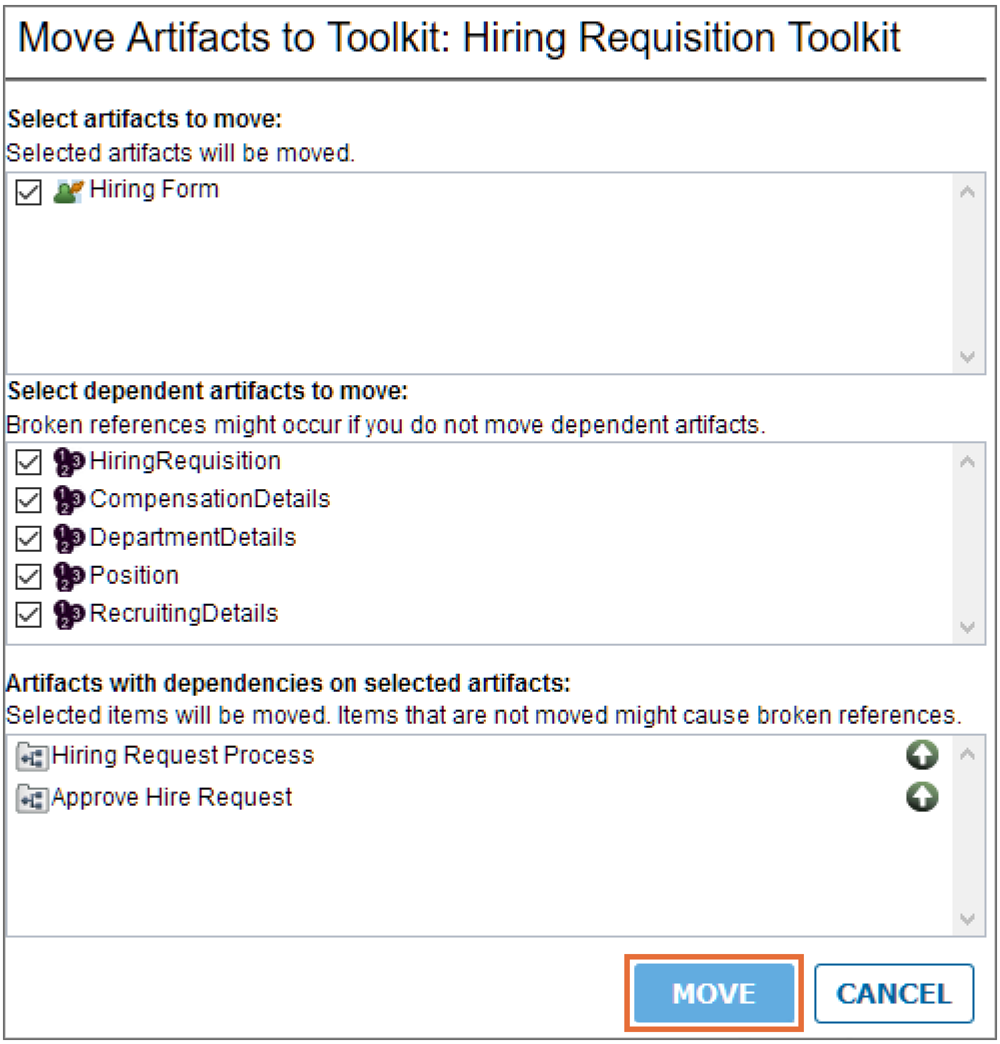
__ d. Click **Move Item to > Other Toolkit**.



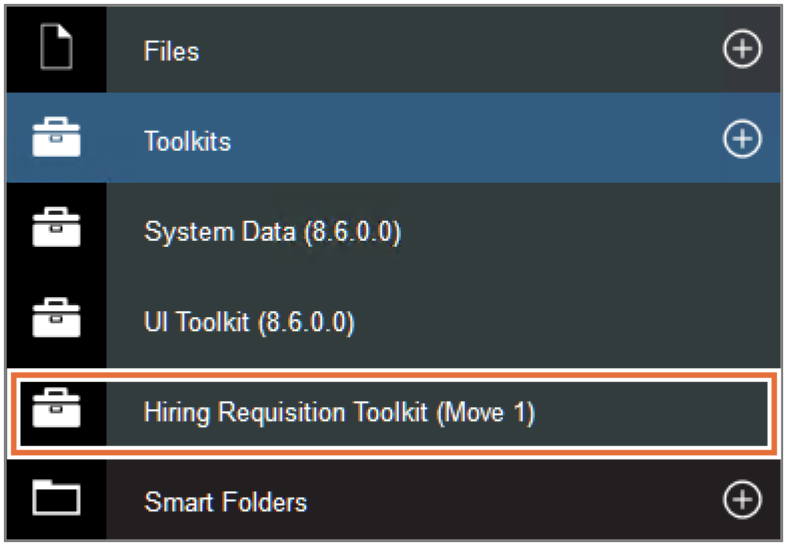
__ e. Select **Hiring Requisition Toolkit**.



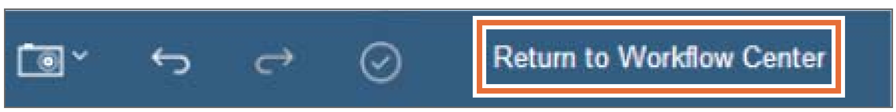

__ f. Keep the defaults and click **MOVE**. The order of the dependent artifacts might be different than in the screen capture.



__ g. All the dependent assets are also moved into the toolkit to include the variables that the client-side human service depends on. A dependency on the toolkit is automatically created inside the process application. In the Library section, expand **Toolkits** to see the Hiring Requisition Toolkit.



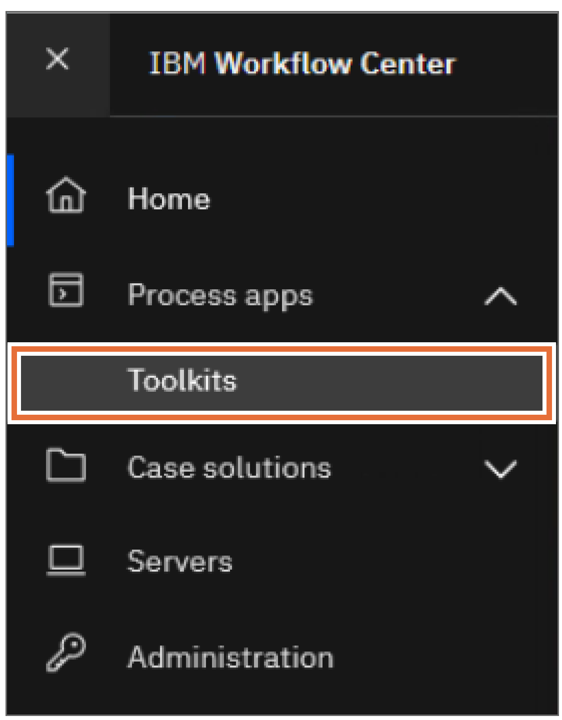

__ h. Return to the **Workflow Center**.



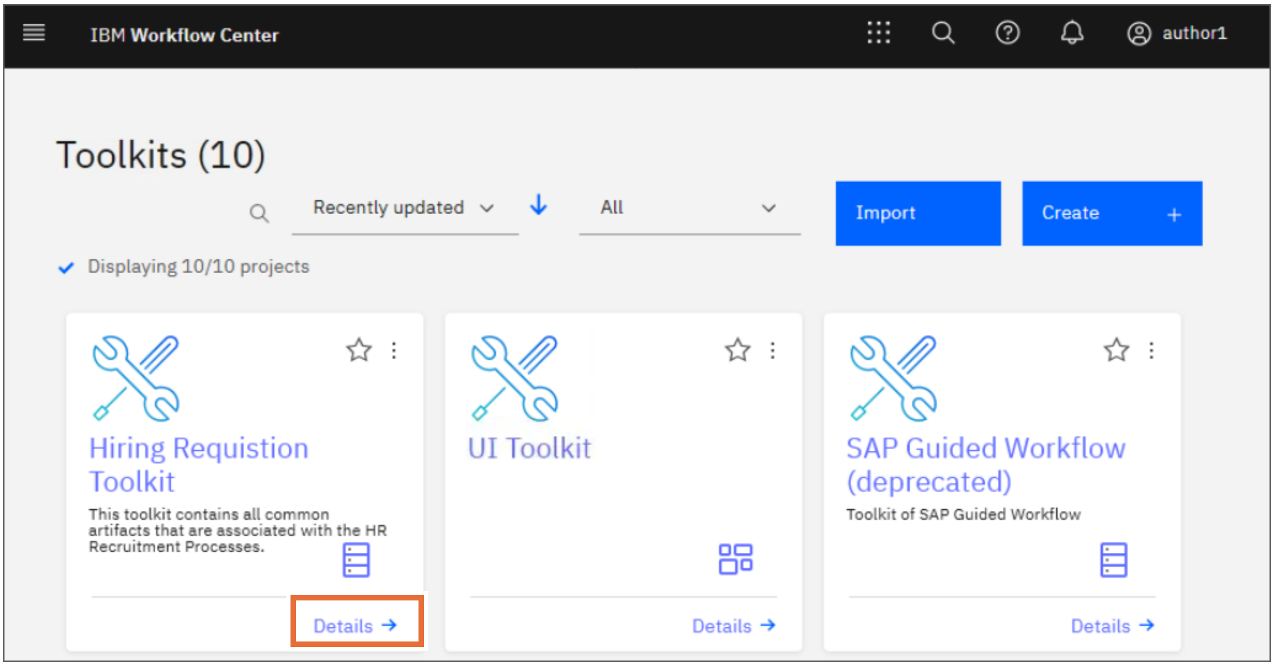

__ 3. Take a snapshot.

__ a. On the Workflow Center main page, click the navigation icon in the upper left and expand the **Process apps** section.

__ b. Click **Toolkits**

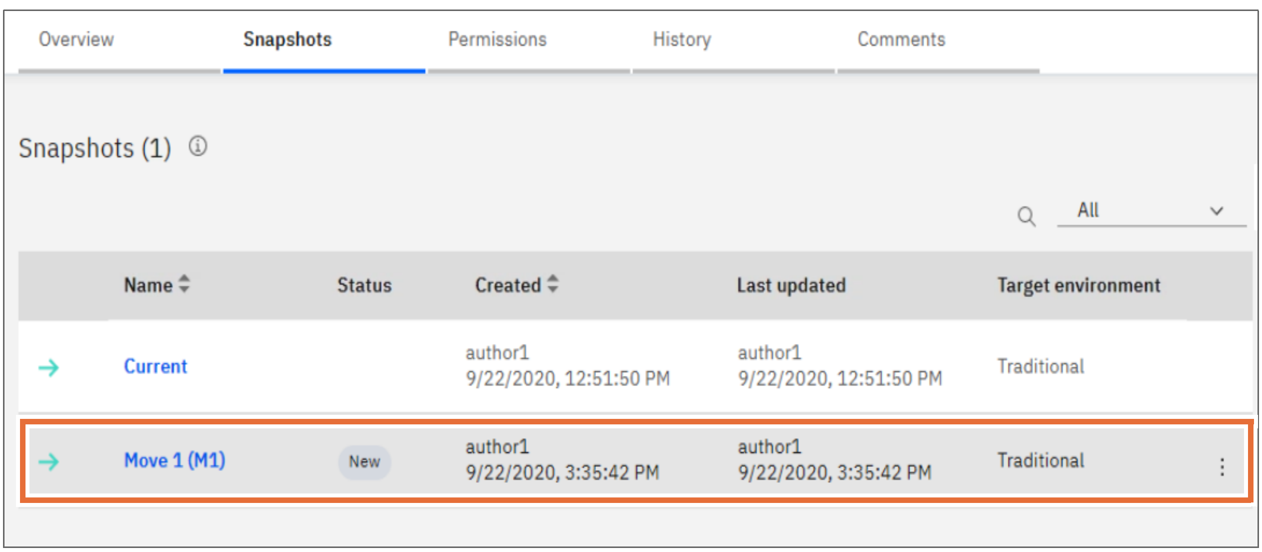


__ c. Click **Details** on the Hiring Requisition Toolkit tile.

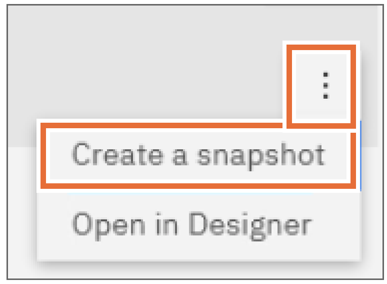



__ d. Select the **Snapshots** tab.

__ e. You notice that a snapshot has already been created. This was done automatically when you moved the assets to the Toolkit.

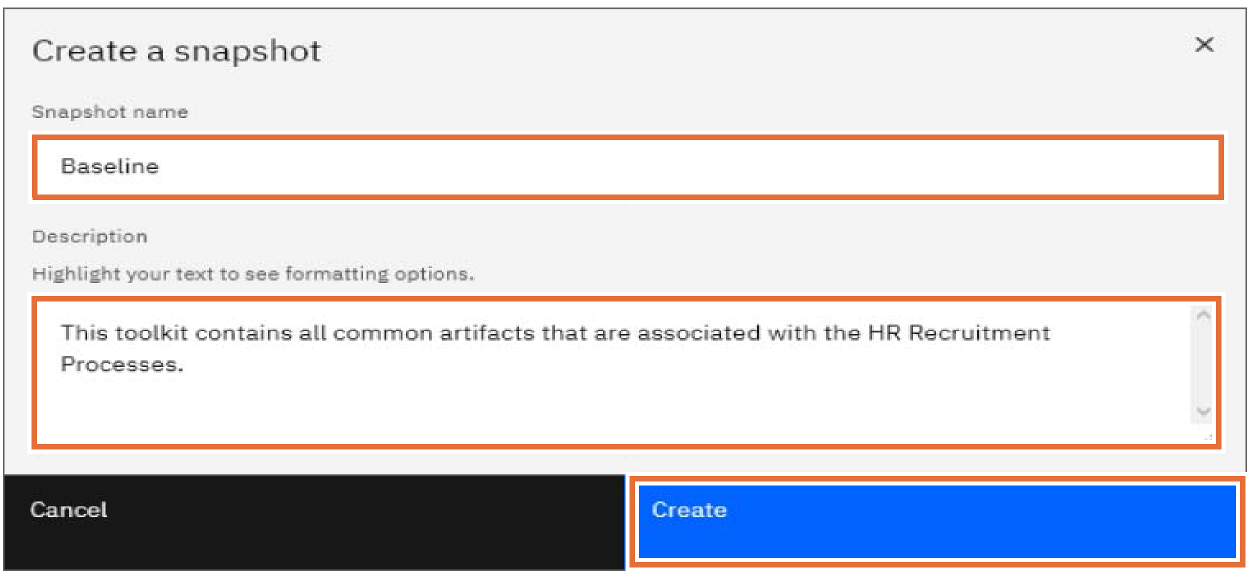


__ f. Open the list of options by clicking the **three dots** on the right of the **Current** snapshot tile and select **Create a snapshot.**

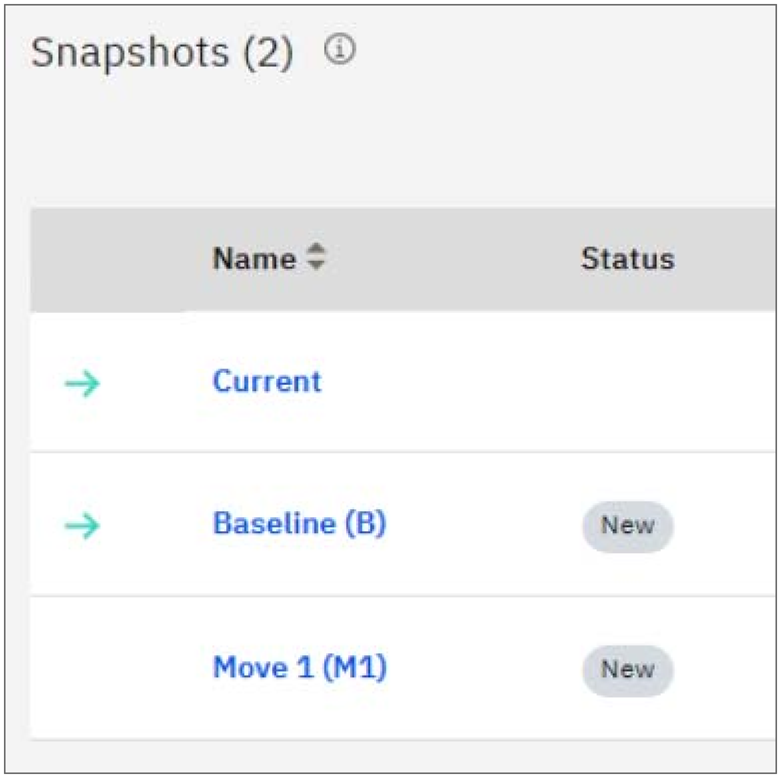



Recall that a snapshot captures the state of the library items within a process application or toolkit at a specific point in time. Snapshots usually represent a milestone or are used for playbacks or for installation. Since Playback 1 is now complete, it is a good time to take a snapshot.

__ g. Name the snapshot `Baseline` and add a description: `This is the baseline snapshot of the Hiring Requisition Toolkit.` Then, click **Create**.



__ h. The snapshot **Baseline** is created.




__ 4. Update the toolkit version on the process application.

Design toolkit hierarchies to avoid mutual dependencies by factoring out common content into its own toolkit so that the other toolkits can refer to it independently.

If you update the dependency on a child toolkit, you need to take a new snapshot of the parent toolkit (or process application) for the change to be effective for whoever uses that toolkit. This can lead to a situation with mutually dependent toolkits.

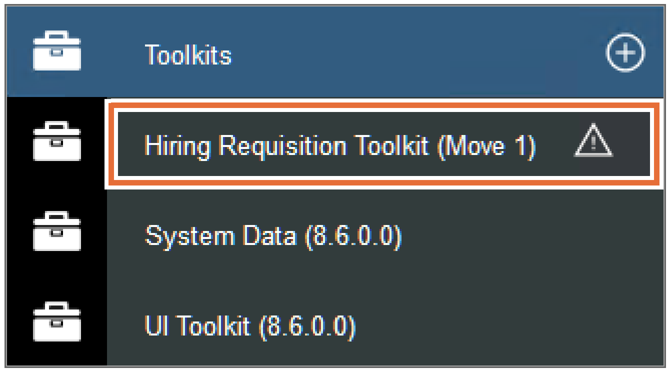
__ a. Click the navigation icon in the upper left and select **Process apps**.

**Troubleshooting**

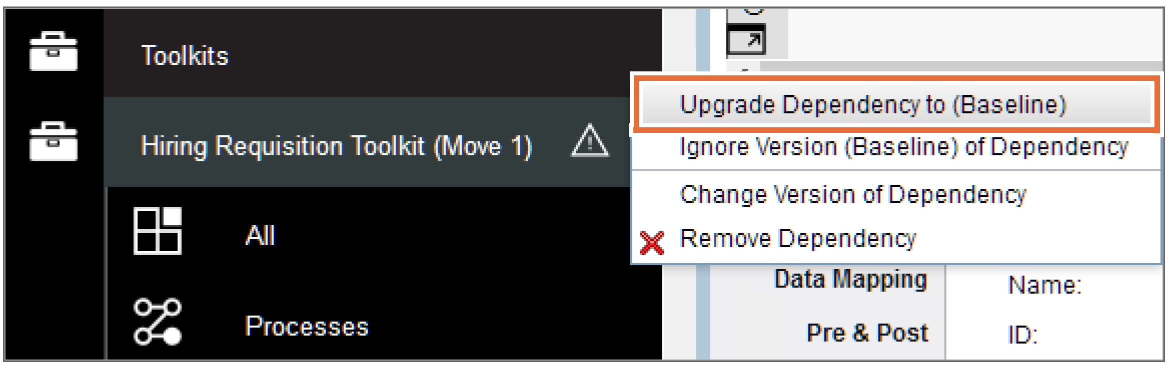
If the Process Apps page displays the Toolkits, wait approximately a minute for the page to refresh.

__ b. Open the **HR Recruitment Process** process application.

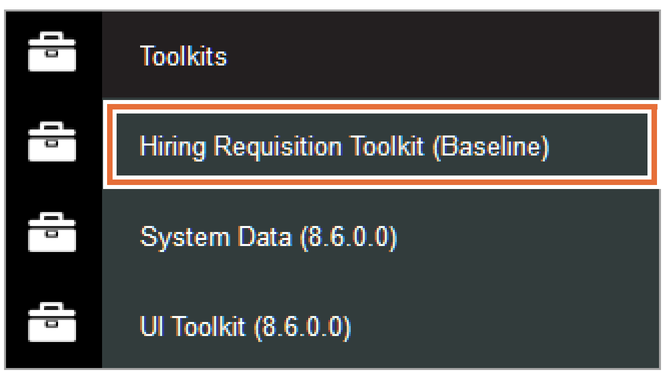
__ c. When you return to the process application, Expand the **Toolkits** category in the Library section. The `Hiring Request Toolkit` shows a warning because the process application is not pointing to the most recent snapshotted version.




__ d. Click the **arrow** next to the Toolkit and select **Upgrade Dependency to (Baseline)**.




__ e. In the Toolkits category, verify that the Hiring Requisition Toolkit (Baseline) is now linked to the HR Recruitment Processes process application and that the warning is gone.




**Information**

Because you created a toolkit and created a dependency, each solution file you import from this point in the training will include the Hiring Requisition Toolkit.

You have completed this exercise.

In this exercise, you completed Playback 1. You conducted a playback session that demonstrates two unique process scenarios. You used the Process Portal to demonstrate the playback from the user point-of-view. After completing Playback 1, you created a snapshot for the toolkit and process application and then exported the process application.

This completes Playback 1. The process, data, and coaches have now all been approved to move on to the next playback. Once all of these components are verified, concentration can be paid to integration with other systems, which is covered in the next playback.

## End of exercise

# Exercise 8. Playback 2: Integrations

## Estimated time

01:30

## Overview

This exercise covers how to create implementation assets to support Playback 2: integrations.

## Objectives

After completing this exercise, you should be able to:

- Create a decision service
- Create and configure a UCA
- Start a process with a message start event
- Use tagging to organize assets
- Query a database to obtain information and populate a list variable
- Create environment variables (ENVs) and exposed process variables (EPVs)
- Change a text control to a single select control

## Introduction

In Playback 2: Integrations, you implement the process interactions and integrations so the process can have all the functions that are needed to complete any process activity. Not all the functions are developed within the process application when it comes to user interactions. Some of the business data can be found, for instance, in systems within an organization. So it is important to be able to retrieve the data, and that is why integrations play a vital role in having a full and robust process application.

Other process interactions involve events within the process model. In this stage of Playback 2: Integrations, you implement those system activities that set business data and process flow data with rules and automations. These intermediate events or start events might need unique event handlers, such as listeners for messages that trigger an event. In Playback 2, you implement, test, and finalize all remaining process application interactions.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

# Exercise instructions

## *Part 1: Create a decision service*

The Hire Request business process is a candidate to include a decision service in the process flow. Decision services allow the business process to make routine decisions that are based on real business rules to speed up the process and eliminate user error.

If the salary is not compliant, the criteria that they must use is to route hiring requisitions for administrator override. The salary compliance depends on the job level. Different job levels have different compliance ceilings. In the as-is process, a user decides. In the to-be process, a user was going to decide, but now they want the system to decide based on business rules.

Your task is to implement that decision service in a process. You created the process model during Playbacks 0 and 1, and in Playback 2, you implement all the services.

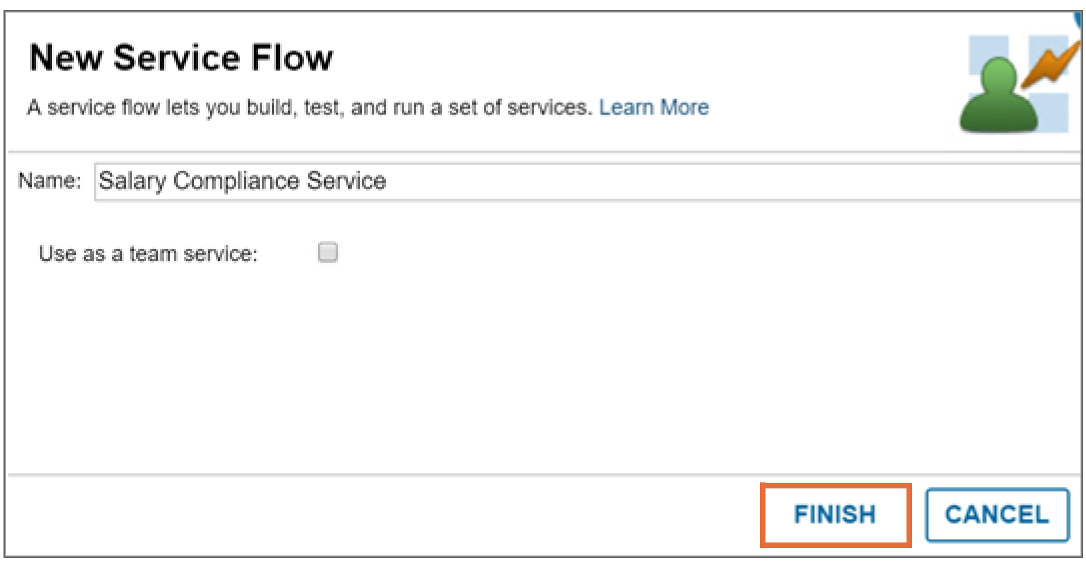
This exercise is long, so a break is included in the middle of the instructions. The instructions include some swapping back and forth and moving around different assets in the library. The sequence is intentionally set to simulate the steps that a developer takes when creating these assets, and reflects real-world development efforts.

__ 1. Create a decision service.

__ a. Open the **HR Recruitment Process** process application.

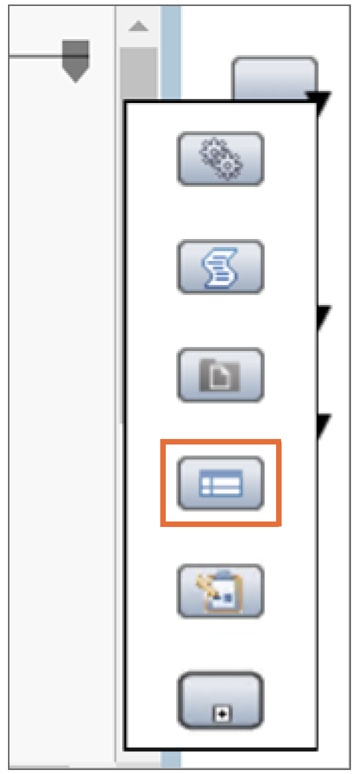
__ b. In the library, click the (**+**) plus sign next to Services. Click **Service Flow** to create a service.

__ c. Name the service `Salary Compliance Service` and click **FINISH**.

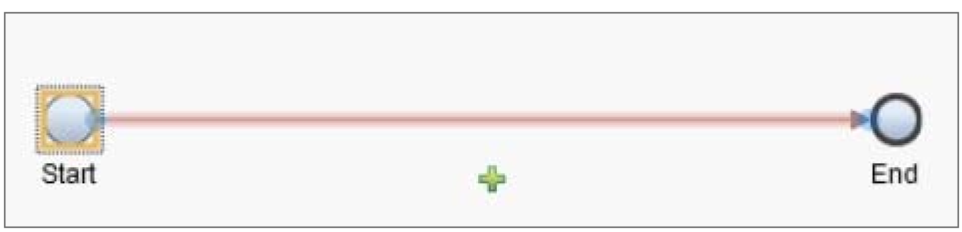


__ 2. Add a Business Action Language rule.

__ a. The service is opened to the Diagram tab. Expand the top activity and drag a **Decision** activity from the service palette onto the sequence flow between the Start and End events.

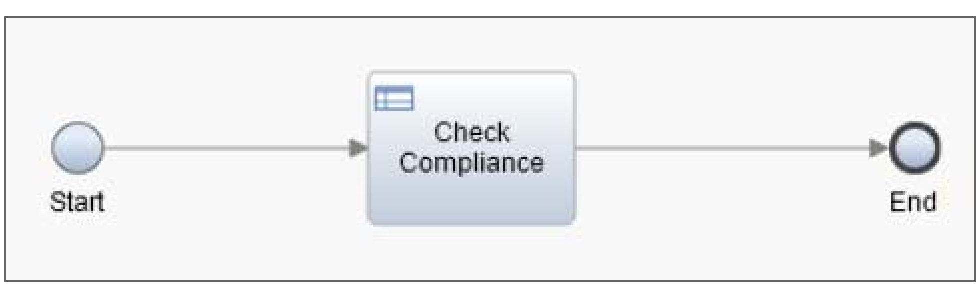


A **green plus (+)** is displayed when you are in an area that the item can be dropped. A **red circle with an (x)** appears if the mouse is hovering in an area where it cannot be dropped. Make sure the sequence line is highlighted and the mouse pointer turns to a **green plus (+)** sign.




__ b. The activity is placed inside the sequence flow.

__ c. Name the Decision: `Check Compliance`



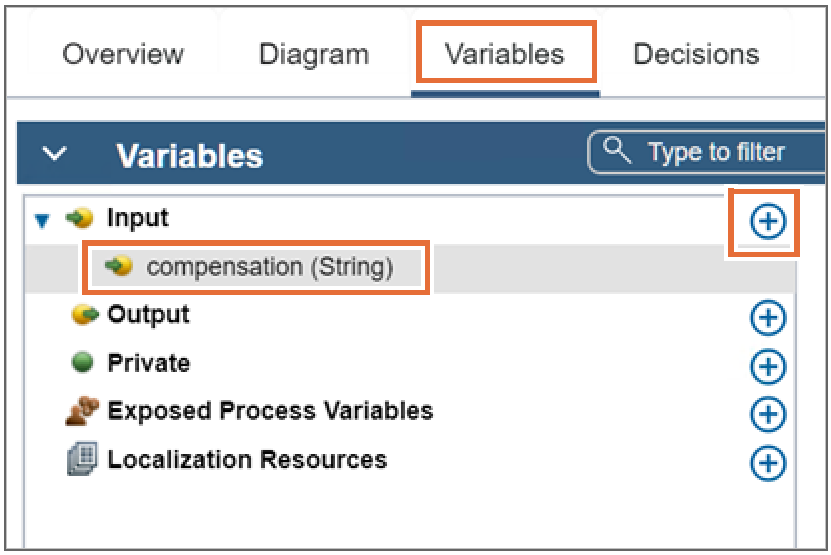
**Reminder**

You can rename artifacts two ways: either through the **General > Common** properties section, or by double-clicking the artifact and renaming it on the canvas.

__ 3. Add variables to the service.

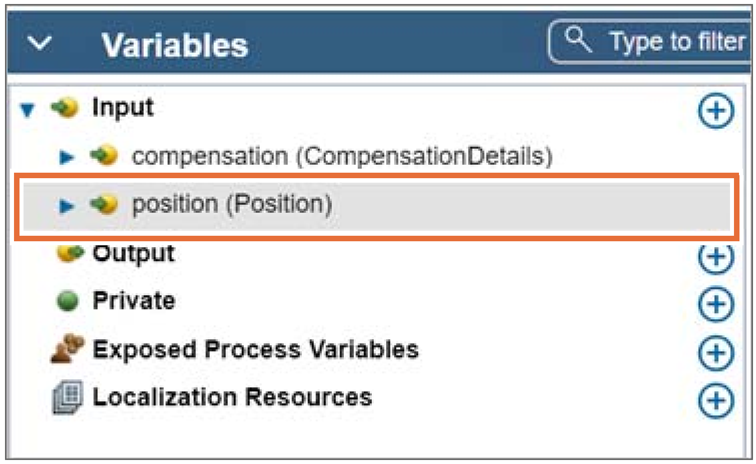
__ a. Click the **Variables** tab.

__ b. Add an input and name the variable: `compensation`

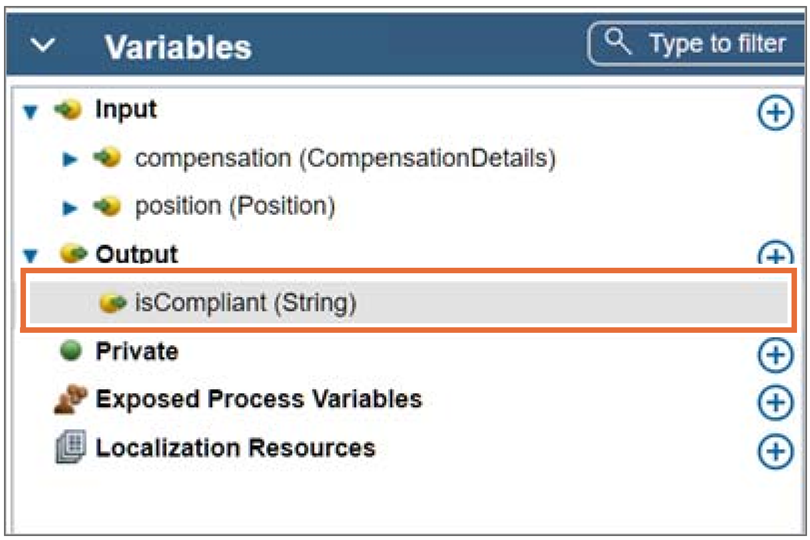



__ c. Click **Select** next to the Variable type. To filter the variable type list, type: `Com` and then click the variable type `CompensationDetails`.

__ d. Add a second input variable named `position` of type **Position**.




__ e. Add an output variable: `isCompliant (String).`

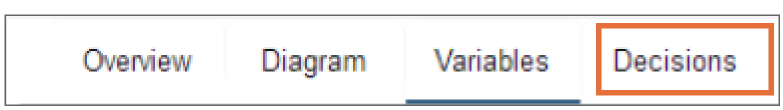



__ f. Save your changes.

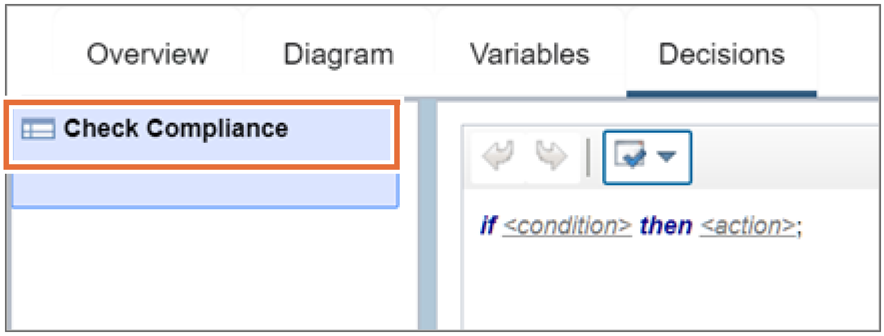
__ 4. Build the condition.

First, you build the condition, then you build the action to be taken when the condition is met.

__ a. Click the **Decisions** tab.



__ b. Click the **Check Compliance** BAL rule on the left.




A default rule is added to the BAL rule editor. The condition and action options are editable. If you click either option, an editor is displayed to assist you with writing the rule. As is the case with the JavaScript editor, you can also use content assist (Ctrl+Space).

The BAL rule is a simple set of if-then statements. In this example, if the salary is over or under a threshold, it is not compliant. Otherwise, it is compliant. The threshold levels are:

- Associate. Minimum salary: 40,000, maximum: 60,000
- Manager. Minimum salary: 50,000, maximum: 75,000
- Director. Minimum salary: 70,000, maximum: 95,000
- President. Minimum salary: 100,000, maximum: 150,000

---

**Troubleshooting**

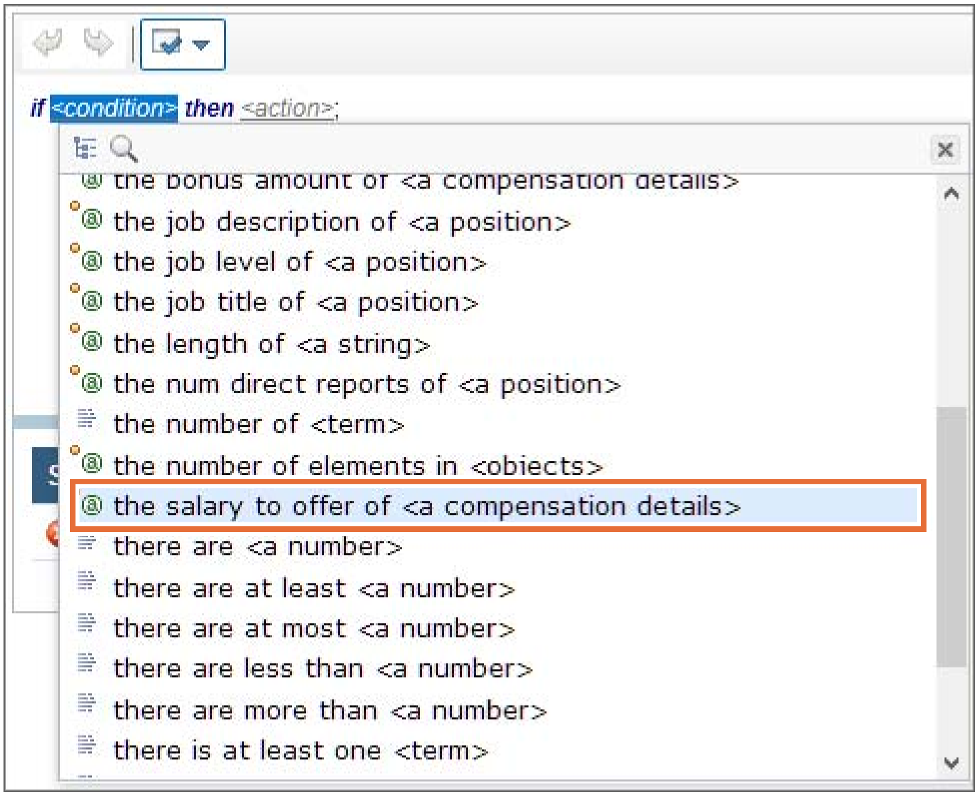
The lab environment might occasionally cause unexpected behavior when using content assist. If the content assist window ever disappears, press the spacebar to see the content assist window display again.

Alternatively, if you are having difficulty with the content assist, you can copy and paste the rules text that is provided in the file `BAL Rules.txt` at the location: `C:\labfiles\Support files\Ex8`

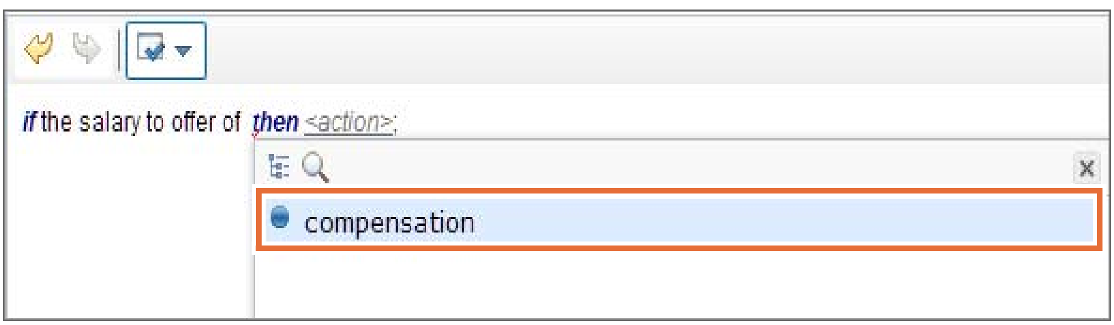
---

__ c. To build the condition, click the **`<condition>`** statement for your rule, and select the following rule piece:

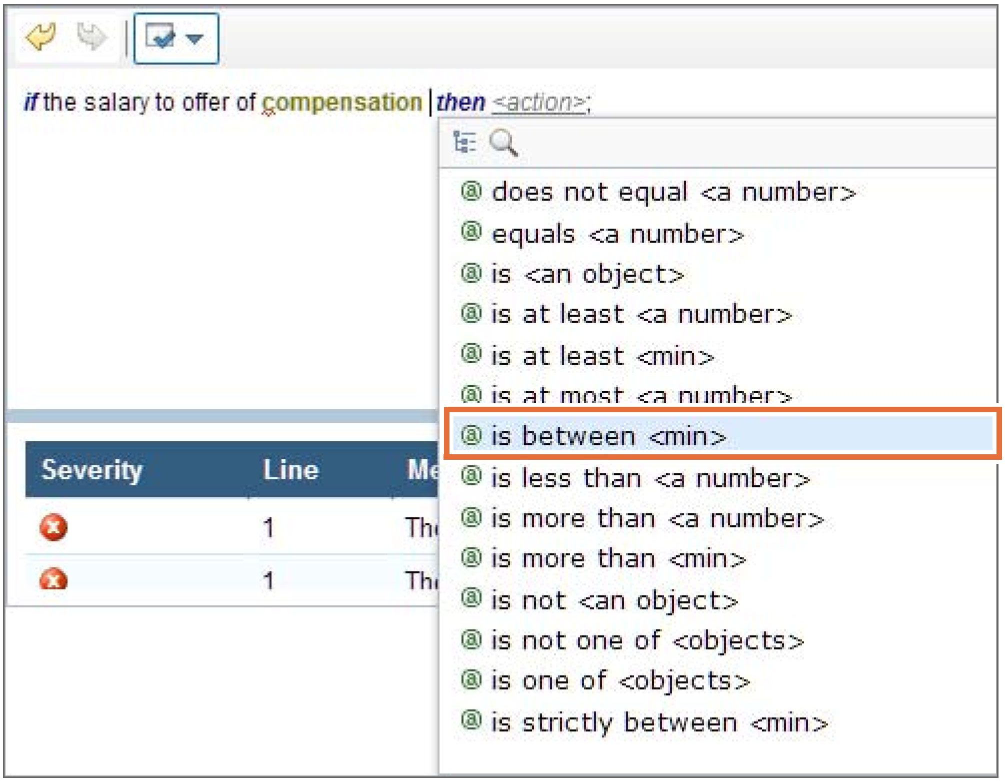
`the salary to offer of <a compensation details>`




__ d. Add a space and use the content assist (Ctrl+Space) to add `compensation`



__ e. Use content assist (Ctrl+Space) to add the following: `is between <min>`




__ f. Use content assist to add: <number>. Immediately after being added, the number turns to 0. Enter `40000` then add a space.

__ g. Use content assist to add `and <max>`

__ h. Enter `60000`

__ i. Add a space and use content assist to add `and <condition>`

__ j. Use content assist to add `the job level of <a position>`

__ k. Use content assist to add `position`

__ l. Use content assist to add `contains <a string>`

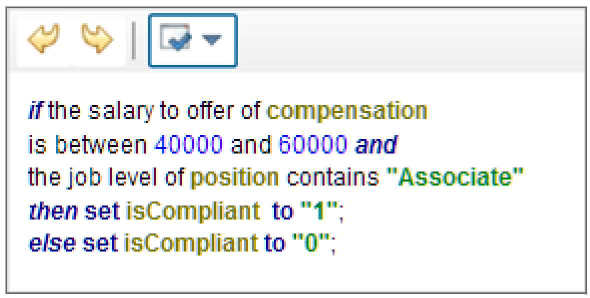
__ m. Enter `"Associate"`

__ n. Save your work.

__ 5. Build the action.

__ a. Click the **<action>** statement of your rule, and select the following rule pieces:

```
set <variable>
isCompliant
to <variable value>
<string>
"1";
<space>
else
set <variable>
isCompliant
to <variable value>
<string>
"0";
```

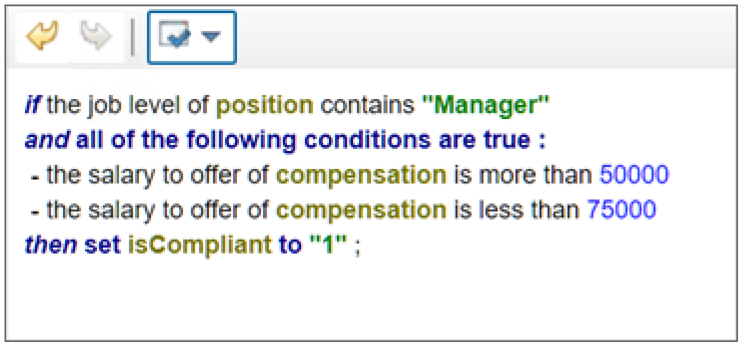



The BAL rule reads much like a natural sentence. You assign a value in the first rule, and the other rules below it evaluate. If one of the rules evaluates to true, then it overwrites the variable.

__ b. Click the **Add action rule** (**+**) symbol to create a second rule.

__ c. Build the next rule by using the previous method to flag the salary as non-compliant when it does not fall within the same range. Alternatively, if you are having difficulty with the content assist, you can copy and paste the rules text that is provided in the file BAL Rules.txt at the location: `C:\labfiles\Support files\Ex8`. Build the rule to read:

```
if the job level of position contains "Manager"
and all of the following conditions are true:
- the salary to offer of compensation is more than 50000
- the salary to offer of compensation is less than 75000
then set isCompliant to "1";
```




__ d. Click the **Add action rule** (**+**) symbol to create the third rule.

__ e. Copy and paste the prior rule and change the variables as required.

```
if the job level of position contains "Director"
and all of the following conditions are true:
- the salary to offer of compensation is more than 70000
- the salary to offer of compensation is less than 95000
then set isCompliant to "1";
```

__ f. Click the **Add action rule** (**+**) symbol to create the final rule.

__ g. Copy and paste the prior rule and change the variables as required.

```
if the job level of position contains "President"
and all of the following conditions are true:
- the salary to offer of compensation is more than 100000
- the salary to offer of compensation is less than 150000
then set isCompliant to "1";
```

**Reminder**

To avoid any errors, the rules text is provided in the file `BAL Rules.txt` at the location: `C:\labfiles\Support files\Ex8`

__ h. Change the position of the different rules to make the first rule you created the top rule, and each subsequent rule underneath the prior one by using the arrows on the right of each rule. You can start by scrolling to the rule for Associate and click the arrow to bring it to the top.


__ i. Verify that all the four rules are created and are placed in the correct order.

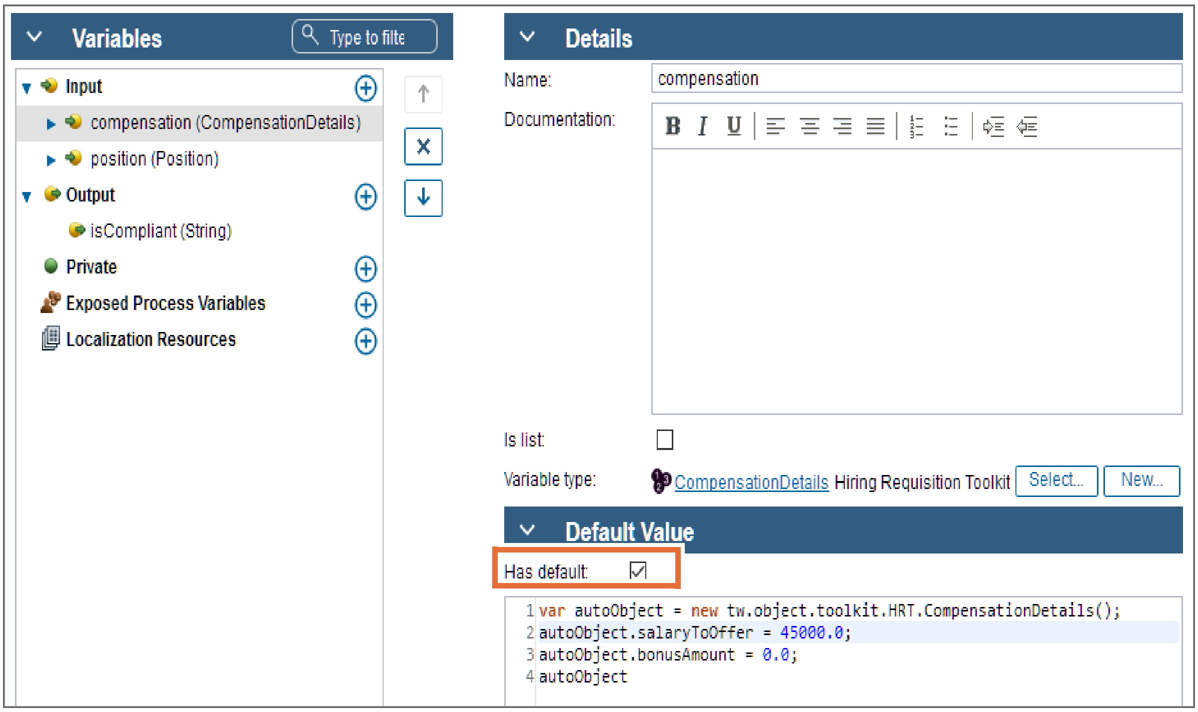
__ j. Save your work.

__ 6. Debug the service: **jobLevel** equals `Associate`.

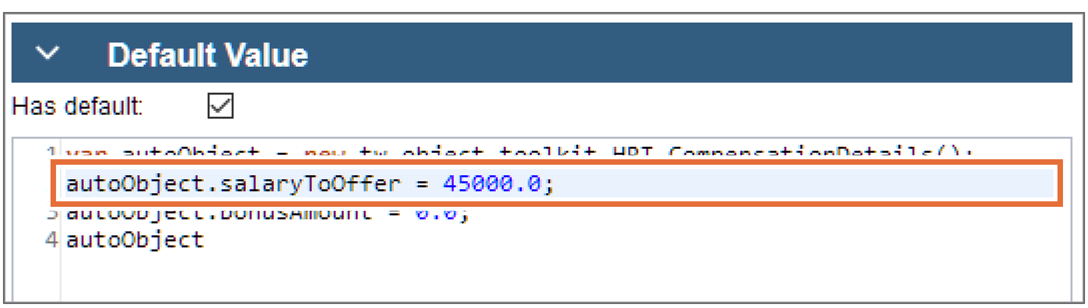
To make sure that the rules are working as expected, debug your service by providing default values to the variables.

__ a. Switch to the **Variables** tab.

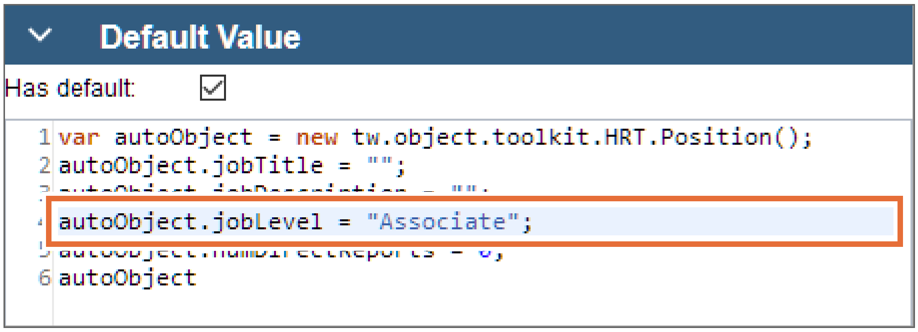
__ b. Click the input variable **compensation (CompensationDetails)**. Select the **Has Default** checkbox under the Default Value section on the bottom right.



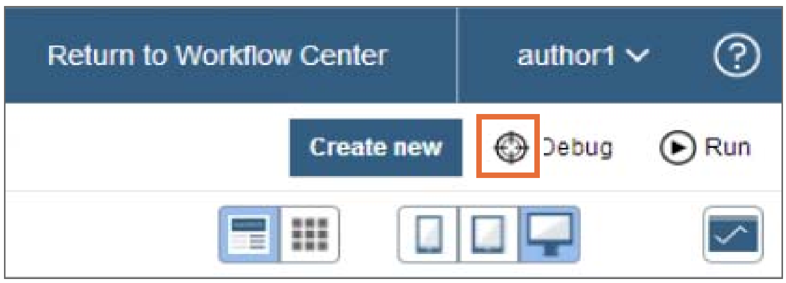

__ c. Change the value for the variable **salaryToOffer** to `45000.0`



__ d. Click the input variable **position (Position)**. Select **Has Default** under the Default Value section on the bottom right. Enter `"Associate"` for the variable **jobLevel**:




__ e. Click the **Debug** button at the top.

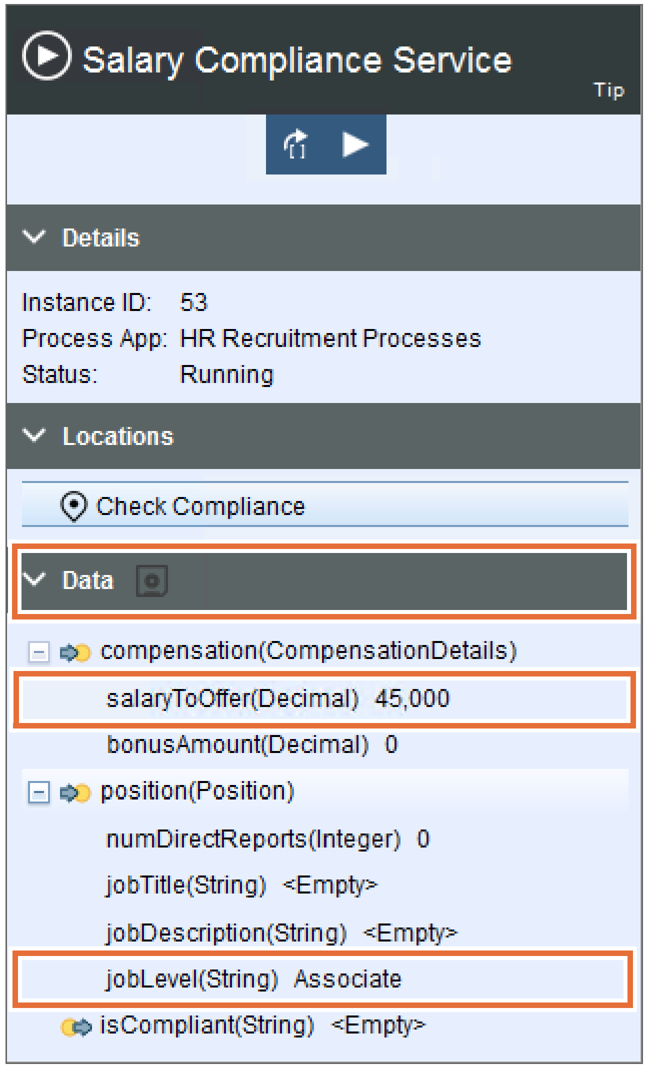



**Hint**

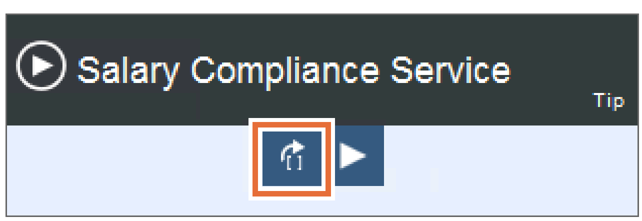
Artifacts are automatically saved when you debug or run a service.

__ f. The view changes from the designer view to the inspector view.

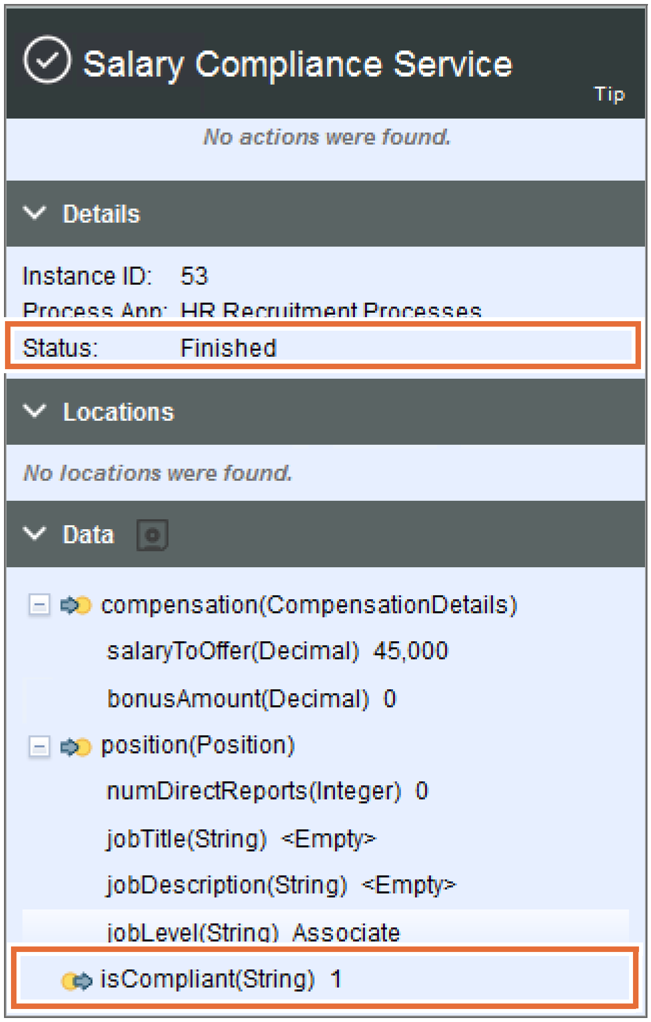
__ g. The debug window opens on the right to show information about the service. Expand the **Data** section on the bottom and verify that the default variables are assigned correctly.




__ h. Click the **Step Over** icon on the top to test the first decision rule with the value for variable **salaryToOffer** as `45000` and the value for variable **jobLevel** as `Associate`.



__ i. The service runs the BAL rule, the service completes, and the status changes to Finished. The correct response with this set of inputs is that the request is compliant. Verify that you get an `isCompliant` variable value of 1 because for the Associate position the salary is in the range 40000 – 60000 and you tested with the value 45000, which makes this request compliant.

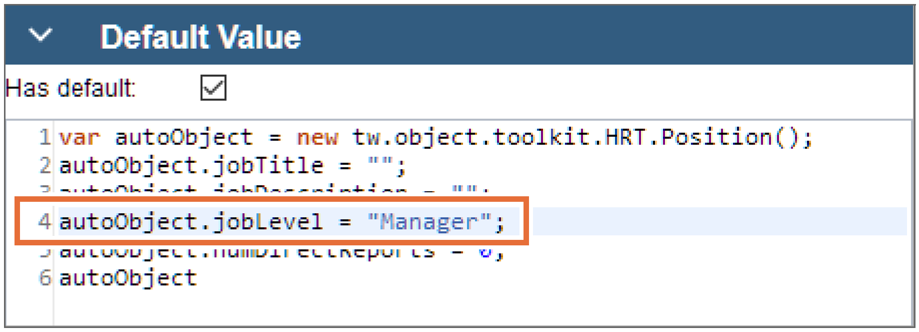



__ 7. Debug the service: **jobLevel** equals `Manager`.

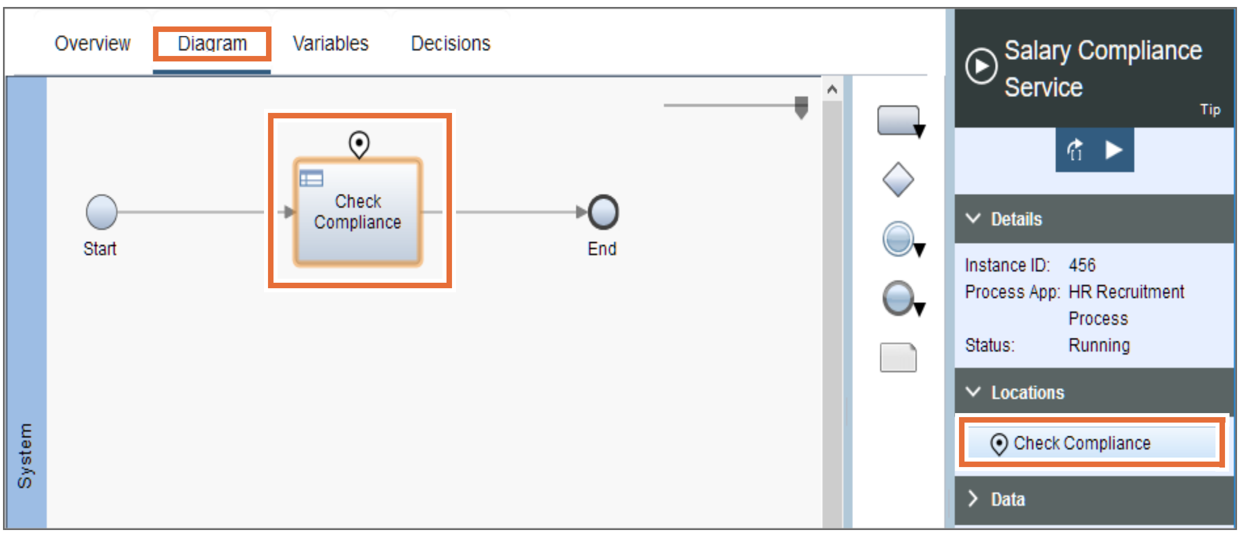
__ a. Click the **Designer** tab to return to the Designer view.


__ b. On the **Variables** tab, select the **position (Position)** input variable

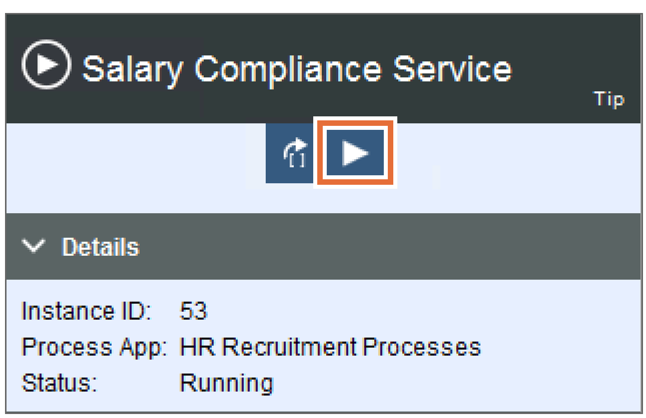
__ c. Change the value of the **jobLevel** variable to `"Manager"` and save the service. Switch to the Diagram tab and click the **Debug** button on the top.




__ d. The information window opens on the right. The token on the Check Compliance service step shows the next step in the service to be run. The token matches the step that is shown in the **Locations** section in the information window.

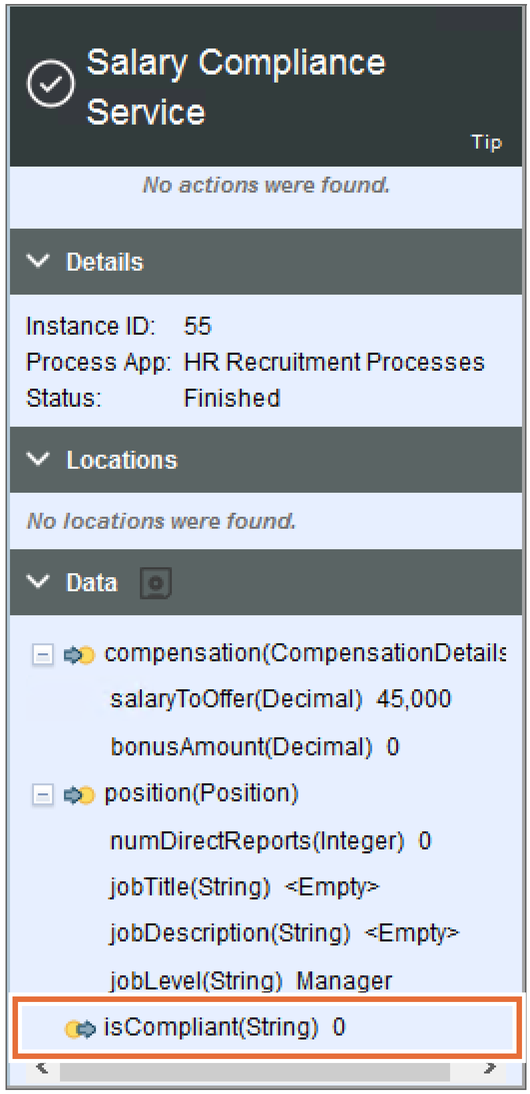



__ e. Expand the **Data** section and click the **Run** button on the top to run the service to completion.




The Step Over button runs a single step in the service where the token is located. The Run button runs the service until a coach is displayed or until the service completes.

__ f. The service request is not compliant. Expand the business objects and verify that you get an `isCompliant` variable value of `0`. This is because the salary for the Manager position is in the range `50000 – 75000` and you tested with salary `45000`, which makes this request non-compliant.




__ g. Return to the **Designer** view of Process Designer.

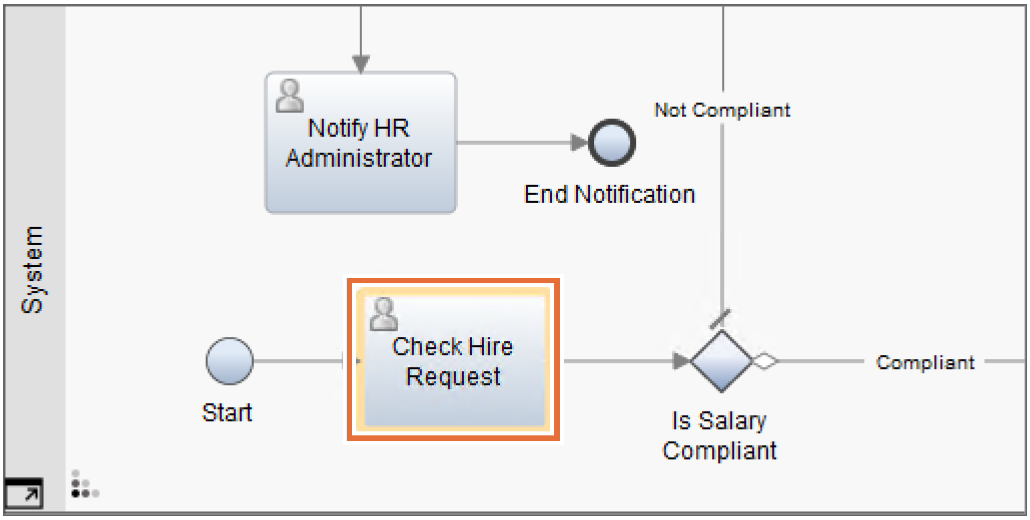
__ h. Click the **Variables** tab and clear the **Has Default** checkbox under the **Default Value** section for the input variables **compensation (CompensationDetails)** and **position (Position)**.

__ i. Save your changes.

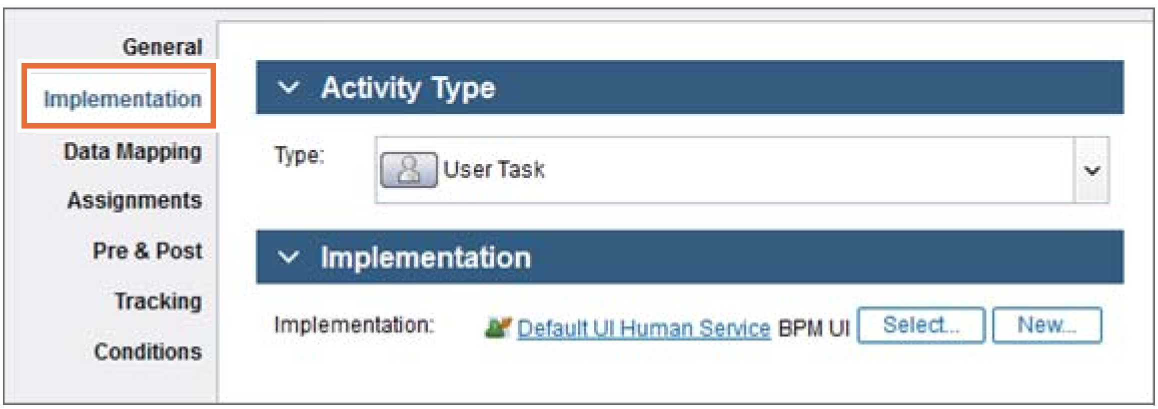
__ 8. Add the rule to the process.

__ a. Open the **Approve Hire Request** process.

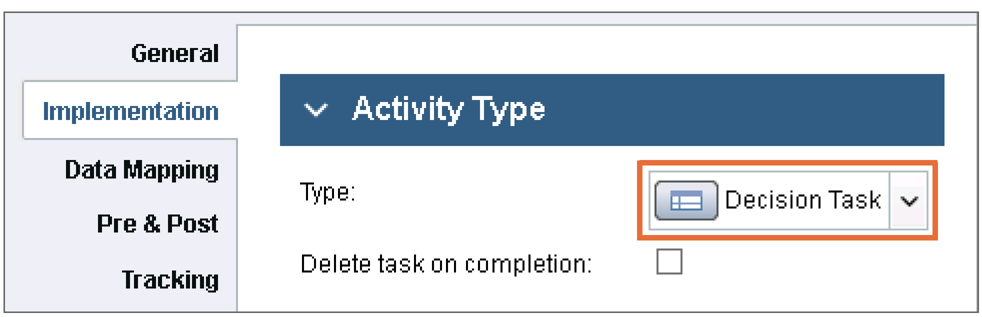
__ b. Select the **Check Hire Request** activity in the `System` lane.



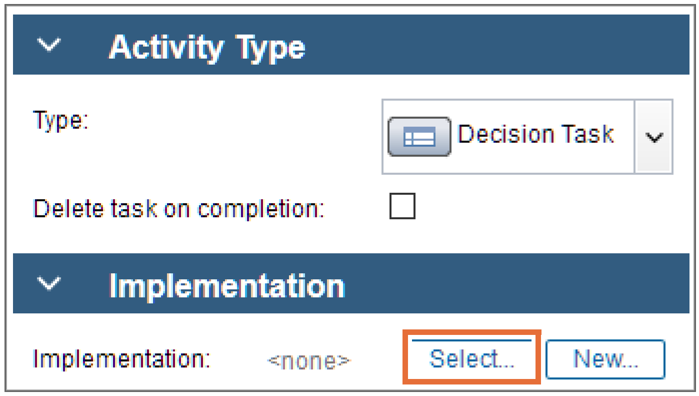

__ c. Click the **Implementation** properties menu.




__ d. Change the activity type to **Decision Task**.



__ e. Click **Select** in the **Implementation** section.




---

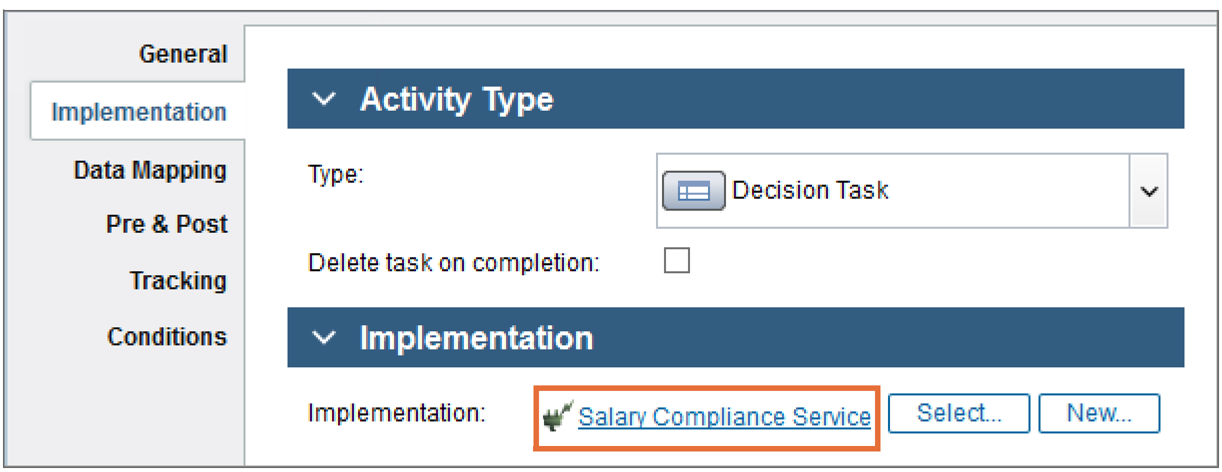
## Information

If you want to run an automated service that does not require routing, you must select the **Delete task on completion** checkbox. This option deletes the task from the system and the process inspector after the task completes. When you select this checkbox, the Process Server does not retain audit data for the task. By default, this checkbox is not selected.

Deleting the task on completion from the task table in the database provides some performance advantages. However, in this case you do not select this option because you want to avoid losing some auditing data, including the details of when that system task ran and how long it took.

---

__ f. Select **Salary Compliance Service**.

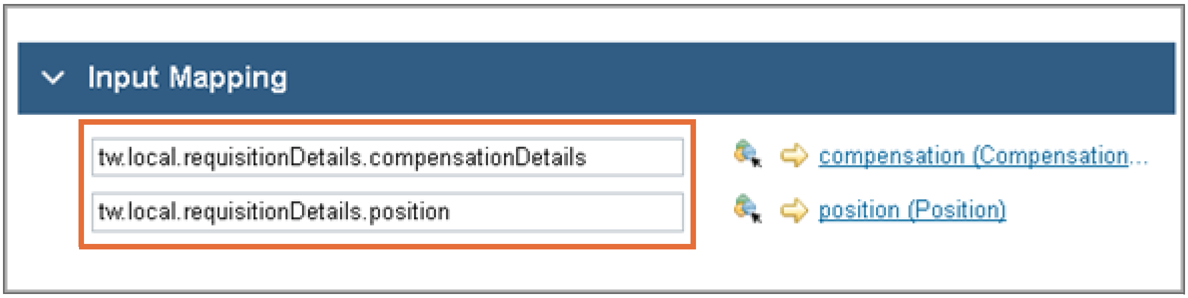



__ 9. Map the variables for the **Check Hire Request** activity.

__ a. Open the **Data Mapping** properties menu.

__ b. Enter `tw.local.requisitionDetails.compensationDetails` for the first input to map it with **compensation**.

__ c. Enter `tw.local.requisitionDetails.position` for the second input to map it with **position**.




__ d. Enter `tw.local.isCompliant` for the output to map it with **isCompliant**.


Notice that you do not need to send the full `requisitionDetails` object into every activity. If a linked process or an activity needs only a small amount of data from the parent object, send in what is necessary. This approach helps the performance of your processes at run time. If you know that variables hold the same data, it also helps to name those variables the same across your processes because it helps with maintenance and troubleshooting.

__ e. Save your work.

## *Part 2: Implement a message start event*

A business requirement has been added specifying the process needs to automatically run every month. Now it is time to implement the process so an instance of the process can be initiated monthly.

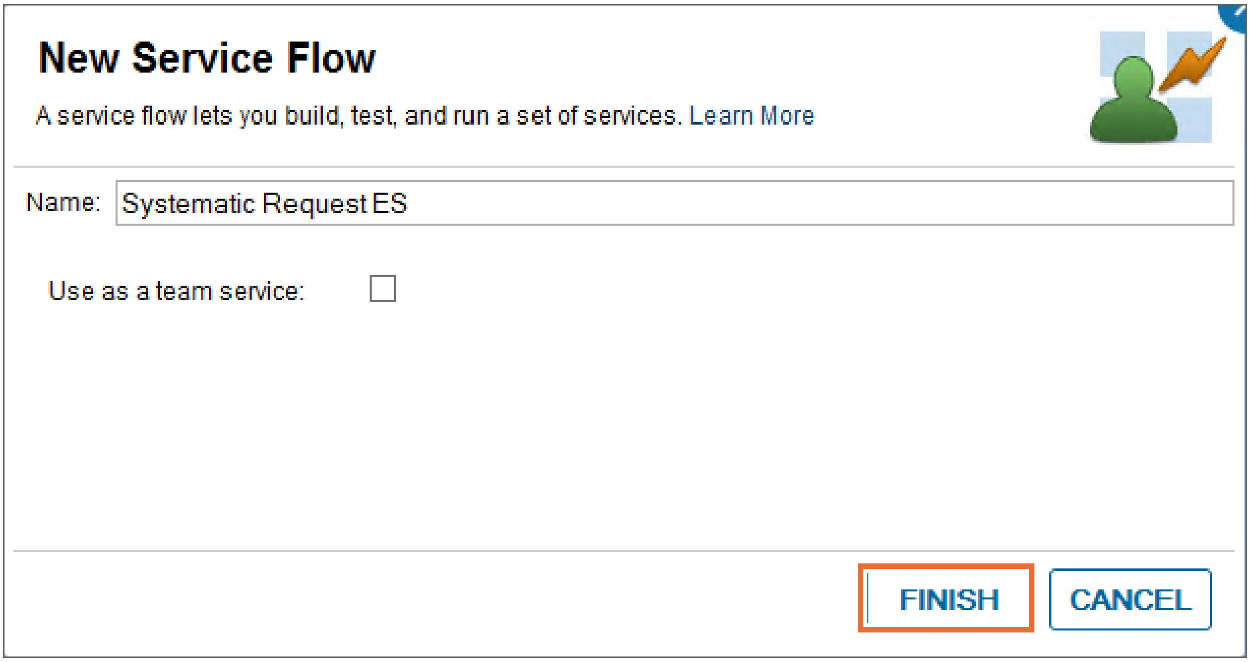
__ 1. Create a service to enable the undercover agent (UCA).

__ a. In the library, click the **(+) plus** sign next to the **Services** menu.

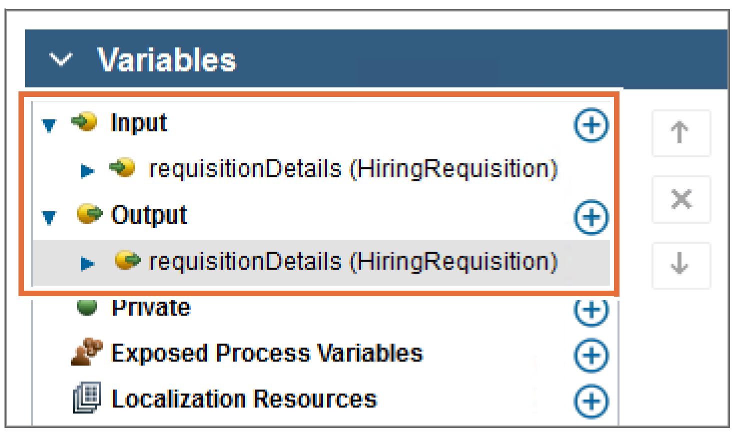
__ b. Click **Service Flow**.

__ c. Name the service: `Systematic Request ES`

__ d. Click **FINISH**.




__ e. Click the **Variables** tab. Continuing from Part 1 of this exercise, create a `requisitionDetails (HiringRequisition)` input and output variable.



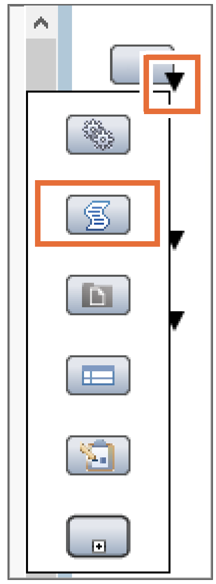

**Important**

Make sure that any single variable that is passed in and out of a service is spelled the exact same for both fields. Remember, variable names ARE case-sensitive. If you have any differences in the variable names, then the system considers them two different variables. This error frustrates many developers when troubleshooting why your variable is not passing out the expected data. Copying and pasting variable names ensures that your variables are the exact same, and reduces problems that result from typographical errors.

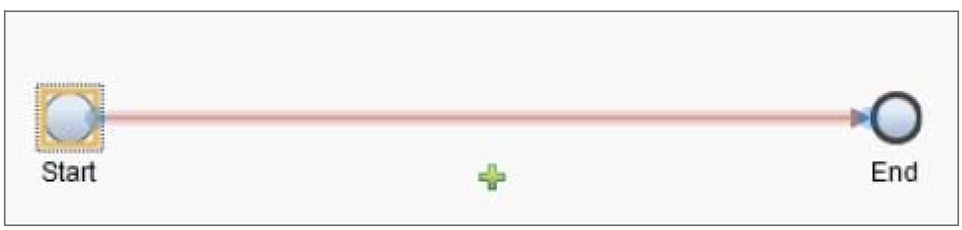
__ f. Save your work.

__ 2. Create a server script to initialize the variables and pass it out of the service.

__ a. Return to the **Diagram** tab. Expand the **Activity** selection and drag a server script from the palette onto the canvas.

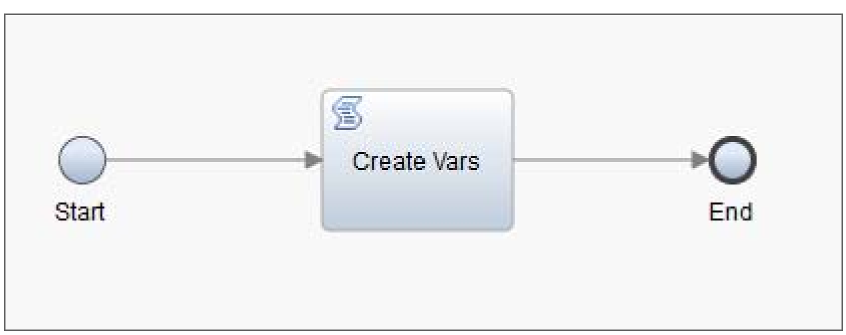


A **green plus (+)** is displayed when you are in an area that the item can be dropped. A **red circle with an (x)** appears if the mouse is hovering in an area where it cannot be dropped. Make sure the sequence line is highlighted and the mouse pointer turns to a **green plus (+)** sign.



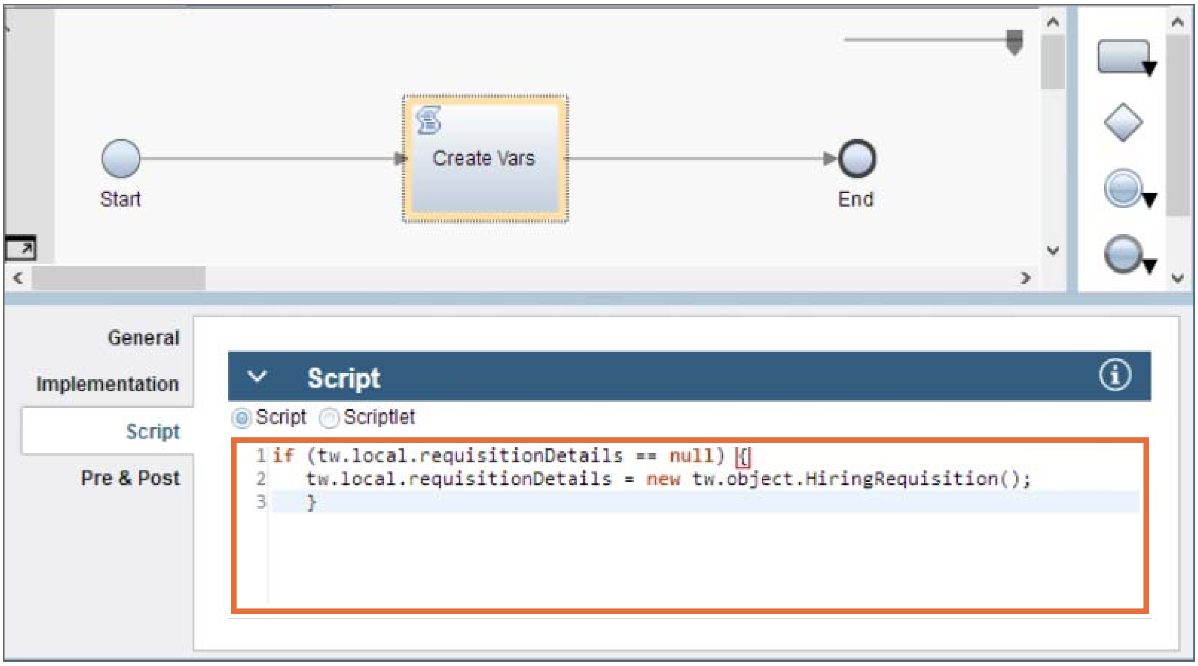

__ b. The activity is placed inside the sequence flow.

__ c. Name the server script: `Create Vars`



__ d. Create a server script to initialize the variables and pass it out of the service. In the **Script** properties menu, copy the script from the `Script1.txt` file that is located in the `C:\labfiles\Support files\Ex8` folder. Optionally, you can enter the following script in the box:

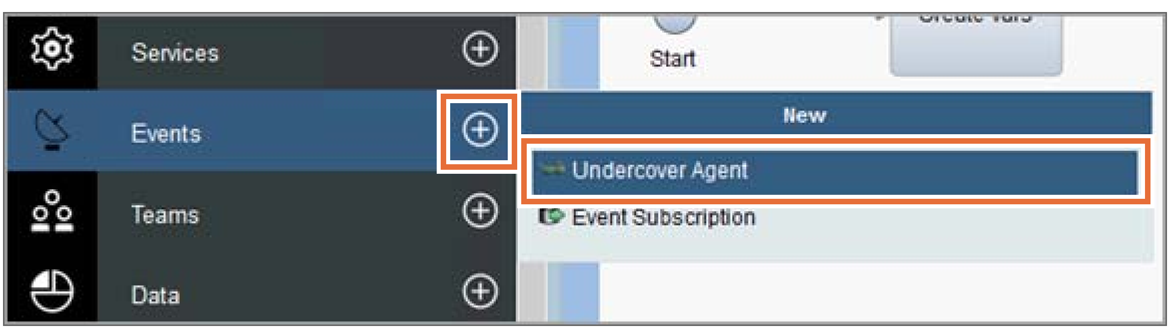
```
if (tw.local.requisitionDetails == null) {
   tw.local.requisitionDetails = new tw.object.HiringRequisition();
   }
```



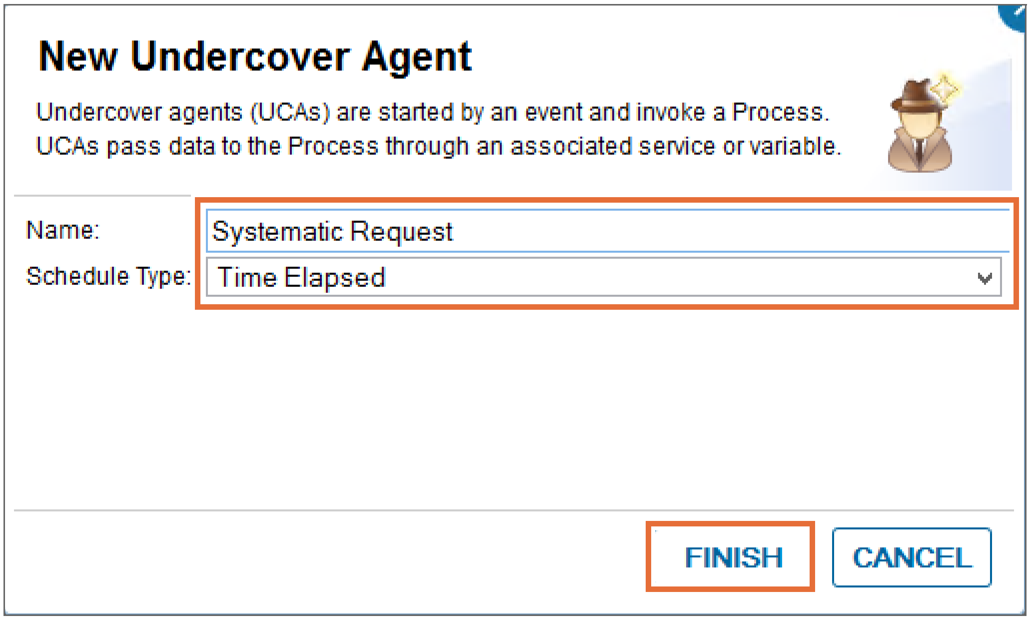

__ e. Save the service.

__ 3. Create and enable an Undercover Agent.

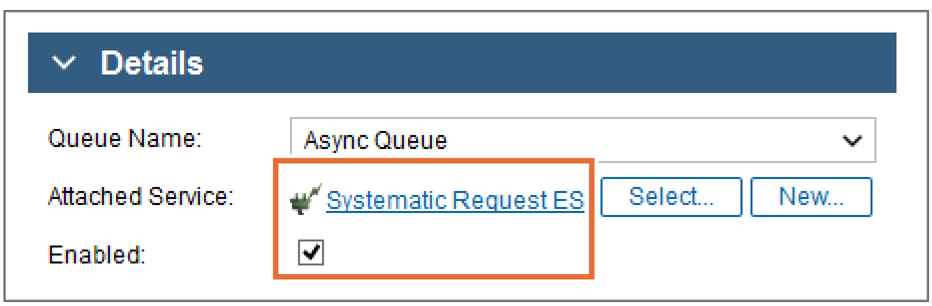
__ a. Click the (**+**) plus sign next to **Events**, and click the **Undercover Agent** option.



__ b. Name the UCA: `Systematic Request`
Make sure that the **Time Elapsed** schedule type is selected. Click **FINISH** to complete the creation of the UCA.




__ c. In the Details section of the UCA settings page on the right, click **Select** next to Attached Service. Select the **Systematic Request ES** service. Verify that the **Enabled** checkbox is selected for the new undercover agent.

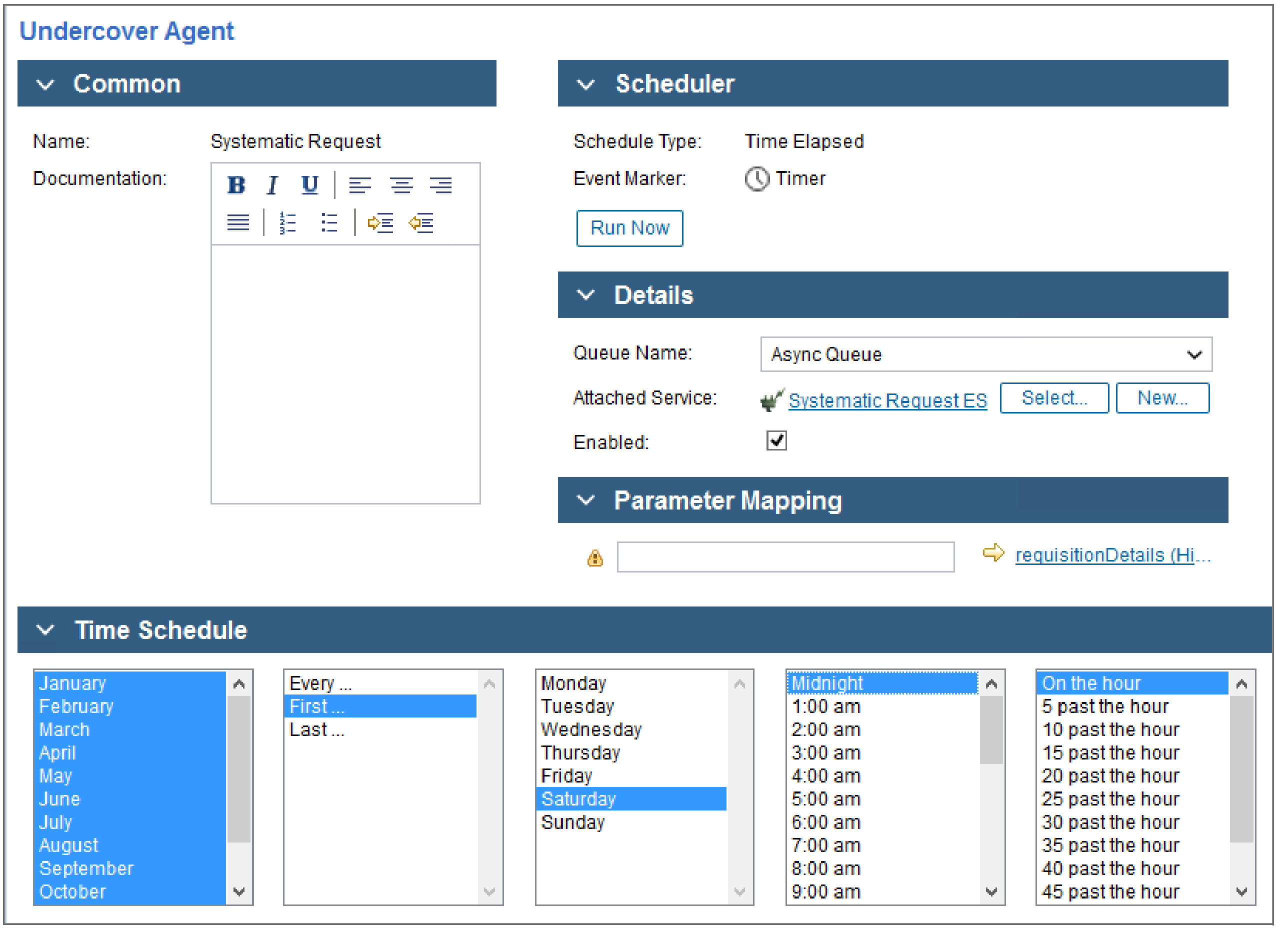



__ d. Create the schedule according to the following script:

`Every month (`**`all months`**`) on the `**`first Saturday`**` at `**`Midnight on the hour`**`.`

Select all the months, then the **First**, **Saturday**, **Midnight**, and **On the hour** options.

This setting generally provides time for the purging service to complete over the weekend.

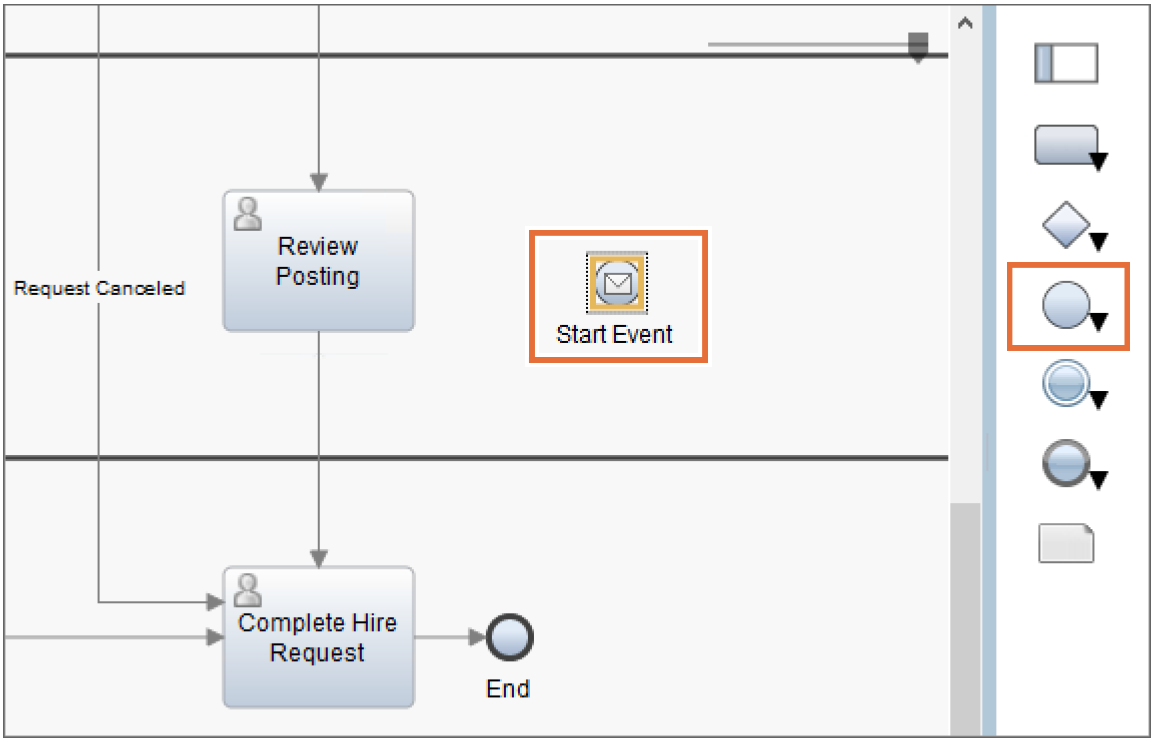



__ e. Save your work.

__ 4. Implement the **Systematic Request** start message event on the **Hiring Request Process** process.

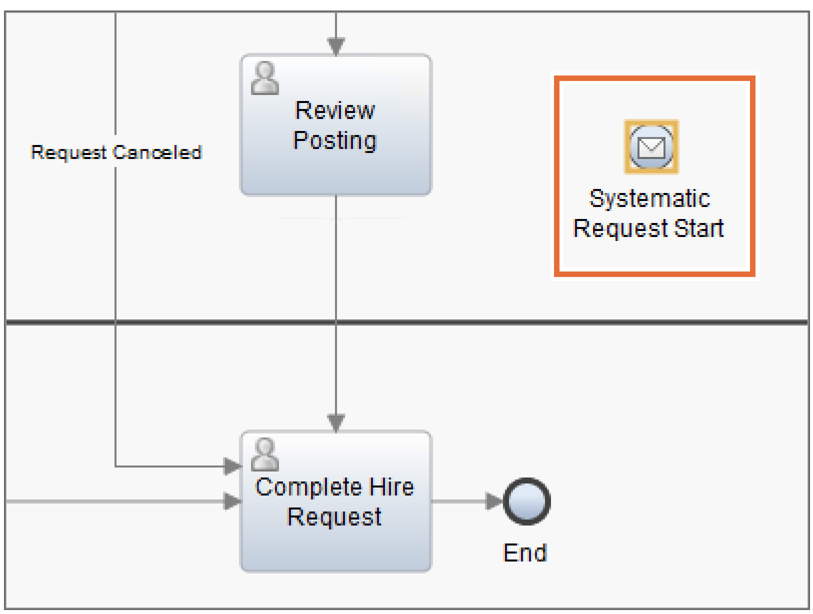
__ a. Open the **Hiring Request Process** process**.**

__ b. Drag a **Start** event from the palette to the process near the **End** event.

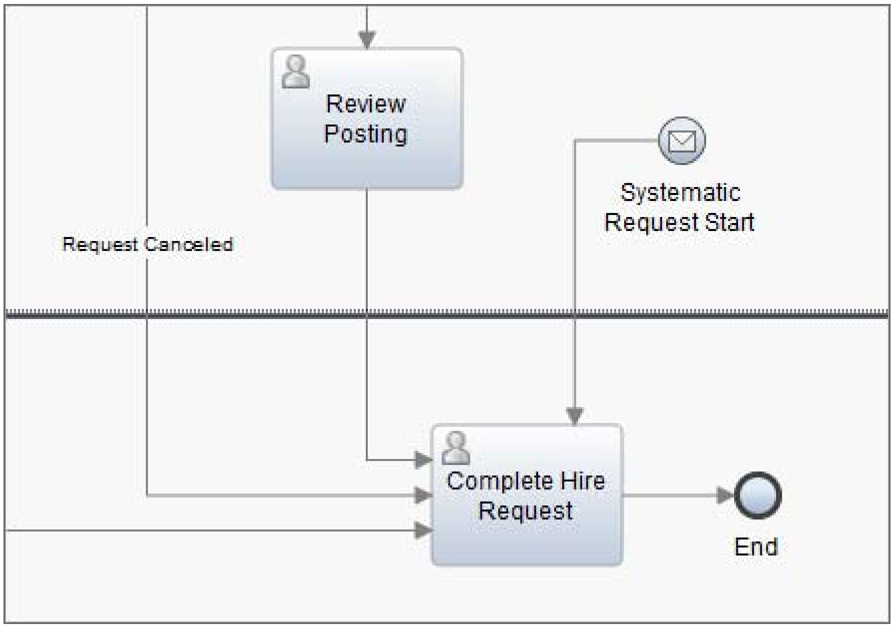



Because a start event is already on the canvas, subsequent start events added to the canvas default to a Start message event. You can also expand the start event in the palette and drag a Start message event to the canvas to get the same result.

__ c. Rename the start message event to: `Systematic Request Start`



__ d. Connect `Systematic Request Start` to the **Complete Hire Request** activity.




__ e. Click the **Systematic Request Start** event, click the **Implementation** properties menu, and then click **Select** next to **Attached message UCA**.


__ f. Select the **Systematic Request** UCA that you created for the activity.

__ g. Save your work.

**Troubleshooting**

If you do not see the Systematic Request UCA, restart Process Designer.

__ 5. Map the `Systematic Request` start event UCA outputs to the process variables.

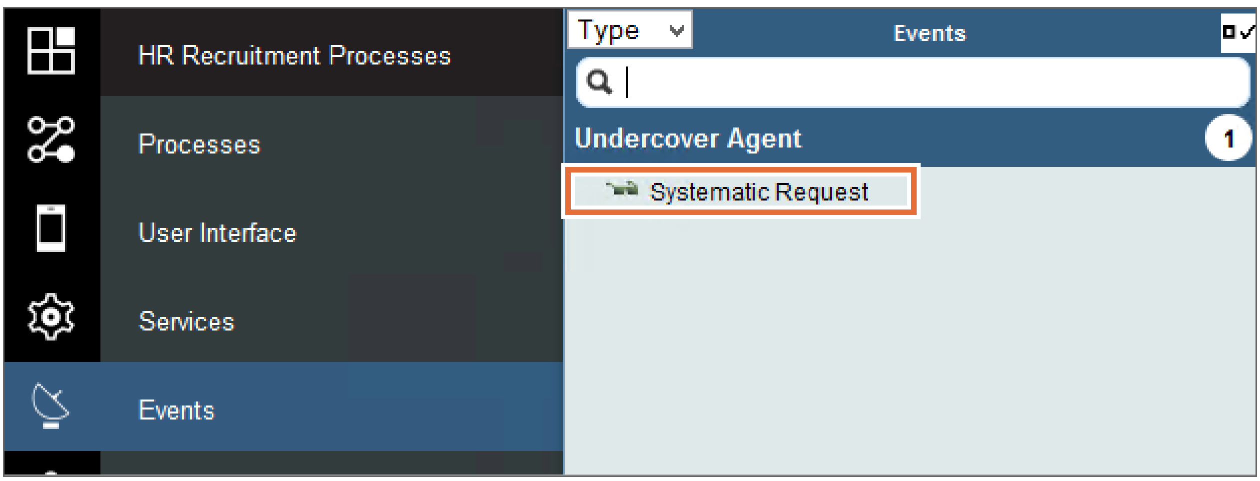
__ a. Click the **Data Mapping** properties menu.

__ b. In the **Output Mapping** section, map the requisitionDetails output variable to the `requisitionDetails (HiringRequisition)` variable.

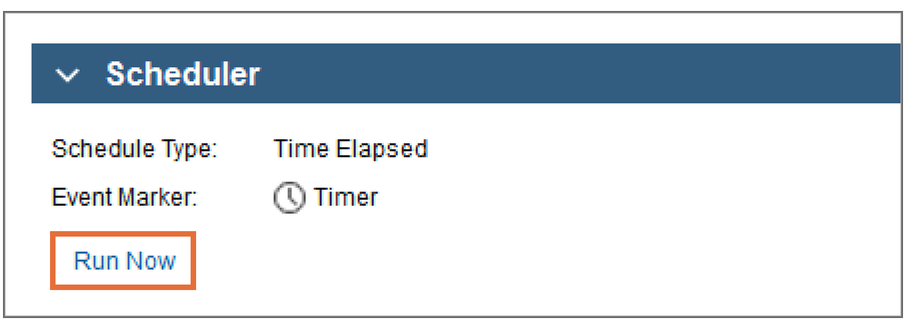

__ c. Save your work.

__ 6. Test the UCA.

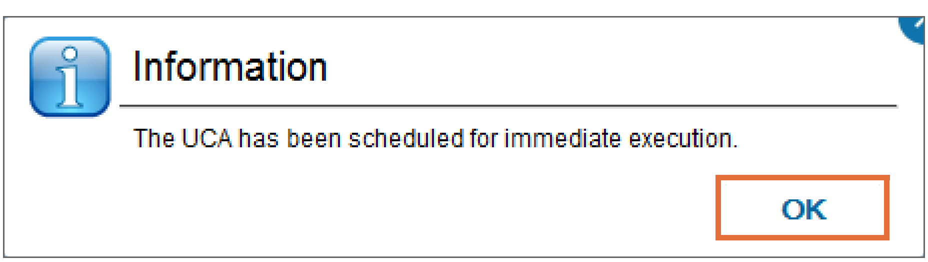
__ a. Open the **Systematic Request** UCA listed in the Events section of the library.



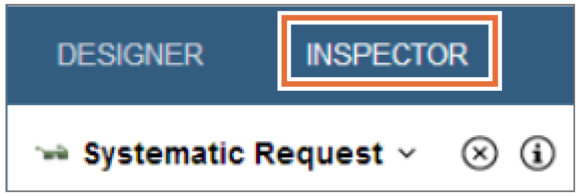

__ b. Click **Run now** in the Scheduler section.



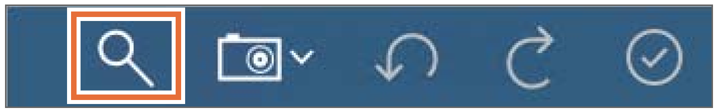

__ c. A dialog box is displayed, which indicates that the UCA is scheduled for immediate execution. Click **OK**.



__ d. Click the **INSPECTOR** view tab.



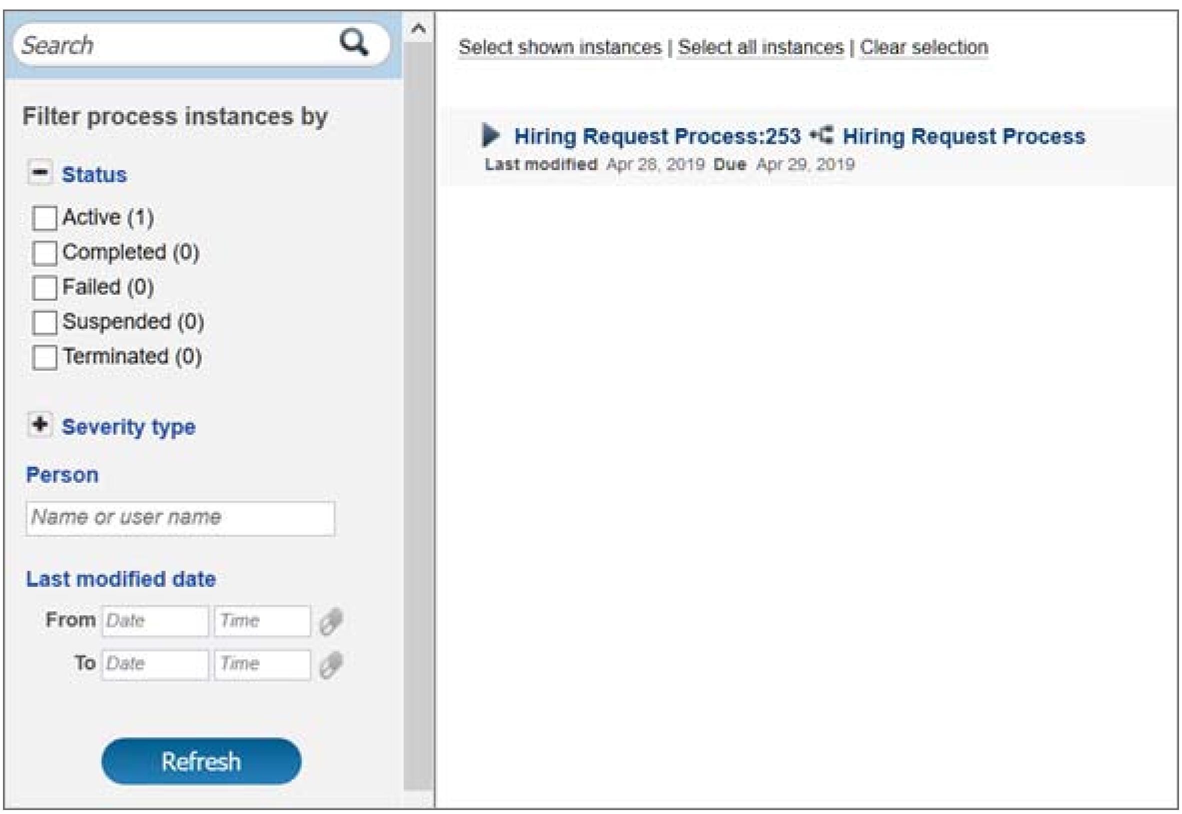

__ e. At the top, click the first icon to **search** process instances.



__ f. On the left, underneath the search filter criteria, click **Search** to view all the instances of all the processes in the Process Center Server.

The Search button changes to Refresh after using it. If you have used it before, it might display Refresh instead of Search.

__ g. If everything was successful, the UCA and Message Start event created a **Hiring Request Process** instance. The instance that you created when you ran the test is the highest numbered instance that is shown in the right frame. This window shows all the instances that ran in Process Designer and their status, although your environment might not match what you see in this screen capture.

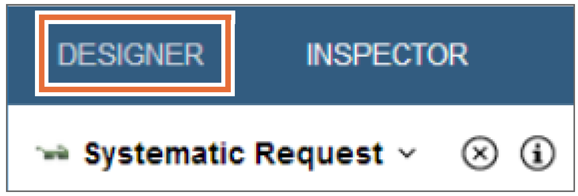


## *Part 3: Apply asset tags*

The Process Designer library now has a number of assets for the **Hiring Request Process** process. You can use asset tagging to organize your existing process application assets. Asset tagging allows developers to accomplish tasks such as associating a UCA with its enabling service.

__ 1. Tag the **Systematic Request Start** service.

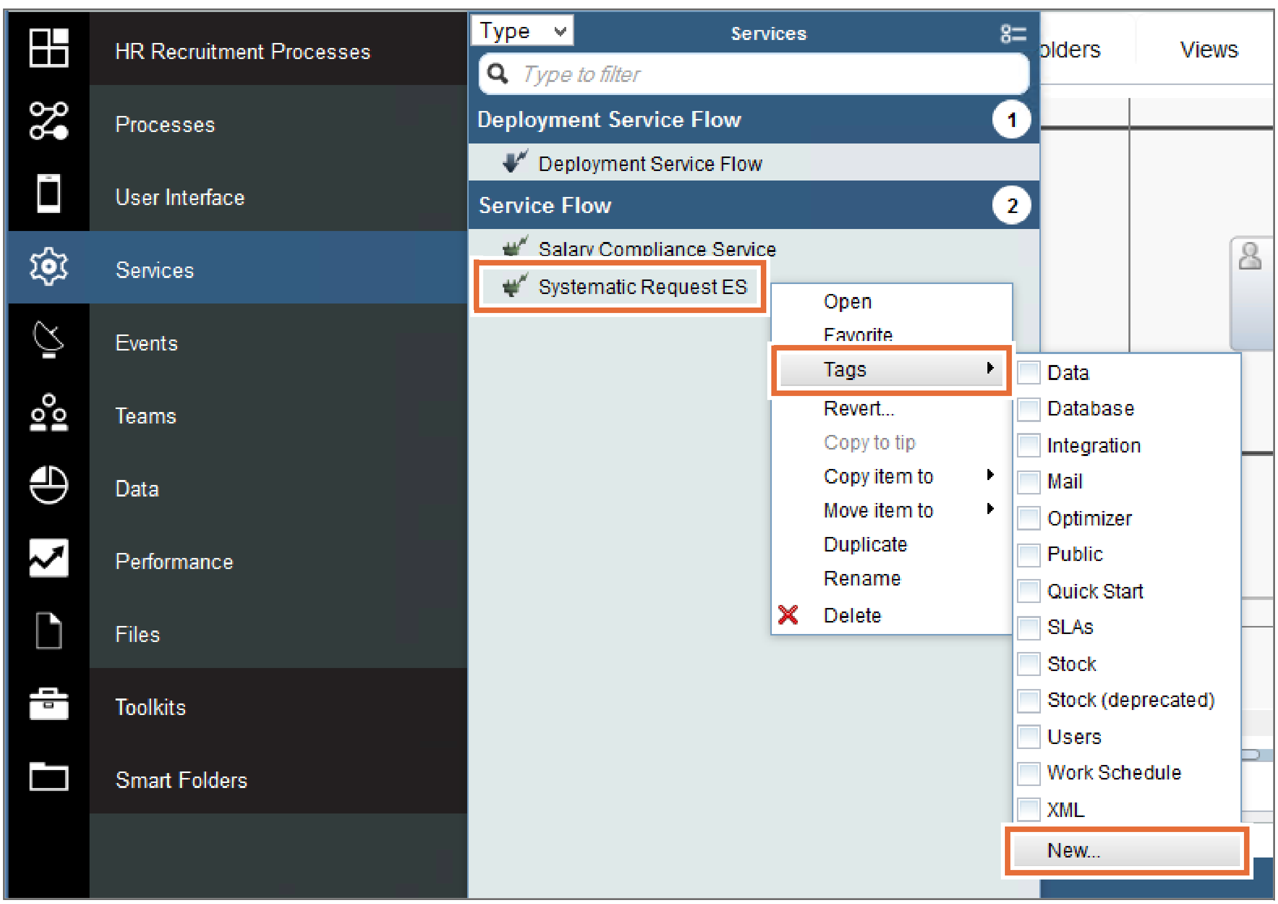
__ a. Click **DESIGNER** to return to the Designer mode of the Process Designer.



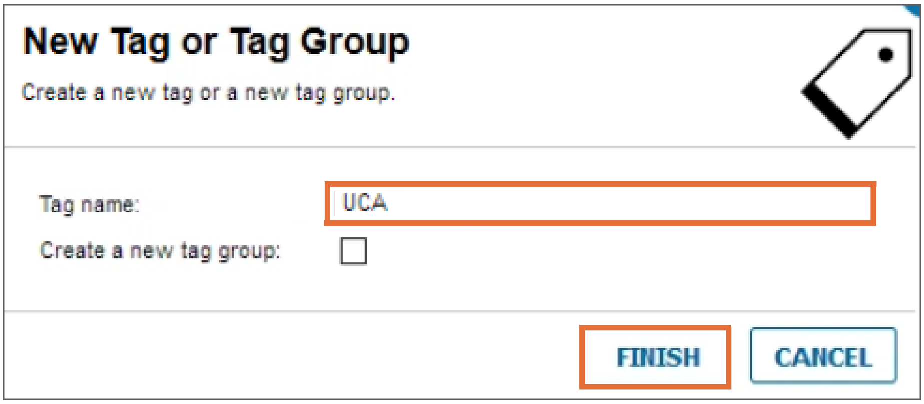

__ b. In the Designer library, open the **Services** category.

__ c. Right-click the **Systematic Request ES** service.

__ d. Click **Tags > New**.



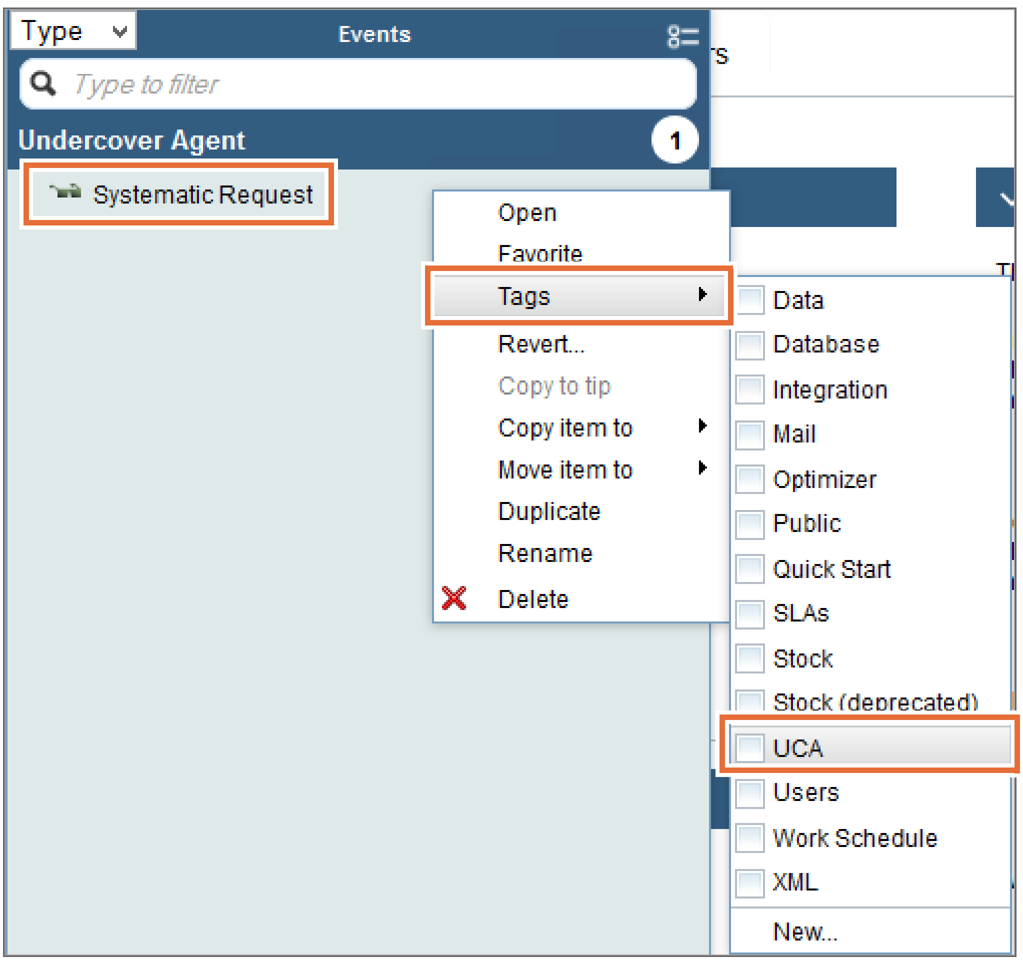
__ e. Name the tag UCA and then click **FINISH**.



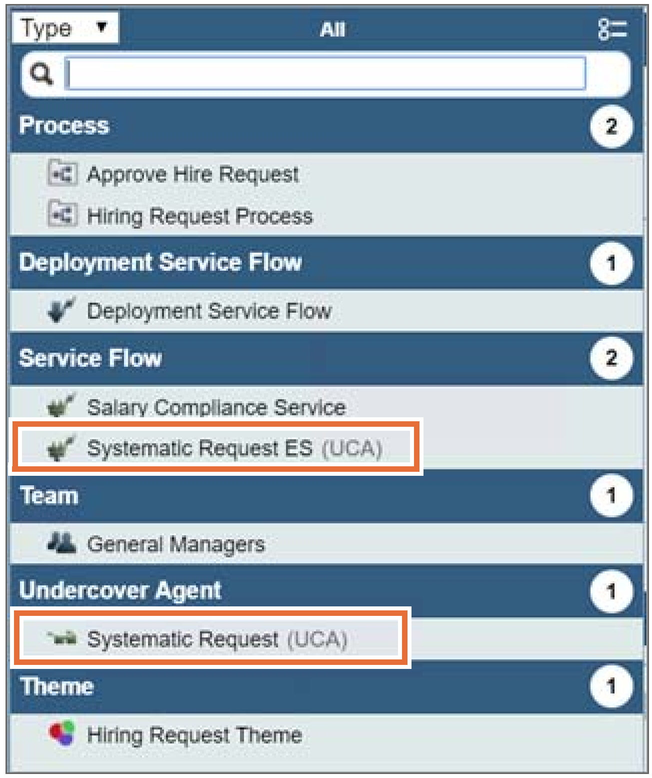

__ f. Return to the library and open the **Events** category.

__ g. Right-click the **Systematic Request** UCA.

__ h. Click **Tags > UCA**.



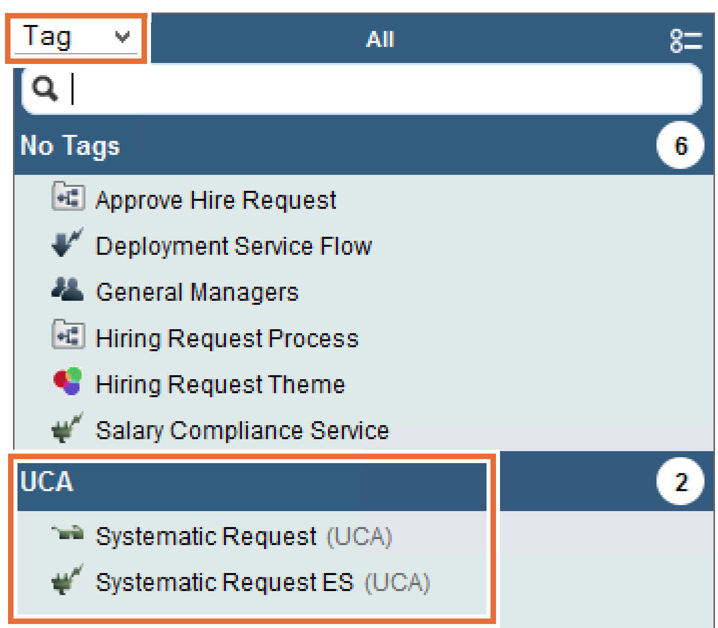
__ i. Click the **HR Recruitment Process** header at the top. This category lists all the artifacts in the library. The two artifacts are now tagged with UCA in parentheses after the artifact name.




__ 2. Group artifacts by tag.

__ a. From the **HR Recruitment Process** menu, you can view all the assets in the library. At the top of the list, the default grouping is by Type. Click the arrow to the right of **Type**. Select **Tag** from the list.

__ b. The artifacts with the tag `UCA` are now grouped in the list.



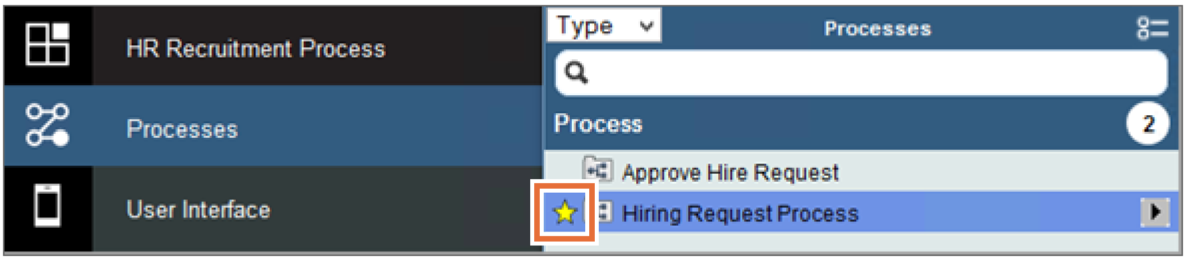
__ c. After viewing the **Name** grouping type, return the grouping type back to **Type**.

__ 3. Create a favorite.

__ a. Click the **Processes** category.

__ b. Hover to the left of the **Hiring Request Process.** The outline of a star appears next to the process name.

__ c. Click the **star** next to the **Hiring Request Process** to designate it a favorite.




__ d. If it is not already expanded, open **Smart Folders** at the bottom of the library. Click **Favorites** to verify that the process is now part of the **Favorites** smart folder.

**Optional**

You have completed part 3 of 5, so you are now half done with this exercise. Now would be a good time to take a 10-minute break.

## *Part 4: Create a service to query a database and populate a list*

In this part of the exercise, you create a service to query a database and populate a list. The current systems require employees to look up different codes and enter them into the system. Because customers use this system, a better approach is to offer a menu with readable options that correspond to the codes in the database. The Job Level and Department Details values make up the menus.

The tables are already established in the TRAINING database as JOBLEVELS. You build a new business object that is called DepartmentDetails that contains the parameters: Division and Department. These parameters match the DIVISIONS and DEPARTMENTS tables in the TRAINING database.

Use an environment variable (ENV) to hold database information for one query, and an exposed process variable (EPV) to hold the database information for the other.

__ 1. Create a service to retrieve the incident categories.

__ a. In the Process Designer library, click the (**+**) plus sign next to **Services** and select **Service Flow**.

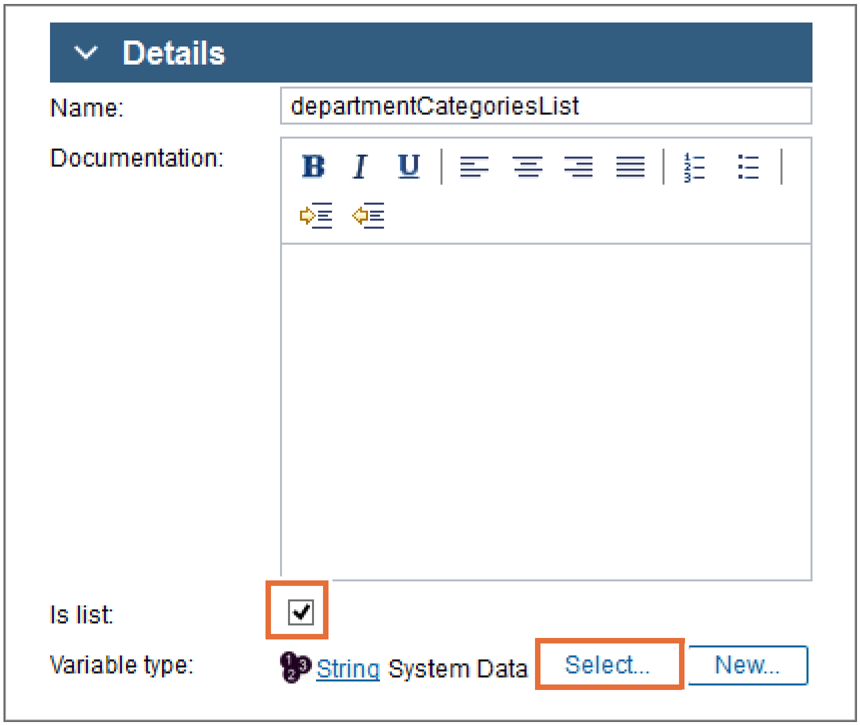
__ b. Name the service `Retrieve Department Categories` and click **FINISH**.

__ 2. Add variables to the service. These variables hold the list data that is retrieved from the database.

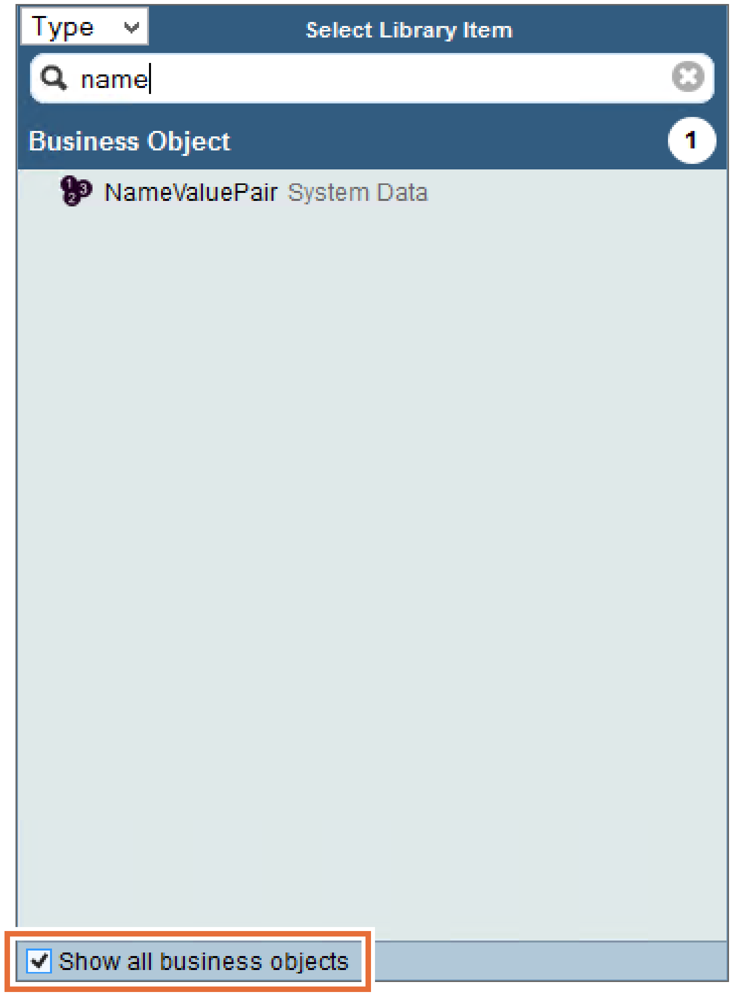
__ a. Click the **Variables** tab.

__ b. Add an output variable named: `departmentCategoriesList`

__ c. Select the **Is List** checkbox and then click **Select** for the variable type.

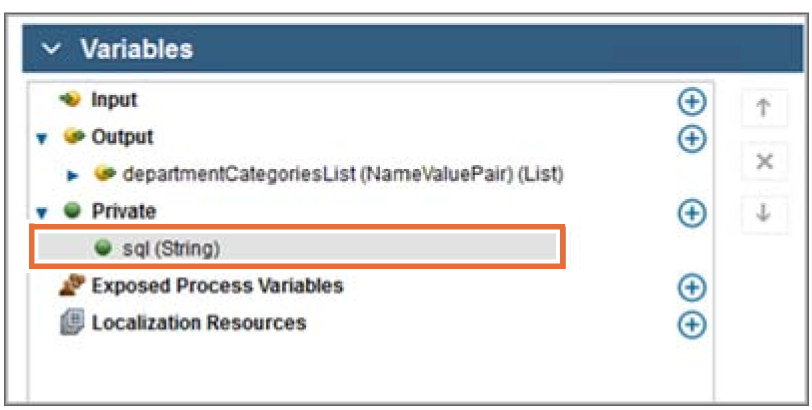


__ d. Select the **Show all business objects** checkbox at the bottom, and select **NameValuePair** from the list.




This variable is used to populate the list of options in a select menu on the coach.

__ e. Now add a **Private** variable: `sql (String)`

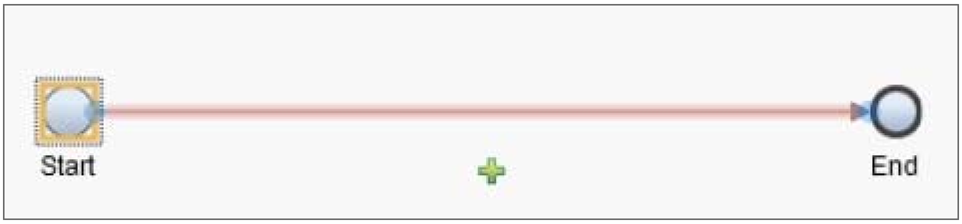



__ 3. Add the elements onto the canvas.

__ a. Click the **Diagram** tab of the service.

__ b. Drag a **Server Script** from the service palette to the canvas.

A **green plus (+)** is displayed when you are in an area that the item can be dropped. A **red circle with an (x)** appears if the mouse is hovering in an area where it cannot be dropped. Make sure the sequence line is highlighted and the mouse pointer turns to a **green plus (+)** sign.

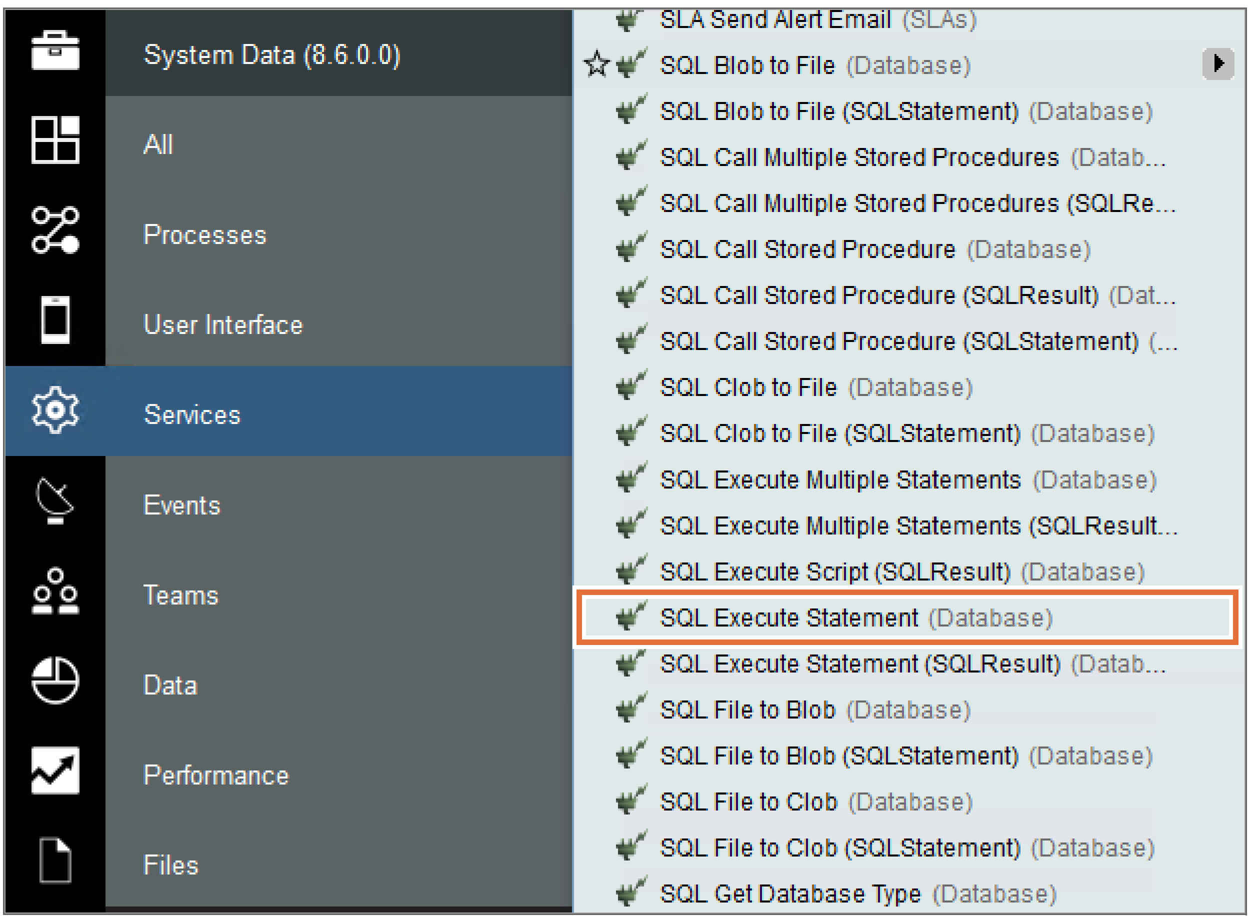



__ c. The activity is placed inside the sequence flow.

__ d. Rename the server script to: `Set SQL`

__ e. Expand the **Toolkits > System Data** toolkit category in the Process Designer library.

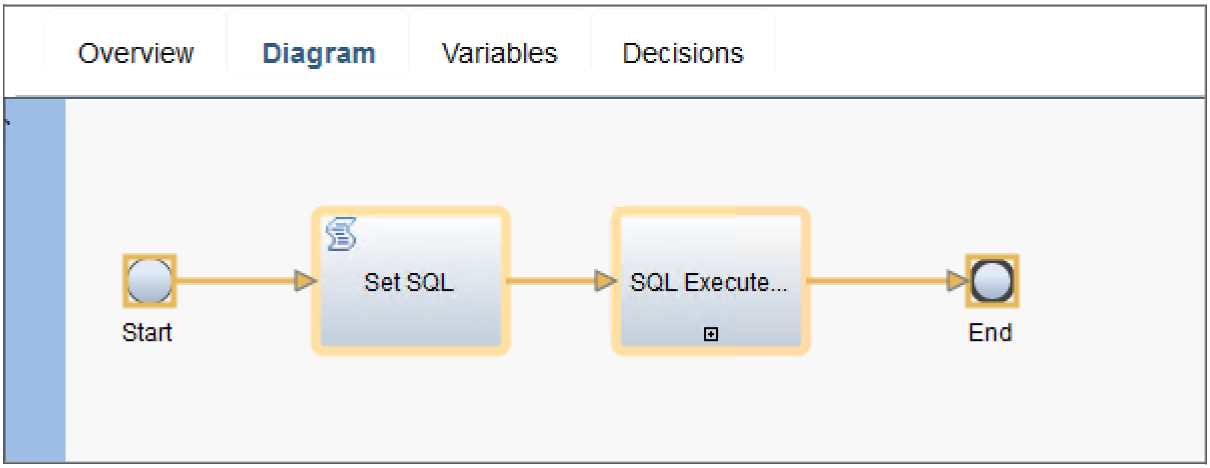
__ f. Select the **Services** category. Drag the **SQL Execute Statement** service onto the sequence flow to the right of the `Set SQL` server scriptlet.




You can find the service that is grouped in with the services that are the Service Flow type, or change to sort by tag and view those items that contain the Database tag. You can also type `SQL Execute` to filter the list to find the service. Be sure to select the correct one as there are several **SQL Execute** services.

If you have problems with dragging the service from the library to the canvas, you can drag a Service Task activity from the palette to the canvas. Then, implement the service by using the SQL Execute Statement service from the System Data toolkit.

__ g. Verify your service flow.



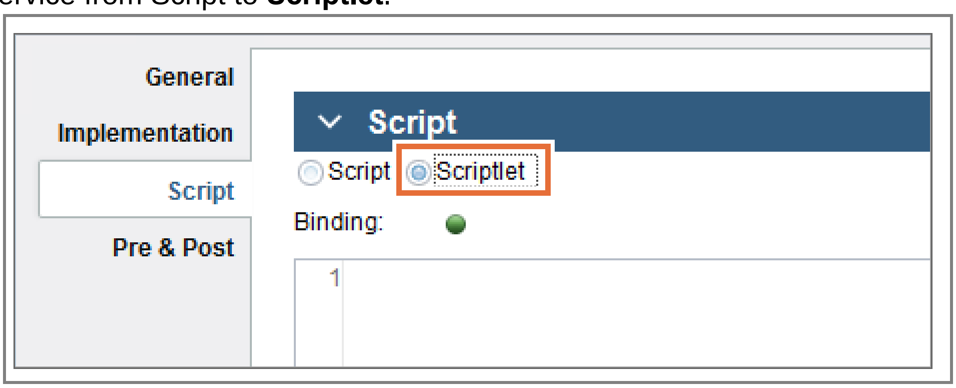

__ h. Save your work.

__ 4. Implement the `Set SQL` server scriptlet.

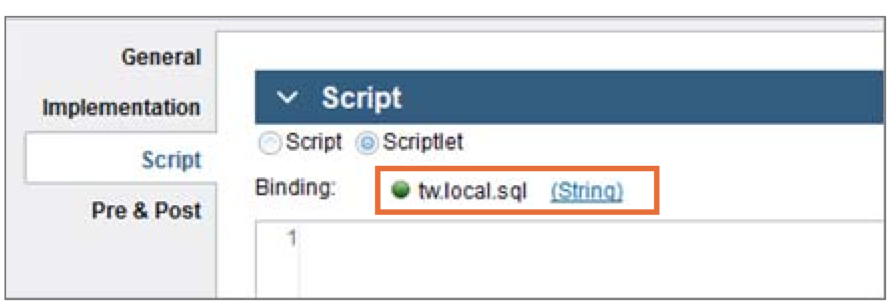
__ a. Click the `Set SQL` server script.

__ b. Click the **Script** properties menu.

__ c. You use a server scriptlet when assigning a value to a single variable. Change the service from Script to **Scriptlet**.

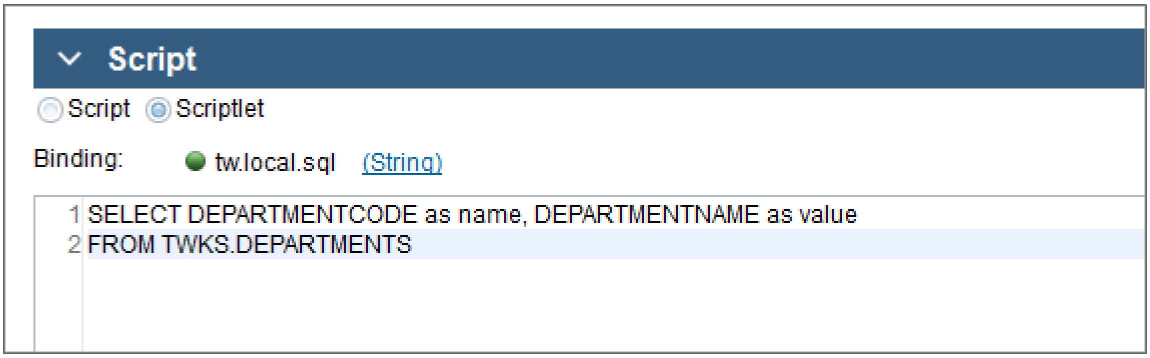



__ d. Click **Select** next to **Binding:** and click the **sql (String)** variable.



__ e. Enter the following SQL query in the script field. The intent is to map the results directly into a **NameValuePair** object, so you must rename the results to enable that direct mapping. Optionally, you can copy the script from the `Script2.txt` file that is located in the `C:\labfiles\Support files\Ex8` folder.

```
SELECT DEPARTMENTCODE as name, DEPARTMENTNAME as value
FROM TWKS.DEPARTMENTS
```

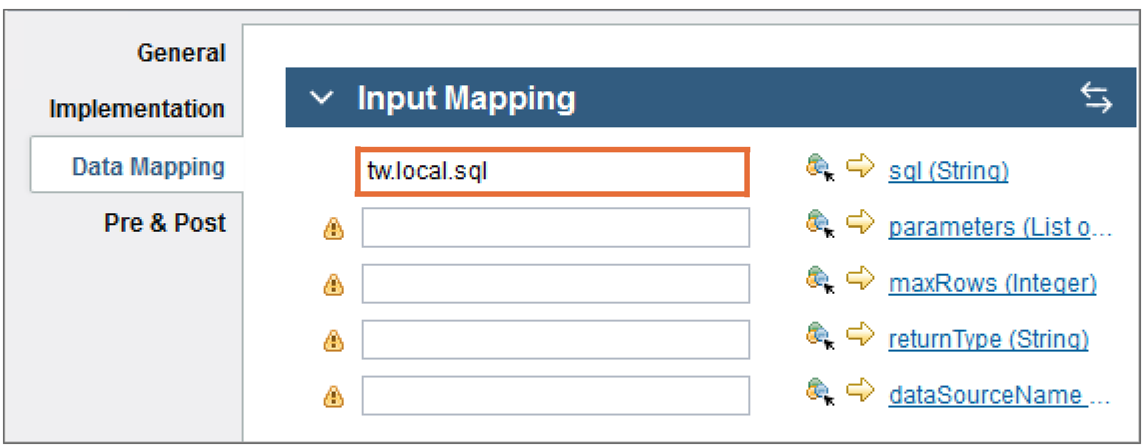



__ 5. Map the inputs and outputs of the `SQL Execute Statement` service.

__ a. Click the **SQL Execute Statement** service step.

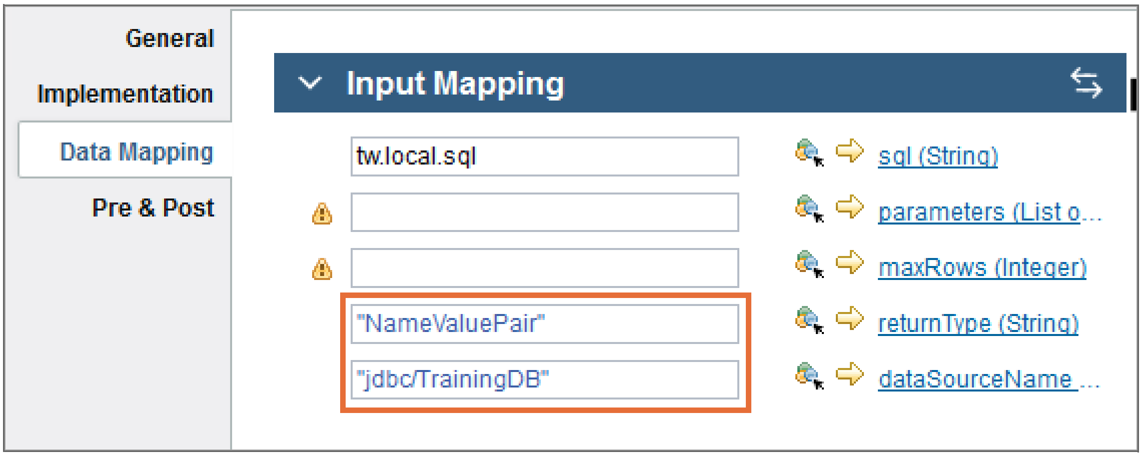
__ b. Click the **Data Mapping** properties menu option.

__ c. In the **Input Mapping** section, map the `tw.local.sql` variable to the `sql (String)` variable.




__ d. For the `returnType (String)` variable mapping, type (include the quotation marks): `“NameValuePair”`

__ e. A JDBC data source was already created for you. In the **dataSourceName** field, type `“jdbc/TrainingDB”` (include the quotation marks).



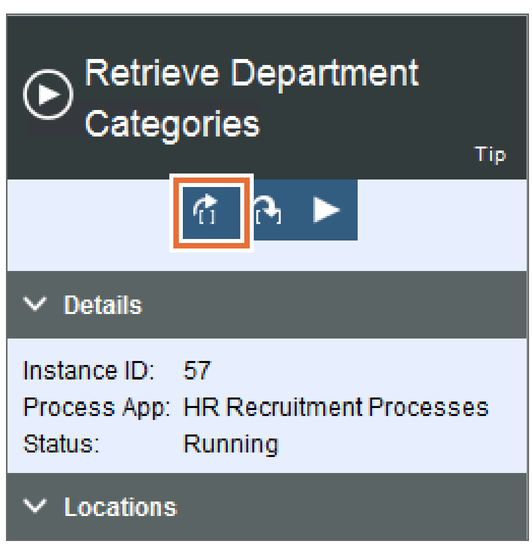

__ f. In the **Output Mapping**, map the `results (ANY)` variable to the `tw.local.departmentCategoriesList` variable.


__ g. Click the **debug** icon to debug the service.

__ h. Make sure that you are on the **Diagram** tab so you see the token move between the steps in the service. Expand the **Data** section on the right and click **Step over**. If needed, you might need to click the Search icon from the top to close the search window.

The first step assigns the string to the sql variable for the database call.

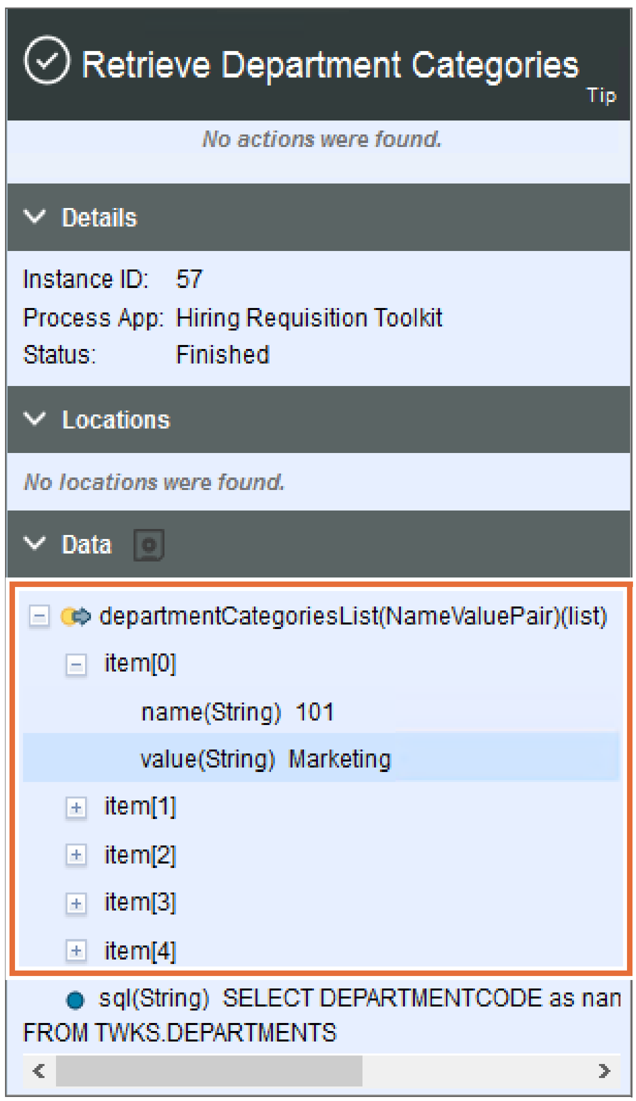
__ i. For the SQL Execute step, click **Step Over** until the status changes to Finished.



**Important**

Two commands in the debugger are for stepping through a nested service: **Step Over** and **Step Into**. When a step in your service is a nested service that contains more than one step, you can use **Step Into** to debug the steps inside the nested service. If you click **Step Over**, the debugger runs all of the steps and moves the token to the end of the nested service, showing only the output after the service is complete.

__ j. Expand the `departmentCategoriesList` in the Data section of the Inspector. The `departmentCategoriesList` variable is populated with the values in the database. Notice that the `departmentCategoriesList` variable contains items 0 - 4 as part of the list. This result means that the database returned five rows, which were mapped to the `NameValuePair` object per the configuration options of the **SQL Execute** service.



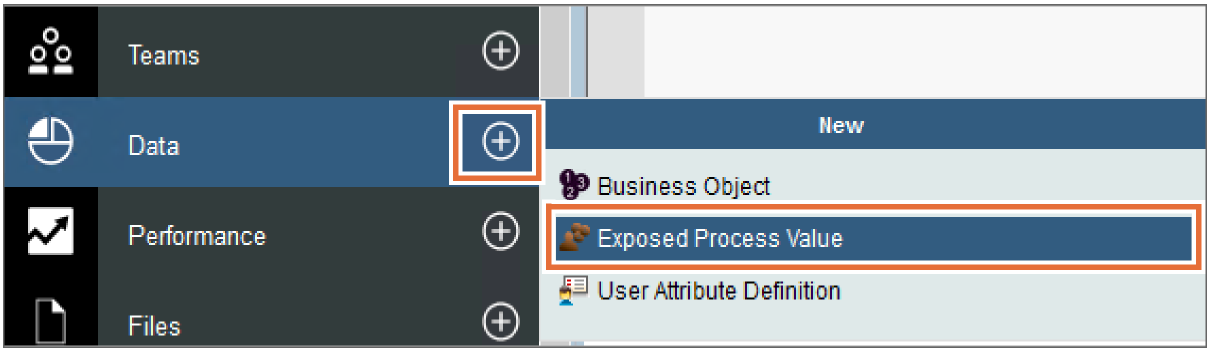

__ k. Click **DESIGNER** to return to the Designer view.

__ 6. Add an exposed process variable (EPV) to manage the data source name.

The data source name should normally be stored as an environment variable. EPVs are designed to store business data because they can be exposed to business users so they can set real-time process variables. In the next two steps, you practice the creation and use of both environment variables and EPVs.

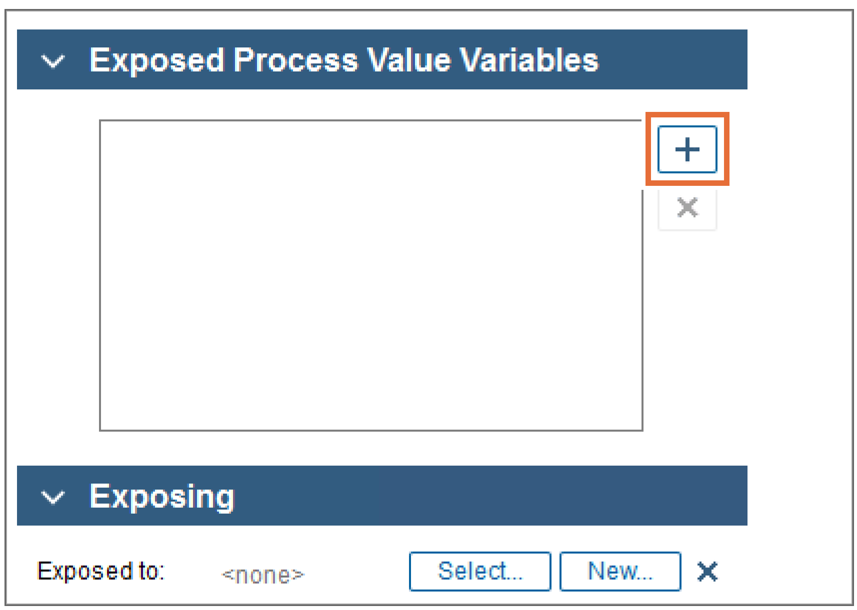
__ a. Click the (**+**) plus sign in the **Data** menu option in the Process Designer library.

__ b. Select **Exposed Process Value**.



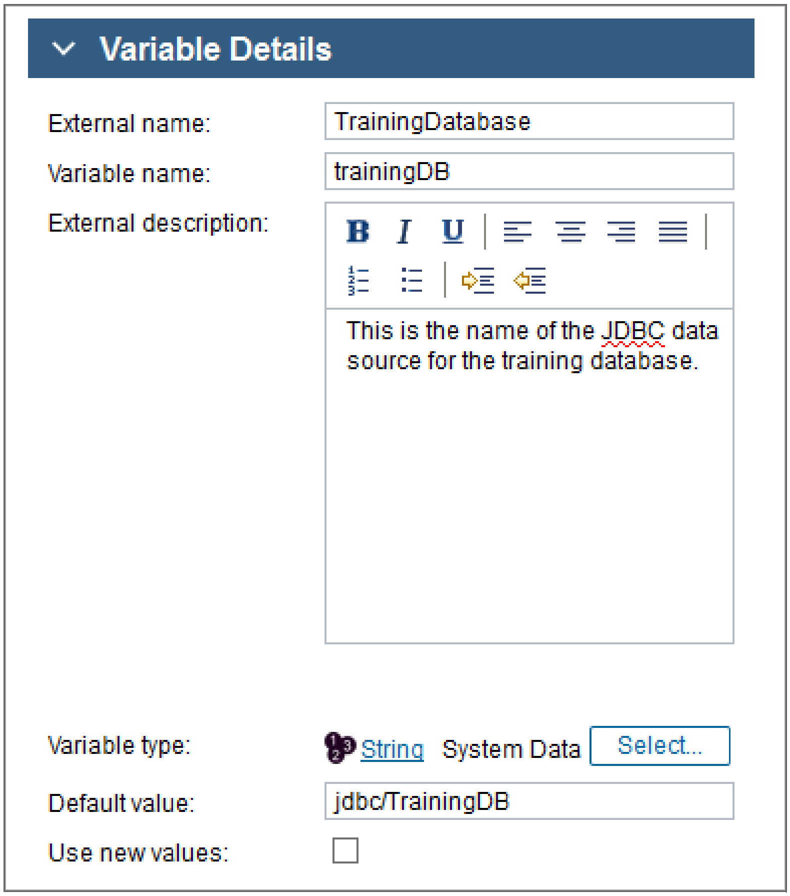

__ c. Name your EPV `DataSource` and click **FINISH**.

__ d. In the **Exposed Process Value** settings, click the (**+**) plus sign in the **Exposed Process Value Variables** section.



__ e. Set the following values in the **Variable Details** section:

- **External Name**: `TrainingDatabase`
- **Variable Name**: `trainingDB`
- **External Description**: `This is the name of the JDBC data source for the training database.`
- **Default Value**: `jdbc/TrainingDB`

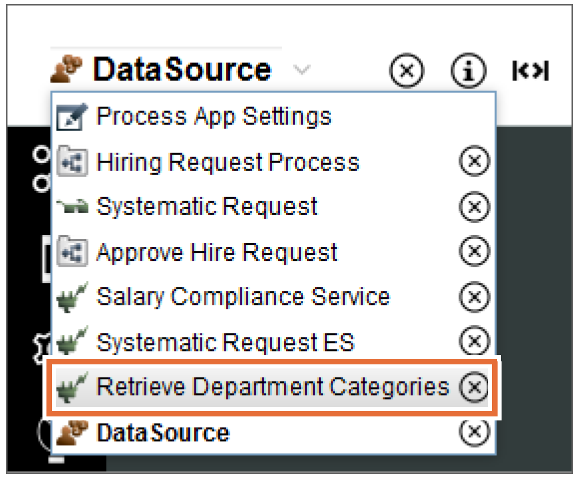



Because the variable type is String, you do not need to enclose the value in quotation marks.

__ f. Save your work.

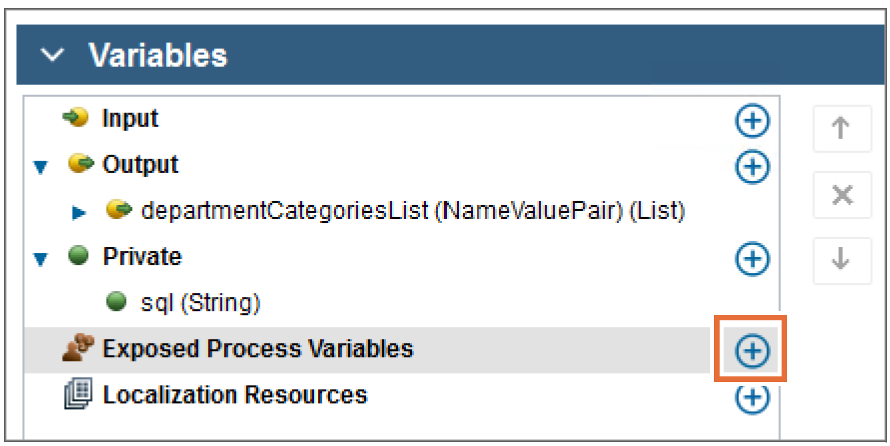
__ 7. Map the EPV

__ a. At the upper left of the designer, open the History menu and return to the **Retrieve Department Categories** service.

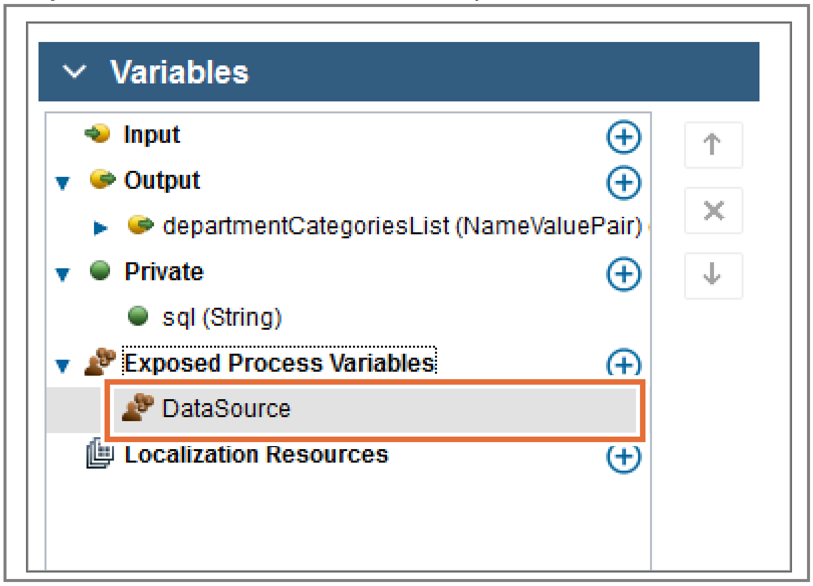



__ b. Open the **Variables** tab.

__ c. Click the (**+**) plus sign next to the **Exposed Process Variables** category in the Variables list, and then select the **`DataSource`** EPV.



__ d. Verify that you added DataSource to the Exposed Process Variables variables list.

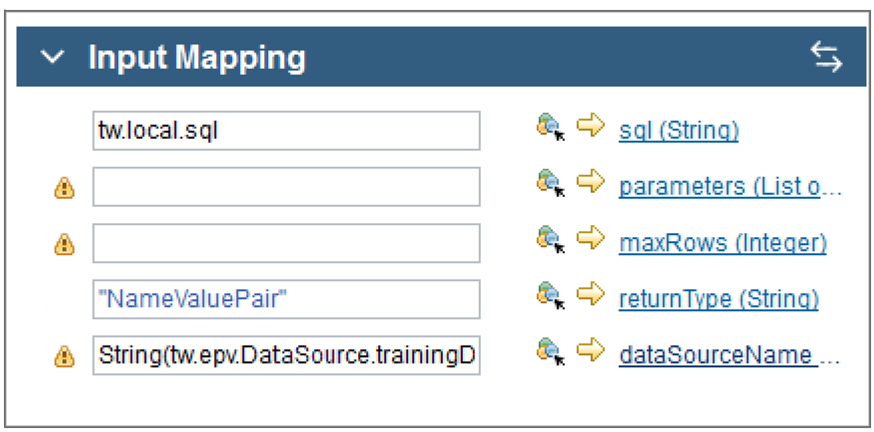



__ e. Click the **Diagram** tab to return to the service objects. Click the **SQL Execute Statement** step.

__ f. In the properties section below, click the **Data Mapping** menu option.

__ g. Map the `trainingDB` EPV to the dataSourceName variable. Enter as the **dataSourceName:** `String(tw.epv.DataSource.trainingDB)`

You must cast your EPV to a String. Replace the string that you previously mapped to the variable with `String(tw.epv.DataSource.trainingDB)`. Although you created the EPV as a String type, the system defines it as an EPV type when you map the variable. Therefore, you must cast the variable to String to use it.

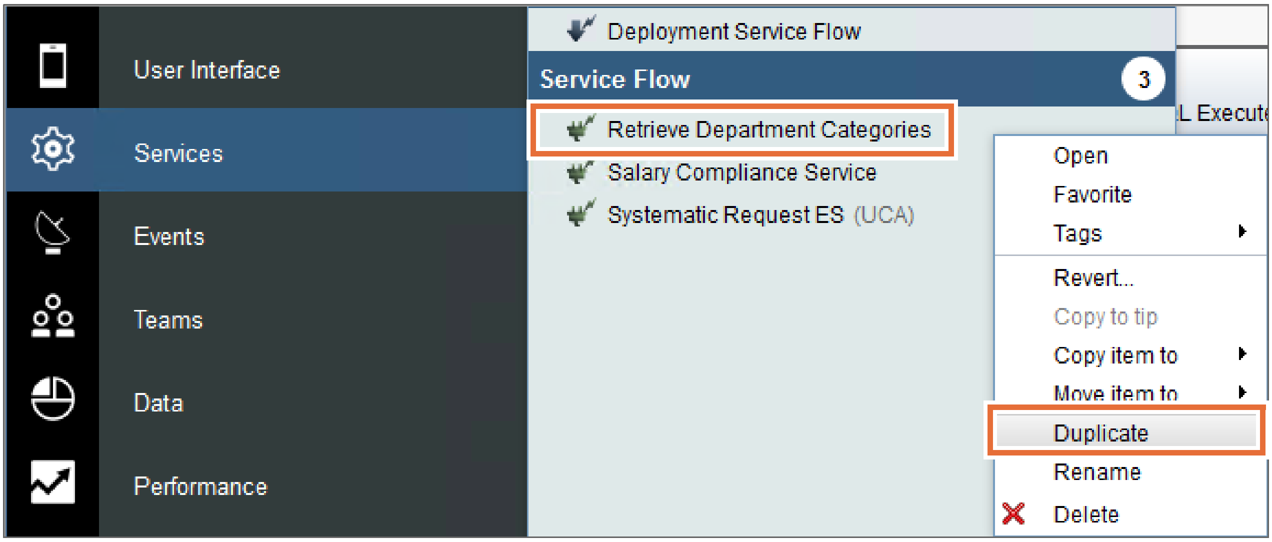



The warning indicator stays because it cannot validate the complex expression. This is normal.

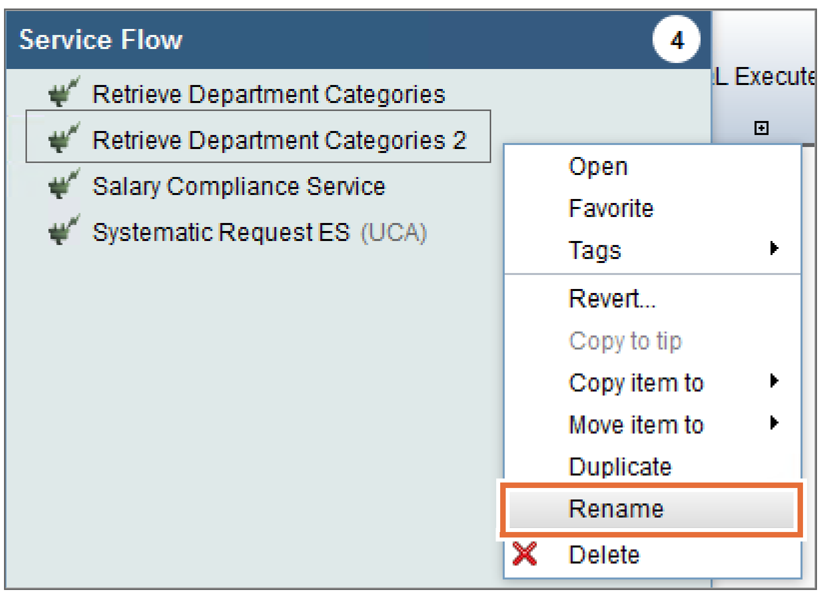
__ h. Save your work.

__ 8. Create a **Service Flow** to retrieve job levels; however, use an environment variable (ENV) to accomplish this implementation.

__ a. In the Designer library, click the **Services** category. Right-click the **Retrieve Department Categories** service flow and click **Duplicate**.




__ b. Right-click the new service and click **Rename**. Name the new service `Retrieve Job Levels.` Click **FINISH**.




__ 9. Change the new service to retrieve job levels.

__ a. Open the new **Services > Retrieve Job Levels** service flow. Because many of these database calls are similar, you just need to change a few things to retrieve the job levels from the database.

__ b. Click the **Variables** tab.

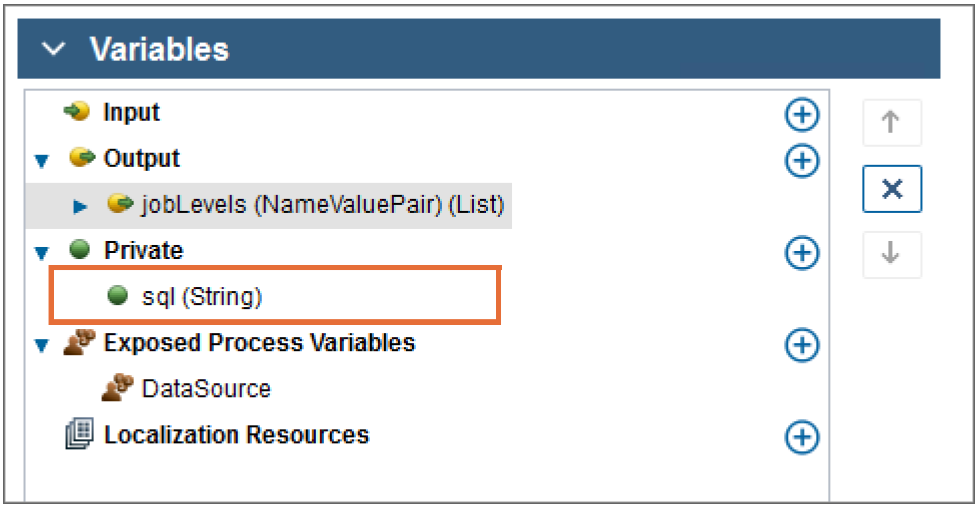
__ c. Click the departmentCategoriesList **Output** variable. Change the variable name to: `jobLevels.`

__ d. Ensure that the **Is List** checkbox is selected, and the Variable Type is **NameValuePair**.

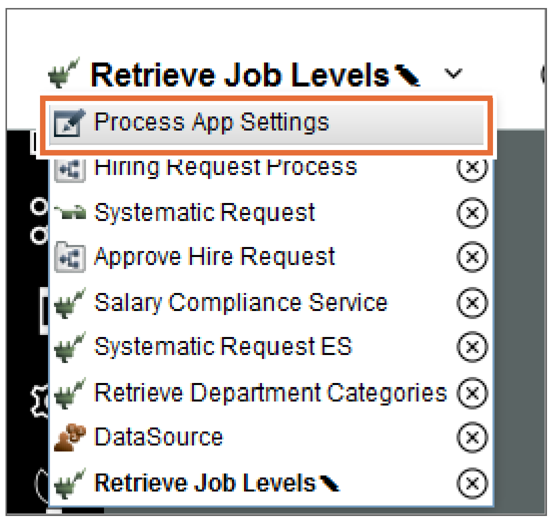
__ e. Save your work.

__ f. Verify the private variable `sql (String)` is still present in the variables list.

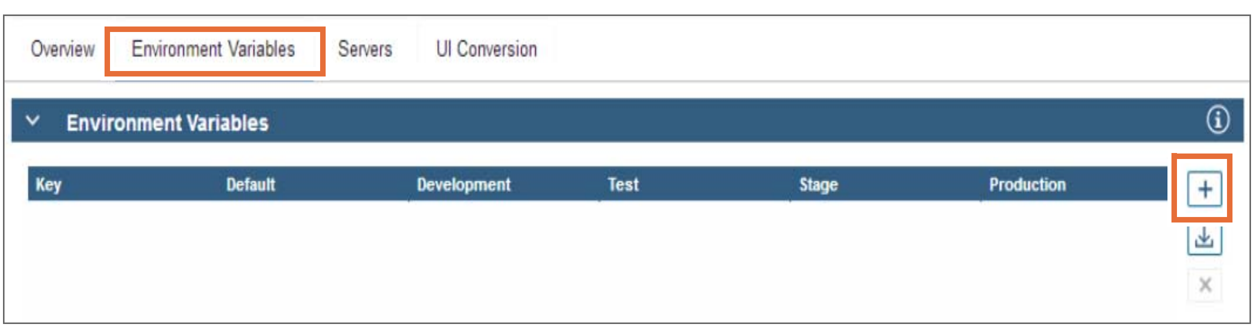



__ 10. Add an environment variable to your process application.

__ a. Open the **History** menu and select **Process App Settings**.

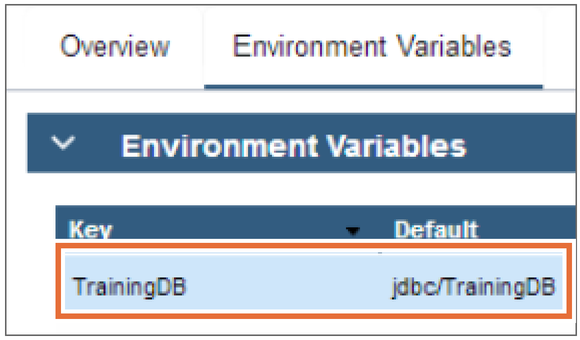



__ b. Click the **Environment Variables** tab and then click the (**+**) plus sign on the right side of the **Environment Variables** section.



__ c. Complete the following values for the environment variable:

- **Key**: `TrainingDB`
- **Default**: `jdbc/TrainingDB`




__ d. Save your work.

__ 11. Implement the `Set SQL` server scriptlet.

__ a. Return to the **Retrieve Job Levels** service flow.

__ b. Click the **Diagram** tab.

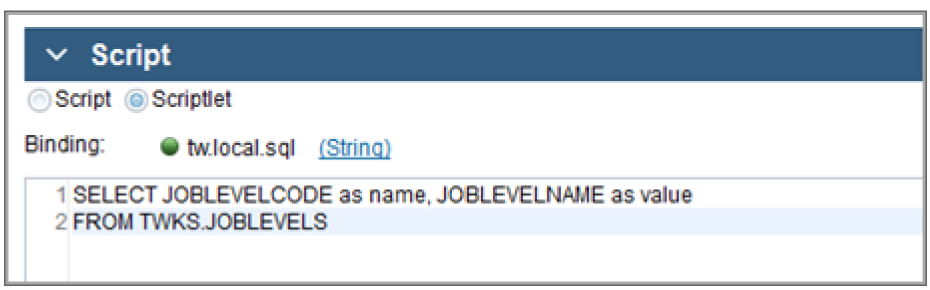
__ c. Click the `Set SQL` server script.

__ d. Click the **Script** properties menu option.

__ e. Verify that the **Scriptlet** type is selected and that the scriptlet is bound to the `tw.local.sql (String)` variable.

__ f. Change the **Implementation** script to:

```
SELECT JOBLEVELCODE as name, JOBLEVELNAME as value
FROM TWKS.JOBLEVELS
```




Alternatively, you can copy the script from the `Script3.txt` file that is located in the `C:\labfiles\Support files\Ex8` folder.

__ g. Save your work.

__ 12. Map the inputs and outputs of the `SQL Execute Statement` service.

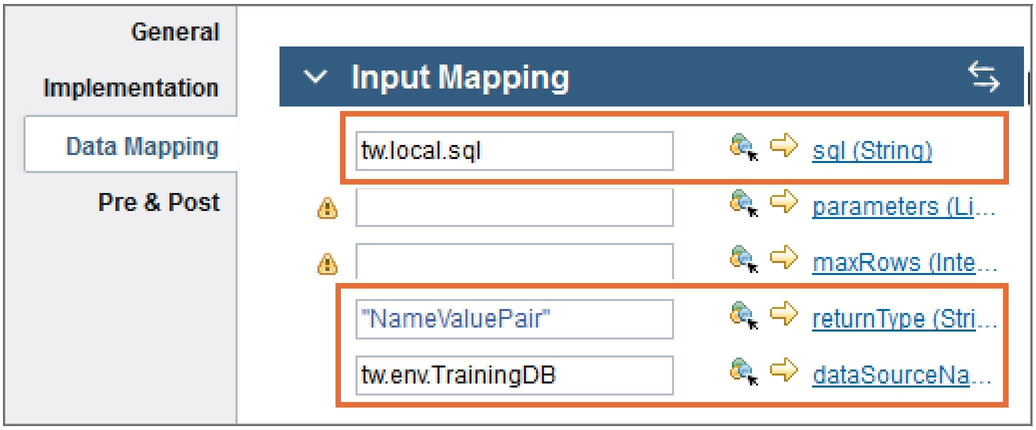
__ a. Click **SQL Execute Statement** on the canvas.

__ b. Click the **Data Mapping** properties menu option.

__ c. Under the **Input Mapping** section, verify that the `sql (String)` input is mapped to the `tw.local.sql` variable.

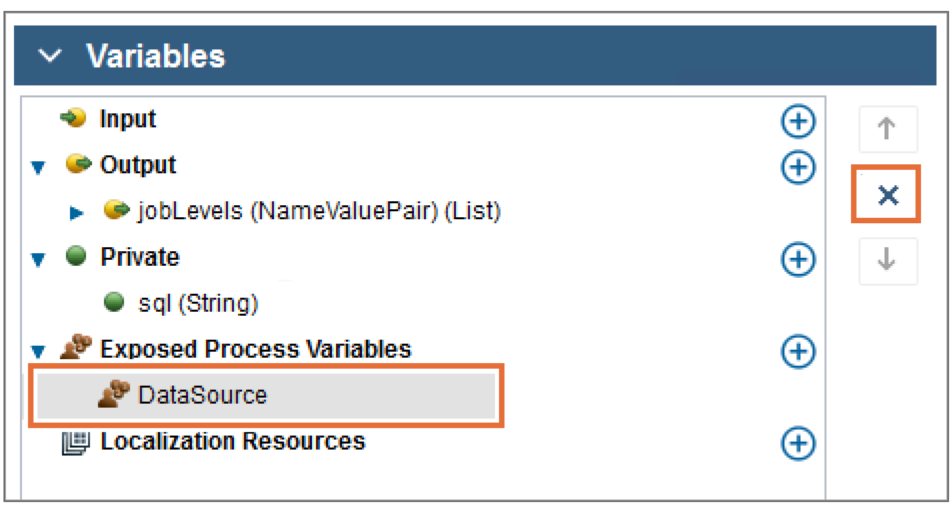
__ d. Verify the `returnType (String)` input is "`NameValuePair`" (include the quotation marks).

__ e. For the `dataSourceName (String)` input, enter: `tw.env.TrainingDB`




__ f. Change the **Output Mapping** section to the `tw.local.jobLevels` variable.


__ g. Now you can remove the DataSource Exposed Process Variable from the Variables tab that was added to the original service and duplicated for this service.




__ h. Save your work.

__ 13. Debug the service to determine whether the data is retrieved from the database correctly.

__ a. Click the **debug** icon above the palette.

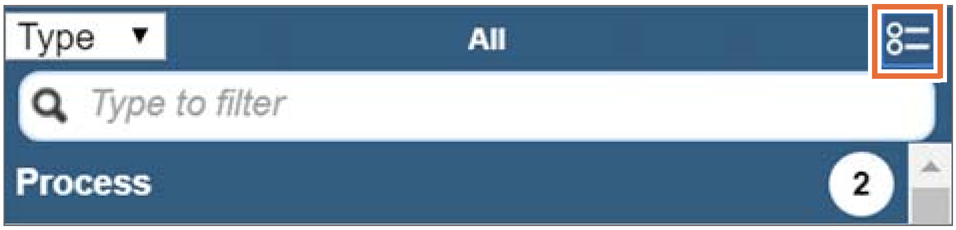
__ b. Click the **Step over** icon twice to step through the service and verify that the database call runs and returns data from the database.


__ c. Click **DESIGNER** to return to the Designer view.

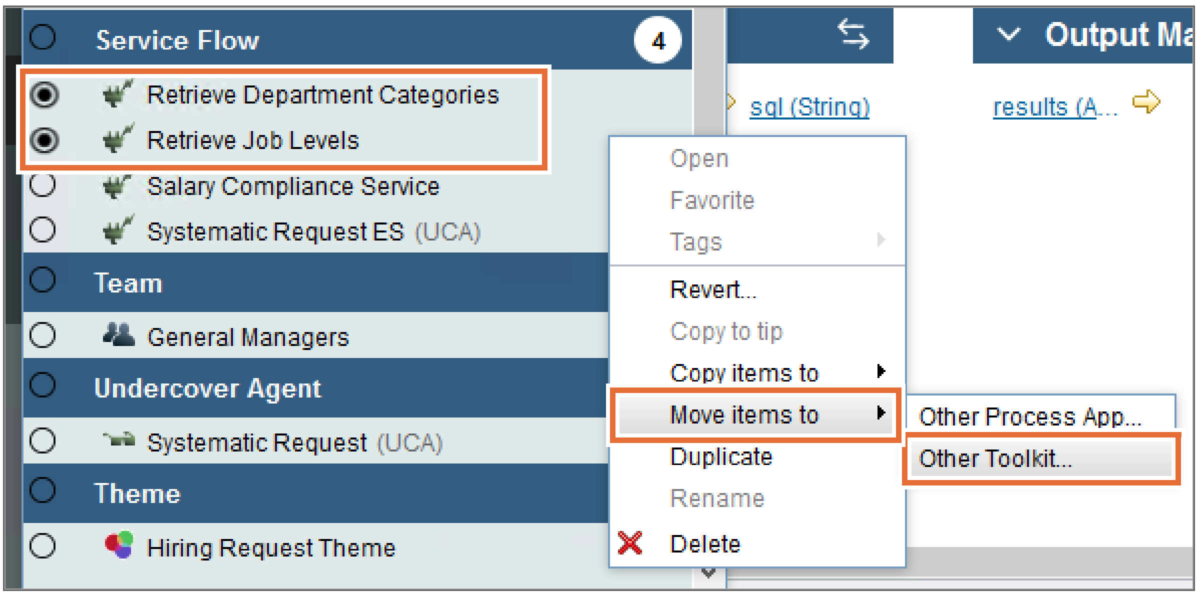
__ 14. To make these services available to other process applications, move the services to the toolkit.

__ a. At the top of the library, click **HR Recruitment Process** to view all the assets in your library.

__ b. On the top of the asset list, click the icon to change to **multi-selection mode**.

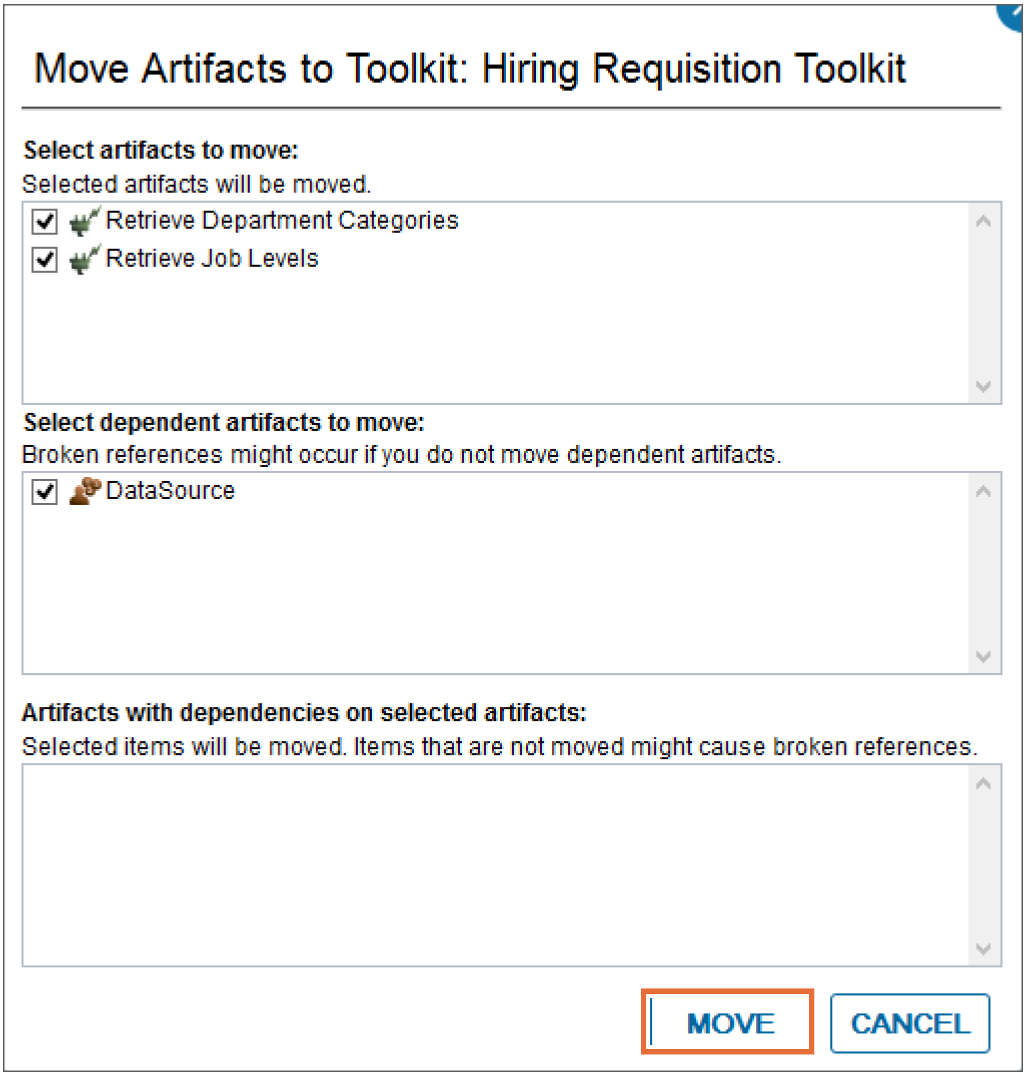



__ c. Radio buttons appear to the left of every asset in the library. Select both the **Retrieve Department Categories** and **Retrieve Job Levels** Service Flow services. Right-click and click **Move items to > Other Toolkit.**



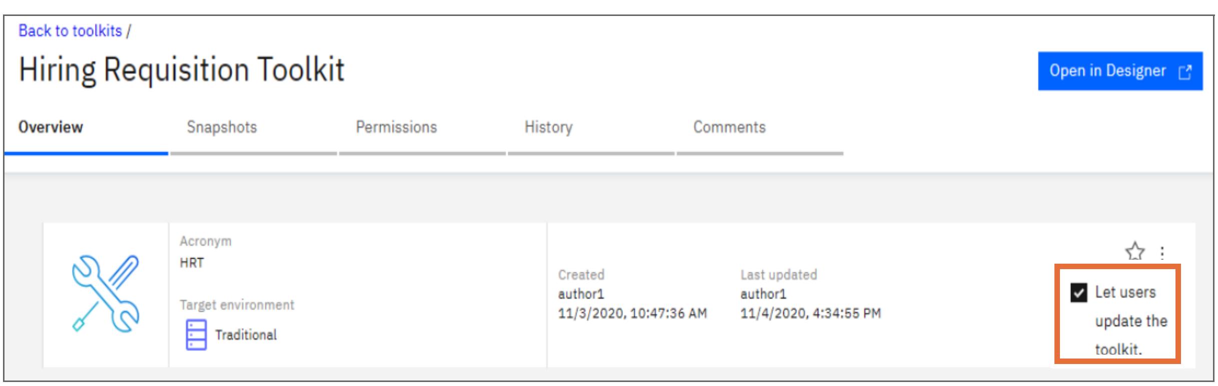

__ d. Click **Hiring Requisition Toolkit**.

__ e. The system identifies the dependencies. Click **MOVE**.



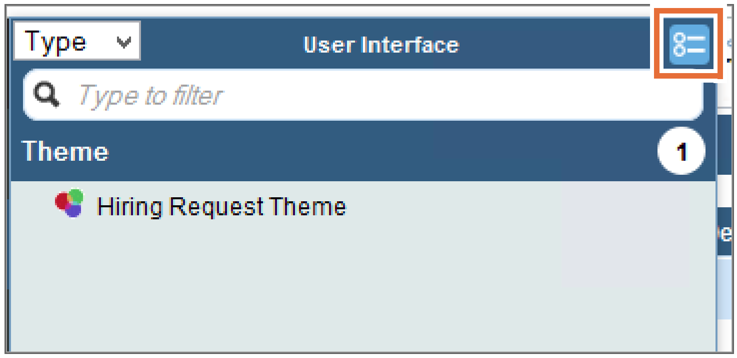

## Troubleshooting

If the Hiring Requisition Toolkit is not listed, verify **Let users update the toolkit** is selected.



__ 15. In the last exercise, you created a Hiring Request Theme. To make this theme available to other process applications, move the theme to the Hiring Requisition Toolkit.

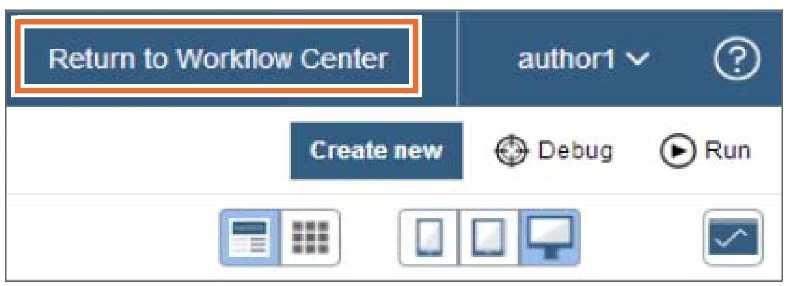
__ a. Open the **User Interface** category in the library. Turn the **multi-select** off for this next move.



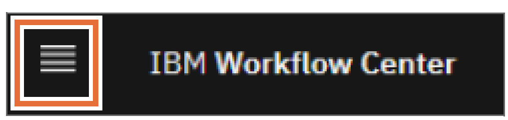

__ b. Right-click **User Interface > Hiring Request Theme** in the library. Click **Move item to > Other Toolkit**, and select the **Hiring Requisition Toolkit**.

__ c. In the Move Artifacts to Toolkit window, click **MOVE**.

__ d. Finally, because the ENV variables are specific to the toolkit or Process App, you need to re-create the environment variable for this toolkit. Click the **Return to Workflow Center** link.




__ e. Click the navigation icon in the upper left.



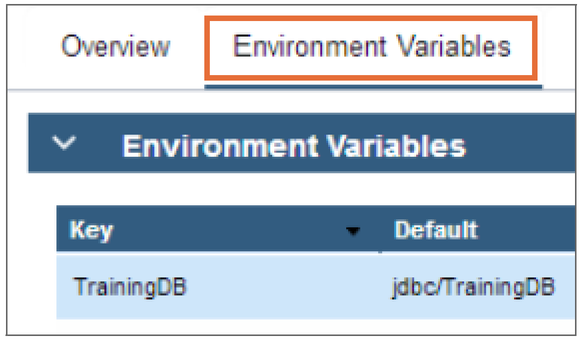

__ f. Select **Toolkits**.

__ g. Click the **Hiring Requisition Toolkit** tile.

__ h. At the top, click the **Environment Variables** tab.

__ i. In the Environment Variables section, add the `TrainingDB` key with the default value of `jdbc/TrainingDB`. This setting is needed to run the Retrieve Job Levels service. Click the (**+**) Add icon from the right to add a variable.

**Key**: `TrainingDB`
**Default**: `jdbc/TrainingDB`



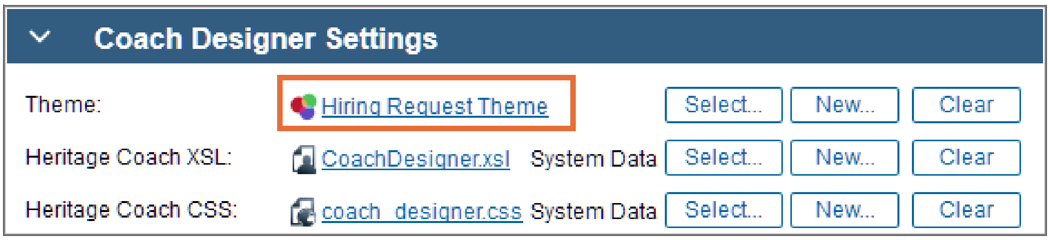

__ j. Save your work.

## *Part 5: Change an input to a single select on a coach*

Until now, any value would be accepted for the Department input on the page. If only certain values are allowed, you must then approach the control in one of the following ways. The first way is to limit the input to only acceptable values by using a selection like the single select on a coach. The second way is through validation of the input data. Many times the solution is a combination of both of these approaches. This section demonstrates how to create a single selection to limit the values that are allowed.

__ 1. Apply the Hiring Request Theme to the Hiring Requisition Toolkit.

__ a. In the **Overview > Coach Designer Settings** section, click **Select** and select the **Hiring Request Theme**.

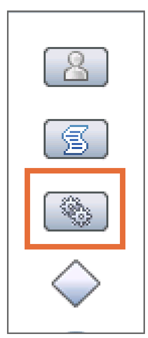



__ b. Save your changes.

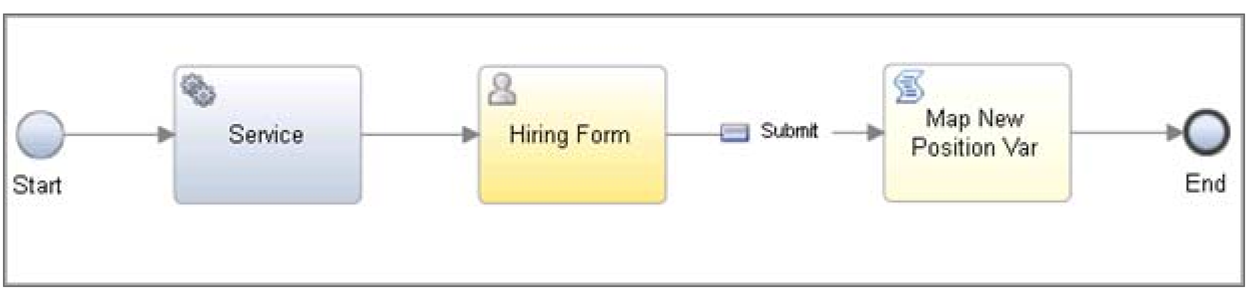
__ 2. Add the **Retrieve Department Categories** service to the **Hiring Form** human service.

__ a. Select **User Interface** from the library and click **Hiring Form**.

__ b. The **Hiring Form** service opens. In the **Diagram** tab, drag a **Service** from the palette onto the **Hiring Form** sequence flow between the **Start** event and the **Hiring Form** coach.

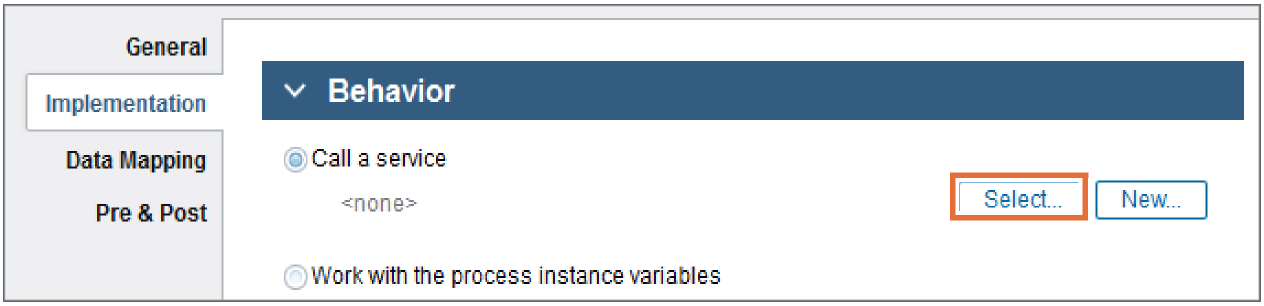


__ c. Verify the flow so that the new service is part of the flow.

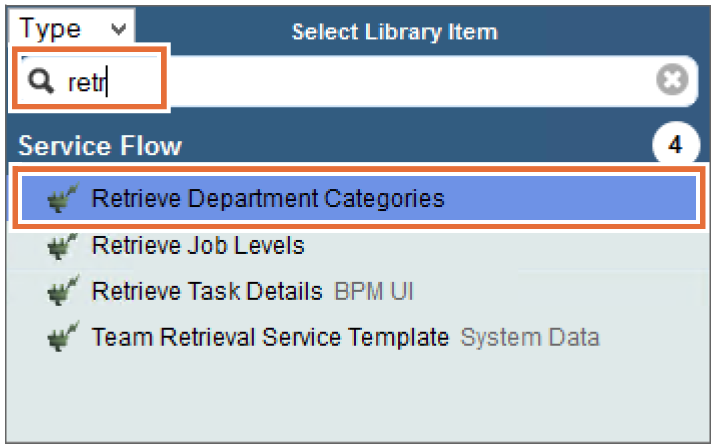



__ d. Rename the new service step to: `Retrieve Department Categories`

__ e. In the **Implementation** properties menu, click **Select** for the **Call a service** setting.



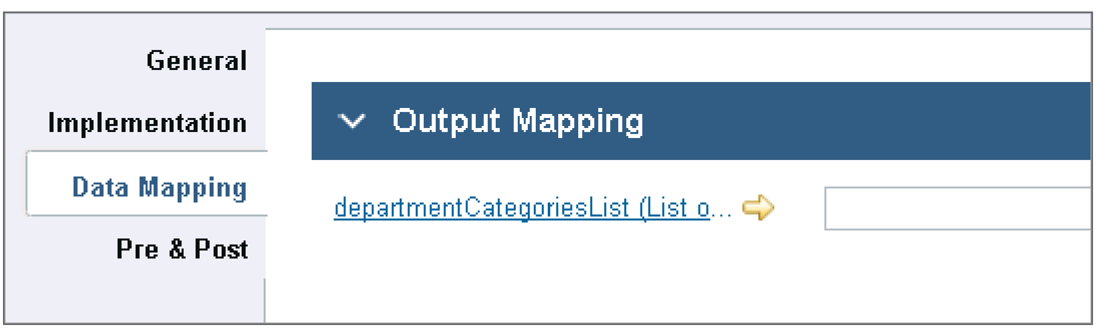

__ f. Select the **Retrieve Department Categories** service that you created.



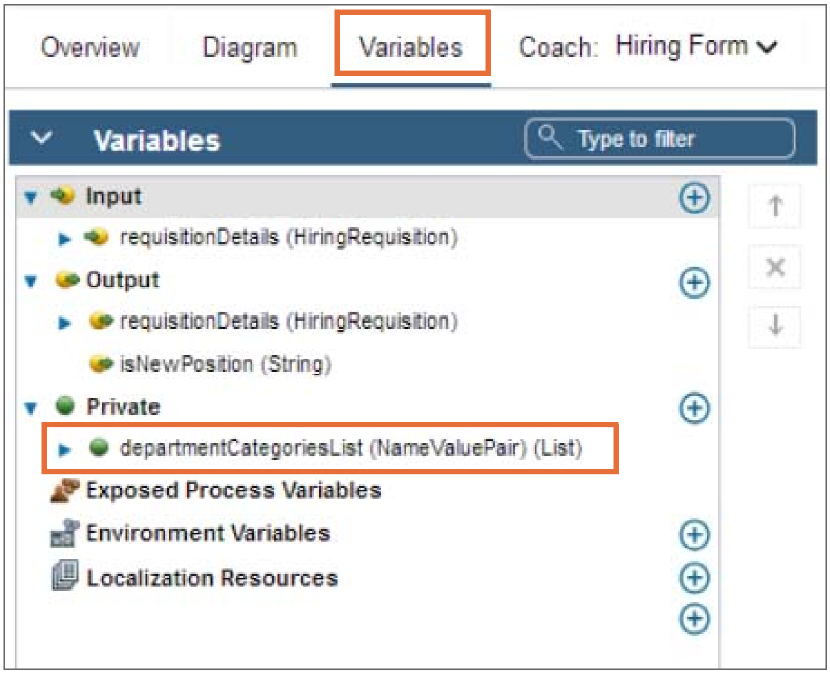

__ g. Save your work.

__ 3. Map the output variable of the service and allow the system to create the private variable.

__ a. In the **Data Mapping > Output Mapping** properties section, note that the service requires you to map a variable to the `departmentCategoriesList` variable.

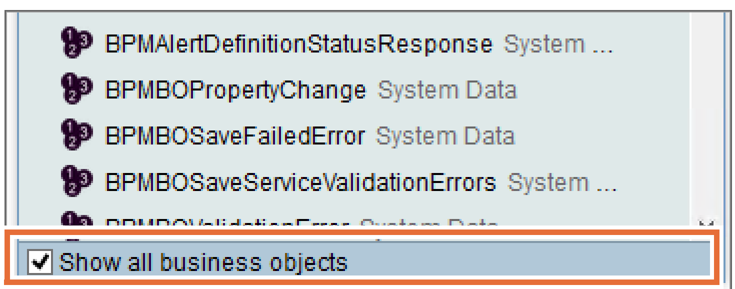


__ b. Click the **Variables** tab. Create a private variable: `departmentCategoriesList (NameValuePair) (List)`. Make sure the variable is configured as a List.




**Reminder**

Select **Show all business objects** to choose the `NameValuePair` variable type.



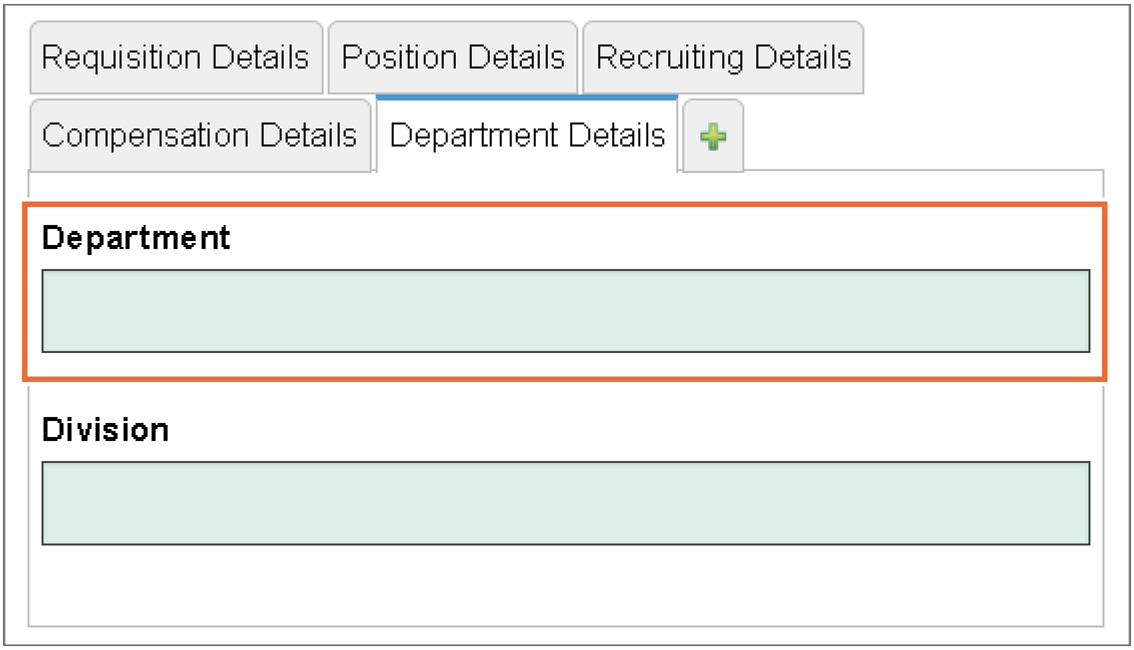

__ c. Return to the **Diagram** tab. Verify that the **Retrieve Department Categories** step on the canvas is selected. In the **Data Mapping** properties section, map the `departmentCategoriesList` variable to the output `tw.local.departmentCategoriesList`.


__ d. Save your work.

__ 4. Change the control on the **Hiring Form** coach to a single select.

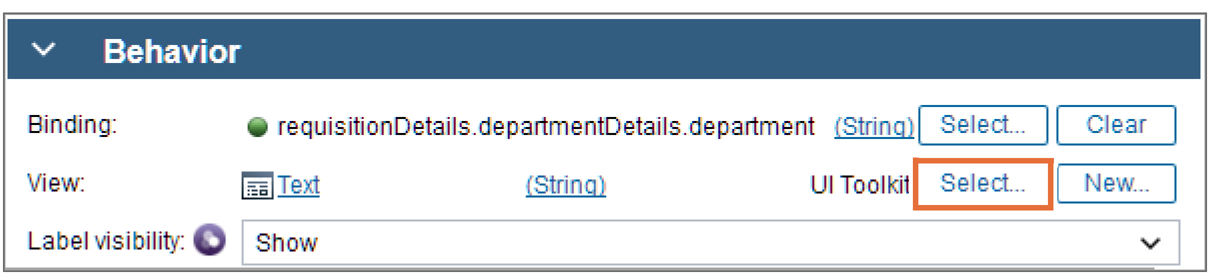
__ a. Click the **Coaches** tab and the **Hiring Form** coach. Initially when you view the coach, the server must generate the CSS and render the new coach on the screen, so a delay might occur before you see the effect.

__ b. From the **Department Details** tab, select the control **Department** that is bound to the department variable.



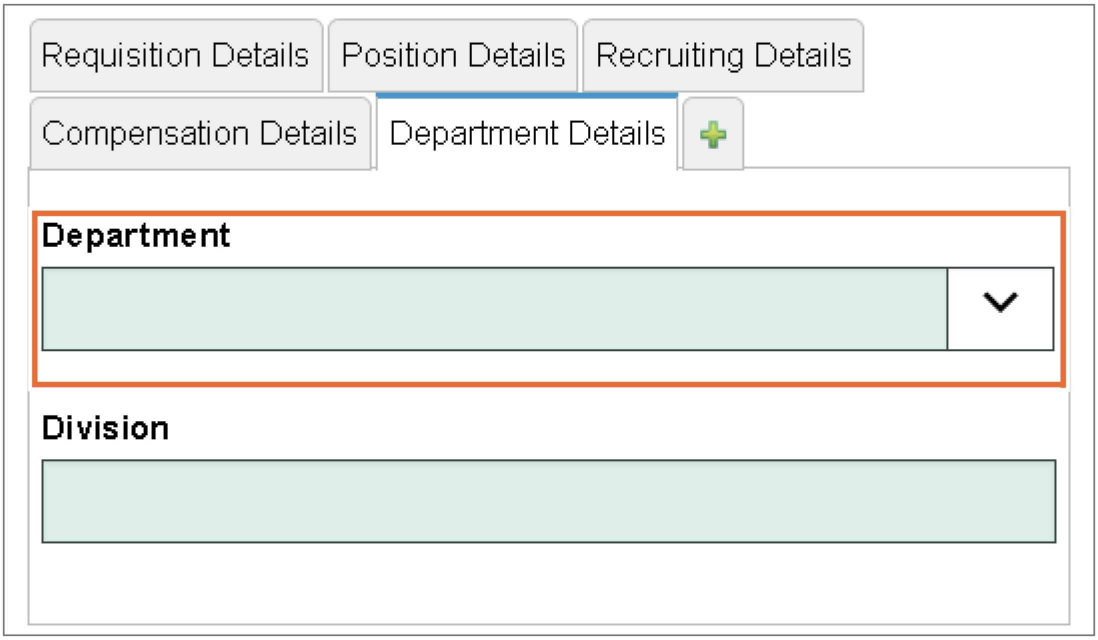

Change the view that the control is bound to. Right now, all the input boxes on the coach are text views. These controls are all variables that are bound to String data types.

__ c. In the **General > Behavior** properties section, click **Select** next to View.




__ d. Select the option **Single Select (UI Toolkit)** coach view.

__ e. After you change the view binding, the control on the palette has a different look. It is now a **Single Select** view control.




__ f. Save your work.

__ 5. The select control returns the single element that is chosen from the list, and you must create a simple list variable to store the selection.

__ a. Click the **Variables** tab of the **Hiring Form** service.

__ b. Create a private variable: `department (NameValuePair)`


__ c. Save your work.

__ 6. Store the selected department to the department variable.

__ a. Return to the Coach.

__ b. Select the **Department** control under the **Department Details** tab.

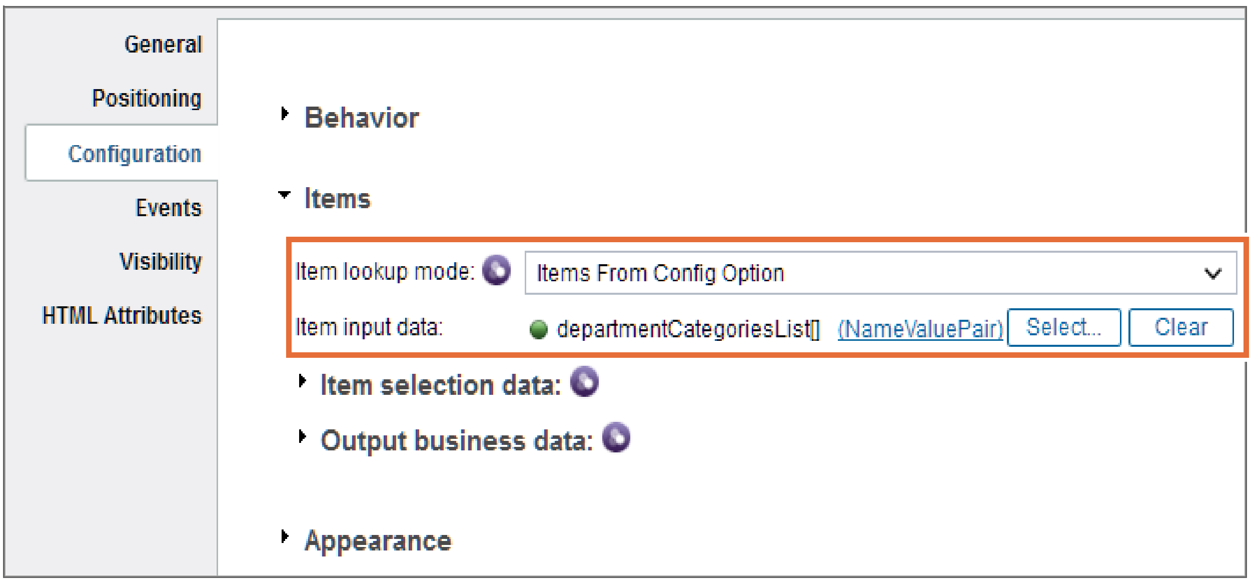
__ c. In the **General > Behavior** properties section, click **Select** next to the **Binding** property.

__ d. Click the new Private variable **department (NameValuePair)** from the variable list.

__ 7. Configure the setting that determines the list of selections available to the user on the `Department` select control.

__ a. Click the **Configuration** properties menu. Expand the **Items** section.

__ b. Change the Item lookup mode to **Items from Config Option**. Click **Select** next to the Item input data setting and click the **departmentCategoriesList (NameValuePair)** variable.




**Information**

Because the control uses the `.name` variable for the **Display Name Property** and the `.value` of the **Value** property settings by default, you do not have to set those settings. If you are using a variable other than a `NameValuePair`, you must assign the values that populate the name and value of the control.

__ c. Save your work.

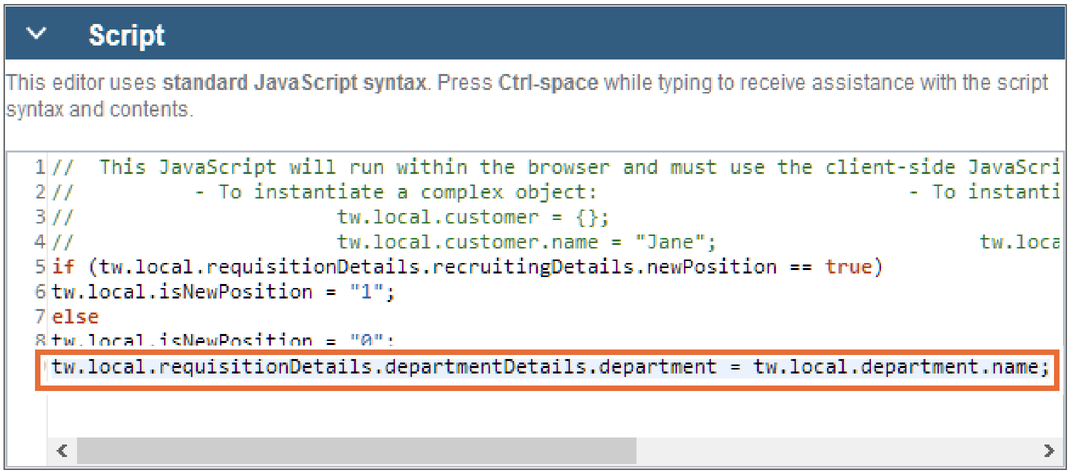
__ 8. The Department selection that is stored in the private `department (NameValuePair)` variable must be mapped to the parameter of the business object. Bind the selection value back to the business object parameter.

__ a. In the **Hiring Form** service, click the **Diagram** tab.

__ b. Change the name of the **Map New Position Var** script to: `Map Vars`

__ c. In the **Script** properties menu, add the following code to the bottom of the script:

```
tw.local.requisitionDetails.departmentDetails.department =
tw.local.department.name;
```




Optionally, you can copy the script from the `Script4.txt` file that is located in the `C:\labfiles\Support files\Ex8` folder

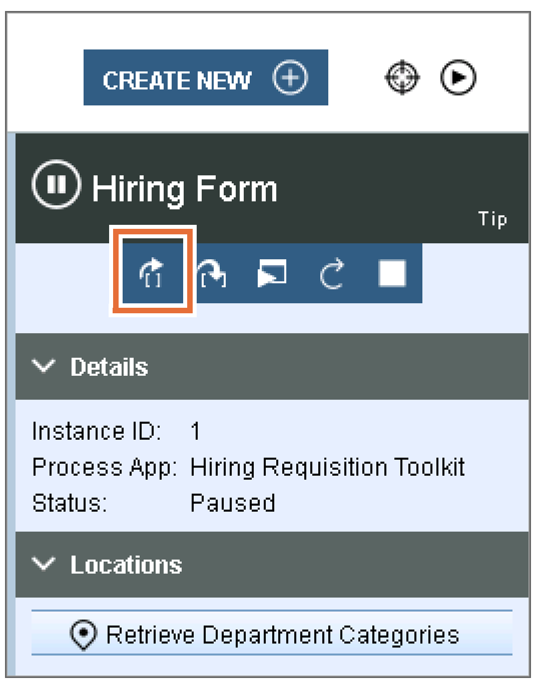
__ d. Save your work.

__ 9. Debug the coach, and verify that the **Department** single select works and binds the data to the variable.

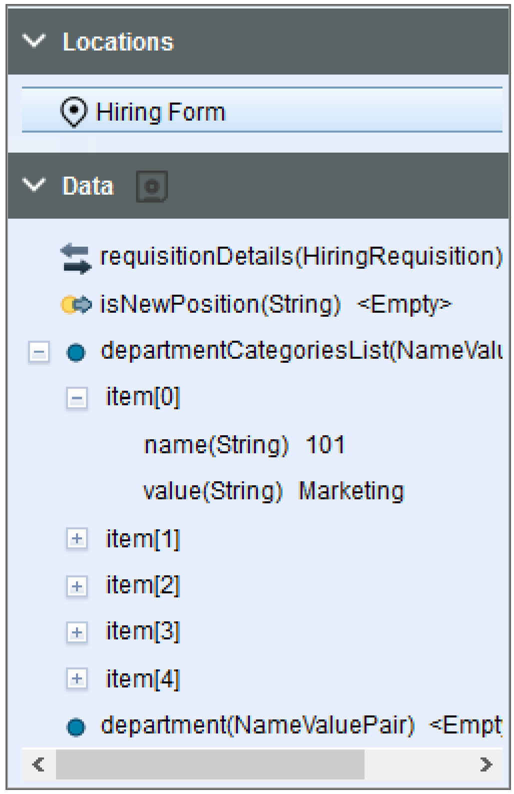
__ a. Click the **Debug** icon.

__ b. When the debug window opens, minimize the new window.

__ c. Click **Step over**.

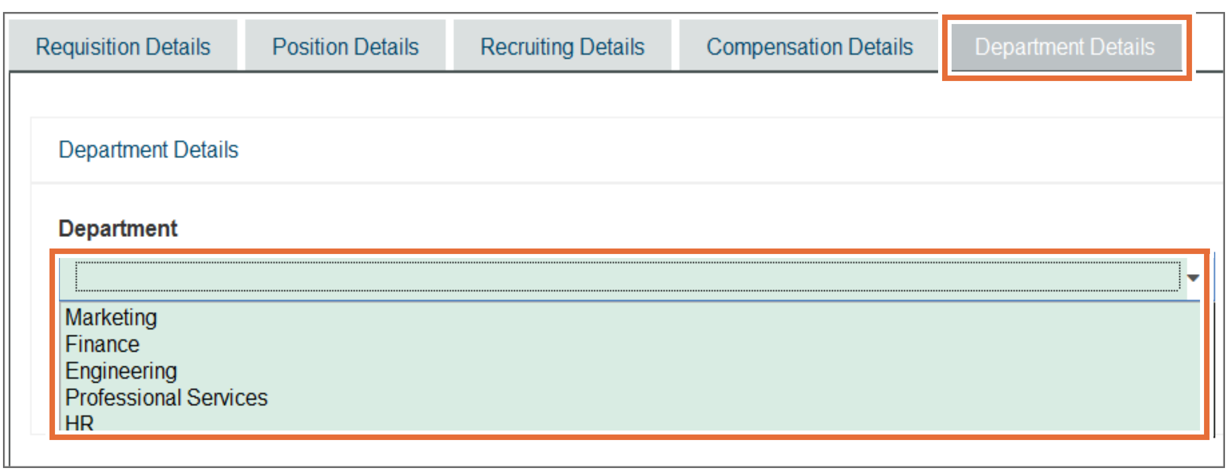


__ d. Verify that the `departmentCategoriesList` variable is populated with data from the integration that you created in the last exercise.

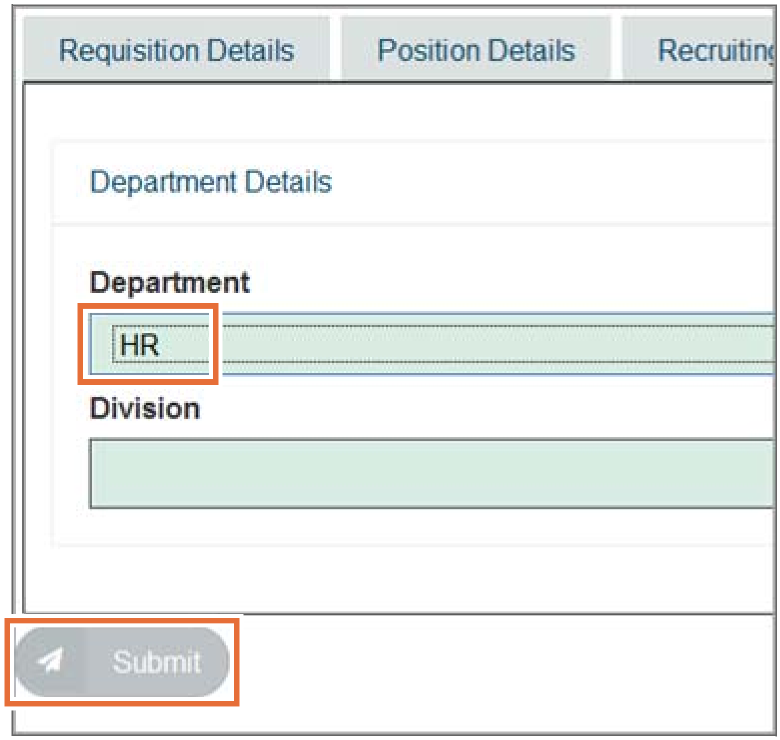



__ e. Click **Step over** to show the coach. The coach is displayed in the Playback window.

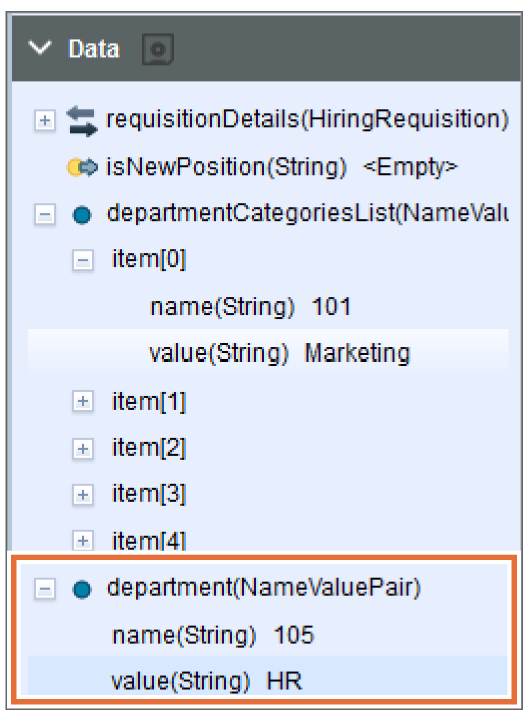
__ f. Click the **Department Details** tab. The database retrieval service now drives the selections that are presented to the user.



__ g. In the **Department** field, select **HR** and click **Submit**.




__ h. Verify that the data is bound to the variable inside the debugger.



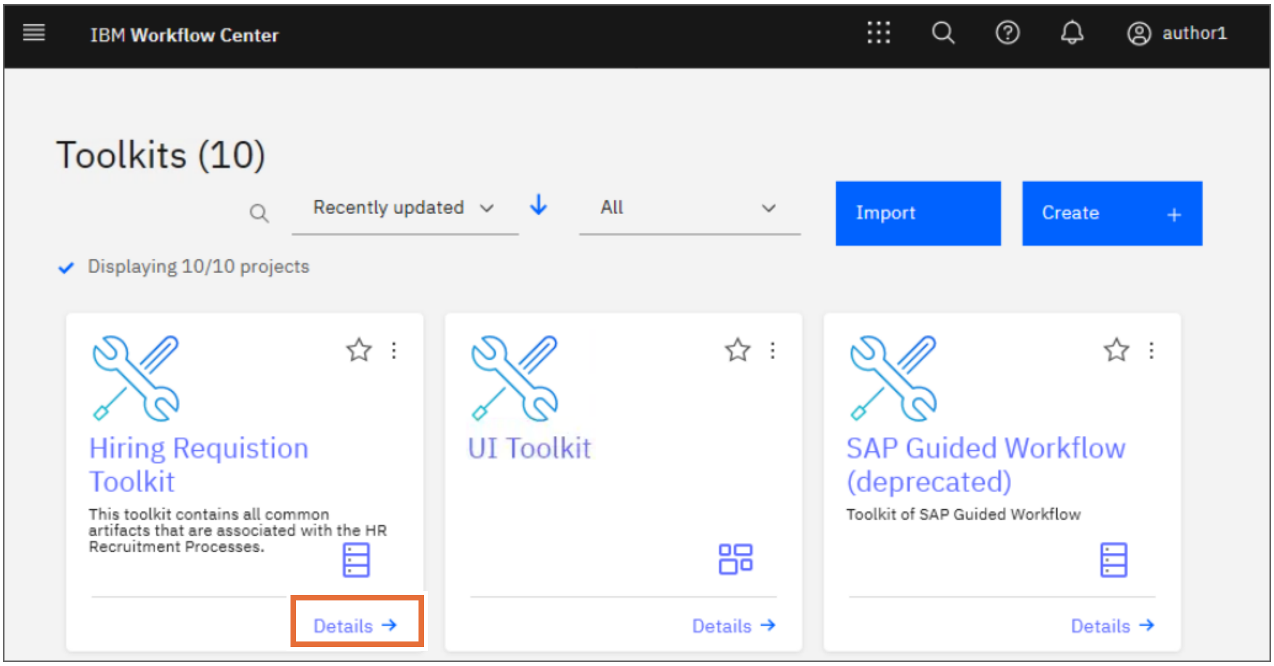
__ i. Click **Step Over**. Verify that the data was mapped into the `requisitionDetails.departmentDetails.department` variable. The `Map Vars` script stores the key for the object, not the value.


__ j. Close the **Debug Hiring Form** browser window.

__ 10. Update the process application HR Recruitment Processes dependency to the most recent version of the Hiring Requisition Toolkit.

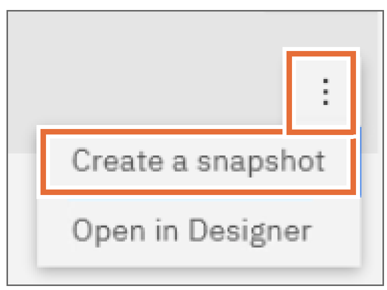
__ a. Return to Process Designer and click **Return to Workflow Center**.

__ b. Click **Details** on the Hiring Requisition Toolkit tile.




__ c. Select the **Snapshots** tab.

__ d. Open the list of options by clicking the **three dots** on the right of the **Current** snapshot tile and select **Create a snapshot.**




__ e. Name the snapshot `Playback 2` and provide the following documentation: `This snapshot includes integrations.` Click **Create**.

__ f. Click the navigation icon in the upper left and select **Process apps**.

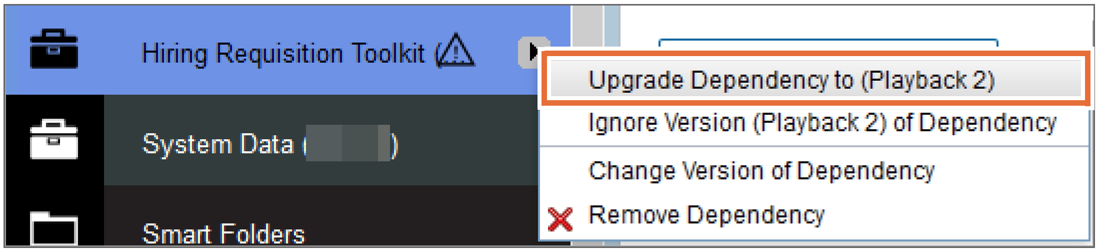
### Troubleshooting

If the Process apps list does not display and still displays the Toolkits, wait a minute for the refresh to occur.

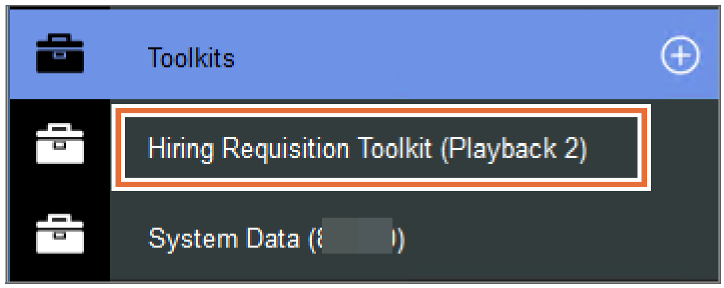
__ g. Open the **HR Recruitment Process** process application.

__ h. Expand the **Toolkits** category in the library.

__ i. Click the arrow next to **Hiring Requisition Toolkit (Baseline)** and click **Upgrade Dependency to Playback 2**.




__ j. The dependency is now upgraded to Playback 2.




__ k. Playback 2: Integrations, is now complete.

__ l. Click **Return to Workflow Center**.

You have completed this exercise.

The first part of this exercise looked at creating a decision service. You created rules in Business Action Language (BAL) to support the service and then implemented the new service in a process. Next, you created a message start event as a trigger to run the process. You organized assets with tags. Finally, you built a query service to read information from a database. The query service is built with environment variables and exposed process variables. You then implemented the query service as a single-select control on the user interface.

## End of exercise

# Exercise 9. Playback 3: Handling errors and deploying your process application

## Estimated time

01:00

## Overview

This exercise covers how to implement error handling in a service.

## Objectives

After completing this exercise, you should be able to:

- Harden a service with a catch exception component
- Create a snapshot for deployment

## Introduction

In this exercise, inside the Process Designer service diagram, you drag a Catch Exception and attach it to a generic service or a human service. You then create a flow from the Catch Exception. Finally, you create a snapshot for deployment.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

# Exercise instructions

## *Part 1: Implement exception handling in a service*

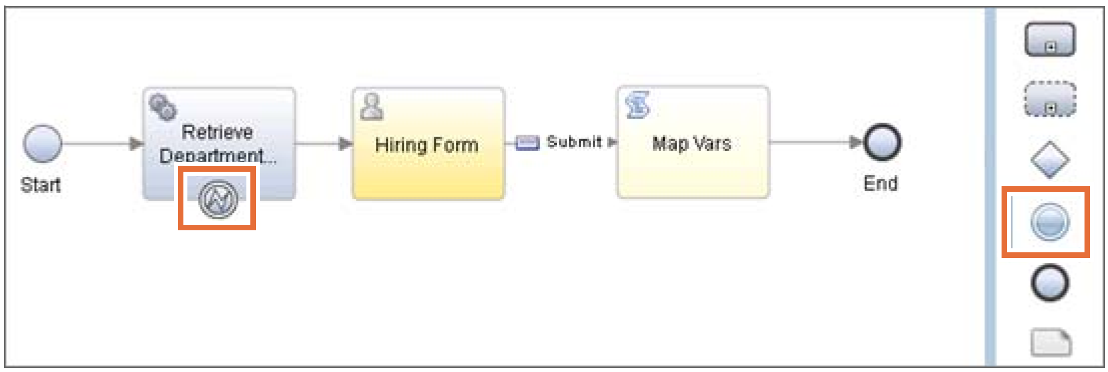
__ 1. Add a Catch Exception event to the Hiring Form service.

__ a. Open the **Hiring Requisition Toolkit**.

Be sure to open the toolkit and not the process application.

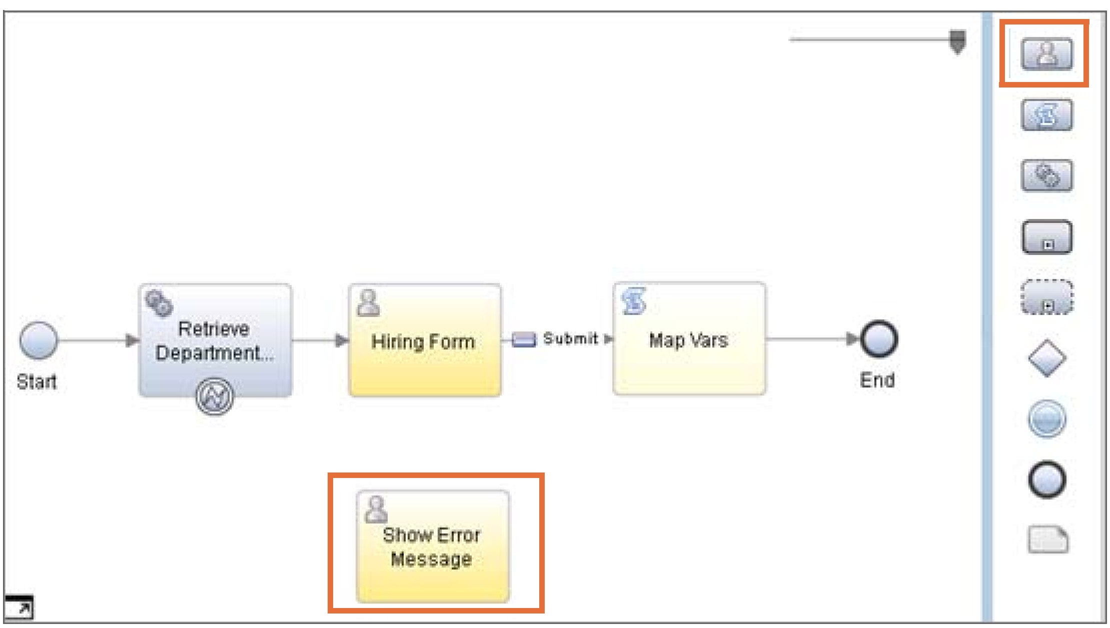
__ b. In the Hiring Requisition Toolkit, open the **User Interface > Hiring Form** human service.

__ c. On the **Diagram** tab, drag an **Intermediate Event** from the service palette on to a control point on the Retrieve Department Categories service.



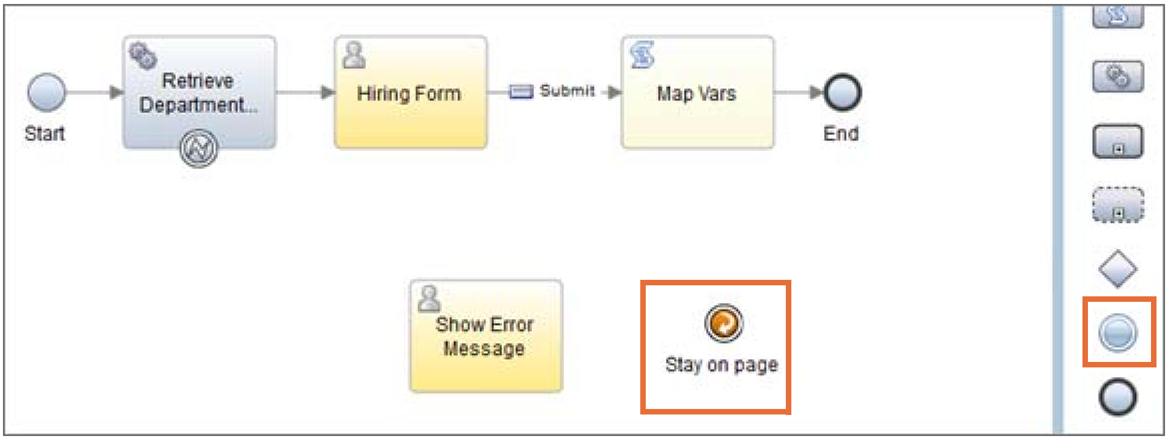

__ 2. Add a coach to display an error message.

__ a. Drag a **Coach** from the service palette onto the canvas. Rename the coach: `Show Error Message`

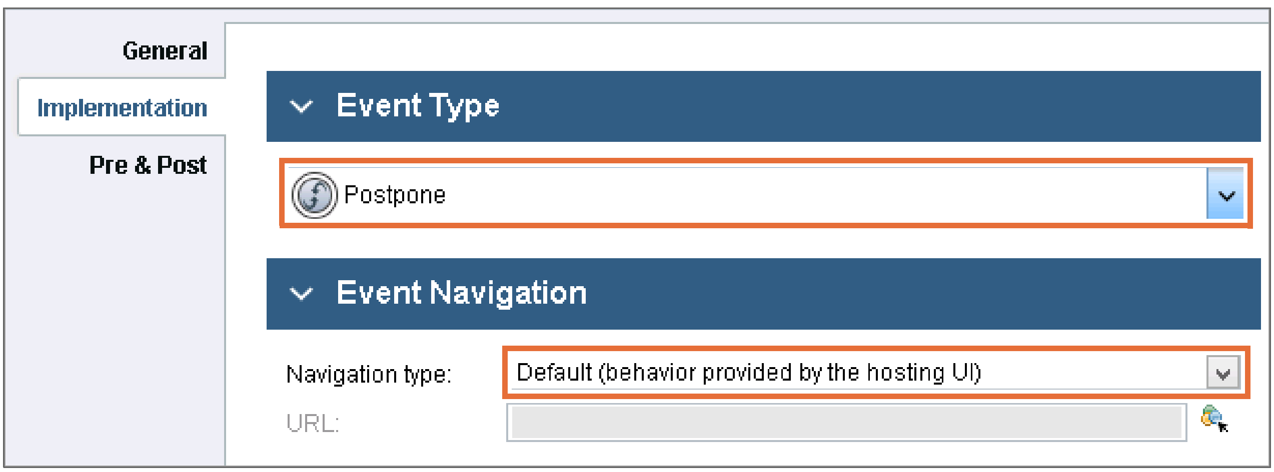


__ 3. Add a postpone task service.

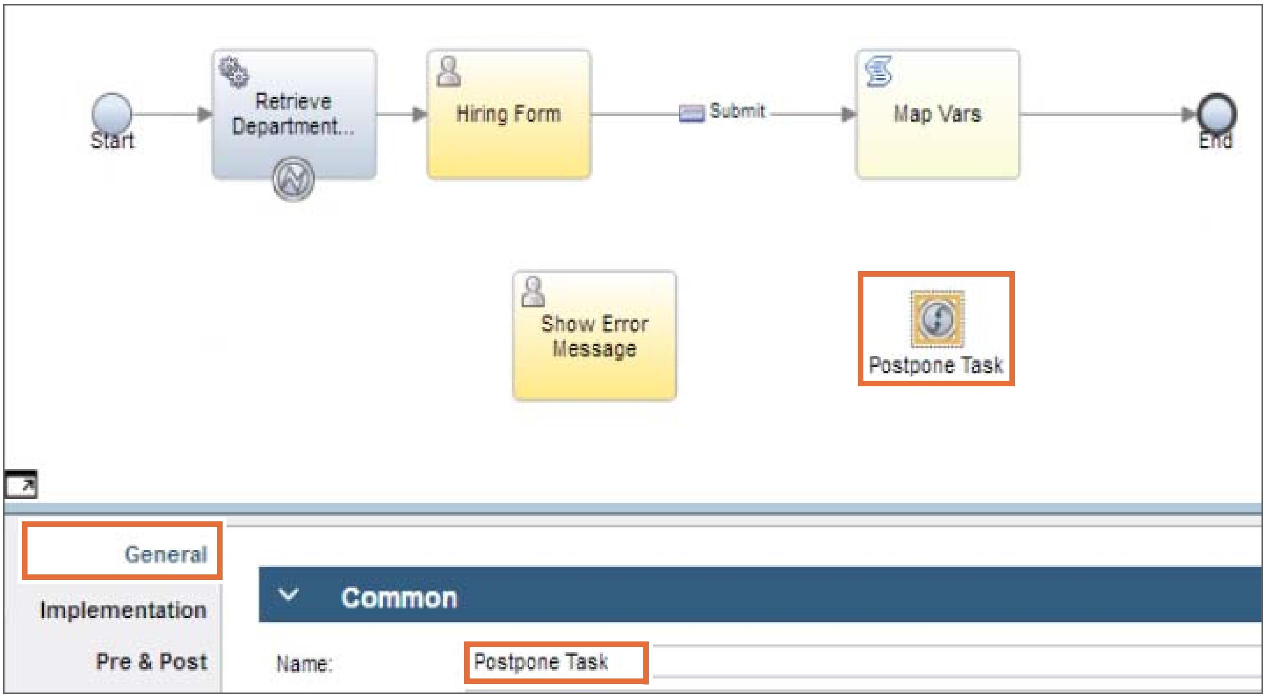
__ a. Drag an **Intermediate Event** from the palette to the right of the **Show Error Message** coach.




__ b. With the new Intermediate Event selected on the canvas, open the **Implementation** properties menu. Change the Event Type to **Postpone**. Leave the Event Navigation as **Default (behavior provided by the hosting UI)**.



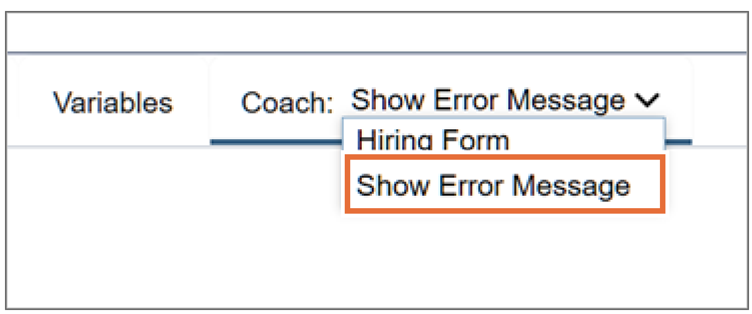
__ c. Open the **General > Common** properties section and rename the new event: `Postpone Task`




__ d. Save your work.

__ 4. Create the error message coach.

__ a. Click the **Coaches** tab and select the **Show Error Message** coach.



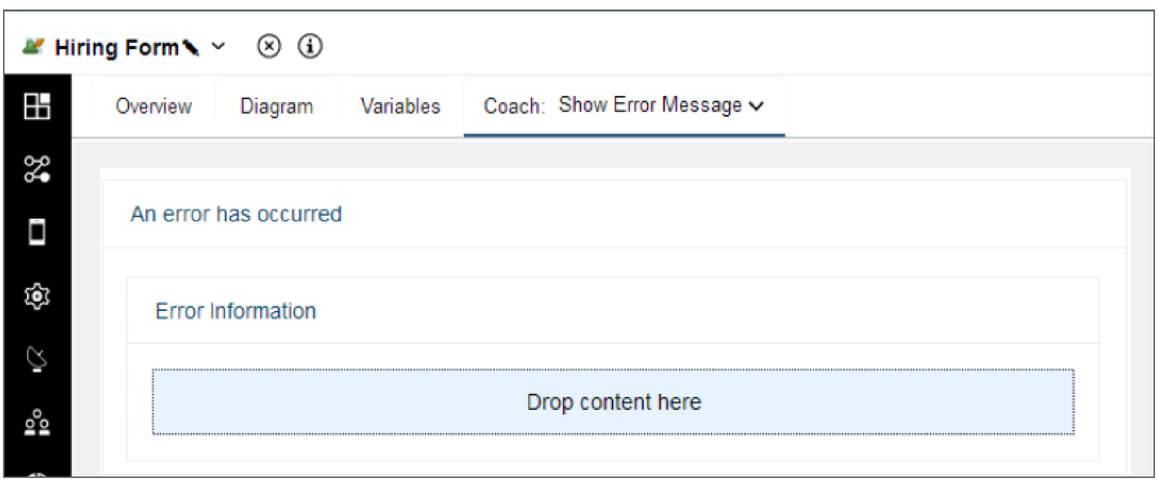
__ b. In the **Starting Point** window, ensure that the **Default coach** is selected, and click **OK**.


__ c. Select **Layout** in the **Building blocks** panel and select **Panel**

__ d. While holding down the left mouse button, drag the **Panel** above the **OK** button. Let go of the mouse button when the top border displays a gold highlight

__ e. Enter `An error has occurred` for the name.

__ f. Add another panel by dragging a panel to the location on the existing panel that displays **Drop content here** and label the panel `Error information`

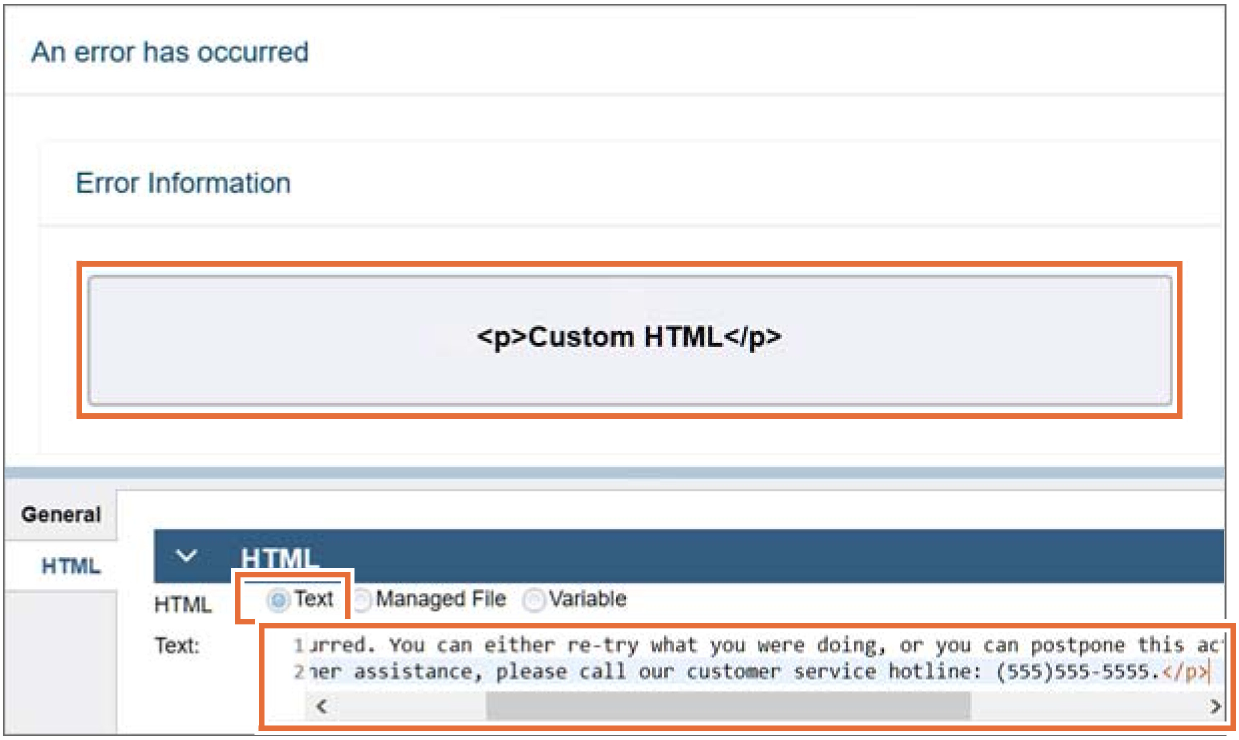



__ g. Select **Uncategorized** in the **Building blocks** panel and select **Custom HTML**

__ h. While holding down the left mouse button, drag the **Custom HTML** control and let go of the mouse button over the section that displays **Drop content here**.

__ i. Under the **HTML** properties menu, select **Text** and add error information to the HTML text block that is provided to the user. Alternatively, you can copy and paste the text that is provided in the file `Script1.txt` at the location: `C:\labfiles\Support files\Ex9`

```
<p>An error has occurred. You can either retry what you were doing, or you
can postpone this activity until a later date.</p>
<p>If you need further assistance, please call our customer service
hotline: (555)555-5555.</p>
```




__ j. Select **Input** in the **Building blocks** panel and select **Button**

__ k. While holding down the left mouse button, drag the **Button** below the **OK** button. Let go of the mouse button when the bottom border displays a gold highlight.

__ l. Change the **OK** button label to: `Retry.`

__ m. Change the second button label to: `Postpone.`

__ n. Save your work.

__ 5. Reconnect the flow on the **Hiring Form** canvas.

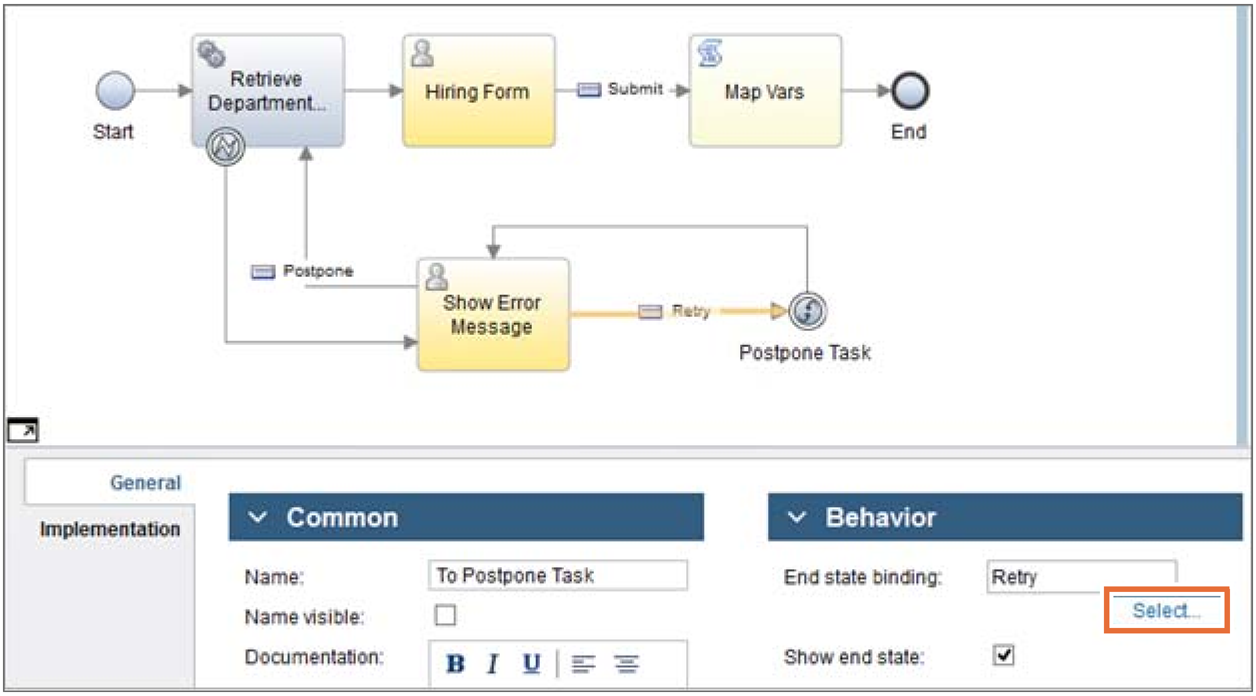
__ a. Click the **Diagram** tab.

__ b. Connect the new elements on the canvas. First, connect the **Error Intermediate Event** event to the **Show Error Message** coach.

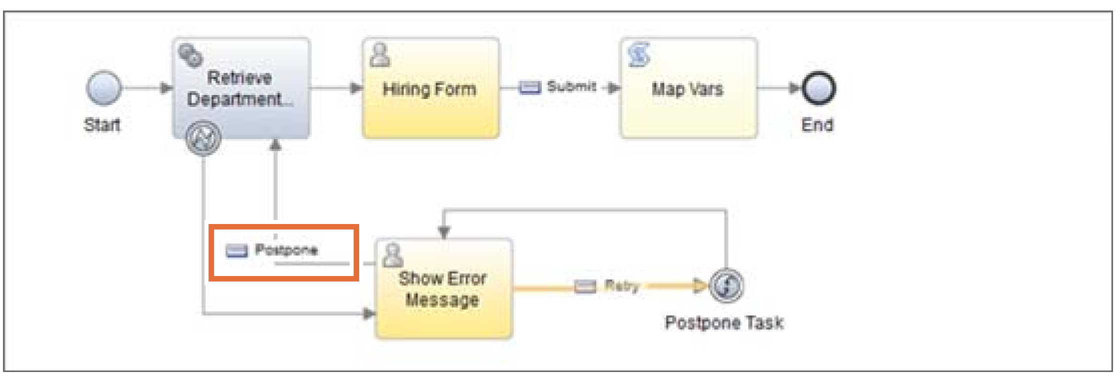
__ c. Next, connect the **Show Error Message** coach to the **Postpone Task** event.

__ d. Third, connect **Postpone Task** back to **Show Error Message**.

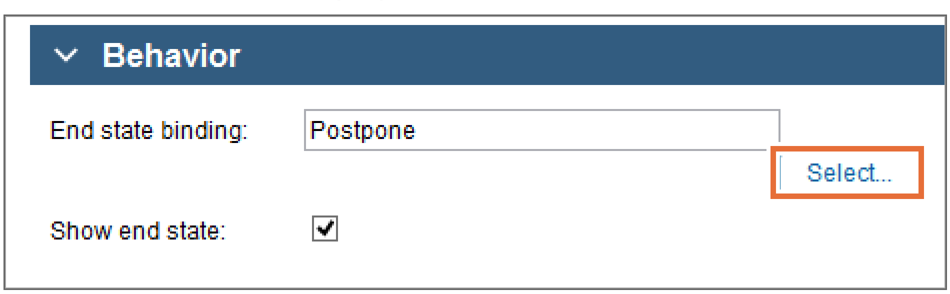
__ e. Finally, connect **Show Error Message** to **Retrieve Department Categories** by the **Retry** control.



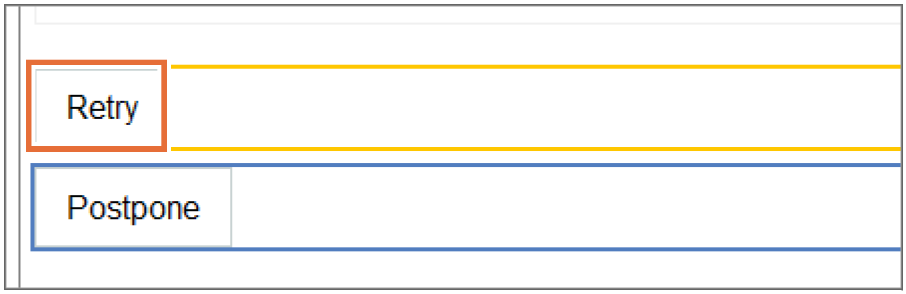

__ f. Notice that the label on the flow is incorrect. The instructions intentionally wired the flows up to the wrong event. In this flow, the **Postpone** button retries the service while the **Retry** button postpones the task. That would be incorrect. After you connect the Show Error Message coach to other activities, the flow editor automatically uses the button controls from the coach. If the controls do not match the flow, you can either reroute the connection arrows or change the end state binding. To change the end state binding, select the **Postpone** flow.



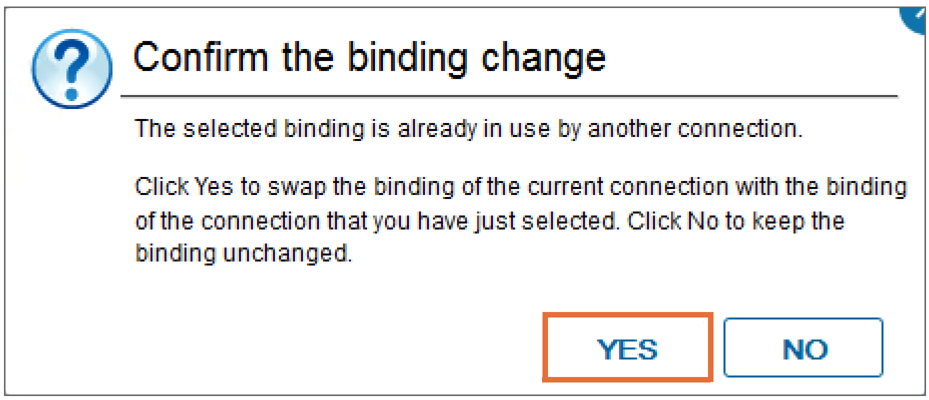

__ g. In the **General > Behavior** properties section, click **Select** next to End state binding.



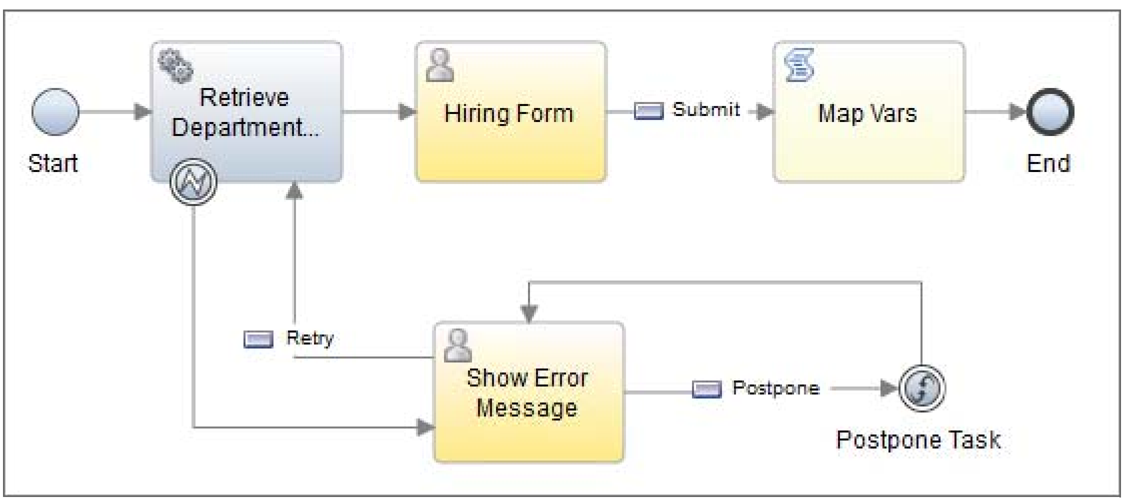
__ h. From the coach dialog box, select the **Retry** button and click **FINISH** at the bottom.




__ i. Click **Yes** to confirm the binding change.




__ j. Ensure that your buttons on your coach correspond to the correct flows on the coach.



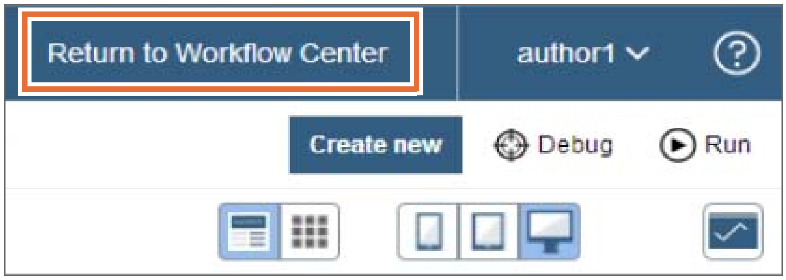

__ k. Save your work.

## *Part 2: Prepare for final snapshot*

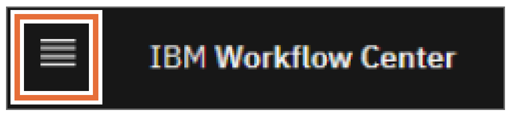
__ 1. Check for any validation errors in the process application.

Before taking the final snapshot of the process application for deployment, it is a good practice to check for any validation errors in the process application.

__ a. Click **Return to Workflow Center**.



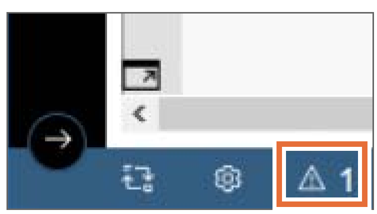

__ b. Click the navigation icon in the upper left.



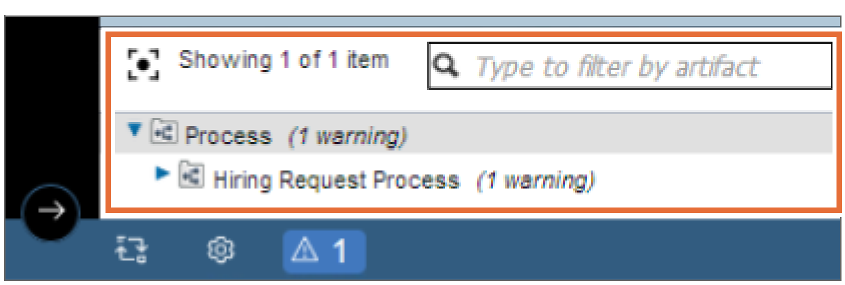

__ c. Select **Process apps**

__ d. Open the **HR Recruitment Process** process application.

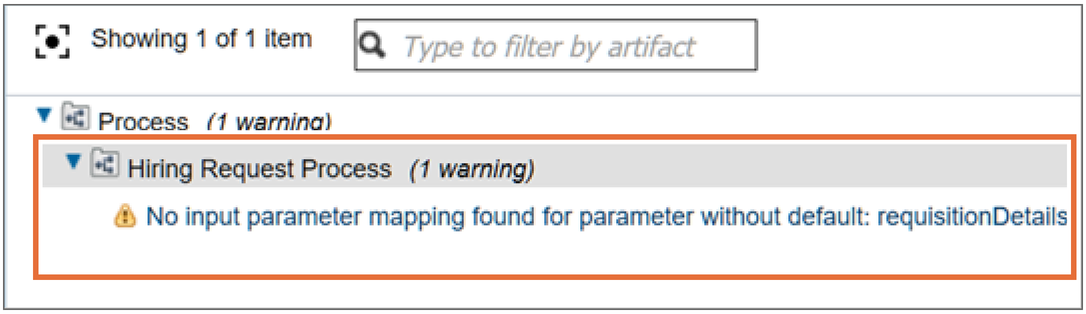
__ e. Validation errors must be resolved before you deploy a process application to a different environment. The instructions thus far introduced an error in the process application. You correct this error before the deployment of your process application. In this example, the Hiring Request Process contains an error. To resolve this error, in the Process Designer footer, click the **Validation errors and warnings** icon to check the name of the process that contains the error. Note that there is a number 1 next to the icon indicating one warning or error.



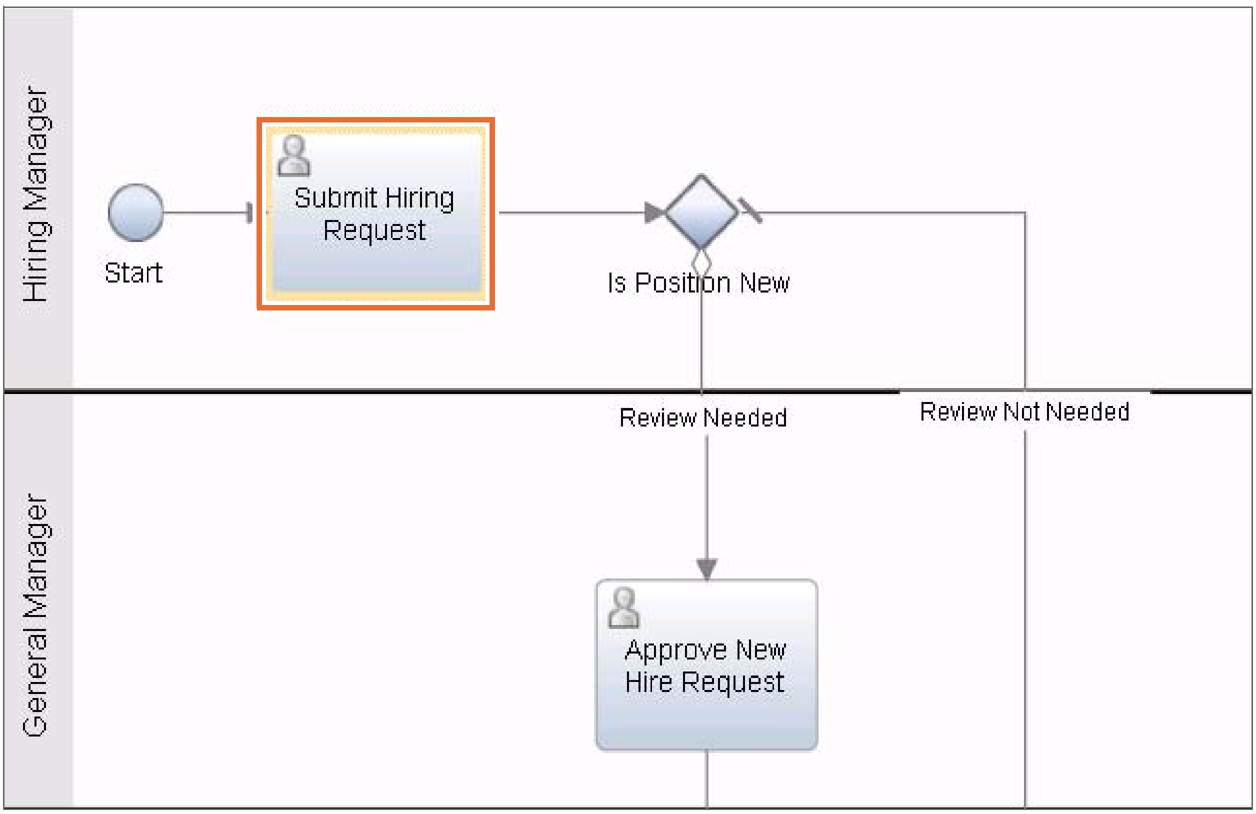

__ f. Clicking the **Validation errors and warnings** icon displays the process that contains the problem.



__ g. Expand the **Hiring Request Process** to view the error message.

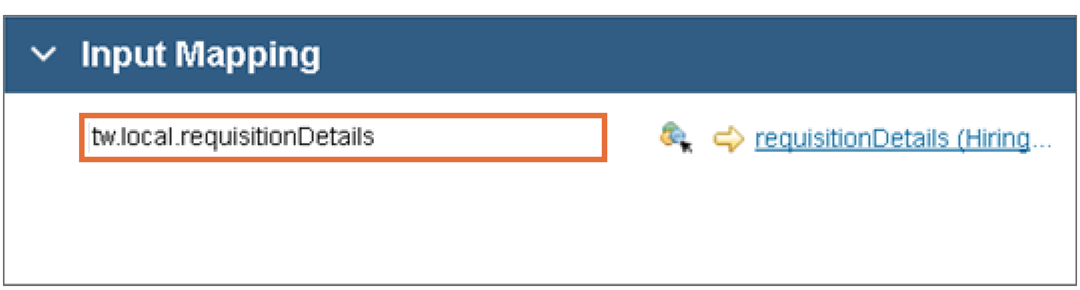



__ h. To resolve the parameter mapping errors, click the error message to directly open **Hiring Request Process** and then click the **Submit Hiring Request** activity.




__ i. Open the **Data Mapping** properties menu.

__ j. In the **Input Mapping** section, map the input to the `requisitionDetails (HiringRequisition)` variable by typing `tw.local.requisitionDetails`.



**Hint**

Because the names are consistent, you can use the auto-map feature for these steps.

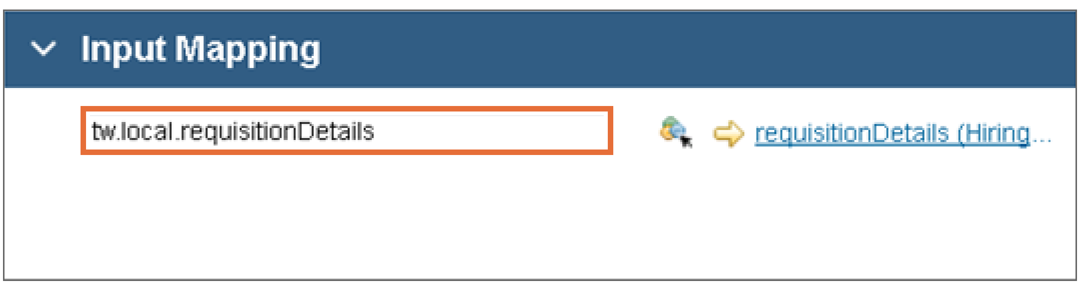
__ k. In the **Output Mapping** section, map the first output to the `requisitionDetails (HiringRequisition)` variable by typing `tw.local.requisitionDetails`.

__ l. In the **Output Mapping** section, map the second output to the `isNewPosition (String)` variable by typing `tw.local.isNewPosition`.


__ m. Select the **Approve New Hire Request** activity.

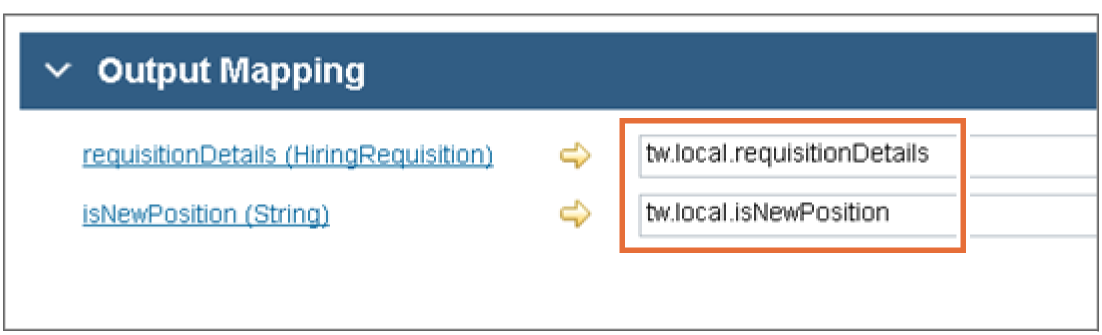
__ n. Open the **Data Mapping** properties menu.

__ o. In the **Input Mapping** section, map the input to the `requisitionDetails (HiringRequisition)` variable by typing `tw.local.requisitionDetails`.



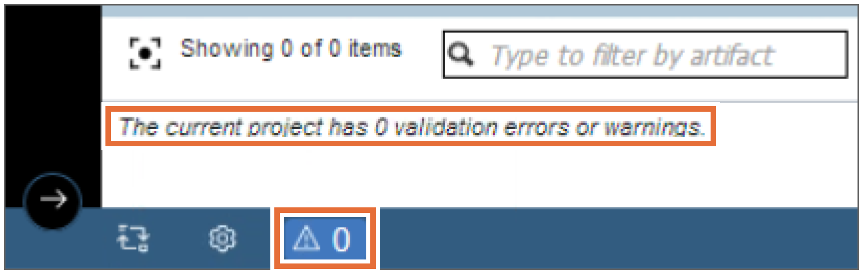

__ p. In the **Output Mapping** section, map the first output to the `requisitionDetails (HiringRequisition)` variable by typing `tw.local.requisitionDetails`.

__ q. In the **Output Mapping** section, map the second output to the `isNewPosition (String)` variable by typing `tw.local.isNewPosition`.



__ r. Save your changes.

__ s. Verify that there are no more any errors or warnings. The Validation and errors icon indicates a zero next to it. Close the validation panel.




## Part 3: Check the history of the process application

__ 1. Before taking a snapshot of the process application for deployment, you can check the history of the actions that you performed on the process application.

__ a. Return to the Workflow Center.

__ b. Click **Details** in the **HR Recruitment Process** process application tile.

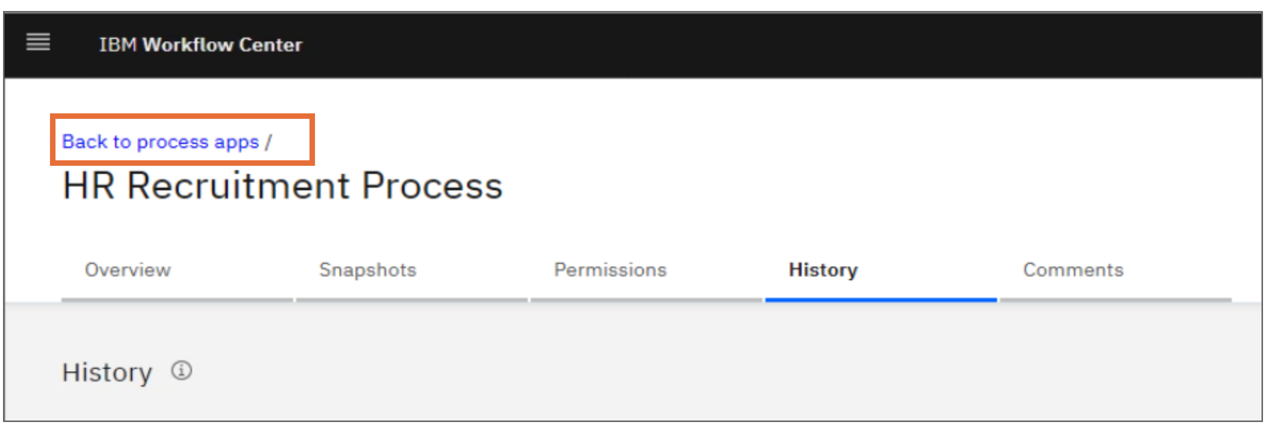
__ c. Click **History**.

__ d. Review the set of actions that you performed on the process application.

## Part 4: Create a snapshot of the process application

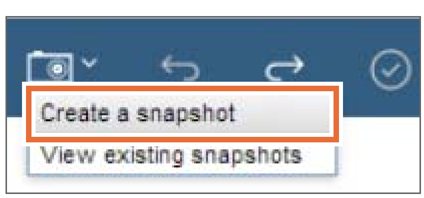
__ 1. Create a snapshot of the **Hiring Request Process** in IBM Process Designer.

__ a. Click the **Back to process Apps** link.

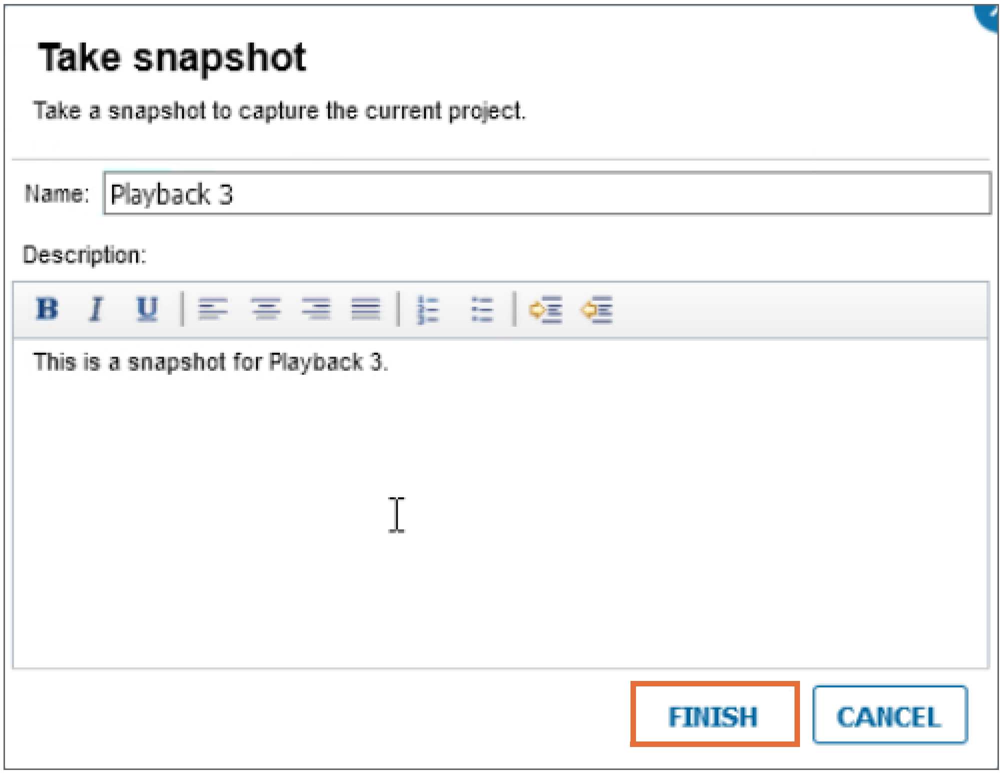



__ b. Open the **HR Recruitment Process** process application.

__ c. Expand the **Snapshot** menu on the top of the Process Designer window. Select **Create a snapshot**.



__ d. Give your snapshot the name `Playback 3` and description: `This is a snapshot for Playback 3.` Click **FINISH**.




__ e. Click **OK** when the pop-up window is shown.

__ f. Return to **Workflow Center**.

__ g. Click **Details** in the **HR Recruitment Process** process application tile.

__ h. Select the **Snapshots** tab

__ i. Verify your snapshot is displayed in the list of snapshots.

__ 2. Create a snapshot of the new toolkit and upgrade the dependency to the new snapshot.

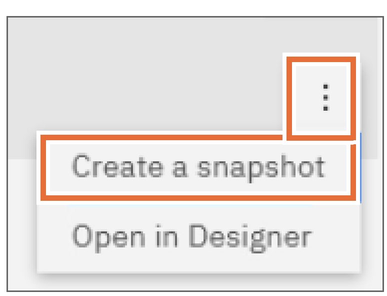
__ a. Click the navigation icon in the upper left and expand the **Process apps** section.

__ b. Click **Toolkits**.

__ c. Click **Details** on the Hiring Requisition Toolkit tile.

__ d. Select the **Snapshots** tab.

__ e. Open the list of options by clicking the **three dots** on the right of the **Current** snapshot tile and select **Create a snapshot.**



__ f. Name the snapshot `Playback 3` and provide the following documentation: `This snapshot includes error handling for a service.` Click **Create**.

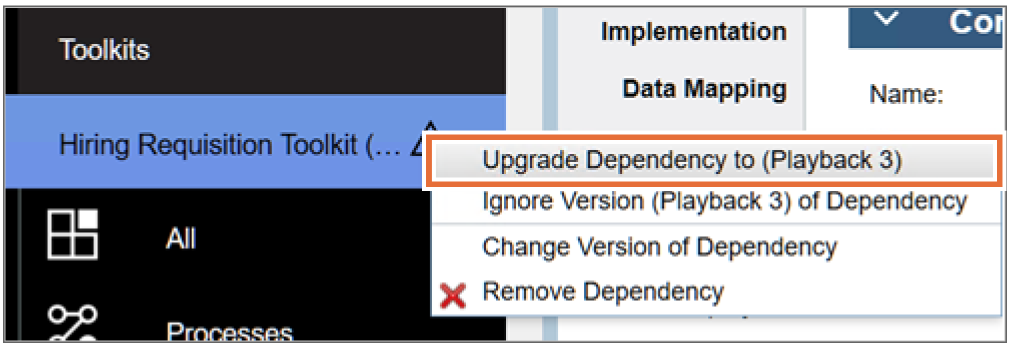
__ g. Open the menu in the top left and select **Process apps**.

**Troubleshooting**

If the Process apps list does not display and still displays the Toolkits, wait a minute for the refresh to occur.

__ h. Open the **HR Recruitment Process** process application in Process Designer.

__ i. Click **Toolkits** in the library. Click the arrow next to **Hiring Requisition Toolkit** and click **Upgrade Dependency to (Playback 3)**.




__ j. Playback 3: Creating error handling for a service is now complete.

You have completed this exercise.

In this exercise, you added a Catch Exception component to capture and manage errors in the process. If any validation errors were present in the process application, you resolved any problems. Next, you looked at the history of the actions that were performed on the process application and created the snapshot for deployment.

## End of exercise

# Exercise 10. Implementing a custom routing solution

## Estimated time

01:00

## Overview

In this exercise, you learn how to implement a custom routing solution.

## Objectives

After completing this exercise, you should be able to:

- Implement custom routing

## Introduction

The company is working hard to meet a deadline and some employees are working on the weekend. General Managers who approve the new hire request do not work on the weekend. Those designated as Weekend Managers should receive work if the date of the request is on the weekend.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

# Exercise instructions

## *Part 1: Add code to determine whether it is the weekend*

Before adding a custom routing option to the **Approve New Hire Request**, the determination of whether the date falls on a weekend needs to be added. To do this, you add a Boolean variable and JavaScript to the **Hiring Form**.

__ 1. Add a Boolean variable

__ a. Open the **Hiring Requisition Toolkit** toolkit

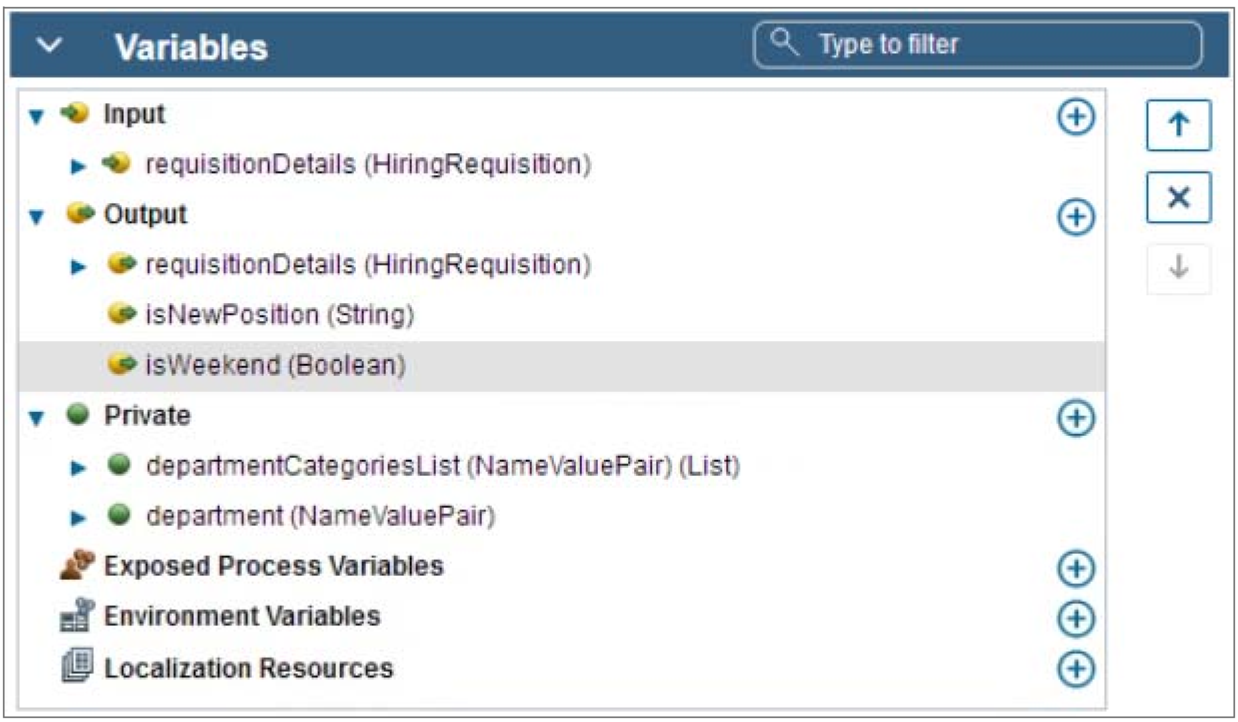
__ b. Open the **Hiring Form** user interface

__ c. Under the **Variables** tab, click the **plus sign (+)** next to Output.

__ d. Enter `isWeekend` for the name

__ e. Click **Select** next to Variable type and select **Boolean** for the type.

__ f. Save your work




__ 2. Enter the JavaScript code to determine if it is the weekend

__ a. Click the **Diagram** tab to switch to the diagram

Because there is already a script task before the End event, you can place the JavaScript in this task.

__ b. Highlight the **Map Vars** script task and select the **Script** tab under the Properties section.

__ c. Enter the following JavaScript at the end of the existing script block

```
//Determine if request date is on the weekend
var dt = new Date(tw.local.requisitionDetails.dateOfRequest);
if(dt.getDay() == 6 || dt.getDay() == 0) {
   tw.local.isWeekend = true; }
else {
   tw.local.isWeekend = false;  }
```

Alternatively, you can paste the script from the file `Script1.txt` under the folder `C:\labfiles\Support files\Ex10`

__ d. Verify your JavaScript code

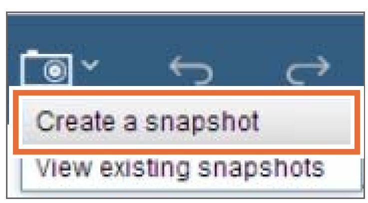
```
//  This JavaScript will run wi
// thin the browser and must use the client-side JavaScript syntax supported by the browser. For example,
//          - To instantiate a complex object:                          - To instantiate a list:
//                      tw.local.customer = {};                                      tw.local.addresses = [];
//                      tw.local.customer.name = "Jane";            tw.local.addresses[0] = {};
if (tw.local.requisitionDetails.recruitingDetails.newPosition == true)
  tw.local.isNewPosition = "1";
else
  tw.local.isNewPosition = "0";
tw.local.requisitionDetails.departmentDetails.department = tw.local.department.name;

//Determine weekend
var dt = new Date(tw.local.requisitionDetails.dateOfRequest);
if(dt.getDay() == 6 || dt.getDay() == 0) {
  tw.local.isWeekend = true; }
else {
  tw.local.isWeekend = false; }
```

__ e. Save your work

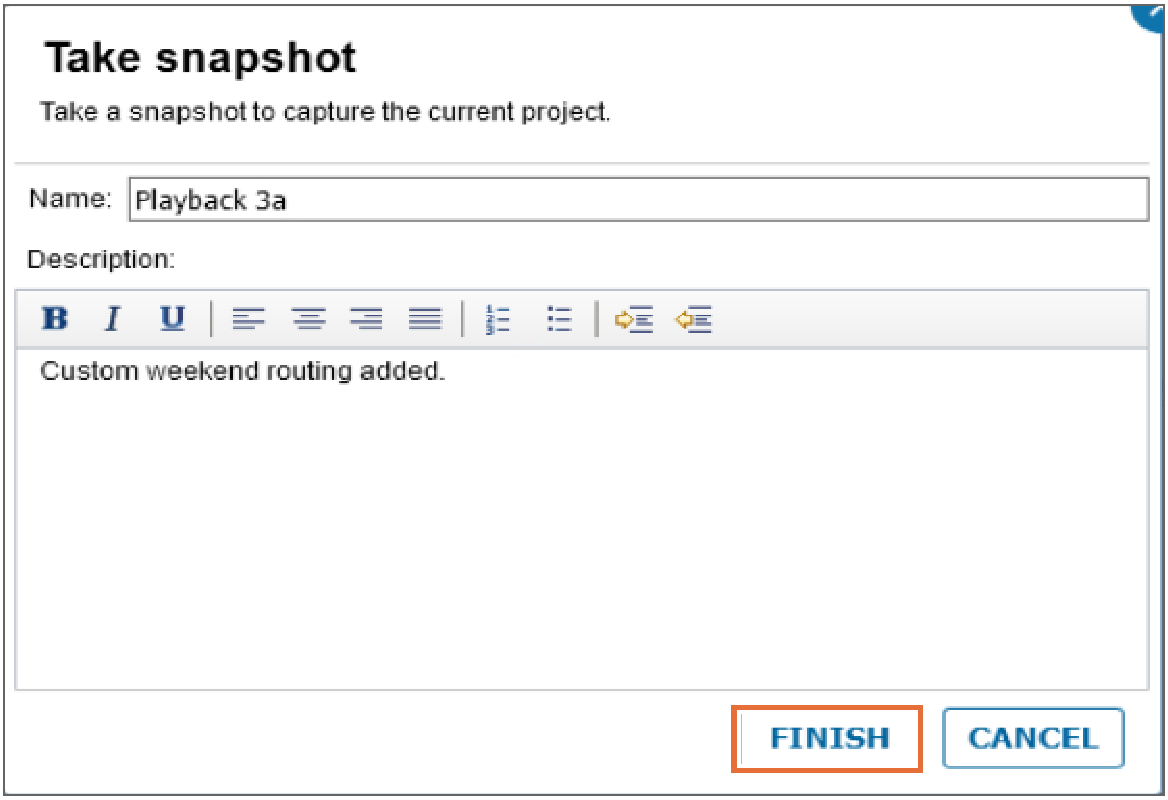
__ 3. Save a new snapshot

This was a last-minute addition. As such, it is considered part of the Playback 3 solution.

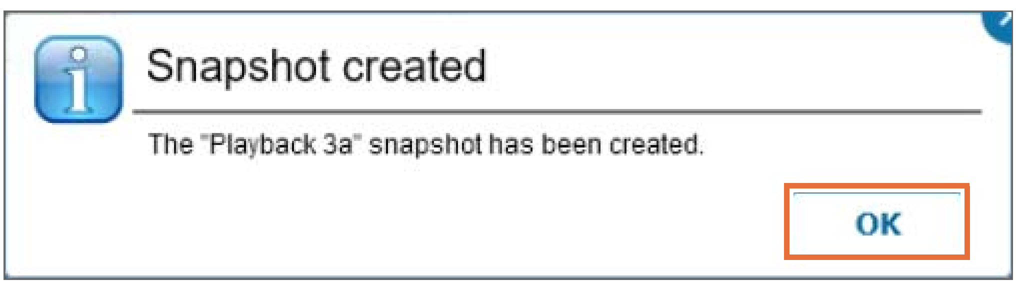
__ a. Click the **camera icon** in the upper right corner and select **Create a snapshot**.



__ b. Enter `Playback 3a` as the name, enter a description, and click **FINISH**




__ c. Click **OK** on the Snapshot Created dialog.




__ d. Click **Return to Workflow Center**.

## *Part 2: Add custom routing to the HR Recruitment process*

In this section, you update **Approve New Hire Request** activity to use custom routing. Before you do that, you add the variable to the process, perform data mapping, and add the **Weekend Managers** team. Before being able to use the updated **Hiring Form**, you must upgrade the dependency to the new snapshot of the **Hiring Requisition Toolkit**.

__ 1. Add a Boolean variable to the process

__ a. Open the **HR Recruitment Process** process application

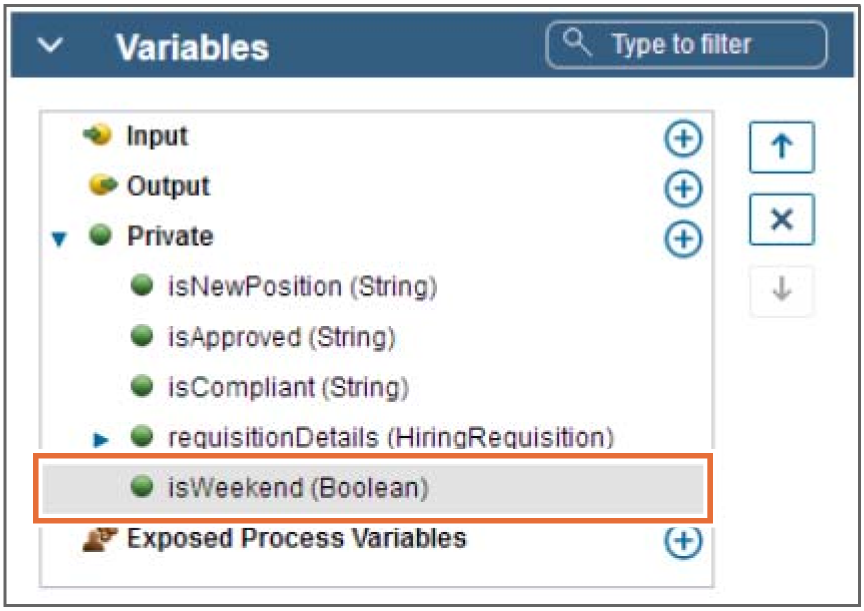
__ b. Open the **Hiring Request Process**

__ c. Open the **Variables** tab

__ d. Click the **plus sign (+)** next to Private to add a variable

__ e. Enter `isWeekend` for the name and set the variable type to Boolean

__ f. Verify the addition of the private variable.

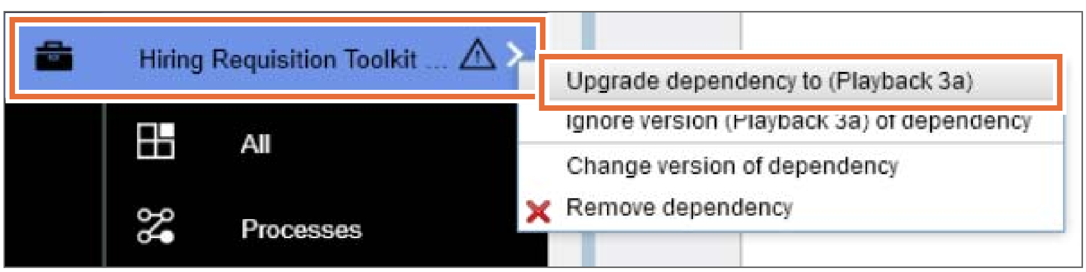



__ g. Save your work.

__ 2. Upgrade to the new **Hiring Request Toolkit** snapshot

__ a. Expand the **Toolkits** category in the Library section. The `Hiring Request Toolkit` shows a warning because the process application is not pointing to the most recent snapshotted version.

__ b. Click the **arrow** next to the Toolkit and select **Upgrade Dependency to (Playback 3a)**




__ 3. Perform data mapping

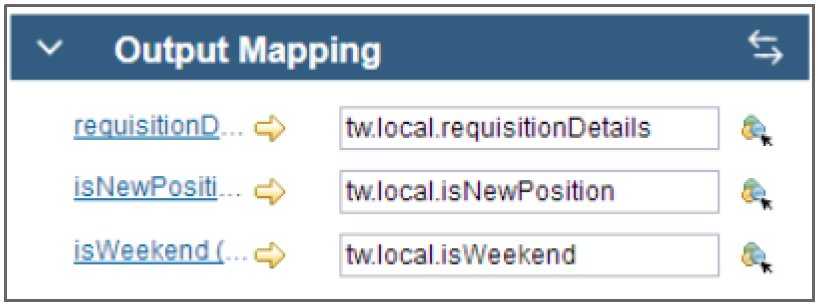
Now that the Boolean variable is added to the process and the toolkit dependency upgraded, you can map the variable from the **Submit Hiring Request** to the process.

__ a. From the Hiring Request Process, click the **Definition** tab.

__ b. Select the **Submit Hiring Request** activity

__ c. Select the **Data Mapping** tab in the properties section below

__ d. Under the Output Mapping section, click the **Variable Picker** next to the `isWeekend` variable and select **isWeekend (Boolean)**

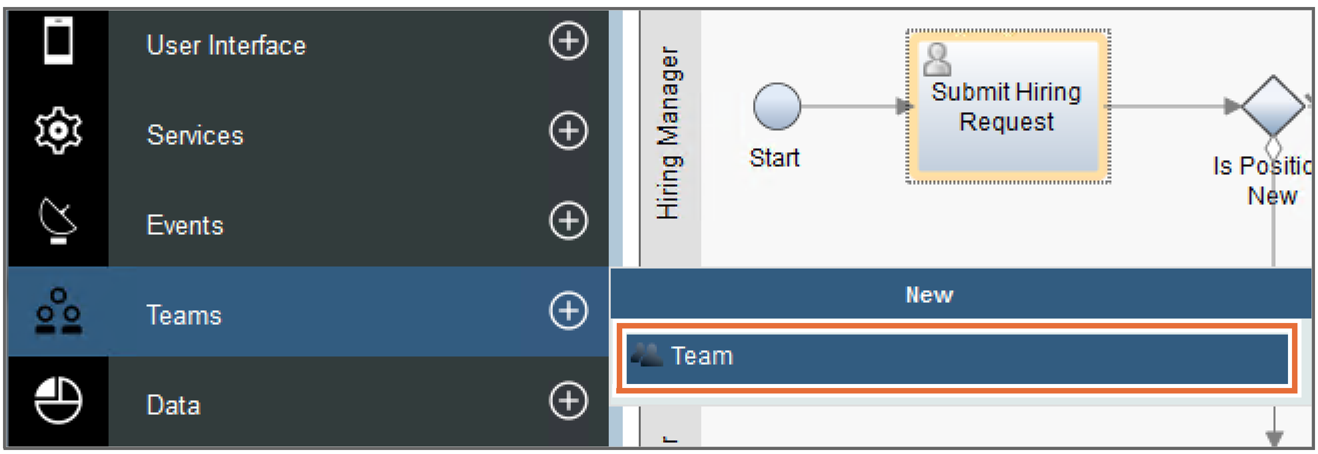



__ e. Save your work

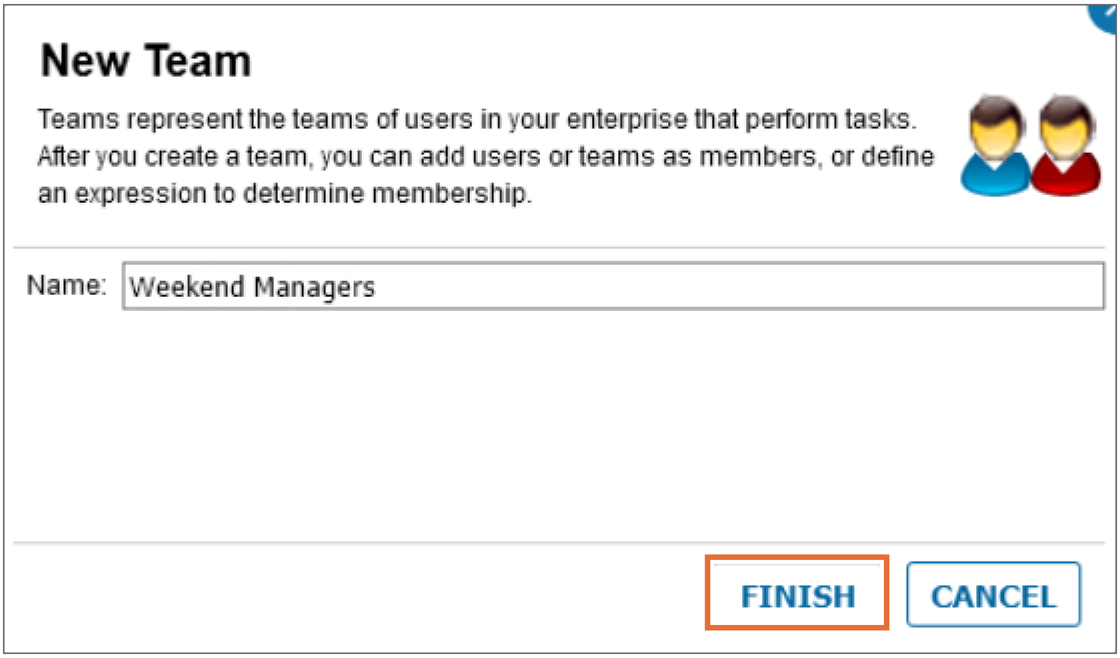
__ 4. Add the Weekend Managers team and add a member

The Weekend Managers team contains the managers that are able to work on weekends.

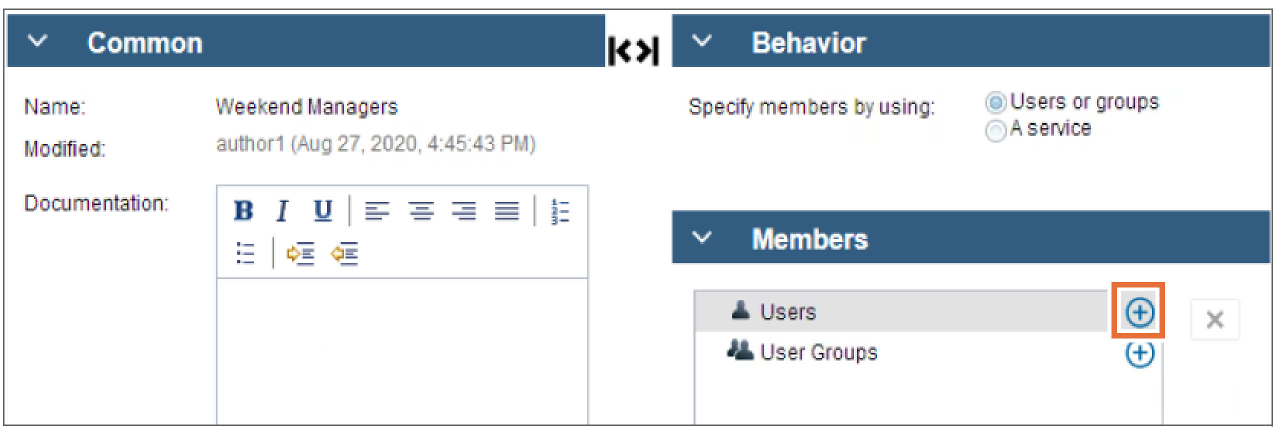
__ a. In the process library, click the (**+**) plus sign next to **Teams**. Click **Team**.



__ b. Name the team `Weekend Managers` and click **FINISH**.

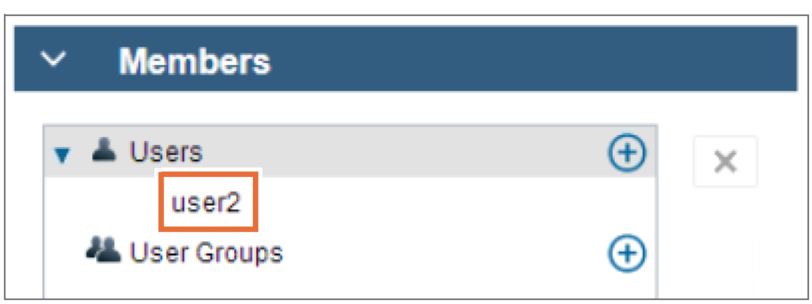



__ c. In the **Members** section, click the (**+**) plus sign next to Users.




__ d. Select **user2** from the list.

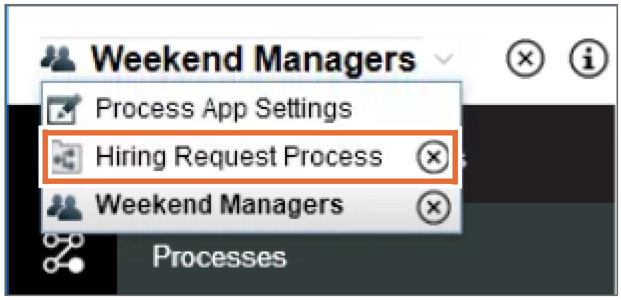
You now have a populated list in the Members section with **user2** for the **Weekend Managers** team.




__ e. Save your work.

__ 5. Apply custom routing to the **Approve New Hire Request** activity assignment

__ a. Return to the Hiring Request Process by opening the **History** menu at the top and selecting it from the drop-down menu




__ b. Select the **Approve New Hire Request** activity

__ c. In the Properties section below, select the **Assignments** tab

__ d. Select **Custom** next to **Assign to**.

__ e. Enter the following expression in the **Expression** field

`tw.local.isWeekend?"TEAM:Weekend Managers":"TEAM:General Managers"`

Alternatively, you can paste the script from the file `Script2.txt` under the folder `C:\labfiles\Support files\Ex10`


__ f. Save your work

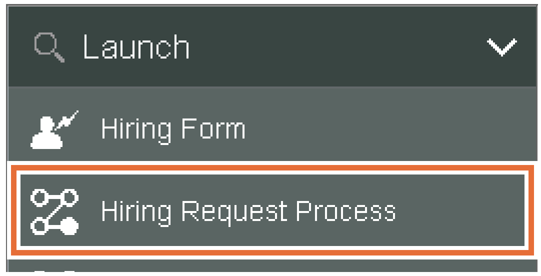
## Part 3: Test the custom routing solution

__ 1. Submit a new request

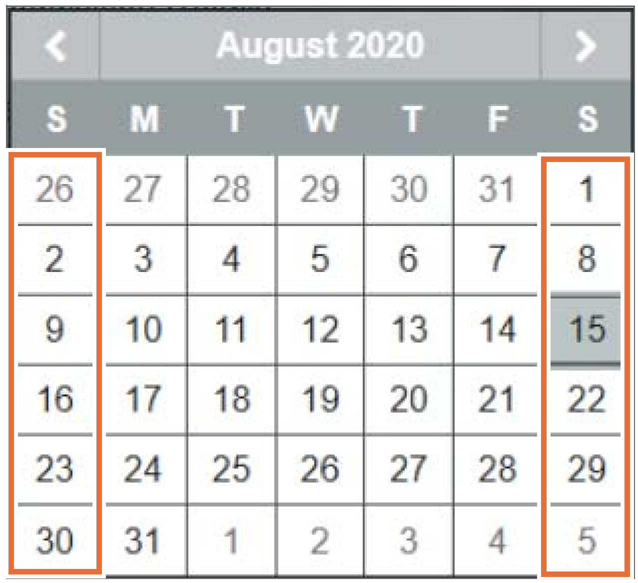
__ a. Open **Process Portal** in another tab.

__ b. The Process Portal opens in a new browser tab. Since you were already logged in to the Workflow Center as **author1**, the session continues and you are logged in to Process Portal as **author1** also.

__ c. Create an instance of the process by clicking **Hiring Request Process** in the Launch section on the left. Wait for the first task to open in the main window.



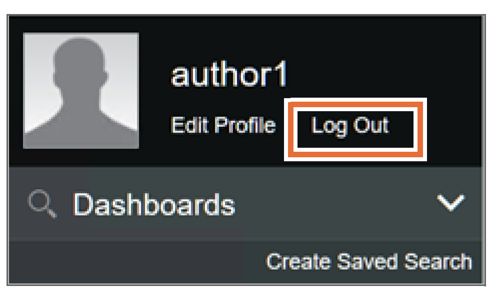

__ d. To send the task to the Weekend Managers group, select a Saturday or Sunday for the **Date Of Request**




__ e. To ensure that the process follows the path to the **Approve New Hire Request** activity, click the **Recruiting Details** tab and place a check mark next to **New Position**

__ f. Click **Submit**

__ g. Log out as **author1**

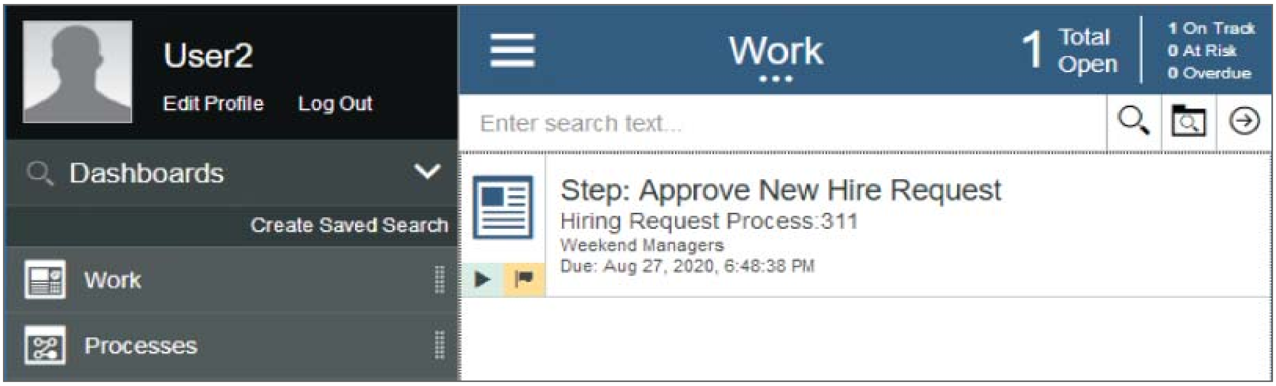



__ 2. Verify that the weekend managers see the task

Recall that `user2` was the only member of the **Weekend Managers** team.

__ a. Log in to Process Portal by using `user2` in the **Username** field and `passw0rd` in the **Password** field

__ b. Verify the user, as a member of the Weekend Managers team, received the **Approve New Hire Request** task




__ c. Log out as **User2**

__ 3. Verify that the General Managers do not see the task

Recall that `user1` was the only member of the **General Managers** team.

__ a. Log in to Process Portal by using `user1` in the **Username** field and `passw0rd` in the **Password** field

__ b. Verify the user, as a member of the General Managers team, did not receive the **Approve New Hire Request** task.

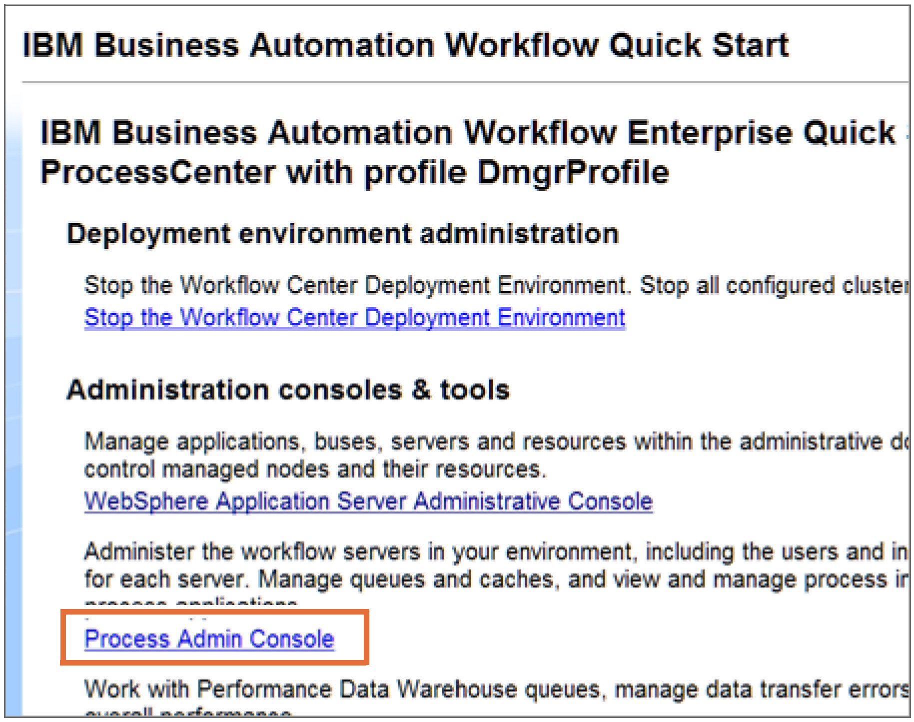

__ c. Log out as **user1**.

## *Part 4: Clean up the environment*

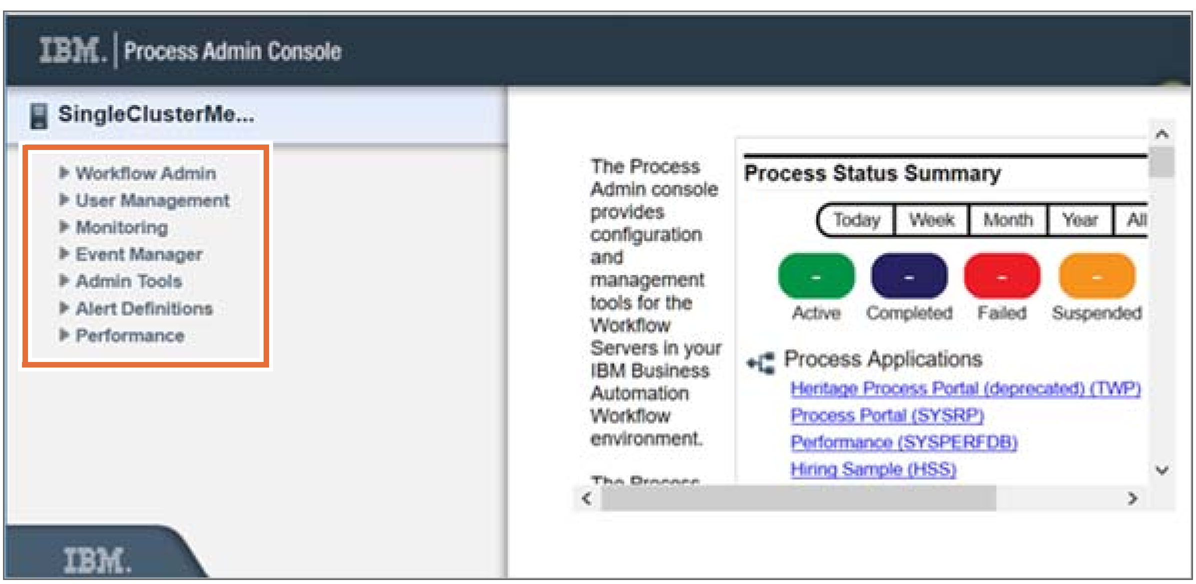
When demonstrating mock scenarios, it is important to reset the environment back to the defaults before moving on.

__ 1. Delete snapshot status changes

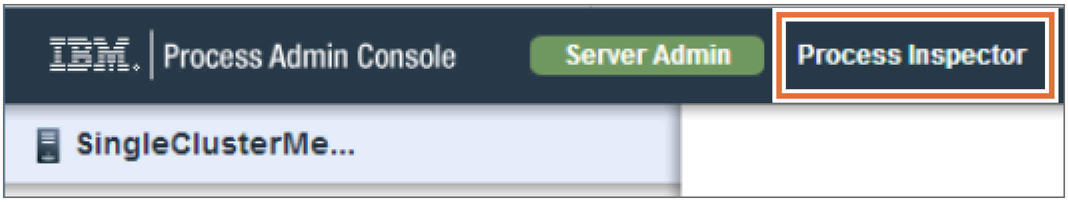
__ a. To access the administration console, go back to the Quick Start window and click the **Process Admin Console** link.




__ b. When the Process Admin console is displayed, note the various links in the left pane.



__ c. Click the **Process Inspector** link in the navigation bar at the top.

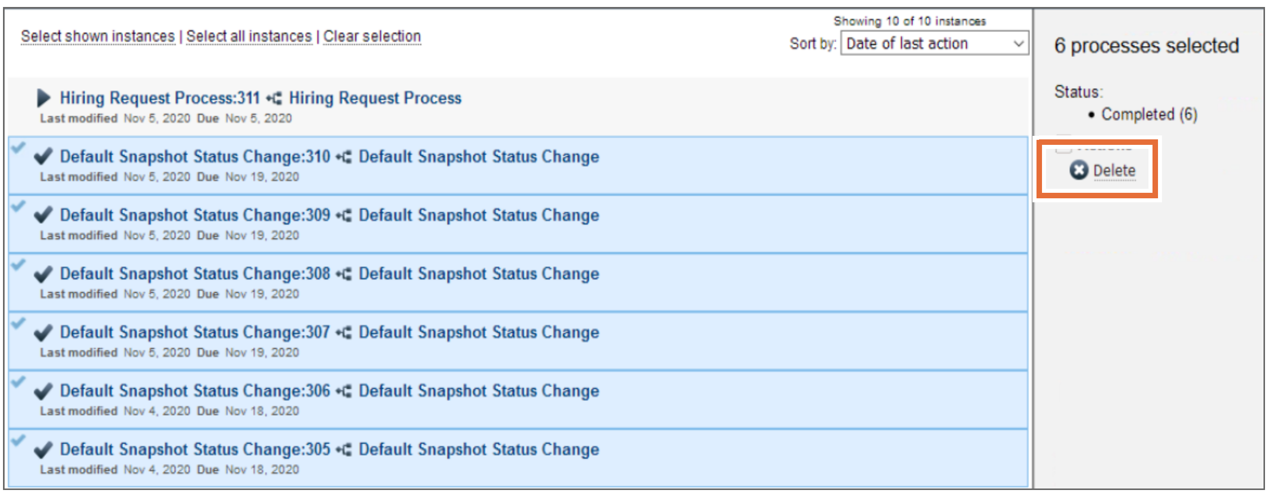



The Process Inspector opens in a new tab.

__ d. Click **Search**

The **Filter process instances by** section is updated and all process applications in the repository are listed on the right with their status. The Search button changes to Refresh.

__ e. Highlight all the snapshot status changes and click **Delete**. The number of snapshots may be different for your environment.

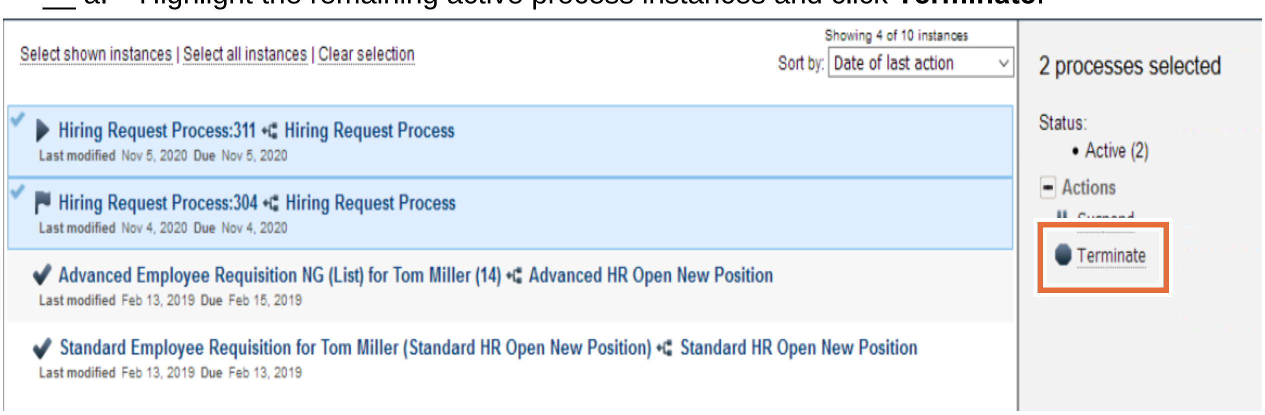



__ f. Click **OK** on the Confirm Bulk Action window.

The snapshot status changes are deleted. Two active Hiring Request process applications are still listed. Your list might have more.

__ 2. Terminate and delete the active processes.

__ a. Highlight the remaining active process instances and click **Terminate**.




__ b. Click **OK** on the Confirm Bulk Action window.

The status of the processes change to Terminated.

__ c. With both terminated processes highlighted, click Delete.

__ d. Click **OK** on the Confirm Bulk Action window.

Both processes are deleted.

You have completed this exercise.

In the first part of the exercise, you modified the Hiring Request Toolkit to add a Boolean variable and JavaScript to the Hiring Request form. Then you added the variable to the process, upgraded the toolkit dependency, and added the weekend managers team to the process. Once that was completed, you were able to configure a custom routing assignment for the Approve New Hire Request activity. Finally, to test that only Weekend Managers receive the task, you created a new request by using a weekend date. To verify that only the Weekend Managers see it, you logged in as a member of the Weekend Managers to verify you see the task. You then logged in as a member of the General Managers team to verify that you did not see the task.

## End of exercise

# Exercise 11. Implementing the “four eyes” policy by using a team filter

## Estimated time

01:30

## Overview

In this exercise, you learn how to implement a “four eyes” policy in a process by using a team filter.

## Objectives

After completing this exercise, you should be able to:

- Implement the “four eyes” policy by using a team filter

## Introduction

The company chief executive officer (CEO) is concerned about the activity in the HR Administrator lane during the approval linked process. The CEO wants to make sure that if an override is required, two people must look at the request, enabling a “four eyes” policy for activities in that lane. You must create a second activity in the lane, and use a routing policy to ensure that the user who completed the first activity in a lane does not claim the second activity in the lane.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

## *Part 1: Modify the process*

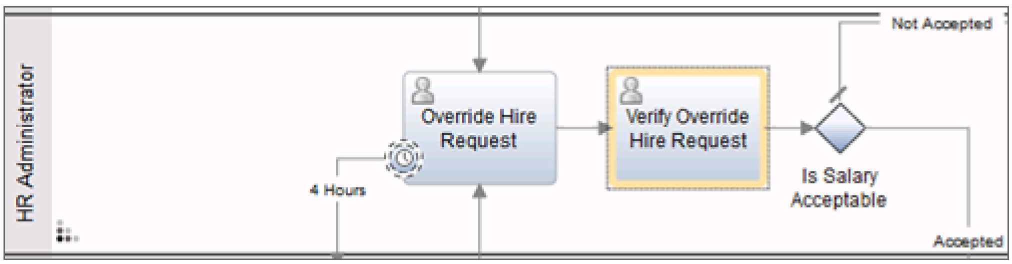
The first steps of this exercise are to change the process to meet the new requirements of the organization. After the process is changed, you will then implement the four eyes policy by using Team Filter services.

__ 1. Create the verify activity to make sure that if an override is required, two people must look at the request, enabling a “four eyes” policy for activities in that lane.

__ a. Open the **HR Recruitment Process** process application.

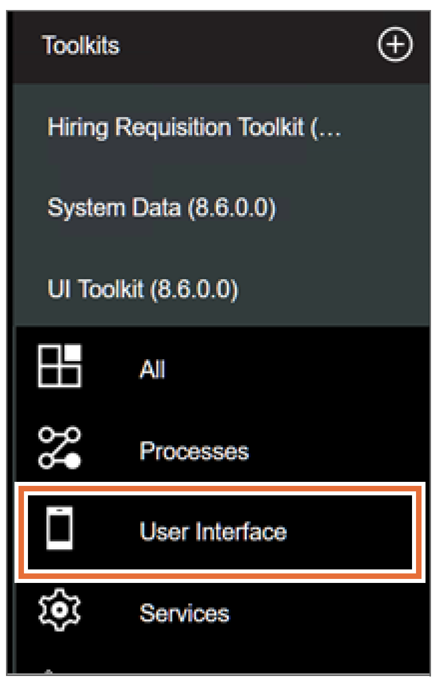
__ b. Open the **Approve Hire Request** process.

__ c. Drag a User Task to the right of the **Override Hire Request** activity and drop it on the sequence flow. Name this activity: `Verify Override Hire Request`

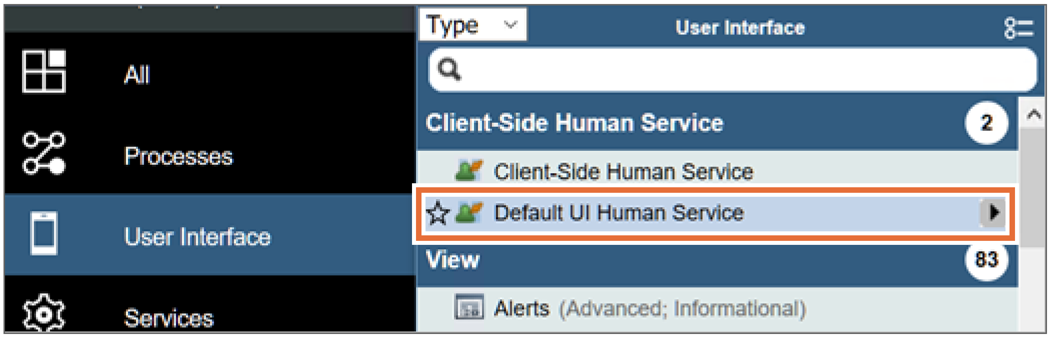



__ 2. Create a service to capture the username during the Override Hire Request step.

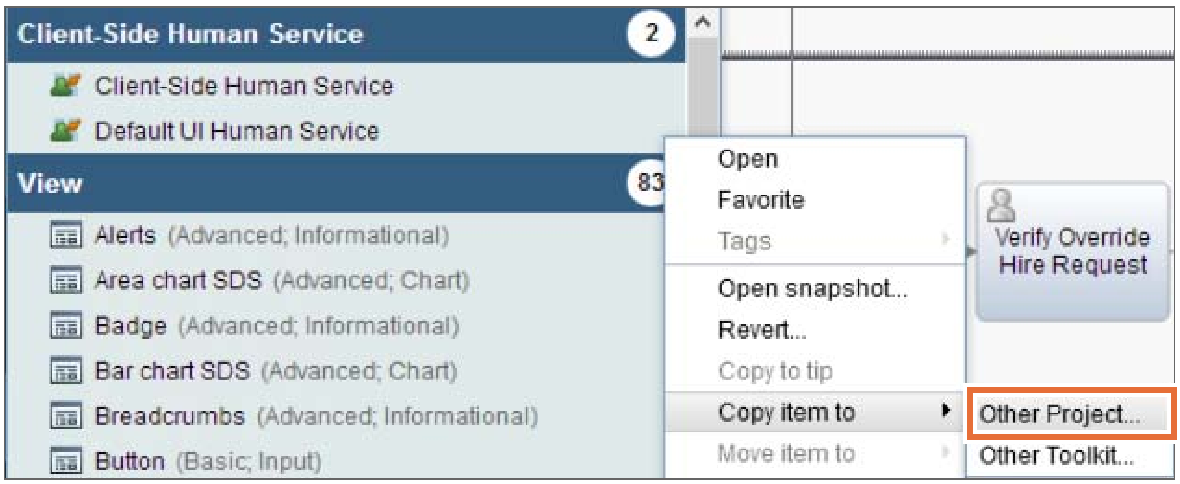
__ a. Expand the **Toolkits** category in the library, and then expand **UI Toolkit** and select **User Interface**.



__ b. Under the Client-Side Human Service category, right-click the **Default UI Human Service**.



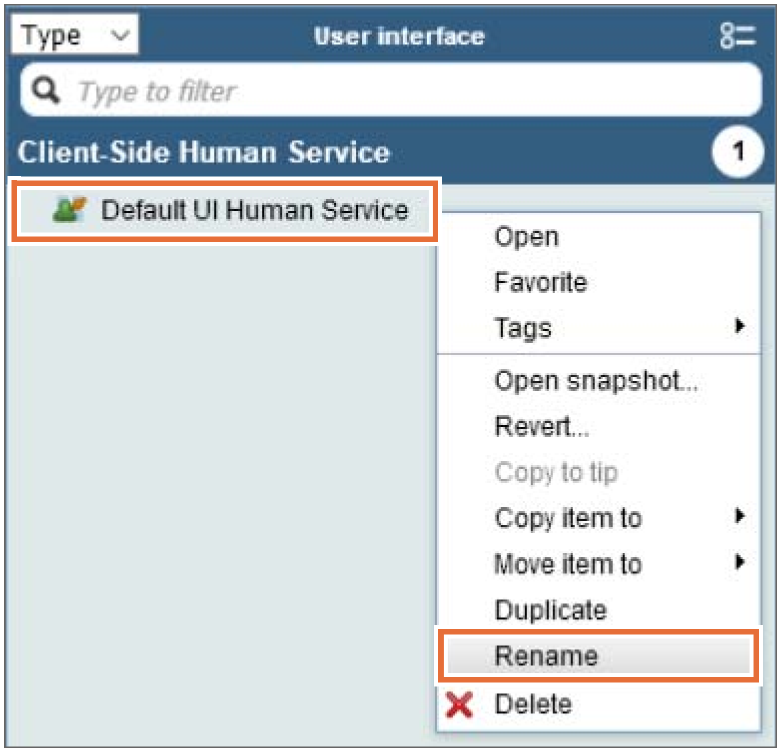

__ c. Select **Copy item to > Other Project**.




__ d. Select **HR Recruitment Process**.

__ e. Click the **Toolkits** menu to collapse the category in the library.

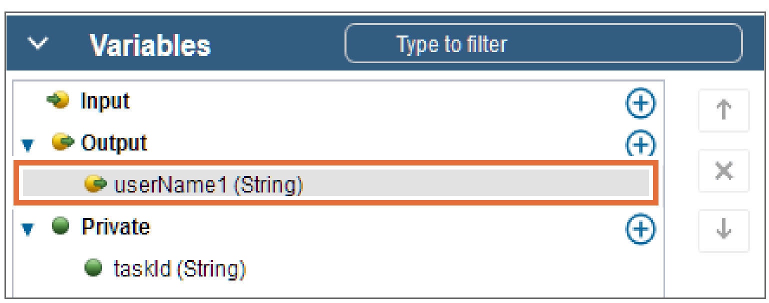
__ f. Scroll up in the library and this time click the **User Interface** menu in the library for the **HR Recruitment Process** process application and right-click the Default UI Human Service you just copied and select **Rename**.



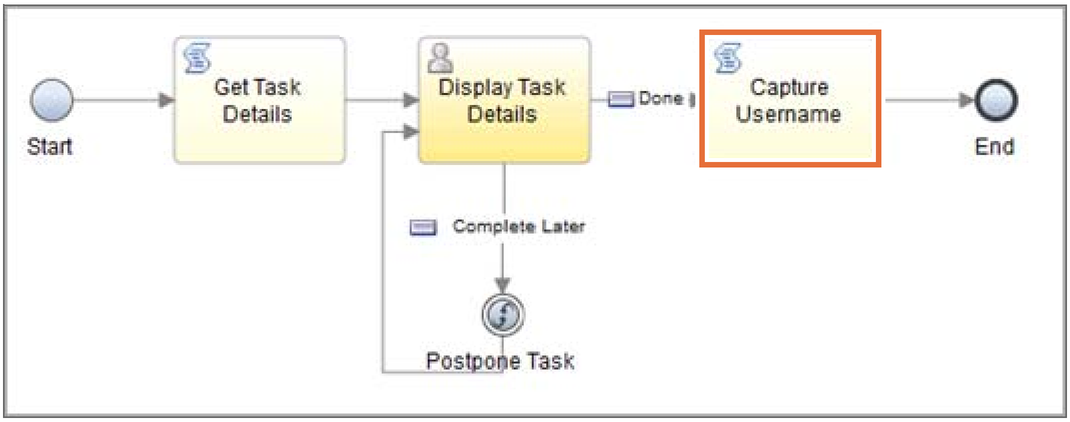

__ g. Rename the **Default UI Human Service** to: `Override Hire Request` and click **FINISH**.

__ h. Open the **Override Hire Request** human service.

__ i. Click the **Variables** tab and add an output variable `userName1 (String)` to the Override Hire Request service to capture the username.



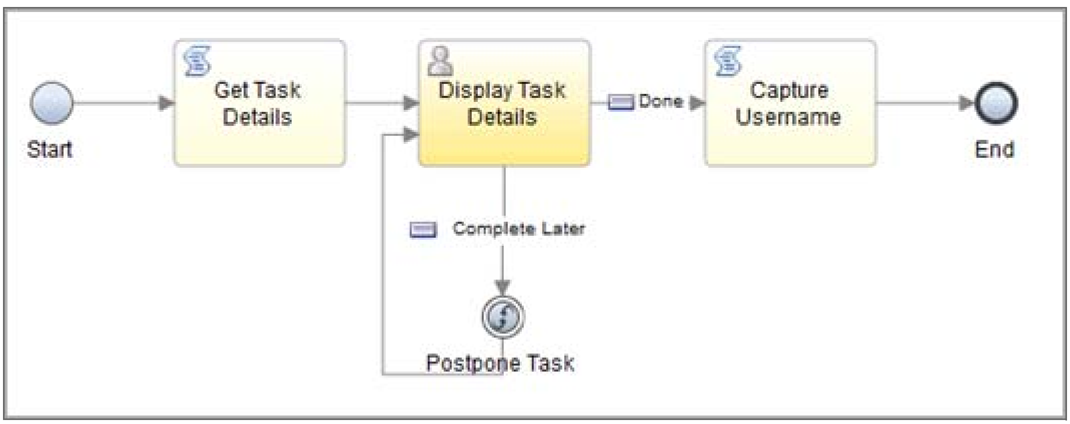
__ j. Click the **Diagram** tab and add a Client-Side Script that is named `Capture Username` to the sequence flow after the **Display Task Details** coach.




__ k. Click the **Capture Username** step.

__ l. Enter the code in the **Script** properties menu to capture the username in the `userName1` variable:

```
tw.local.userName1 = tw.system.user.name;
```

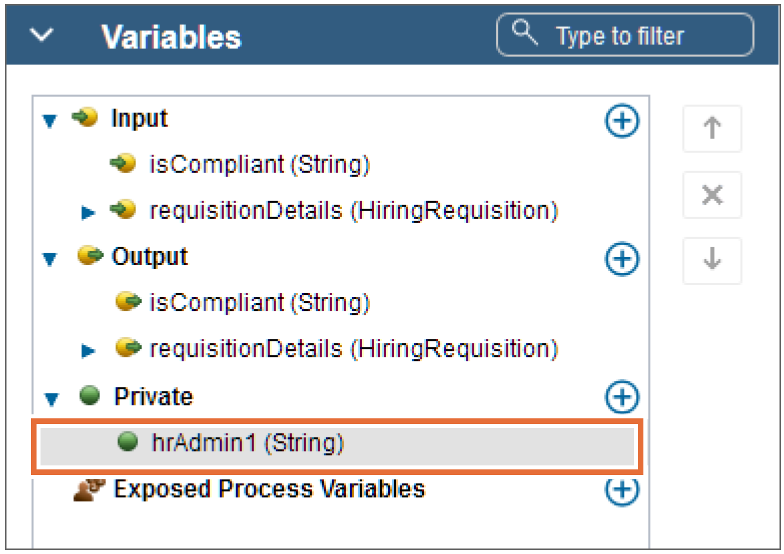



__ m. Save your work.

__ 3. Implement the new service on the process.

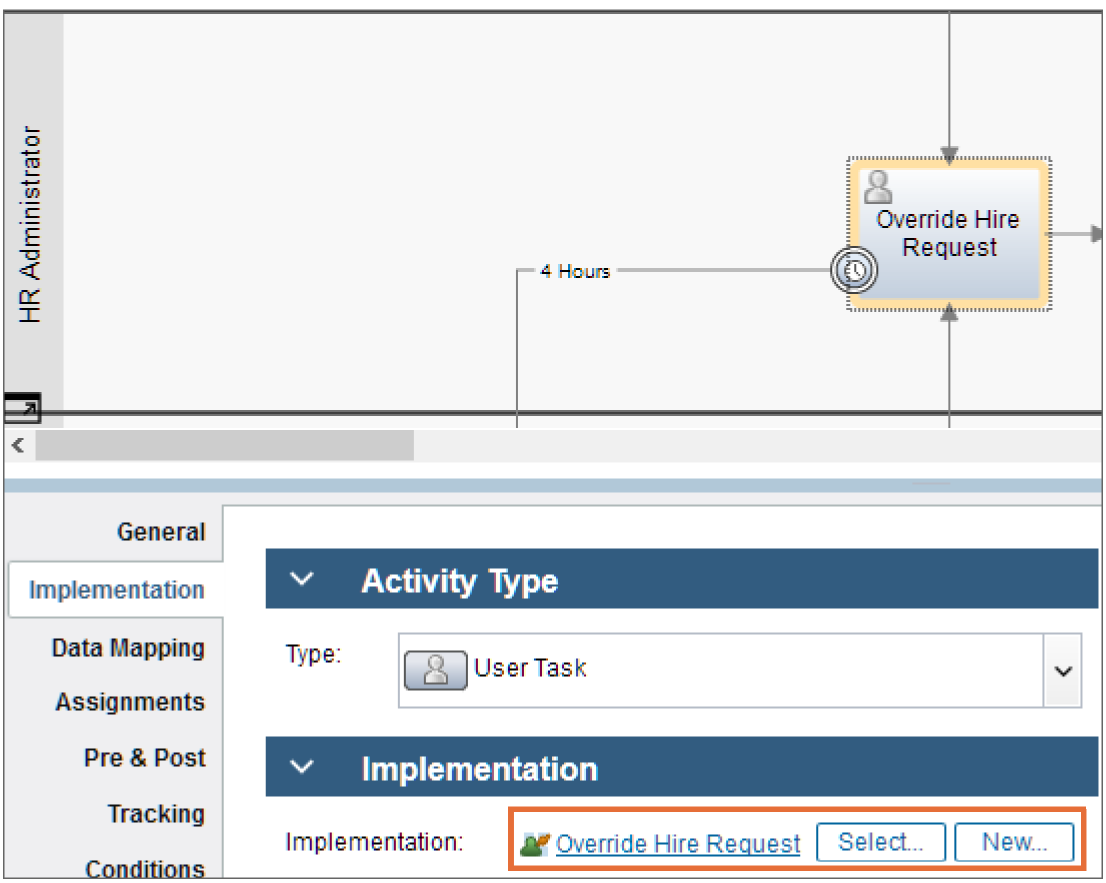
__ a. From the history menu, open the **Approve Hire Request** linked process.

__ b. On the **Variables** tab, create a private variable: `hrAdmin1(String)`




__ c. Click the **Definition** tab.

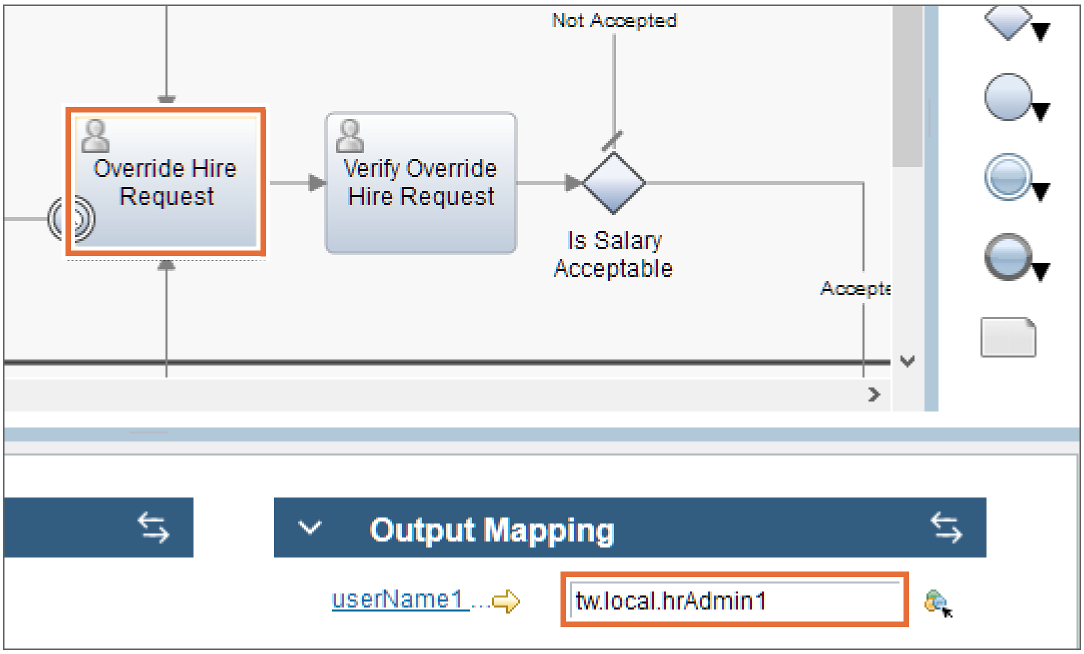
__ d. Implement the activity with the **Override Hire Request** client-side human service in the **Implementation** section. Click the **Override Hire Request** in the HR Administrator lane and click the **Implementation** properties menu. Select the Override Hire Request human service for the implementation.



__ e. Repeat the last step to implement the **Verify Override Hiring Request** activity with the same **Override Hire Request** client-side human service.

__ f. Click the **Override Hire Request** activity.

__ g. Open the **Data Mapping** properties menu and map the **userName1** output variable to the `tw.local.hrAdmin1(String)` private variable.




__ h. Map the **userName1** variable of the **Verify Override Hire Request** activity to the `tw.local.hrAmdin1 (String)` private variable.

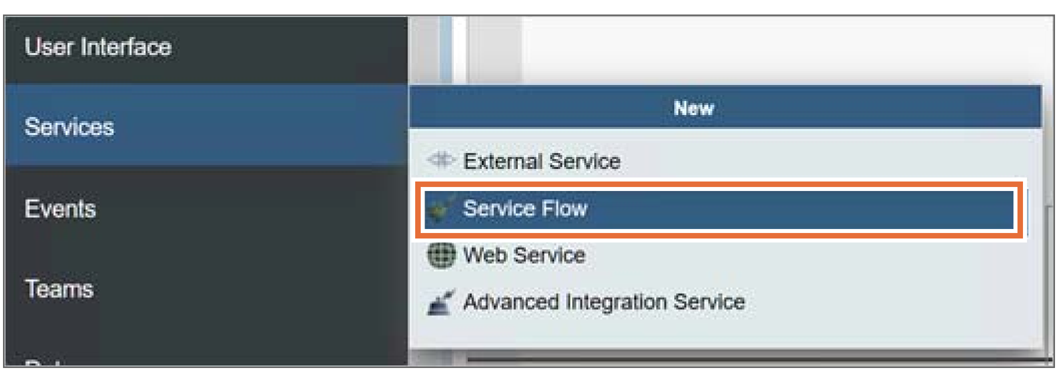
__ i. Save your work.

## Part 2: Create the team filter service

The four eyes principle requires that two people approve an activity. To implement the Four eyes policy, you need to know who the previous approver for the activity was, then filter that person out using Team Filter services.

__ 1. Create a team filter service to assign the second task to the correct pool of users.

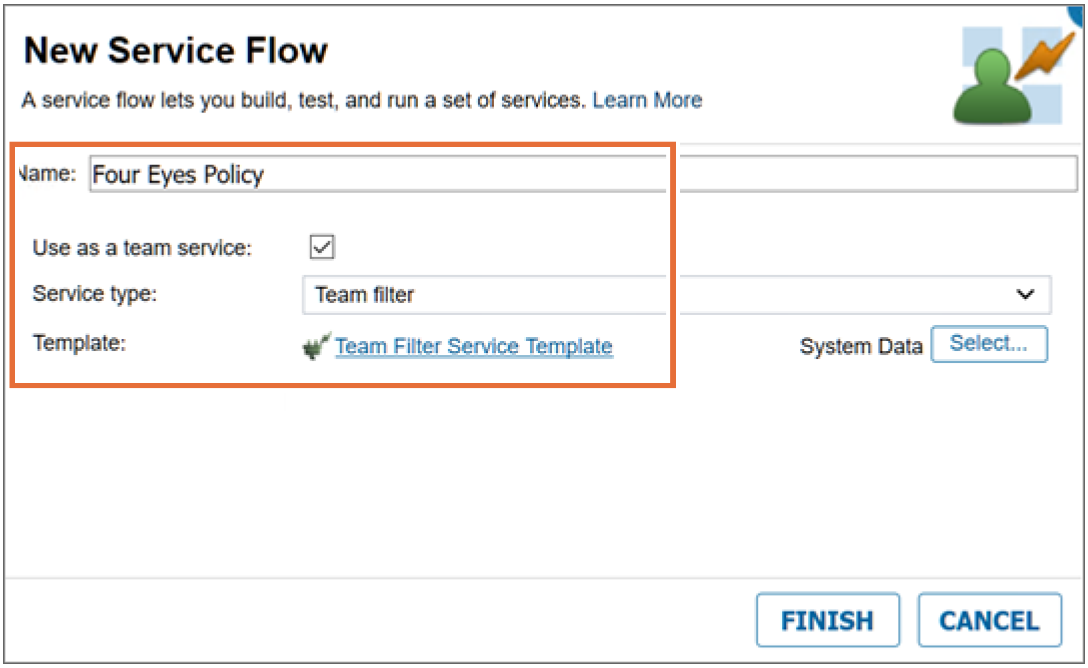
__ a. In the library, click the **plus sign (+)** next to the **Services** category, and then click **Service Flow**.



__ b. Name the service: `Four Eyes Policy`

__ c. Select the **Use as a team service** option.

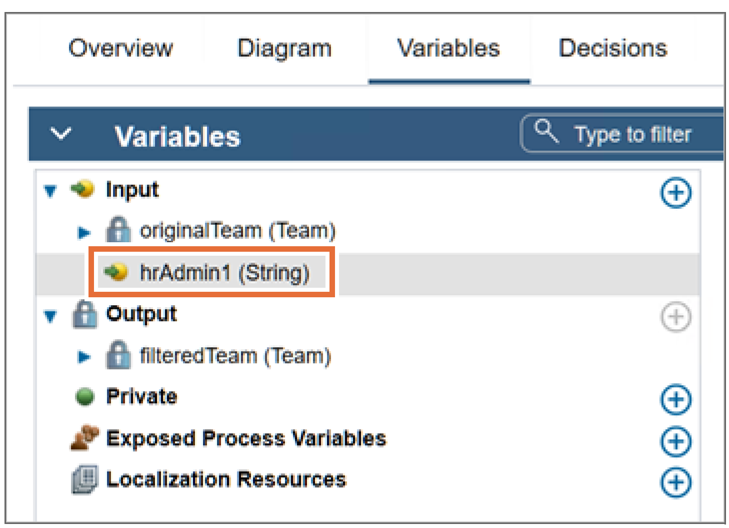
__ d. Select **Team filter** for the **Service type** and verify that the **Team Filter Service Template** is set as the **Template**. This template is part of the System Data toolkit. The template defines the inputs and output variables that are required for this type of service.




__ e. Click **FINISH**.

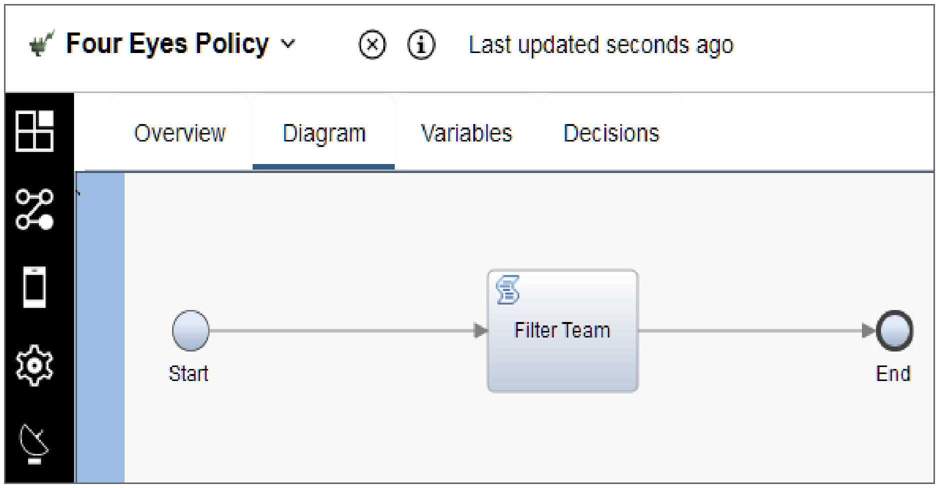
__ f. In the Four Eyes Policy service flow, click the **Variables** tab.

__ g. Add an input variable: `hrAdmin1 (String)`




__ h. Click the **Diagram** tab.

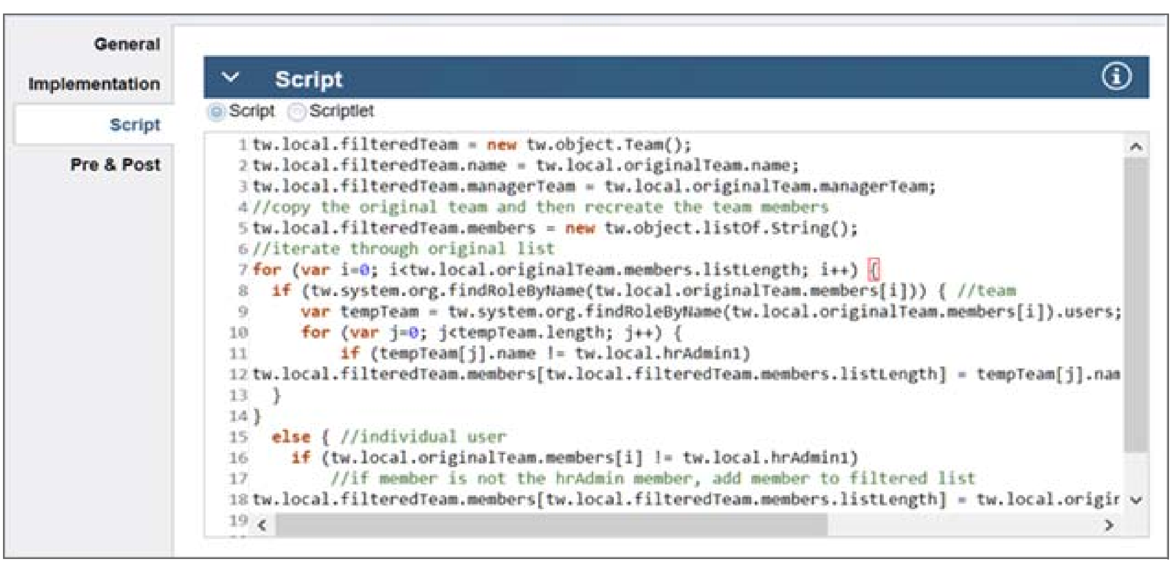
__ i. Drag a Server **Script** onto the sequence flow and name the it: `Filter Team`




__ j. Click the **Filter Team** server script, and then click the **Script** properties tab. Leave the default as **Script**. Do not use a Server **Scriptlet** for this task.

__ k. Filter the team by adding the following script. Alternatively, you can copy and paste the script that is provided in the file `FilterJS.txt` text file at the location: `C:\labfiles\Support files\Ex11`

```
tw.local.filteredTeam = new tw.object.Team();
tw.local.filteredTeam.name = tw.local.originalTeam.name;
tw.local.filteredTeam.managerTeam = tw.local.originalTeam.managerTeam;
//copy the original team and then recreate the team members

tw.local.filteredTeam.members = new tw.object.listOf.String();
//iterate through original list

for (var i=0; i<tw.local.originalTeam.members.listLength; i++) {
    if (tw.system.org.findRoleByName(tw.local.originalTeam.members[i])) {
//team
        var tempTeam =
tw.system.org.findRoleByName(tw.local.originalTeam.members[i]).users;
        for (var j=0; j<tempTeam.length; j++) {
            if (tempTeam[j].name != tw.local.hrAdmin1)

tw.local.filteredTeam.members[tw.local.filteredTeam.members.listLength] =
tempTeam[j].name;
        }
}
    else { //individual user
      if (tw.local.originalTeam.members[i] != tw.local.hrAdmin1)
          //if member is not the hrAdmin member, add member to filtered
list

tw.local.filteredTeam.members[tw.local.filteredTeam.members.listLength] =
tw.local.originalTeam.members[i].toString();
    }
}
```



__ l. Save and close the Four Eyes Policy service.

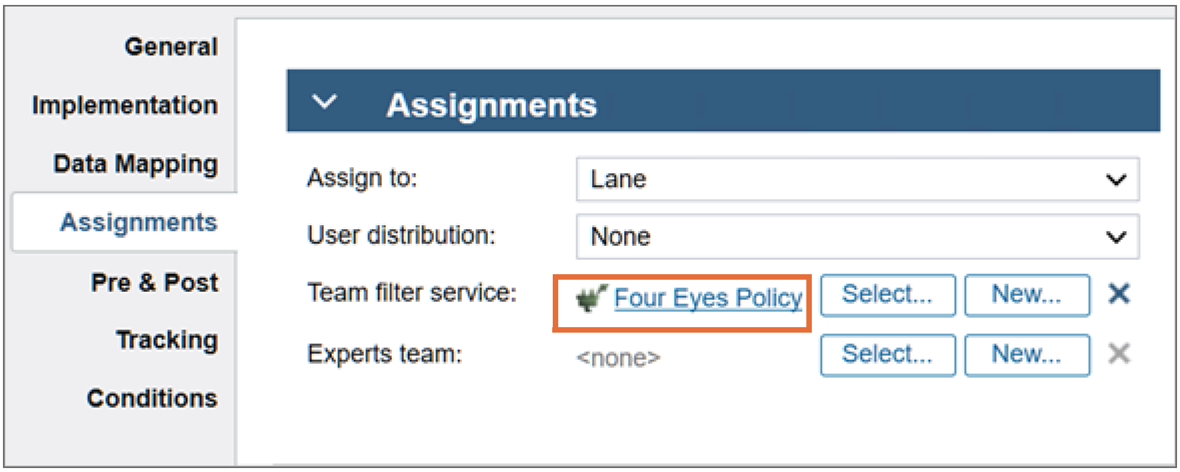
## *Part 3: Apply the team filter service to the activity assignment*

__ 1. Apply the team filter service.

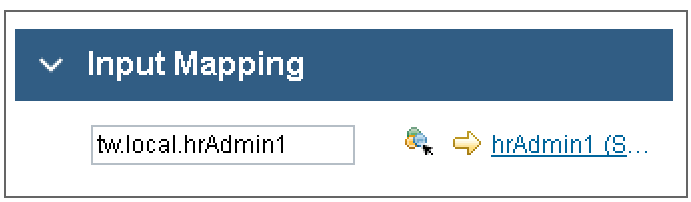
__ a. Return to the **Approve Hire Request** process.

__ b. Click the **Verify Override Hire Request** activity on the process.

__ c. From the **Assignments** properties menu, select the **Four Eyes Policy** as the Team Filter service.




__ d. In the input mapping section for the team filter service on the right, map the `tw.local.hrAdmin1` variable.



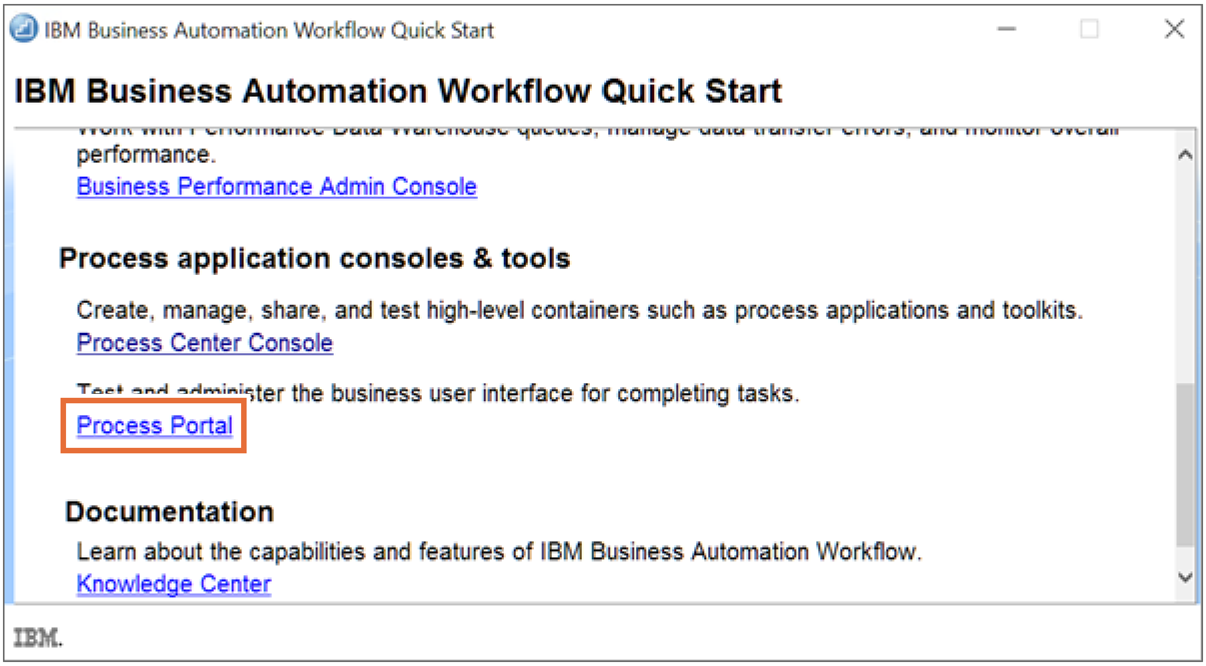

__ e. Save your work.

## Part 4: Test the four eyes policy

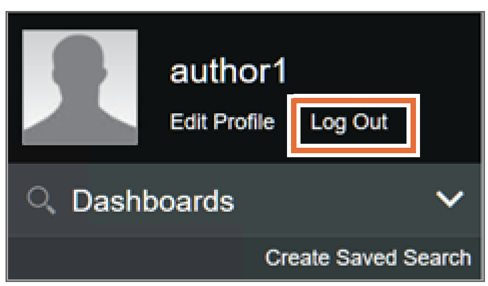
You now test the four eyes policy in the process portal.

__ 1. Log in as `user1` and start the process.

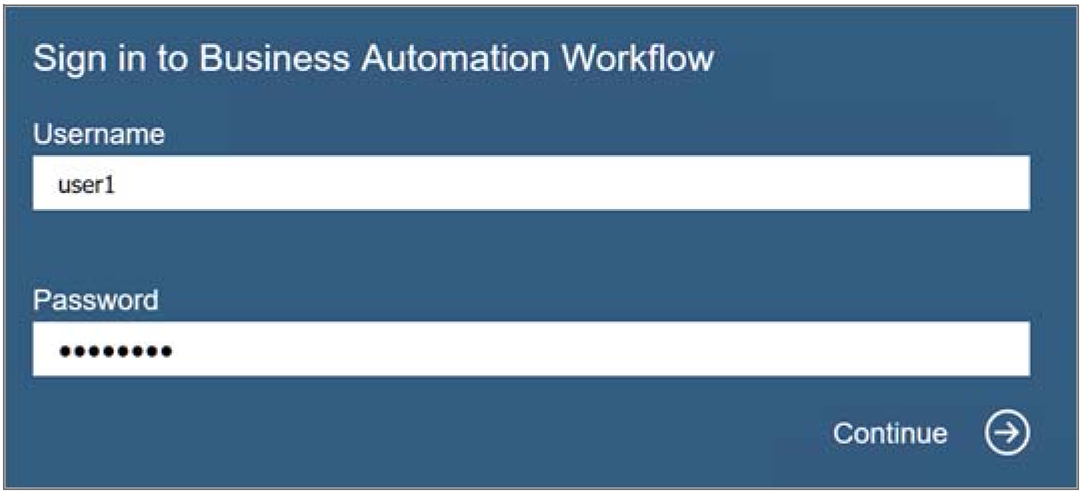
__ a. If not already open, click the link to the **Process Portal** by using the Quick Start menu.



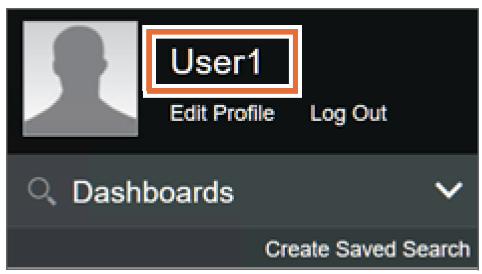

__ b. The Process Portal opens in a new browser tab. Since you were already logged in to the Workflow Center as **author1**, the session continues and you are logged in to Process Portal as **author1** also. For the team filtering to work successfully, you need to be logged in as a non-administrator user. If logged in as **author1**, log out of Process Portal by clicking **Log Out**.



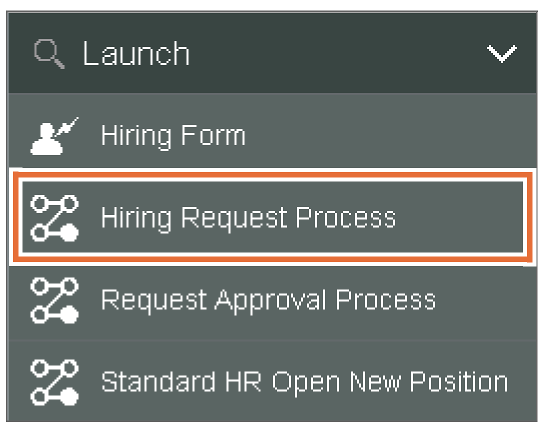
__ c. Log in to Process Portal using `user1` in the **Username** field and `passw0rd` in the **Password** field. Click **Continue**.




__ d. Verify that **User1** is listed at the upper left corner.




__ e. Create an instance of the process by clicking **Hiring Request Process** in the Launch section on the left. Wait for the first task to open in the main window.

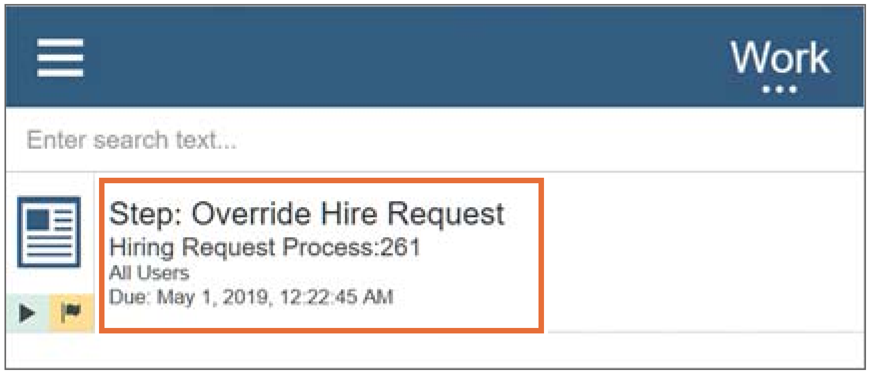



__ f. Complete the first task by creating a non-compliant requisition. Use the following values for the variables on the form (settings inside the browser in parentheses):

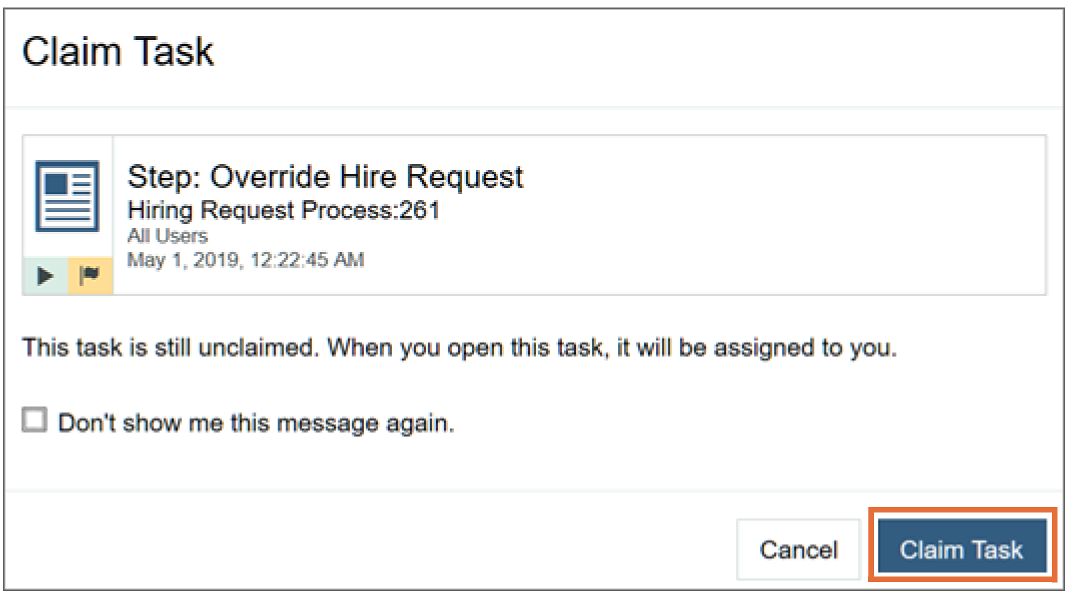
- `tw.local.requisitionDetails.position.jobLevel` (**Position Details > Job Level**) enter: `Manager`
- `tw.local.requisitionDetails.recruitingDetails.newPosition` (**Recruiting Details > New Position**) set to: `false` (not selected)
- `tw.local.requisitionDetails.compensationDetails.salaryToOffer` (**Compensation Details > Salary to Offer**) enter: `100000`

__ g. Click **Submit**.

__ h. When the portal refreshes, the new task **Override Hire Request** activity is shown in the work dashboard. This task is the first task in the four eyes requirement. Click the **Step: Override Hire Request** link to run the task.




__ i. Click **Claim Task**.



__ j. In the Override Hire Request page, click **Done**.

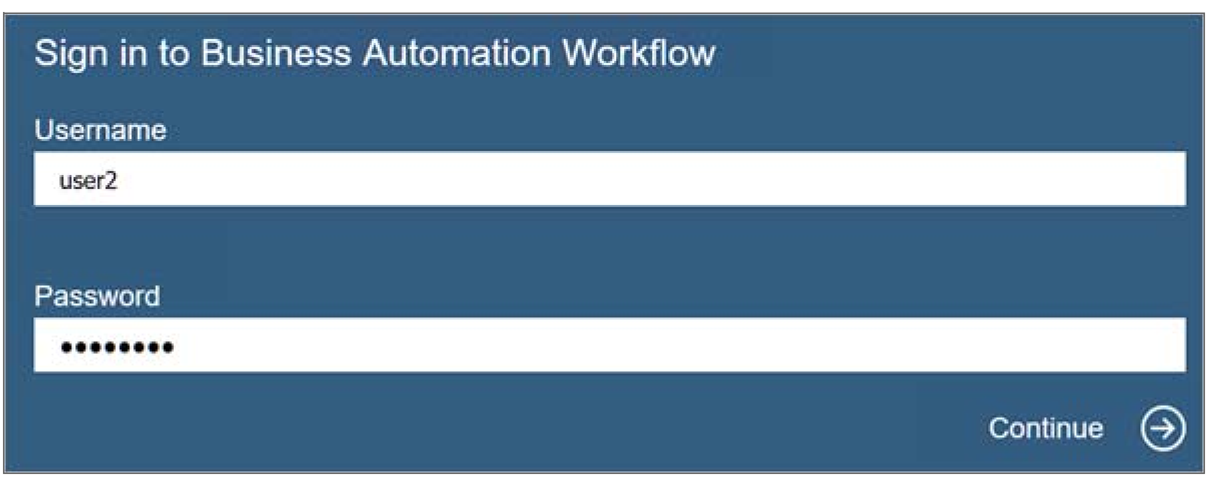

This coach is just a placeholder, and would be replaced with an approval form. You are trying to test the assignment, so ignore the placeholder.

__ k. Wait for the work dashboard to refresh. No other tasks are shown in the dashboard even though a second task is created and assigned to the All Users team, but the current user `user1` is filtered from the list. As a result, you do not see any task that is listed in the dashboard.

__ l. Click **Log Out** to logout of the Process Portal.

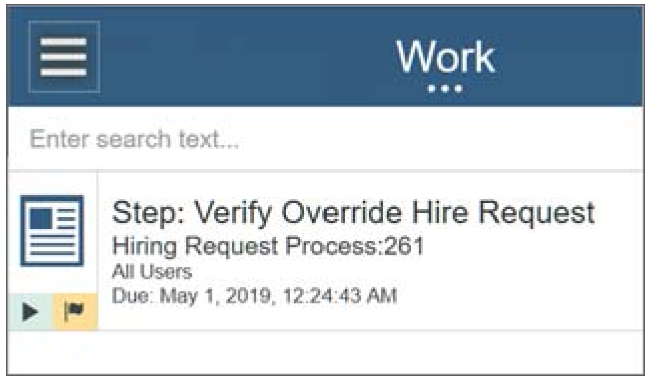
__ 2. Login as user2 and claim the **Verify Override Hire Request** task

__ a. Log back into the Process Portal, this time as user2. Enter `user2` in the **Username** field and `passw0rd` in the **Password** field. Click **Continue**.




The task is shown in the Work page. The **Verify Override Hire Request** task is assigned to “All Users,” which is misleading. Although the role might seem to contain all users, it is a filtered list. The original approver (user1) is filtered from the All Users list. The filter is the reason why the task is not listed in the user1’s work dashboard. The task is assigned to a team that filtered out the user who completed the override hire request task. In this example, user `user1` completed the task of **Override Hire Request**, so `user1` is filtered out and cannot complete the **Verify Override Hire Request**.

__ b. Click **Verify Override Hire Request**.




__ c. Click **Claim Task**.

__ d. In the Verify Hire Request coach that opens, click **Done**.


User2 successfully completed the **Verify Override Hire Request** task.

__ e. In the library on the left, click the link to **Log Out** of the process portal.

__ f. Close the Process Portal.

You have completed this exercise.

In the first part of the exercise, you modified the process and service to capture the first reviewer user ID. Next, you modified the activity on the process and created a routing policy. You tested the routing policy and verified that it filtered out the user from the list of participants available to claim the task.

## End of exercise

# Exercise 12. Building a cancellation pattern

## Estimated time

01:00

## Overview

In this exercise, you learn how to implement a cancellation pattern.

## Objectives

After completing this exercise, you should be able to:

- Implement a cancellation pattern in a process application
- Implement an undercover agent (UCA) to cancel the hiring request

## Introduction

A general manager requests the ability to cancel a hiring request while waiting for approval from the approvers. In this exercise, you create a cancellation message listener inside IBM Business Automation Workflow.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

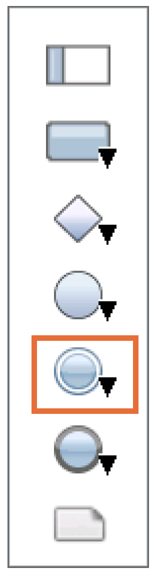
## *Part 1: Create the event*

__ 1. A cancellation event must be attached to the Approve Hire Request linked process.

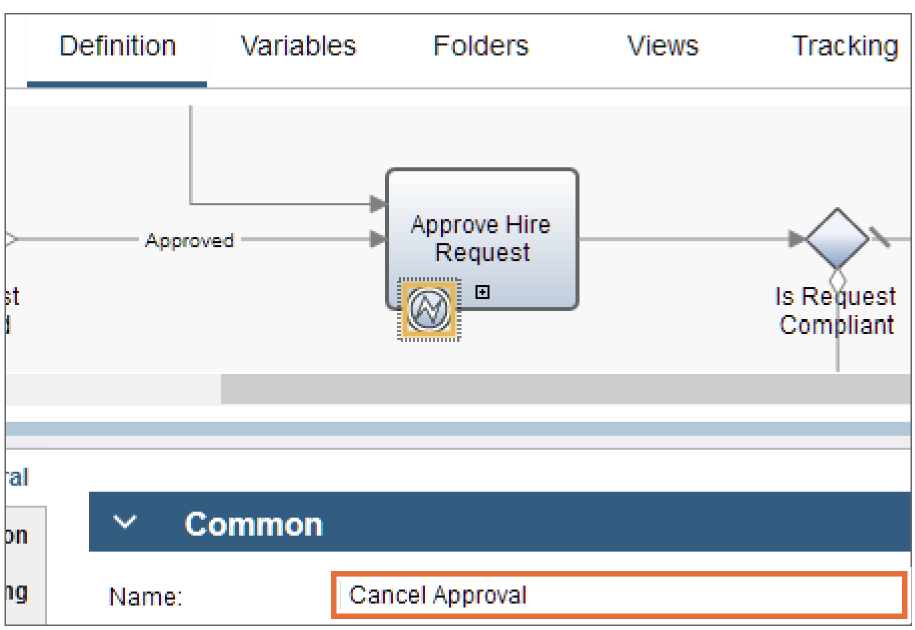
__ a. Open the **HR Recruitment Process** process application.

__ b. Open the **Hiring Request Process**.

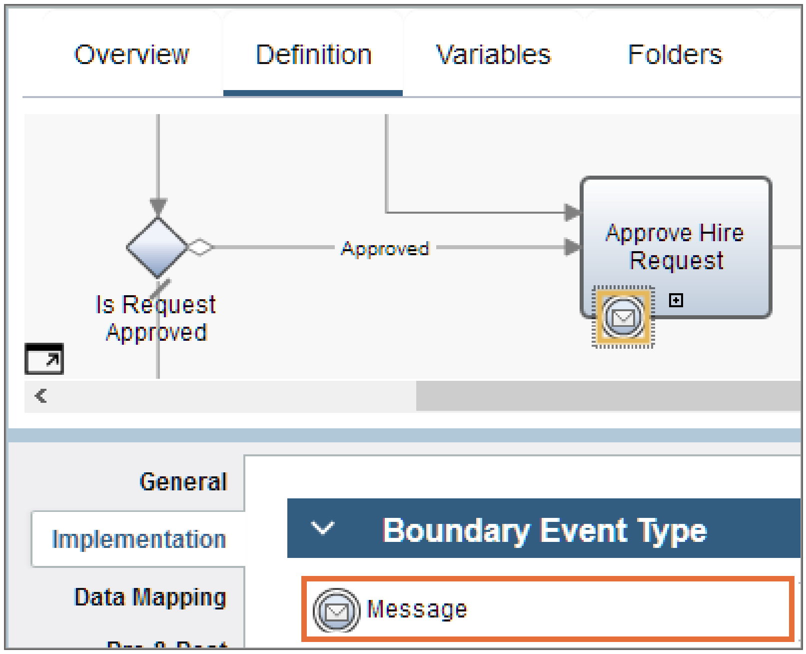
__ c. Drag an **intermediate event** from the palette and place it on one of the attachment points on the **Approve Hire Request** linked process. Make sure that you attach the intermediate event to the **Approve Hire Request** linked process and not the **Approve New Hire Request** activity.



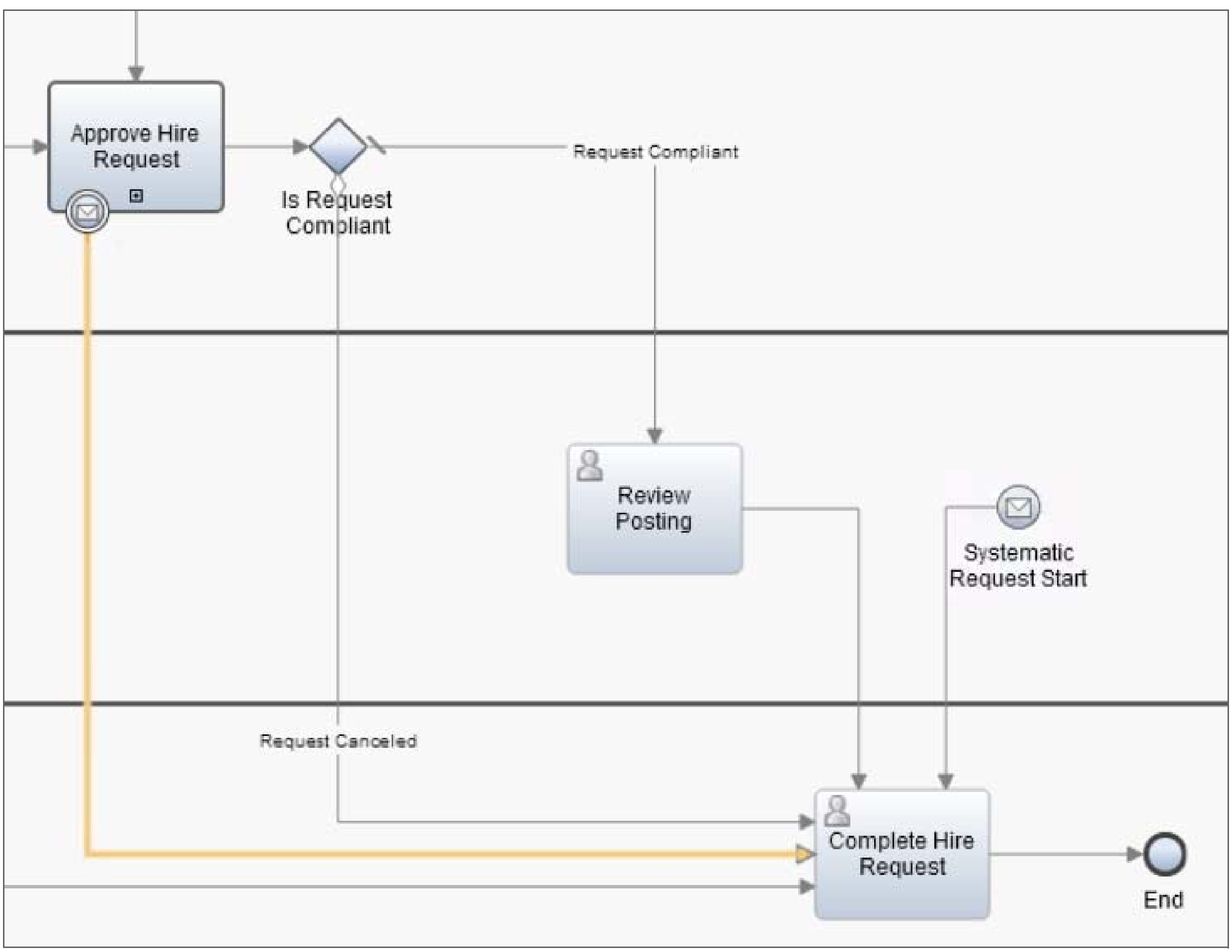
__ d. Verify that the attached intermediate event is selected. Change the name of the event to: `Cancel Approval`.



__ e. From the **Implementation** properties menu, change the Boundary Event Type to **Message**.




__ f. Create a flow from the attached message intermediate event to the **Complete Hire Request** system lane activity.



__ g. Save your changes.

**Important**

If an event changes the flow of a process similar to a cancellation event, you should also include some cleanup steps. This approach ensures that the process participants know about the event, and participants can react to the event as part of the process. Merely stopping or terminating an instance is not an ideal approach.

## *Part 2: Create an undercover agent*

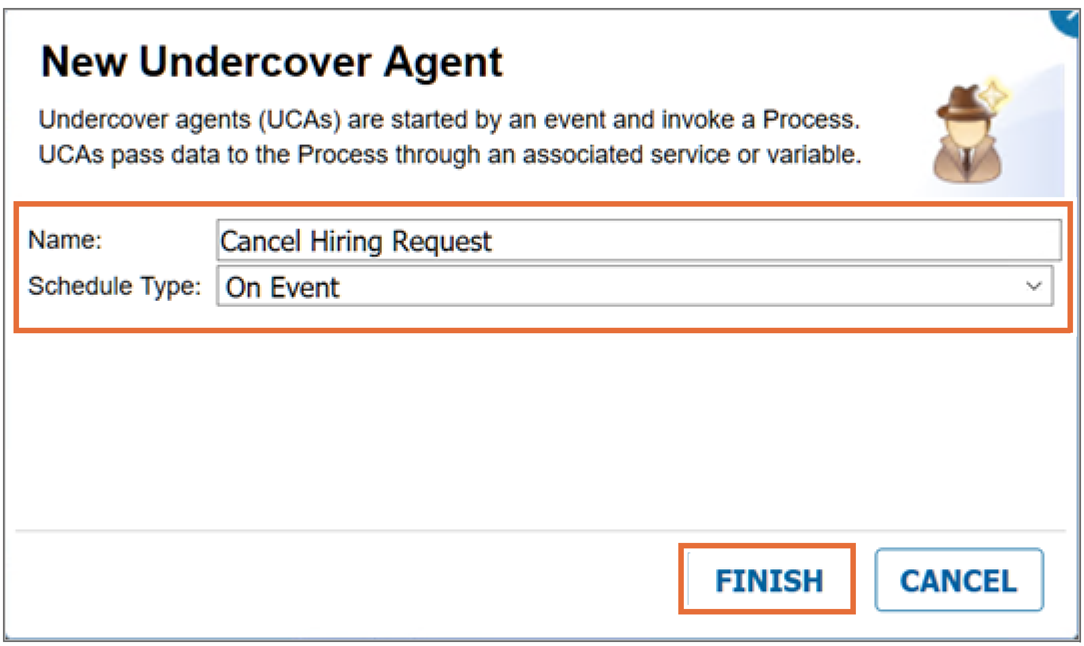
__ 1. Add an undercover agent in the library.

__ a. In the library, click the **plus sign (+)** next to the **Events** category, and click **Undercover Agent**.

__ b. Name the undercover agent: `Cancel Hiring Request`

__ c. Set the Schedule Type to: **On Event**.

__ d. Click **FINISH**.



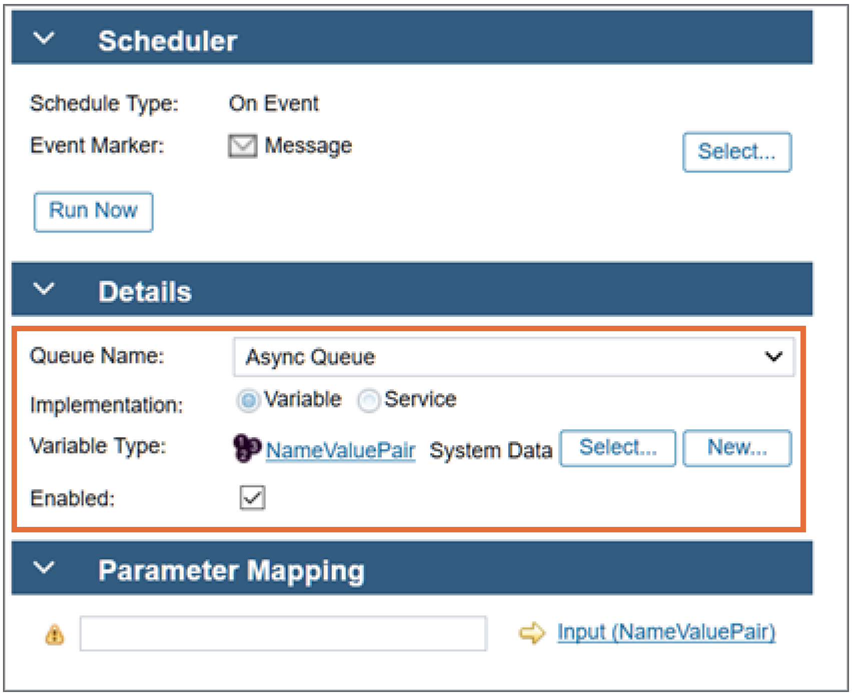

__ 2. Configure the undercover agent.

__ a. Under the **Details** section, keep all the default settings.

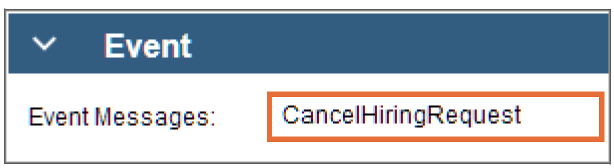
__ b. Verify that the **Implementation** type is set to `variable`.

__ c. Verify the default **Variable Type** is set to `NameValuePair`.

__ d. Verify that the **Enabled** checkbox is selected.




__ e. In the Event section at the bottom, change the Event Message to: `CancelHiringRequest`.



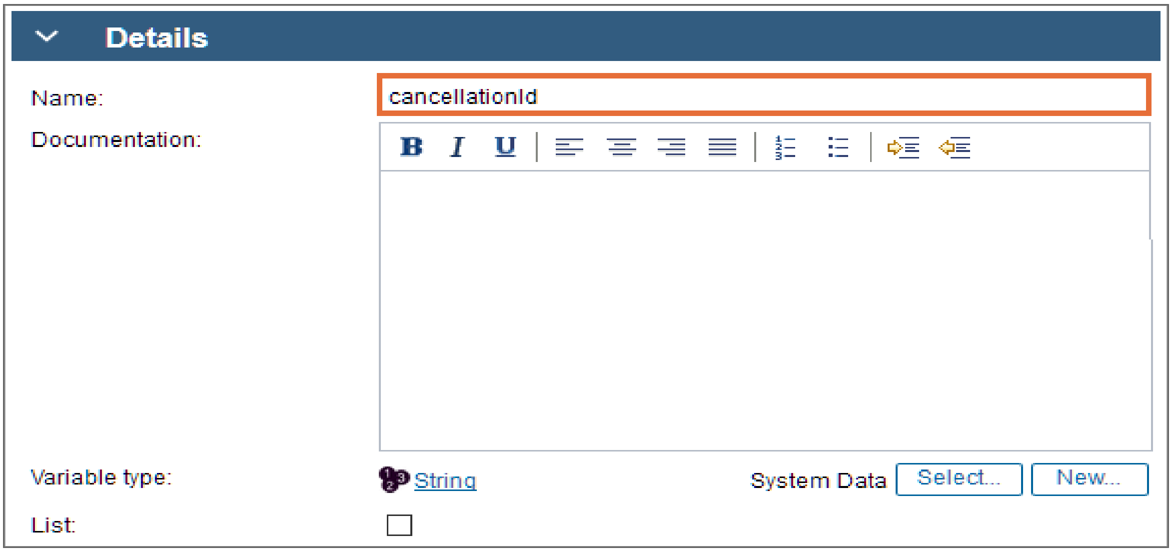

__ f. Save your work.

## *Part 3: Implement the boundary event with the UCA*

__ 1. Create a variable to correlate the UCA.

__ a. Switch to the **Hiring Request Process** in the Designer.

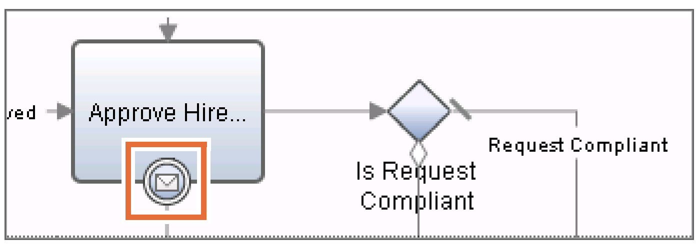
__ b. On the **Variables** tab, create a private variable: `cancellationId (String)`




__ 2. Configure the boundary event.

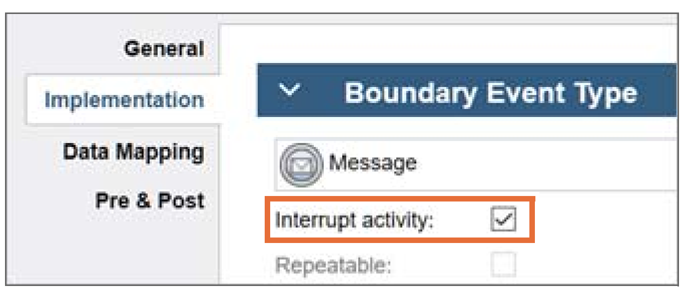
__ a. Click the **Definition** tab.

__ b. Click the **Approve Hire Request** boundary event.

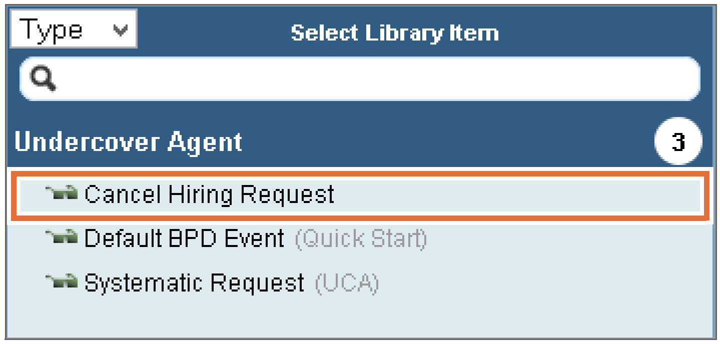



__ c. Click the **Implementation** properties menu.

__ d. Verify that the **Interrupt activity** checkbox is selected. The interrupt activity option consumes the token of the attached activity when the event is triggered. Because it is a linked process, even if numerous tokens exist inside the linked process, all the tokens are closed when the event is triggered.



__ e. In the **Event Properties** section, click **Select** next to the **Attached message UCA** and select **Cancel Hiring Request**.



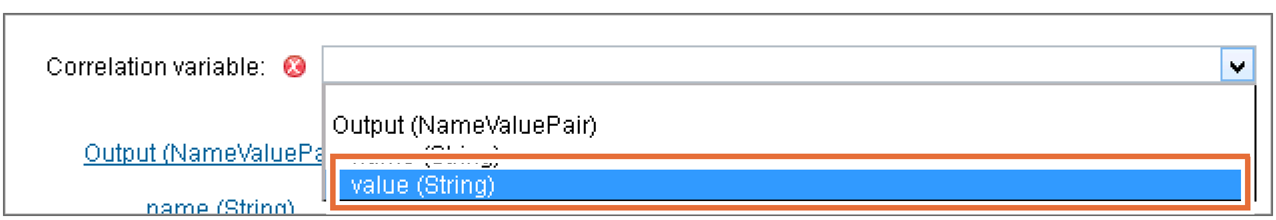

__ f. In the **Event Properties** section, select the **Consume message** checkbox. Consume message allows correlation with only a single process instance. If you configure multiple event listeners to correlate on the same message, then all events trigger if the **Consume message** checkbox is not selected.


__ g. Save your changes.

__ 3. Map the elements of the **Output (NameValuePair)** to simple variables in the process.

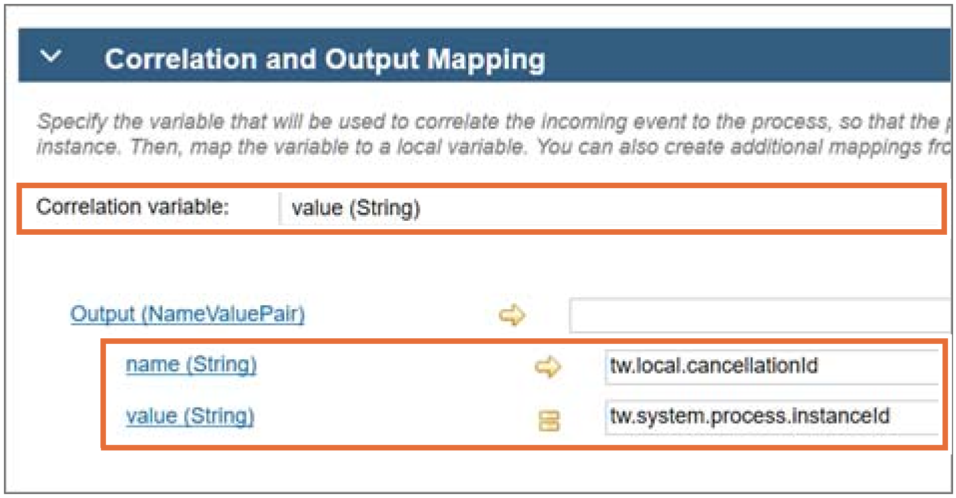
__ a. Click the **Data Mapping** properties menu.

__ b. From the **Correlation variable** list, select the **value (String)** variable.




__ c. Map the **Output.name (String)** value to the `tw.local.cancellationId` variable.

__ d. Map the Output.**value (String)** to the `tw.system.process.instanceId` variable.




The instance must know which message is targeting it. The messages sent to the UCA must contain a value that matches the instance ID of this process instance in order for this instance to receive the message. When the `Output.value` and the instance ID correlate, the intermediate event processes the message, and a token flows down the path. The user ID that canceled the process is also captured and mapped to a local variable.

__ e. Save your work.

## Part 4: Create the wrapper service

__ 1. Create a service to trigger the UCA.

__ a. In the library, click the **plus sign (+)** next to the **Services** menu and create a **Service Flow**.

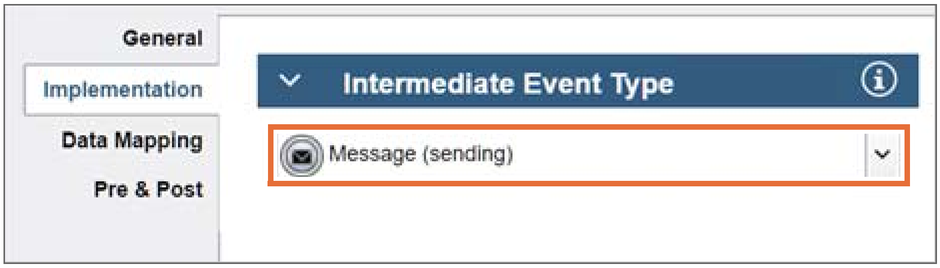
__ b. Name the service: `Cancel Hiring Request Trigger`

__ c. Click **FINISH**.

__ d. Drag a **Message (Sending) Intermediate Event** on to the sequence flow.

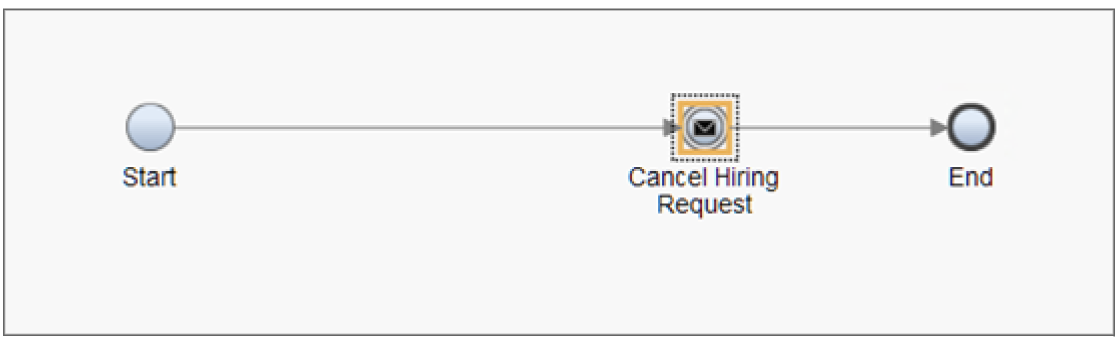
__ e. Select the **Implementation** properties section below.

__ f. Verify the implementation type is set to **Message (sending)**.

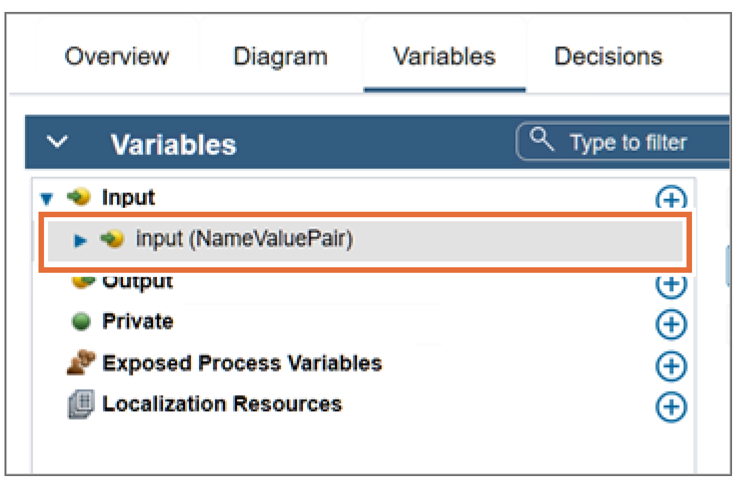



__ g. In the **Event Properties** section on the right, click **Select** next to **Attached message UCA** and select **Cancel Hiring Request**.


__ h. Rename the intermediate message event: `Cancel Hiring Request`.




__ i. On the **Variables** tab, create an input variable: `input (NameValuePair)`




__ j. Save your changes.

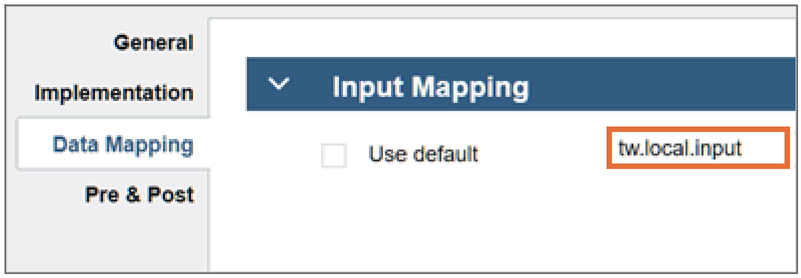
__ 2. Map the variable.

__ a. Click the **Diagram** tab.

__ b. Click the **Cancel Hiring Request** event.

__ c. Click the **Data Mapping** properties menu.

__ d. Map the input of the UCA to the `tw.local.input` variable.



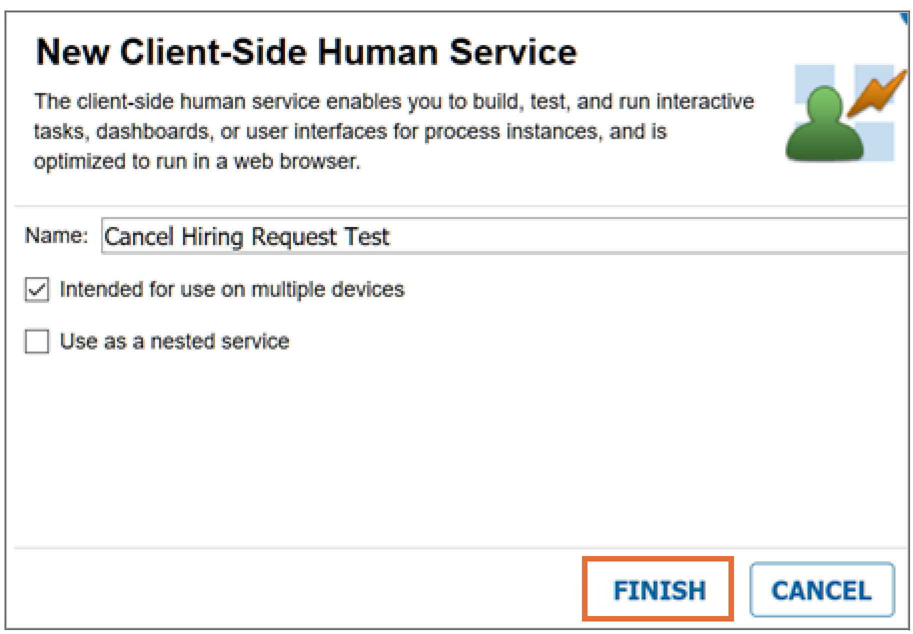

__ e. Save your work.

## Part 5: Create a test harness

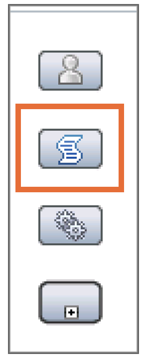
__ 1. Add a client-side human service to test the UCA.

__ a. In the library, click the **plus sign (+)** next to **User Interface**. Click **Client-Side Human Service**.

__ b. Name the service: `Cancel Hiring Request Test`. Select the **Intended for use on multiple devices** checkbox. Click **FINISH**.



__ c. Drag a Client-side script from the palette and place it next to the Start event on the sequence flow. Name it: `Create NVP`



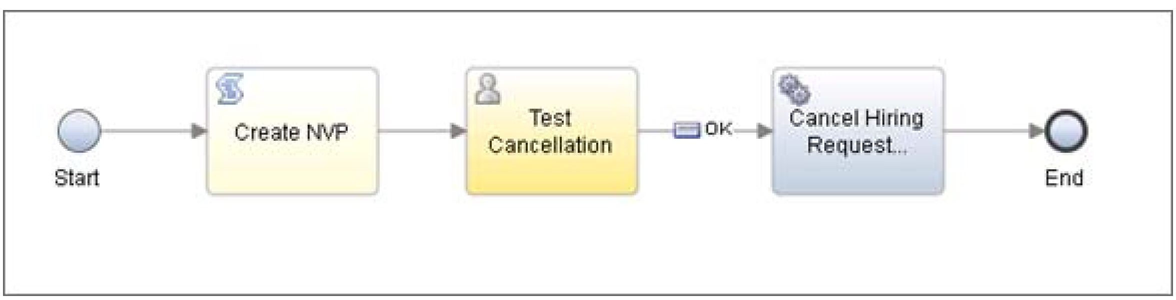
**Information**

The abbreviation for name-value pair is NVP for this step.

__ d. Select the coach on the canvas and name it: `Test Cancellation`

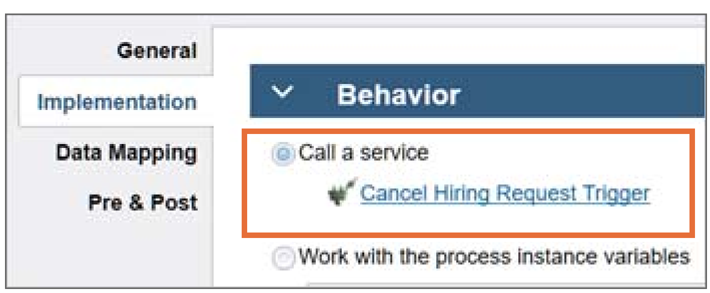
__ e. Drag a service task from the palette onto the sequence flow to the right of the coach and name it: `Cancel Hiring Request Trigger`

__ f. Connect the flows.




__ g. Select the **Implementation** properties menu.

__ h. For the **Call a service**, click **Select**, and choose the **Cancel Hiring Request Trigger** service.

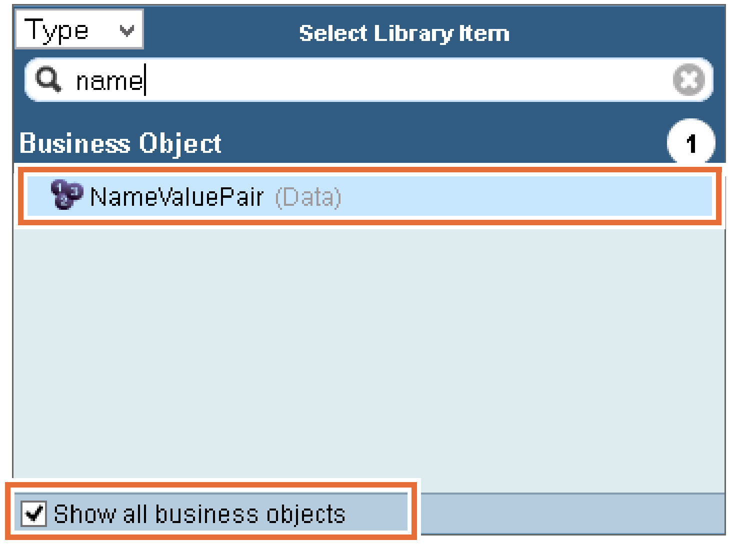



__ i. Save your changes.

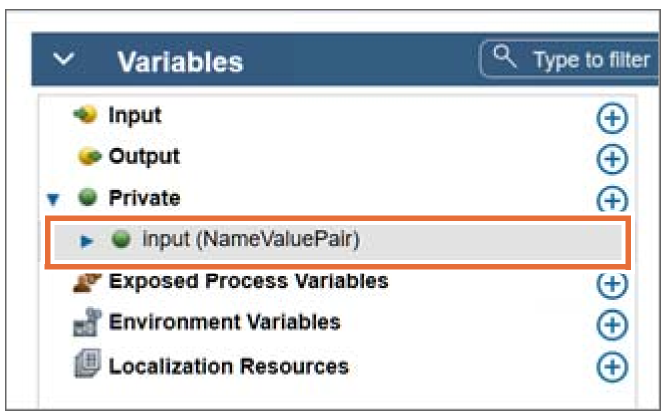
__ 2. Implement the Create NVP step on the canvas.

__ a. On the **Variables** tab, create a private variable: `input`.

__ b. Assign the variable type to **NameValuePair**.




__ c. Verify that the variable is created with the **NameValuePair** type and save your work.

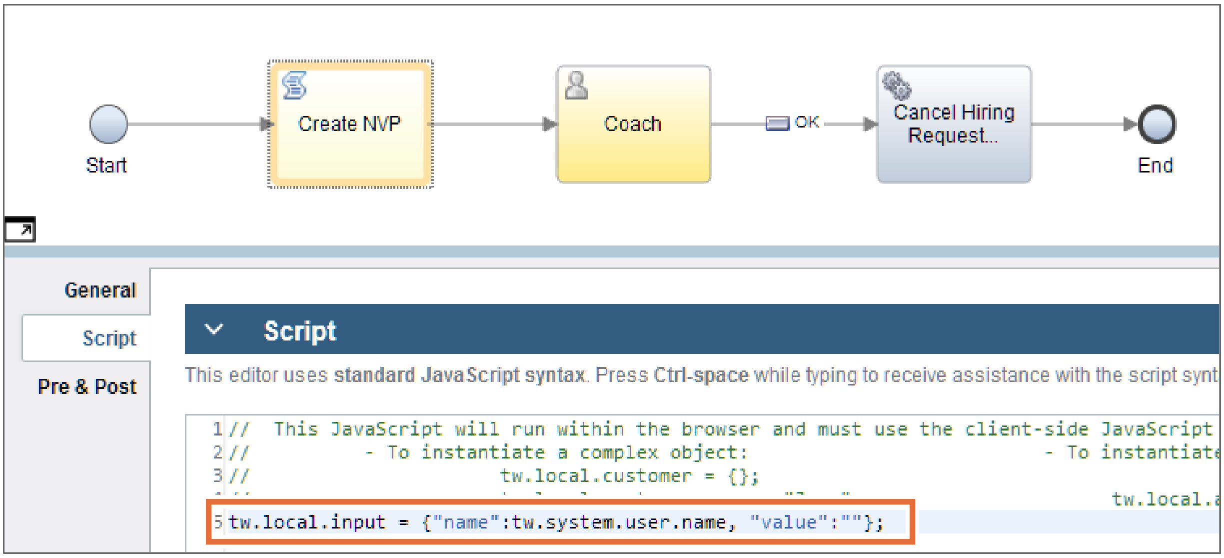



__ d. Click the **Diagram** tab.

__ e. Select the **Create NVP** script, and click the **Script** properties menu.

__ f. Copy the following code and paste it into the code block. Alternatively, you can copy and paste the script that is provided in the file `CreateNVP.txt` text file at the location: `C:\labfiles\Support files\Ex12`.

```
tw.local.input = {"name":tw.system.user.name, "value":""};
```



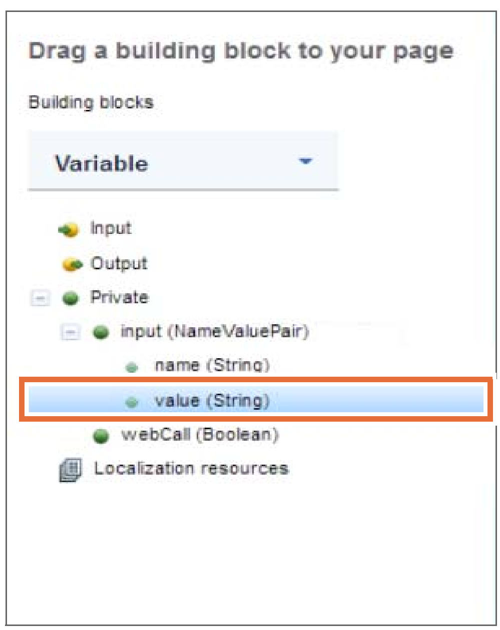

__ g. Save your changes.

__ 3. Build the Test Cancellation coach.

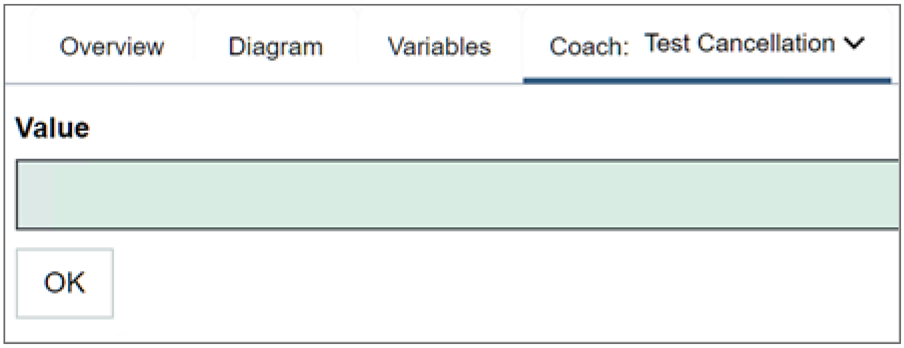
__ a. On the **Coaches** tab, click the **Test Cancellation** coach.

__ b. Select **Default coach** as the starting point and click **OK**.

__ c. Select **Variable** in the **Building blocks** panel and select **Private > input > value**



__ d. While holding down the left mouse button, drag the `input.value` variable to the top of the **OK** button. Let go of the mouse button when the top border displays a gold highlight



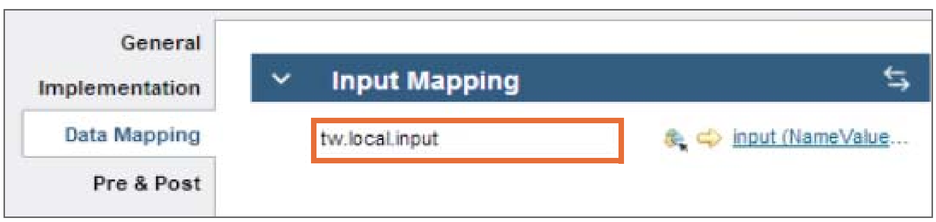

__ e. Change the input field label to: `Instance ID`

__ 4. Map the input variable for the Cancel Hiring Request Trigger step on the canvas.

__ a. Open the **Diagram** tab.

__ b. Click the **Cancel Hiring Request Trigger** service.

__ c. Click the **Data Mapping** properties menu, and map the `tw.local.input` private variable to the input.



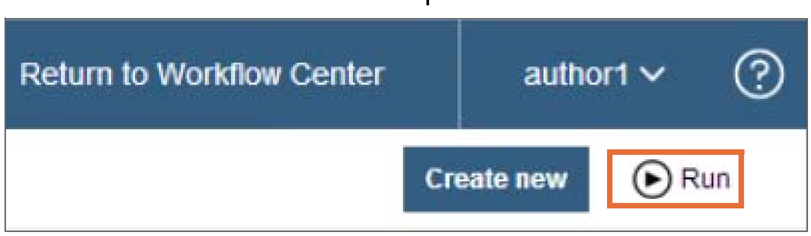

__ d. Save your work.

## *Part 6: Test the cancellation UCA*

__ 1. Create an instance of the Hiring Request Process.

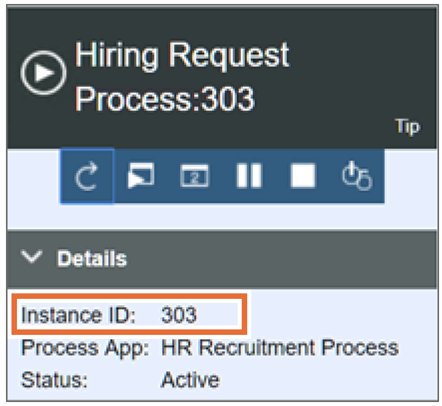
__ a. Open the **Hiring Request Process**

__ b. Click **Run** to create an instance of the process.




__ c. The perspective changes to the **Inspector** tab, and the information bar opens on the right.

__ d. The **Instance ID** in the Process Inspector is listed on the right. Make a note of this value. You use this number in the next step to cancel the instance. Your instance ID will be different than the one in the screen capture.




__ 2. Complete the first activity in the process.

__ a. To complete the first activity and get to the linked process, make sure that the **Locations** section is expanded.

__ b. Click **Start** to run the Submit Hiring Request task from the Inspector information bar.


__ c. Use the following values for the variables on the form:

- Position Details > Job Level set to: `Manager`
- Recruiting Details > New position set to **false** (not selected)
- Compensation Details > Salary to offer set to `100000`

__ d. Click **Submit**.

__ e. When the service completes, close the coach browser window.

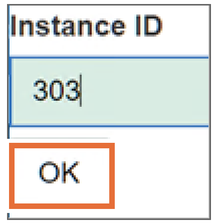
When the Inspector refreshes, a token is associated with the Approve Hire Request linked process. Next, you cancel the linked process and close all the tokens that are associated with the linked process by sending a message to the attached intermediate message event.

__ 3. Trigger the cancellation event on the linked process.

__ a. Open the **User Interface > Cancel Hiring Request Test** client-side human service.

__ b. Click **Run**.

__ c. Input the instance ID that you noted earlier. Click **OK**.

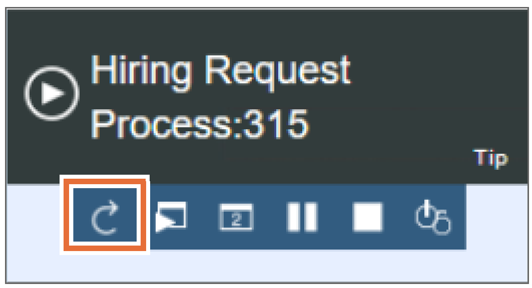



__ d. When the service completes, close the window.

__ 4. Verify that the linked process is canceled.

__ a. Return to the **Hiring Request Process** in the Inspector view.

__ b. Refresh the Process Inspector.




__ c. Verify that the message consumed the token for the **Verify Override Hire Request** linked process and all tokens that are associated with the linked process are closed.

__ d. In the Locations section on the right, click **Start** to run the Complete Hire Request activity.


__ e. Click **Done**

__ f. Close the browser window.

__ g. Return to the **Designer** tab.

All the tasks complete and the process consumes all the tokens in the linked process. The process flow travels out of the attached message event to the system lane activity. After the Complete Hire Request activity completes, the token moves to the End event and the process is complete. The instance status changes to **Completed**.

You have completed this exercise.

In the first part of the exercise, you created an attached intermediate event on the process. You implemented it as an intermediate message event. You created the enabling service, created the on-demand UCA, and then implemented the message service. You then created a service to wrap the UCA and created a test harness to trigger the UCA on the coach.

## End of exercise

# Exercise 13. Building web service connections

## Estimated time

01:00

## Overview

In this exercise, you learn how to build inbound and outbound web service connections.

## Objectives

After completing this exercise, you should be able to:

- Create an event-based undercover agent
- Build an inbound web service connection
- Build an outbound web service to message the inbound web service

## Introduction

The project sponsor identified a requirement to expose a cancellation service to an external system. The ability to cancel a request must be enabled internally, but also must be available to systems that integrate externally with IBM Business Automation Workflow.

You must create a cancellation message listener as an inbound web service inside IBM Business Automation Workflow. Then, create a web service to test or simulate an external system, which connects to the inbound web service. This activity gives you practice in creating web services and calling external SOAP services. Even though the message is targeting the localhost, the approach and concepts that are used in this activity are the same for an external system.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

# Exercise instructions

## *Part 1: Create an inbound web service to trigger the UCA*

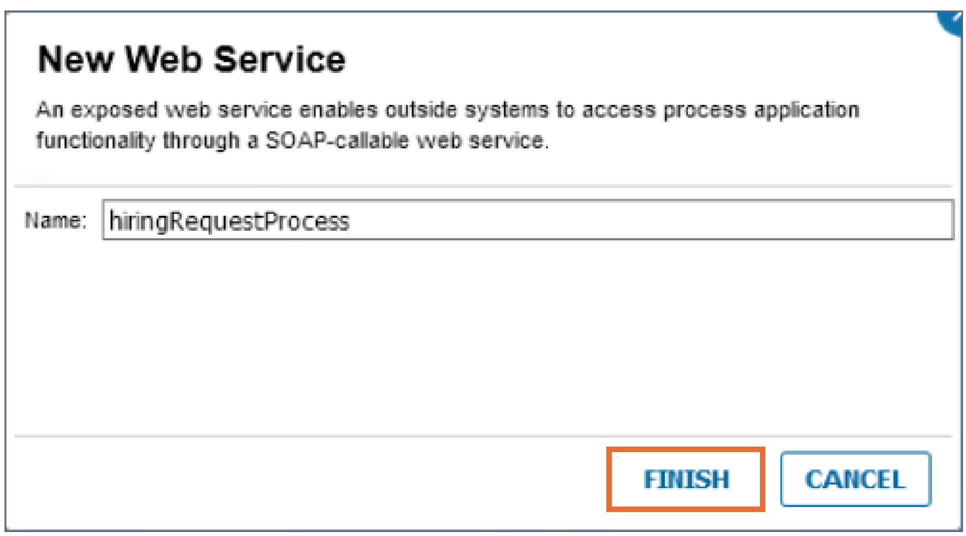
__ 1. Create an inbound web service.

__ a. Open the **HR Recruitment Process** process application.

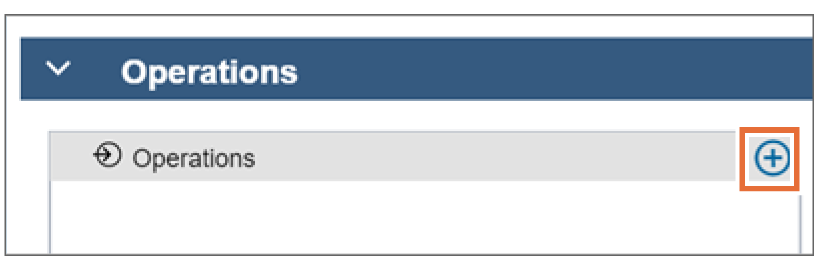
__ b. In the Process Designer library, click the **plus sign (+)** next to the **Services** category.

__ c. Click **Web Service**.


__ d. Name the web service: `hiringRequestProcess` and click **FINISH**.

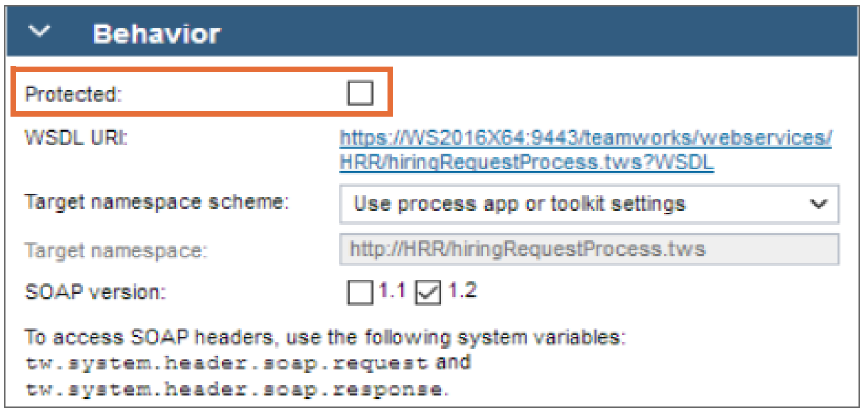



__ e. Click **Add** (plus icon) under the **Operations** section and select the **Cancel Hiring Request Trigger** service.



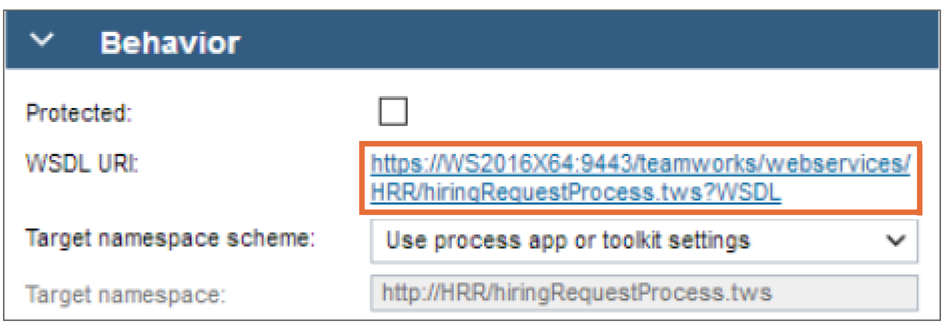
__ f. Under the **Operation Detail** section, change the **Operation Name** to: `cancel`


__ g. Under the **Behavior section**, verify that the **Protected** checkbox is not selected.




__ h. Save your work.

__ i. Click the **WSDL URI** link.



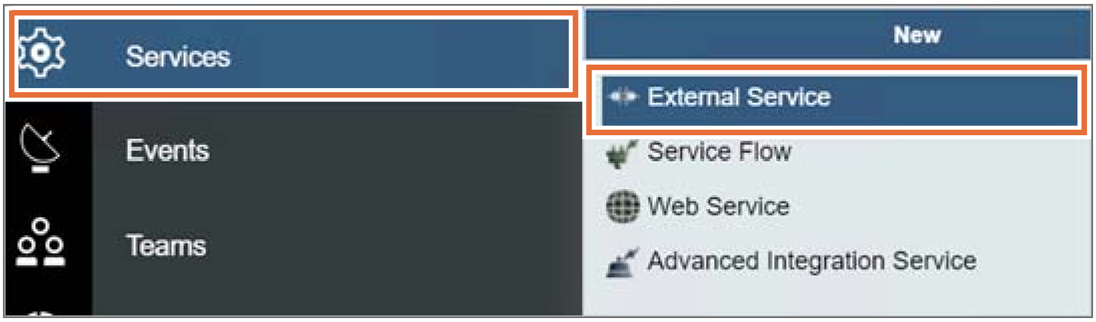

__ j. The WSDL is loaded in your browser. The information about the web service you created is shown to include the methods available, the variable inputs and outputs, and metadata about the service call. Highlight the address in the browser address bar, right-click, and click **Copy**.


The URL is used to define the server setting for the **Web Service Integration** component in the outbound integration in the following steps.

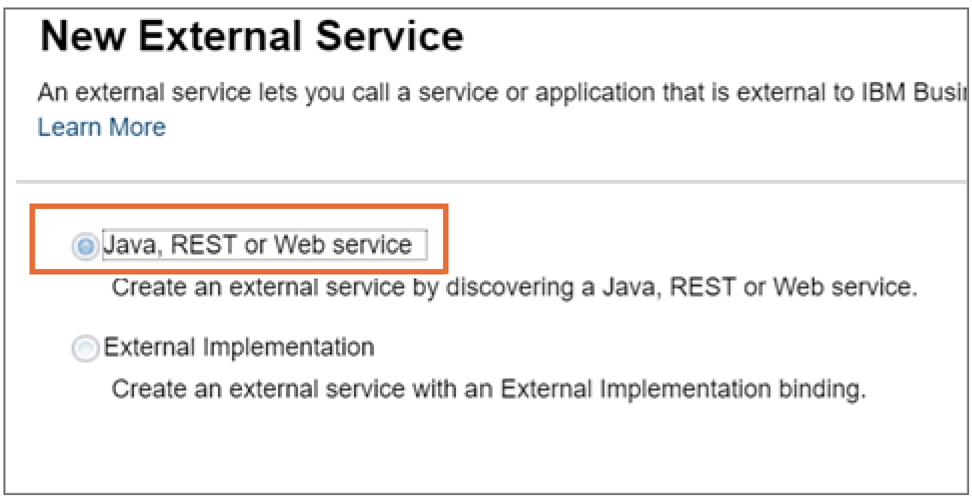
## *Part 2: Test the inbound integration by creating an outbound integration*

__ 1. Create an outbound integration.

__ a. In the library, click the **plus sign (+)** next to the **Services** category, and select **External Service**.



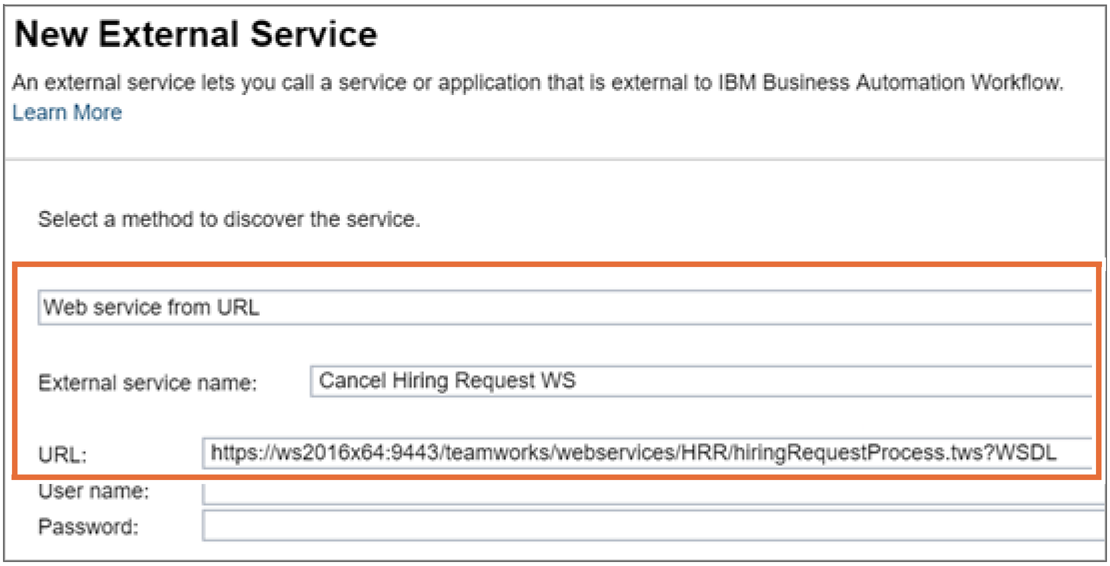

__ b. Leave the **Java REST or Web service** option selected and click **NEXT**.




__ c. Select **Web Service from URL** as the method to discover the service.

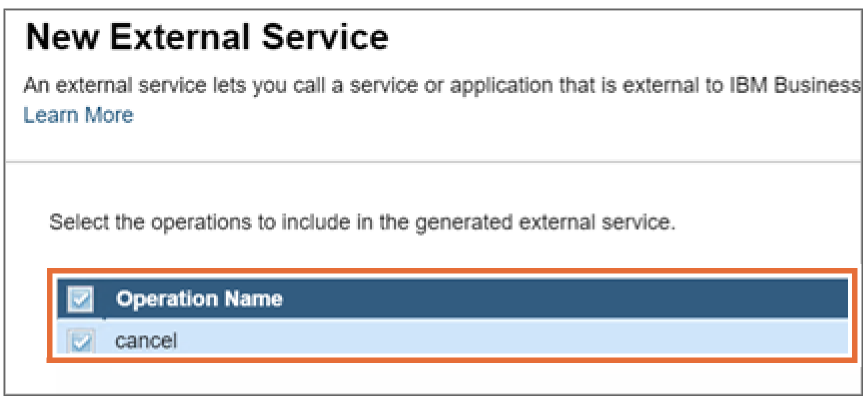
__ d. For the **External service name**, enter `Cancel Hiring Request WS`

__ e. In the URL field, enter the URL that you copied earlier. If needed, switch to the browser tab and copy it.

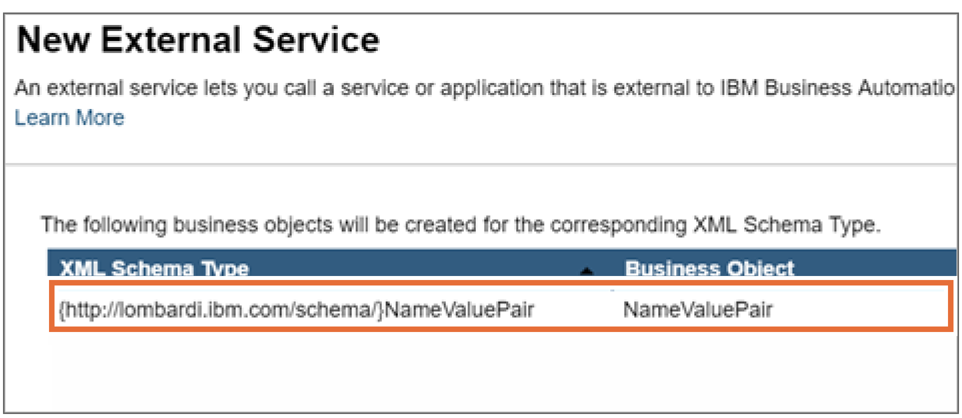



__ f. Click **NEXT**.

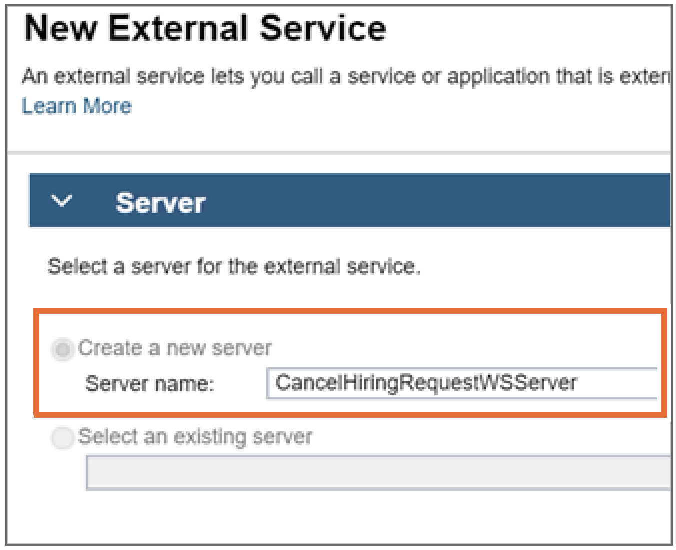
__ g. Leave the default operation selected and click **NEXT**.




__ h. Leave the defaults for the request and response wrappers (unchecked) and the business objects to be created and click **NEXT**.



__ i. Leave the default new server name and click **FINISH**.



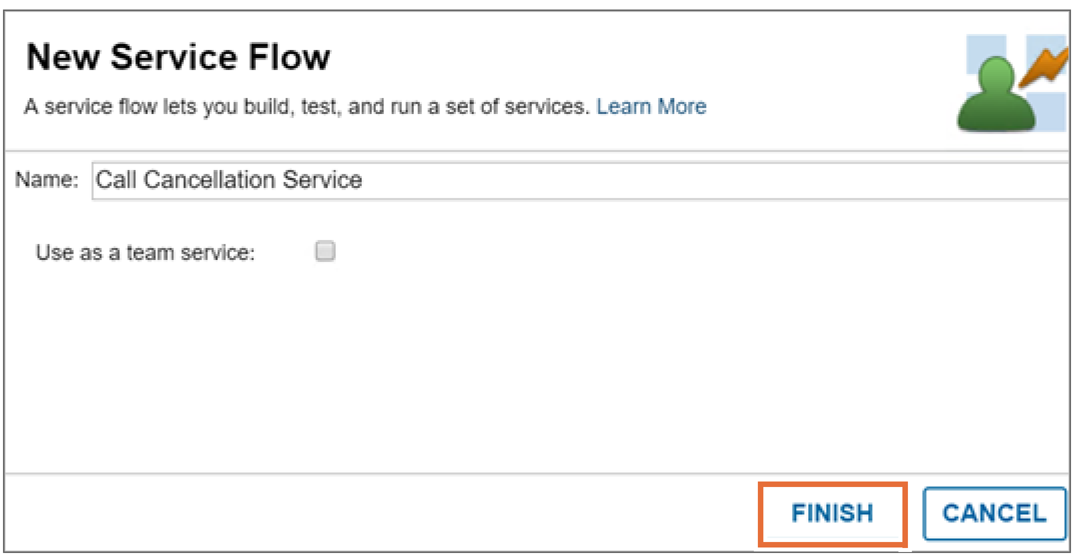

The external service configuration is displayed after you click **FINISH**.

__ 2. Create the service flow for the web service.

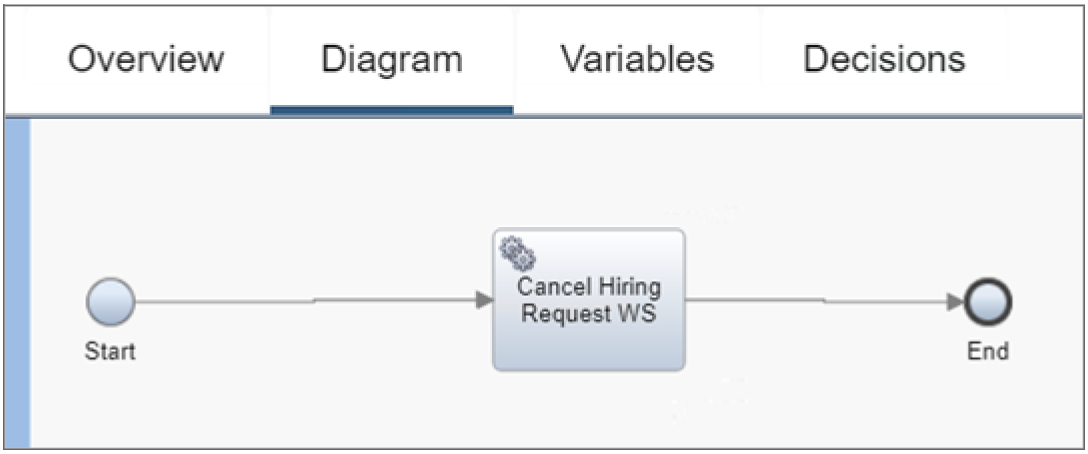
__ a. In the library, click the **plus sign (+)** next to the **Services** category, and select **Service Flow**.

__ b. Name the Service flow: `Call Cancellation Service`

__ c. Click **FINISH**.



__ d. In the library, click **Services**, locate **Cancel Hiring Request WS** and drag it to the sequence flow for the **Call Cancellation Service** flow.



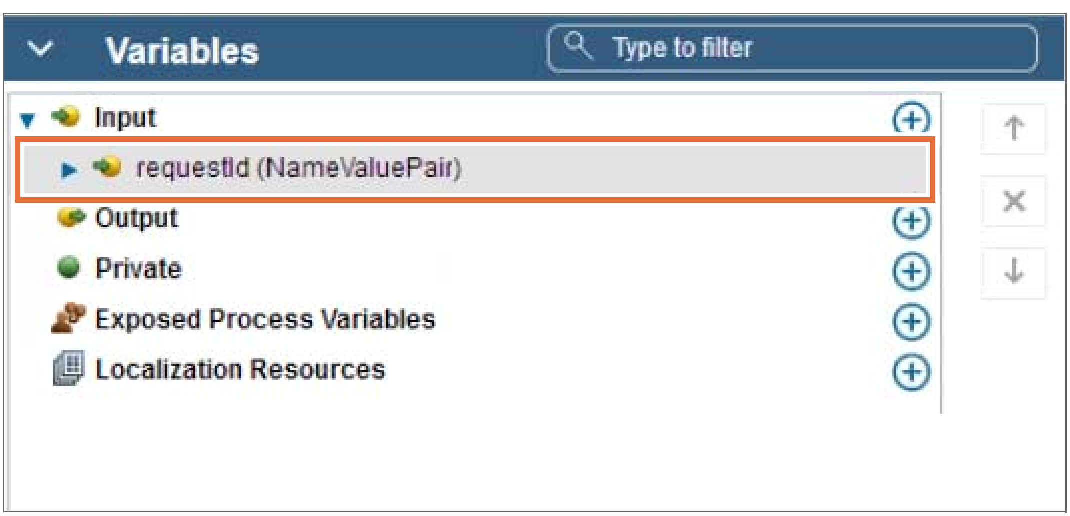

**Troubleshooting**

If you cannot drag the service to the canvas, close the browser and reopen the process application.

__ e. With the **Cancel Hiring Request WS** service highlighted, click the **Implementation** properties menu.

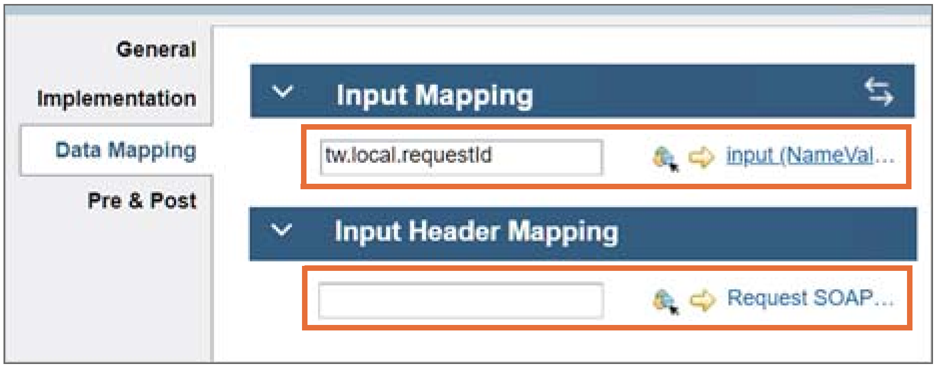
__ f. In the Implementation section, change the **Operation** value to `cancel`.


__ g. On the **Variables** tab, add an input variable named: `requestId (NameValuePair)`



__ h. Click the **Diagram** tab. Click the **Cancel Hiring Request WS** component. Open the **Data Mapping** properties menu.

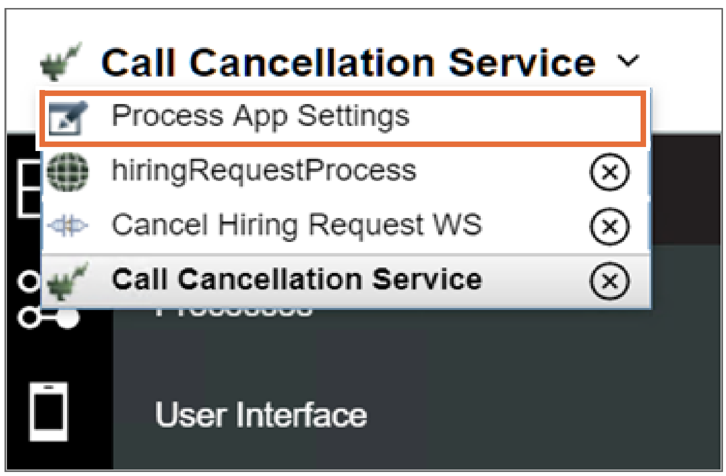
__ i. In the Input Mapping section, map the `tw.local.requestId` variable to the input of **Cancel Hiring Request WS**. Leave Input Header Mapping and Output Header Mapping blank.



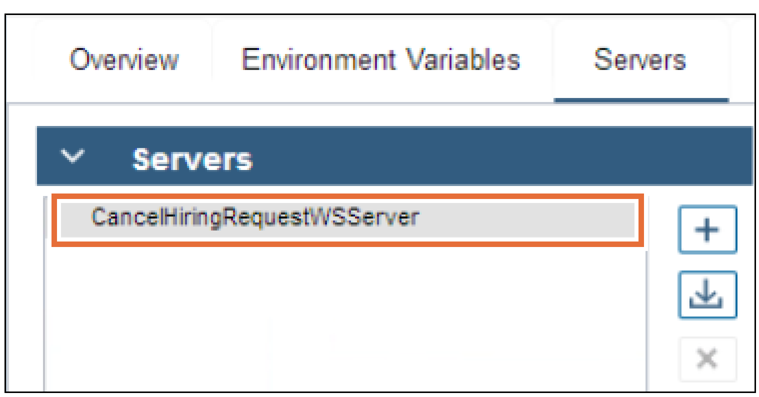

__ j. Save your work.

__ 3. Verify the new server that you just added.

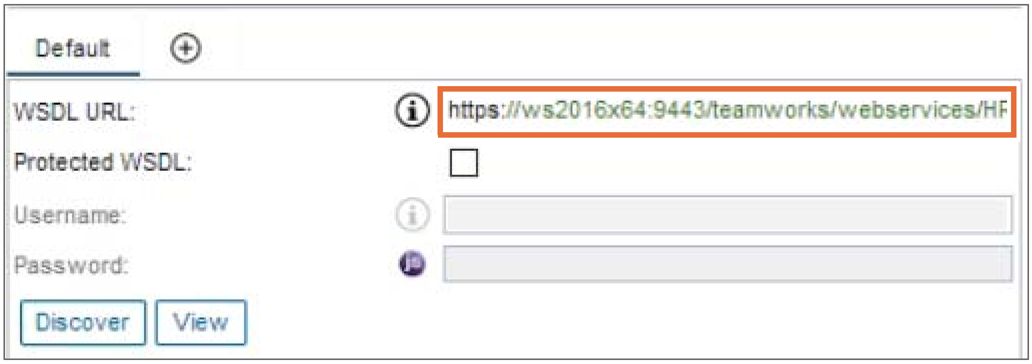
__ a. Click **Process App Settings**.




__ b. Click the **Servers** tab and verify that the new server **CancelHiringRequestWSServer** is listed.



__ c. In the **Details** section, scroll to the bottom and examine the WSDL URL that you used earlier to create the web service.

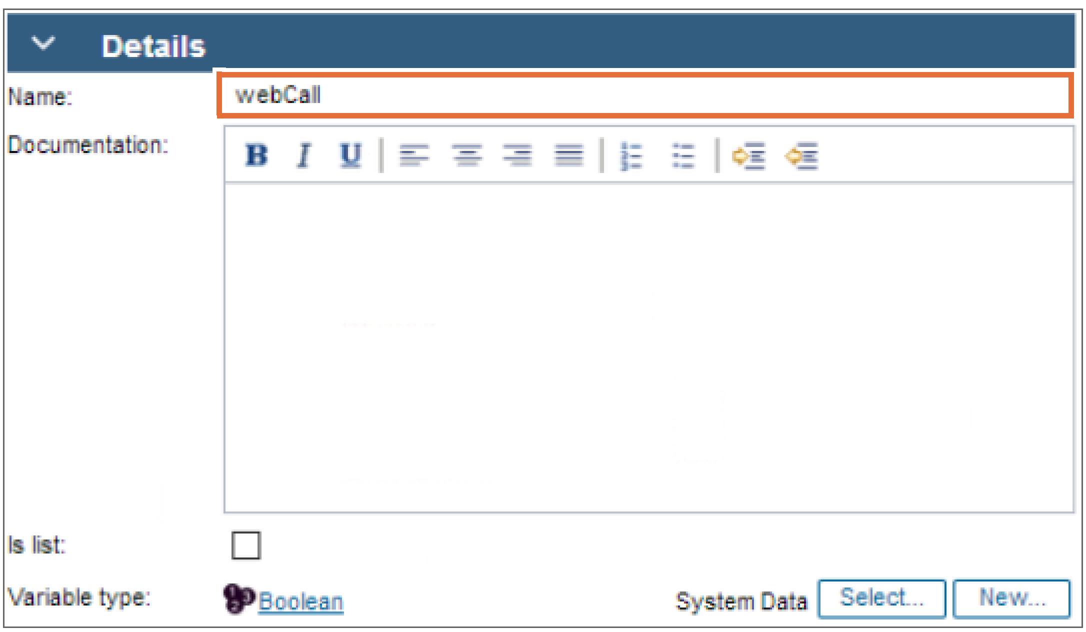



## Part 3: Test the integrations

__ 1. Modify the Cancel Hiring Request Test service to cancel an instance with a UCA or a web service call.

__ a. Open the **User Interface > Cancel Hiring Request Test** client-side human service in the Process Designer.

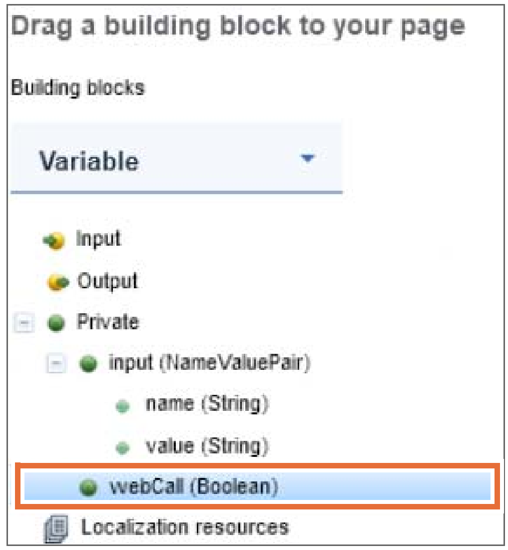
__ b. Add a private variable: `webCall (Boolean)`



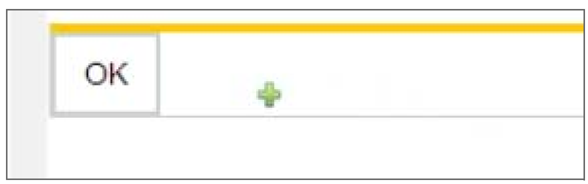

__ c. Click the **Coaches** tab and click the **Test Cancellation** coach.

__ 2. Create a control on the page to allow users to select the type of test to perform.

__ a. Select **Variable** in the **Building blocks** panel and select **Private > webCall**

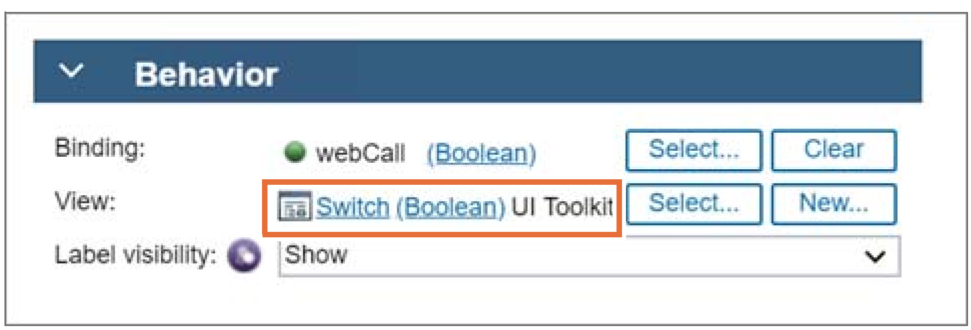



__ b. While holding down the left mouse button, drag the `webCall` variable to the top of the **OK** button. Let go of the mouse button when the top border displays a gold highlight.




__ c. Change the **Web Call** control label to `Type of Test`.

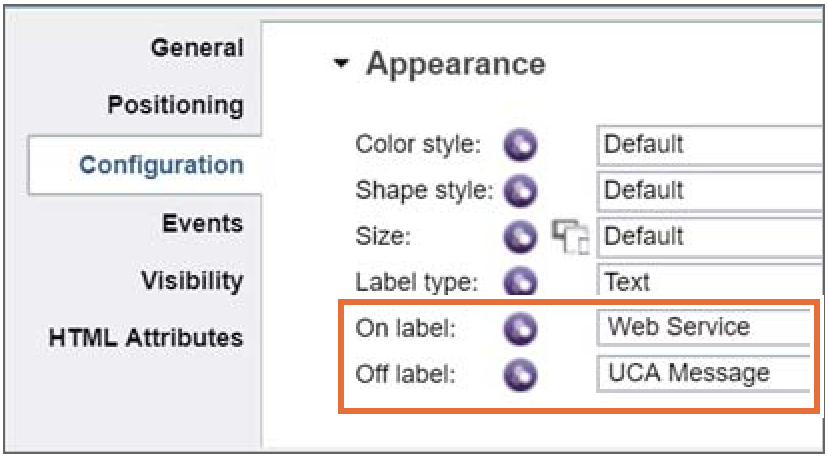
__ d. In the **General > Behavior** section, click **Select** for View and change the type to **Switch**.




__ e. Click the **Configuration** menu and expand **Appearance**.

__ f. Set **On label:** `Web Service`

__ g. Set **Off label:** `UCA Message`




__ h. Save your work.

__ 3. Add the **Cancel Hiring Request WS** service to the service flow.

__ a. Click the **Diagram** tab.

__ b. Drag a **service** from the palette right above the Cancel Hiring Request service step on the canvas.

__ c. Name it: `Cancel Hiring Request WS`

__ d. Under the **Implementation > Behavior** section, ensure that **Call a service** is selected.

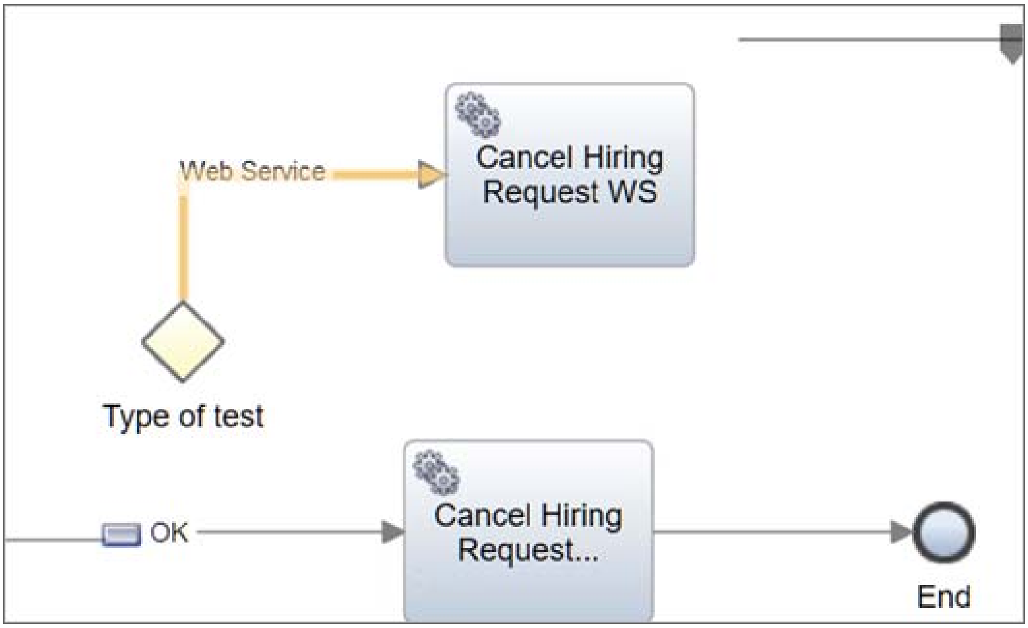
__ e. Click **Select** and under the Integration Service category, select **Call Cancellation Service**.


__ f. Open the **Data Mapping** properties menu, and map the `tw.local.input` variable to the **requestId** input variable.

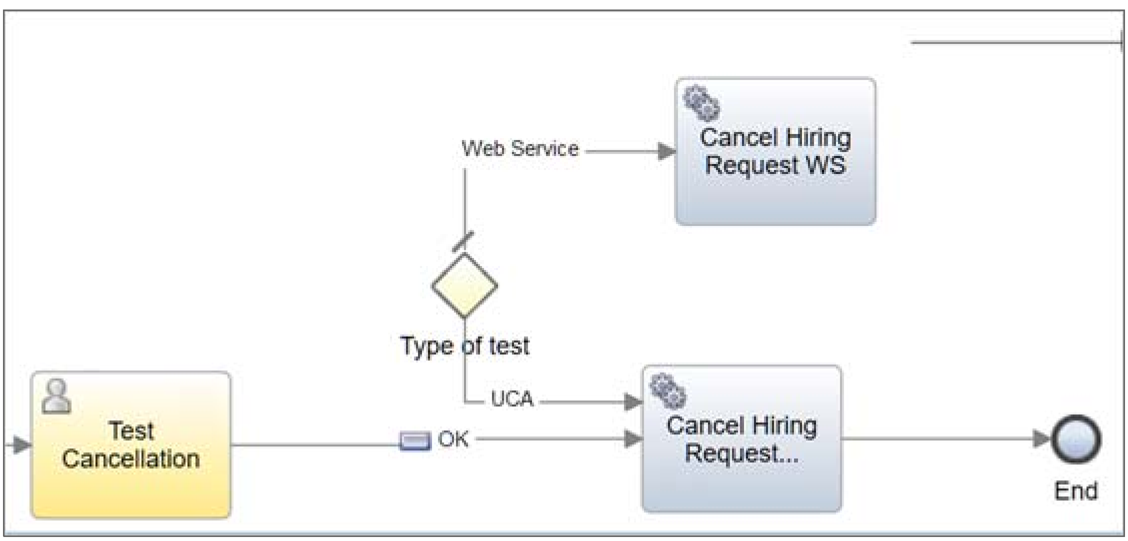
__ 4. Route the flow dependent on which type of test the user selects.

__ a. Add an **Exclusive Gateway** to the right of the Test Cancellation coach, and name it: `Type of test`

__ b. Create a flow from the gateway to **Cancel Hiring Request WS**, and name this flow: `Web Service`. Select the **Name visible** checkbox.



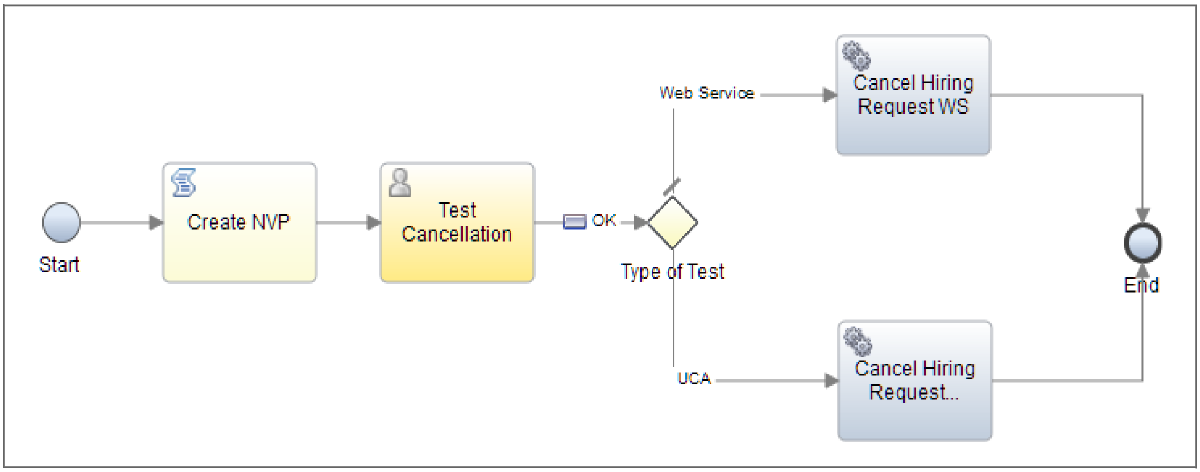

__ c. Create a second flow from the gateway to the **Cancel Hiring Request Trigger** service, and name this flow: `UCA`. Select the **Name visible** checkbox.




__ d. Delete the flow from the **Test Cancellation** coach to the **Cancel Hiring Request Trigger** Service**.**

__ e. Connect the flow from the **Test Cancellation** coach to the **gateway**.

__ f. Connect a flow from **Cancel Hiring Request WS** to the **End** event.

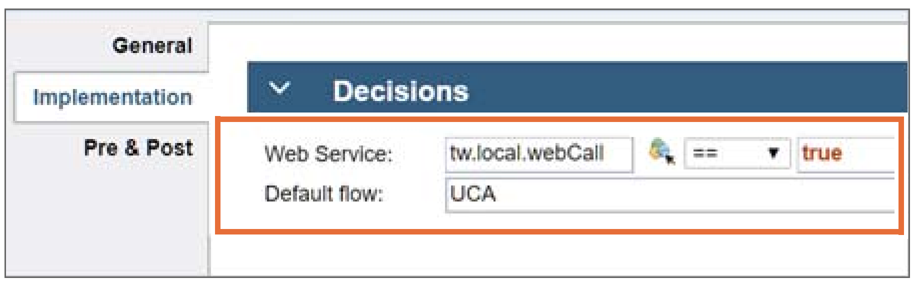



__ 5. Configure the gateway.

__ a. Click the **Type of test** gateway. Click the **Implementation** properties menu.

__ b. Implement the decision gateway by making the **UCA** flow the Default Sequence Flow. This flow connects to the Cancel Hiring Request Trigger service.

__ c. Enter `tw.local.webCall` in the Web Service path, select == from the operands menu, and enter `true` in the last field.



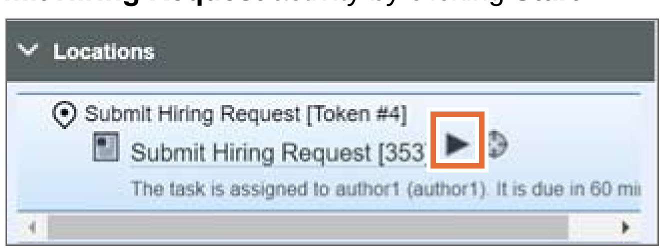

__ d. Save the service.

__ 6. Test the outbound web service call.

__ a. In the Process Designer, open the **Hiring Request Process**.

__ b. Create an instance of the process by clicking **Run**.

__ c. Run the **Submit Hiring Request** activity by clicking **Start**.




__ d. Use the following values for the variables on the form:

- **Position Details > Job level** set to `Manager`
- **Recruiting Details > New Position** set to **false** (not selected)

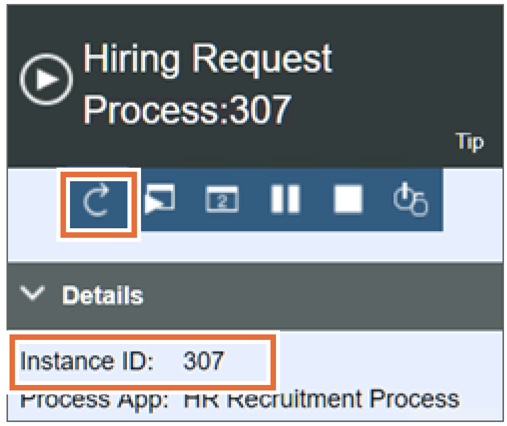
- **Compensation Details > Salary to offer** set to `100000`

__ e. Click **Submit**.

__ f. Close the browser window.

__ g. Refresh the instance details until you see that the **Override Hire Request** task appears in the Locations section of the information window.

__ h. Note the **Instance ID** in the Details section.

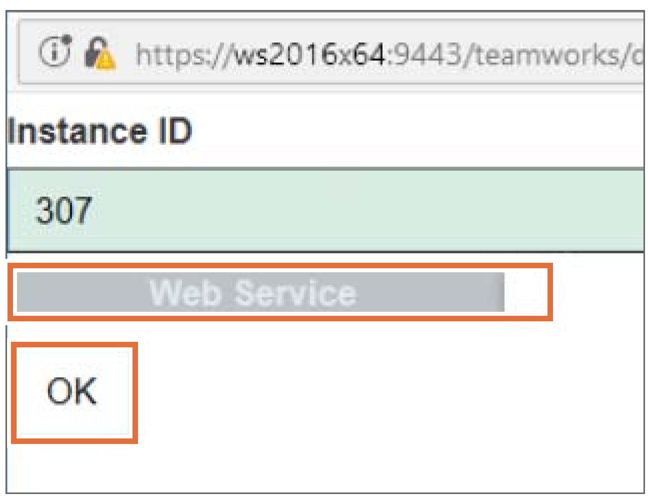



__ i. Open the **Cancel Hiring Request Test** client-side human service.

__ j. Click **Run**.

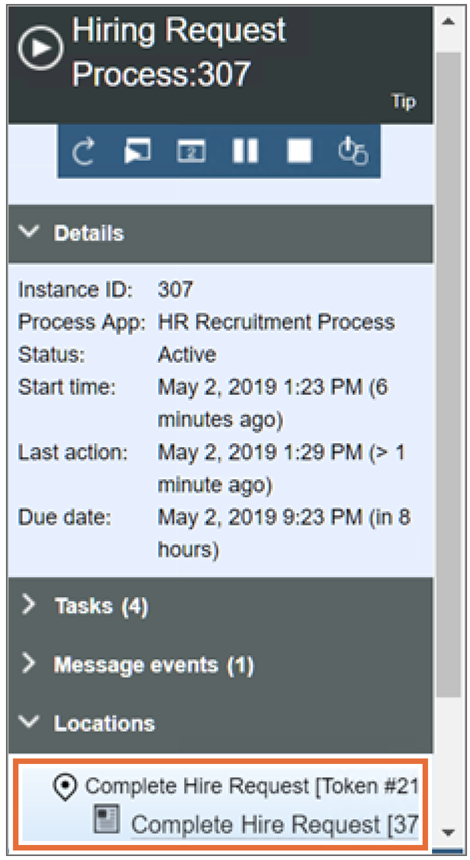
__ k. Enter the Instance ID and then click the **Web Service** option.

__ l. Click **OK**.




__ m. Close the service window and return to the **Hiring Request Process**.

__ n. Refresh the instance details window. Verify that the message event consumes all the tokens inside the linked process and moves the process down the message event flow. The instance moves from the **Override Hire Requests** task and is at the **Complete Hire Request** step. Feel free to complete the task.




You have completed this exercise.

In the first part of the exercise, you created an inbound integration by using an existing generic system service to implement the service. You viewed the WSDL in the browser. You then created an outbound web service to test the inbound web service integration. You configured the outbound web service, discovered the endpoint methods, and selected the method to call. Finally, you modified an existing test harness to call both the internal UCA and the outbound web service to test the ability to cancel an in-flight process.

## End of exercise

# Exercise 14. Handling content events in a process

## Estimated time

01:30

## Overview

In this exercise, you learn how to implement a content event in a process.

## Objectives

After completing this exercise, you should be able to:

- Use the CMIS capabilities of IBM Business Automation Workflow
- Implement a content event in a process
- Use the BPM document store to add a document to a process
- Use the Target Object Store to share documents between a BPM solution and a Case solution
- Build a simple HR Case solution and create an activity to start the Hiring Request Process in the HR BPM solution
- Demonstrate integration between the Case solution and the BPM solution

## Introduction

In this exercise, you create a content start event that starts the Hiring Request Process when a document is received. You also create a startable service that provides the capability to add a document to the BPM document store.

After building the solution to use the BPM document store, you then configure the solution to use the Target Object Store and demonstrate sharing documents between a Case solution and a BPM solution. You then build a simple Case solution and add the Hiring Request Process as an activity. You also demonstrate the integration between a Case solution and the BPM solution by adding a document to the Case and verifying that the Hiring Request Process was started.

## Requirements

Successful completion of the prior exercise is required. If you did not complete the prior exercise, you can import the solution for the prior exercise to perform this exercise.

## Index

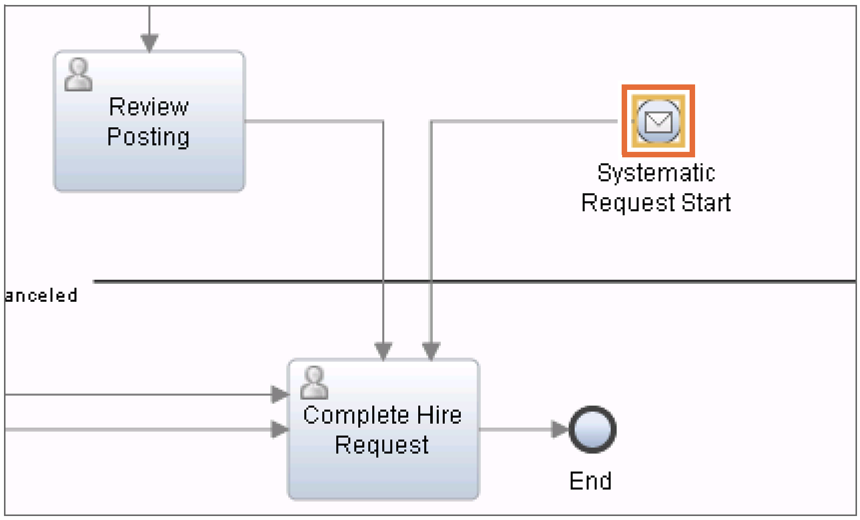
# Exercise instructions

## *Part 1: Implement a document start event on a process by using the BPM document store*

__ 1. Change the name of the start event on the process.

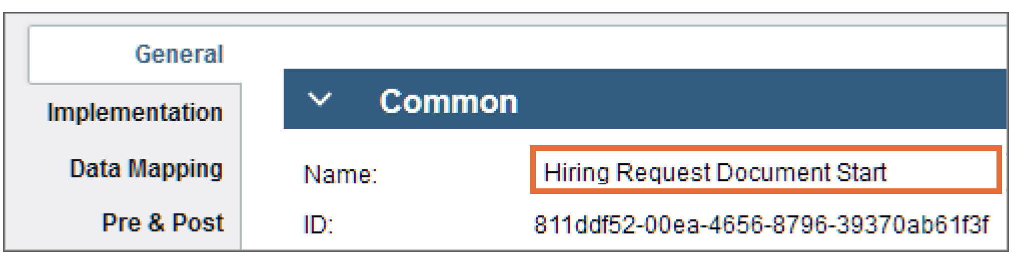
__ a. Open the **HR Recruitment Process** process application.

__ b. Open the **Hiring Request Process**. Click the **Systematic Request Start** event.




View the process on the **Definition** tab. The process has a Systematic Request Start that creates an instance of the process on a periodic basis. Management wants an employee application document that is uploaded to the repository to automatically trigger an instance of the process.

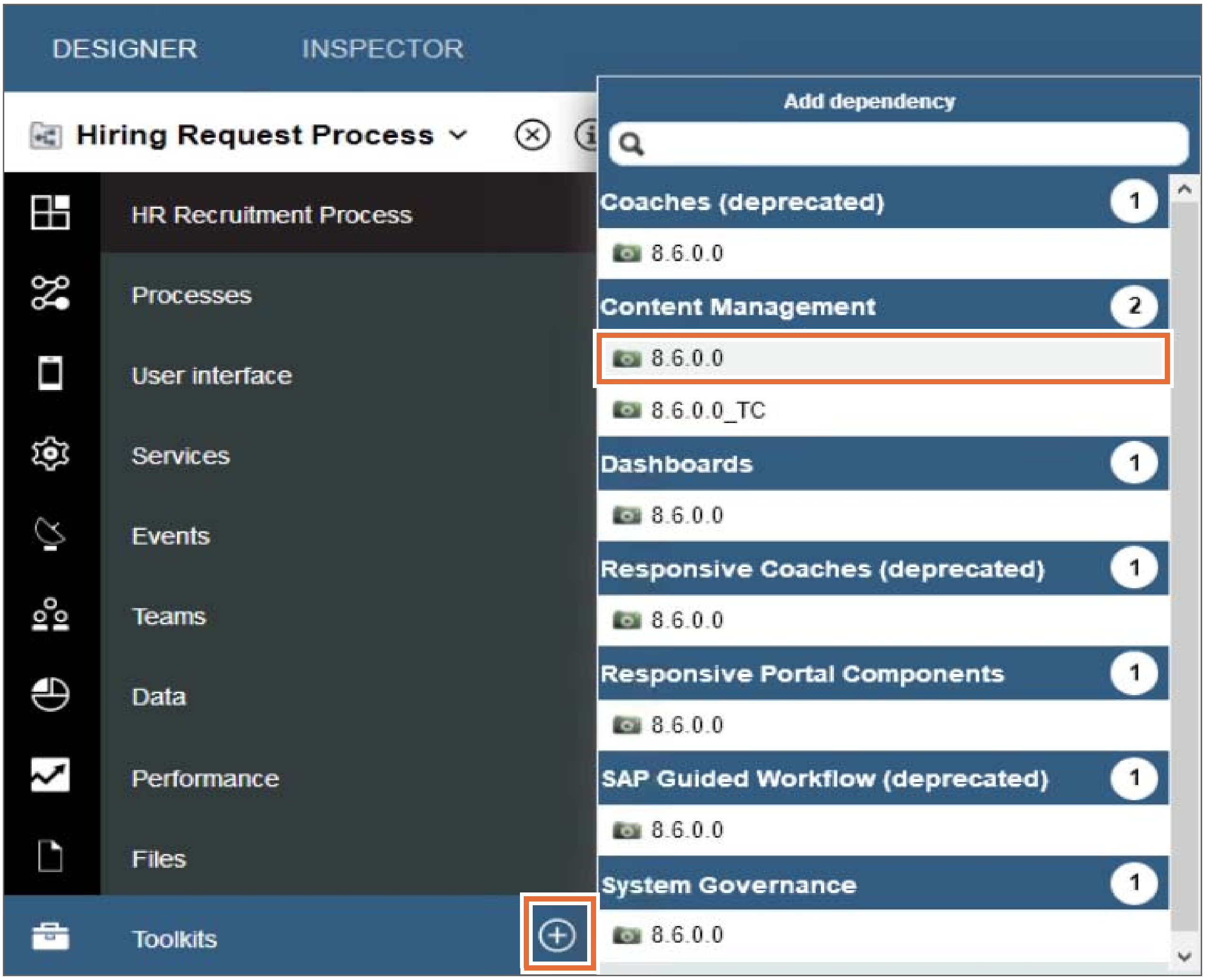
__ c. From the **General** properties menu, change the name of the event to `Hiring Request Document Start`.




__ d. Save your work.

__ 2. Create a dependency to the Content Management toolkit snapshot.

__ a. Open the library on the left and click the **plus sign (+)** next to the Toolkits category, then click the **8.6.0.0** snapshot under Content Management.

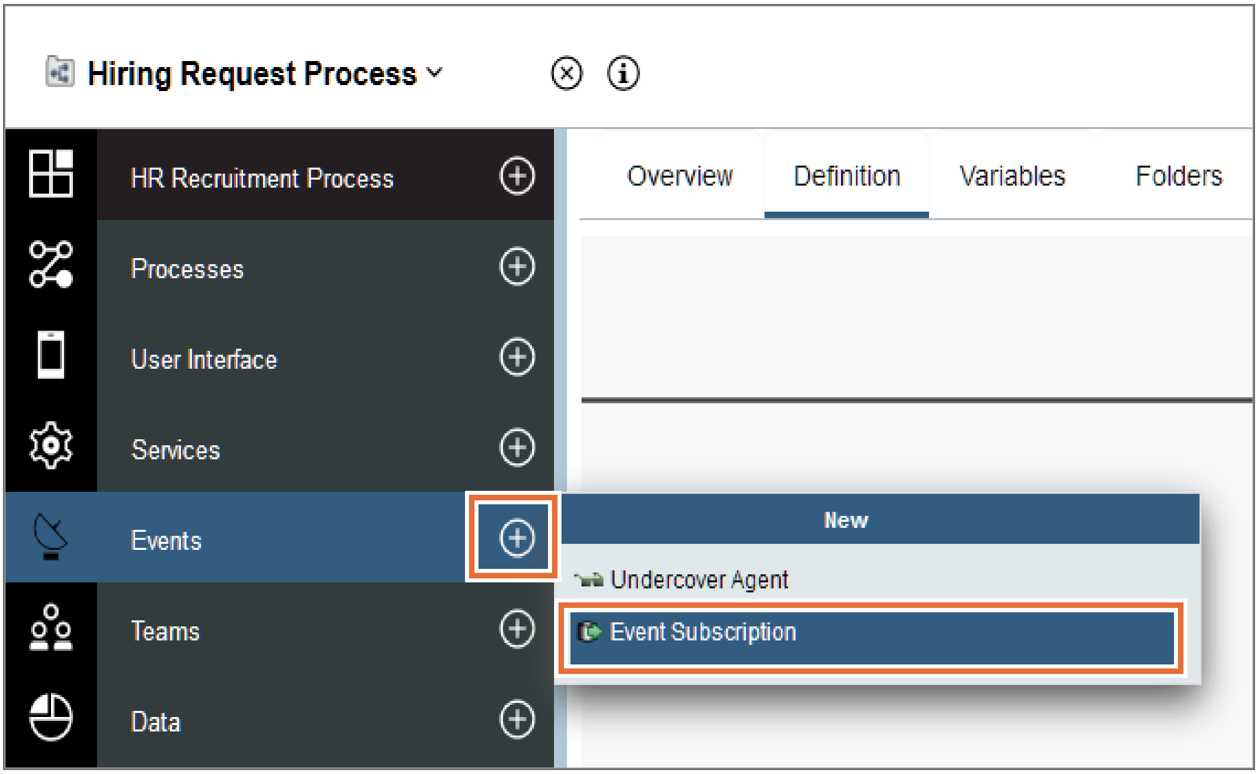



**Information**

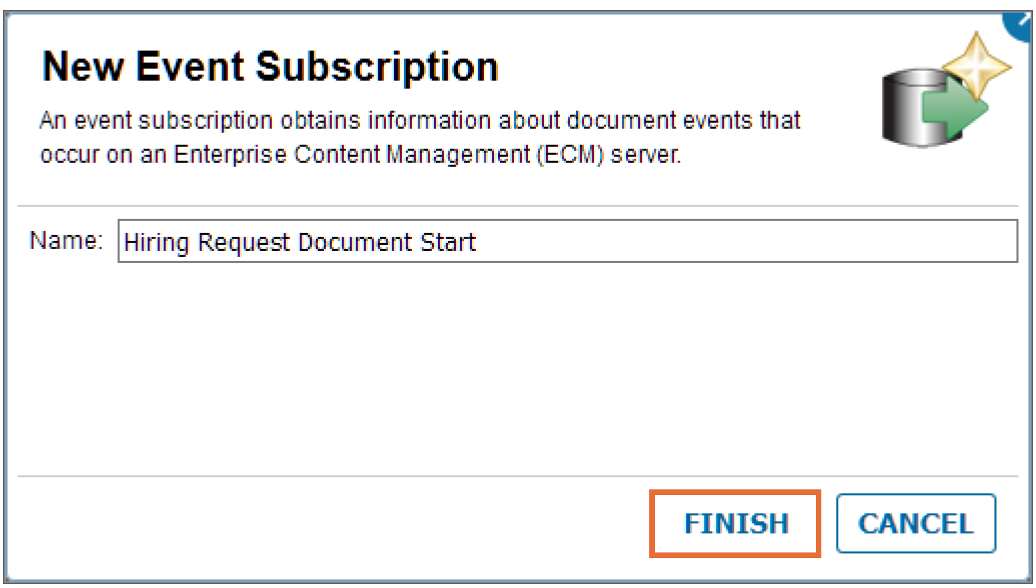
In IBM Business Automation Workflow, all artifacts are generally supported in the traditional WebSphere runtime environment. However, not all artifacts are supported in the container runtime environment, such as most deprecated artifacts (like BPDs, integration services and some other services). You can use either version 8.6.0.0 or version 8.6.0.0_TC of the content management toolkit in a traditional environment. However, you must use version 8.6.0.0_TC of the toolkit when implementing in a container runtime environment.

__ 3. Create a new event subscription.

__ a. In the library, click the **plus sign (+)** next to Events and then click **Event Subscription**.



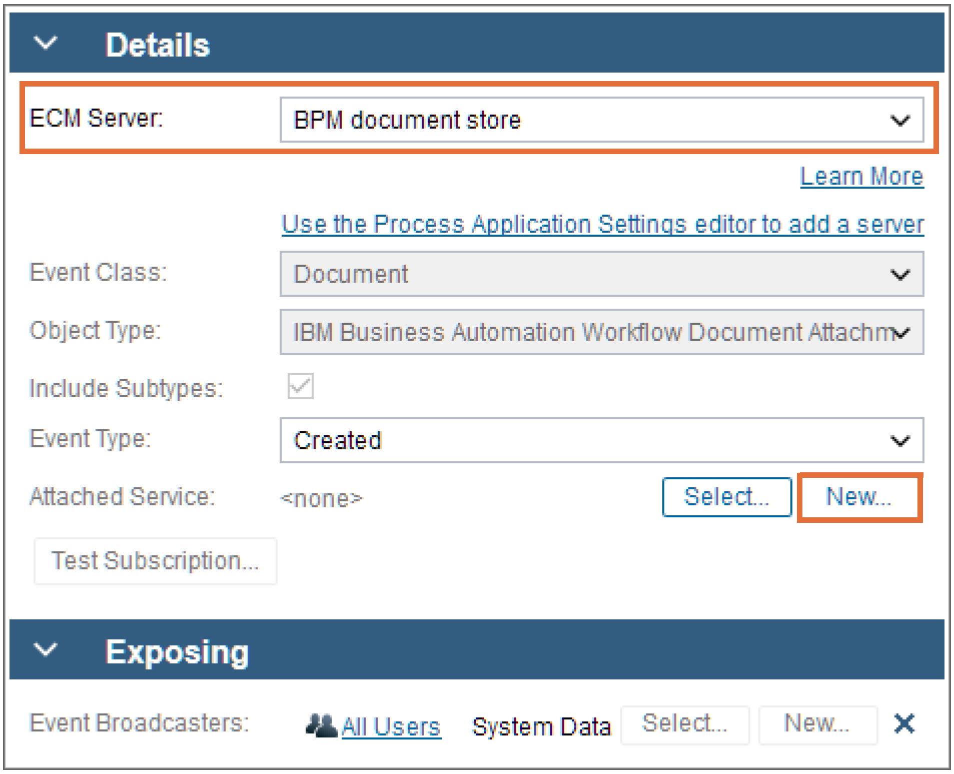

__ b. Name the event subscription `Hiring Request Document Start` and click **FINISH**.




__ c. The event subscription opens in Process Designer. On the right, in the **Details** section, select **BPM document store** for the **ECM Server** setting. Leave the rest of the settings as the default.

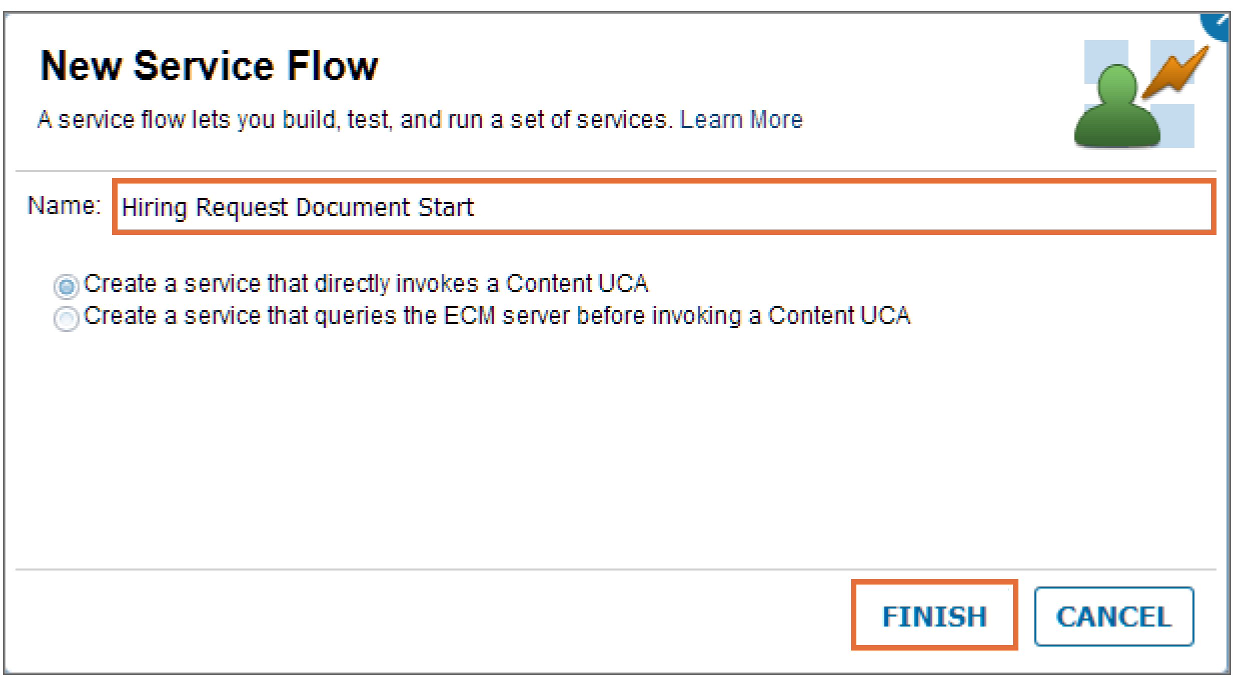
__ 4. Create and attach the Service Flow to invoke the Undercover agent.

__ a. At the bottom of the Details section, click the **New** button next to Attached Service.

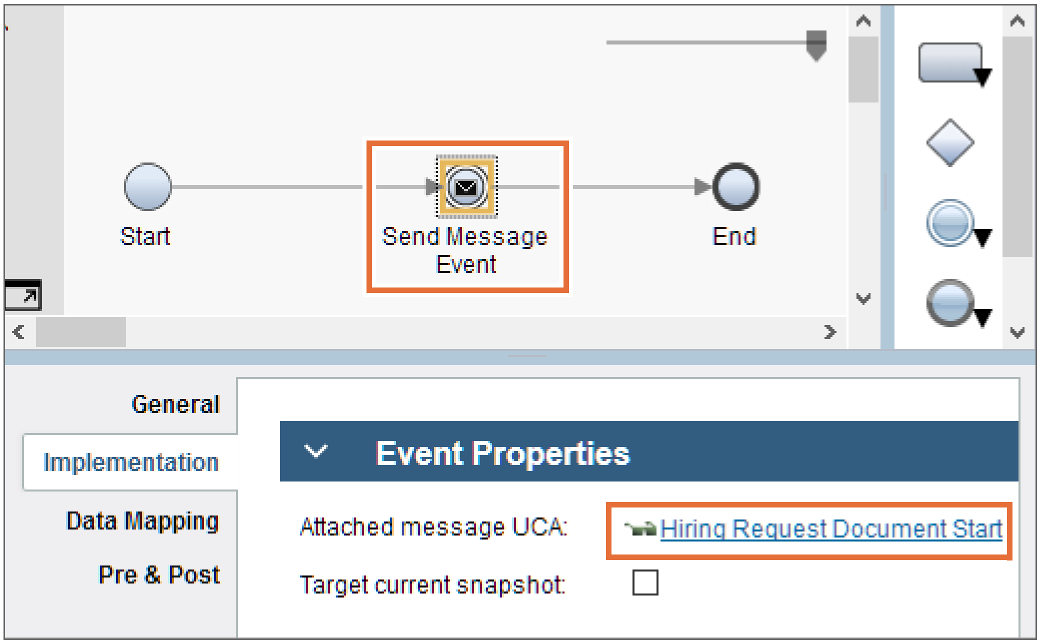



__ b. Name the New Service Flow as `Hiring Request Document Start`.

__ c. Leave the options as default and click **FINISH**.

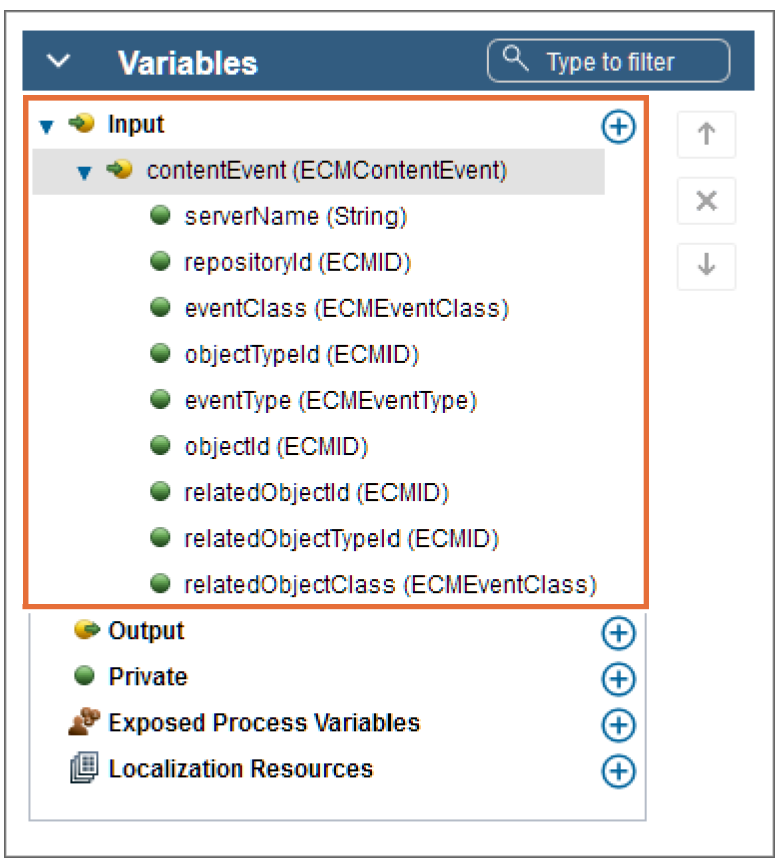


When you chose the first option, the system creates a fully implemented service and opens it in IBM Process Designer. The service consists of a Start event, an End event, and an intermediate Send Message Event that invokes the new content undercover agent (UCA).




__ d. Click the **Variables** tab.

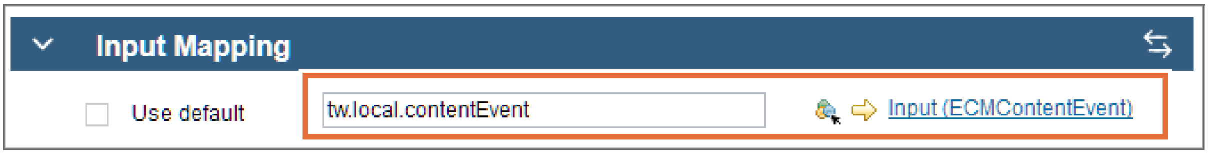
The system created an input variable `contentEvent(ECMContentEvent)`. The system requires this type of variable when the content start event is triggered. When you trigger the UCA, it creates an instance of a process.



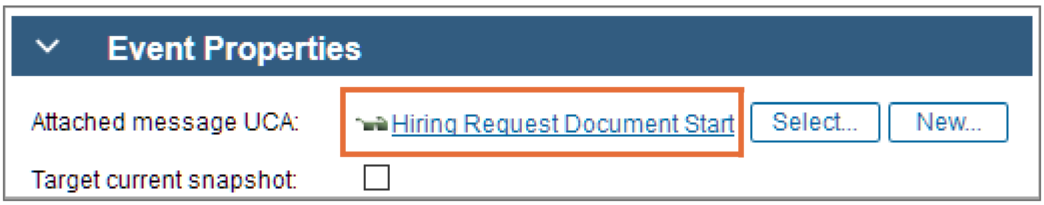
__ e. Return to the **Diagram** tab.

__ f. Click the **Send Message Event** on the canvas.

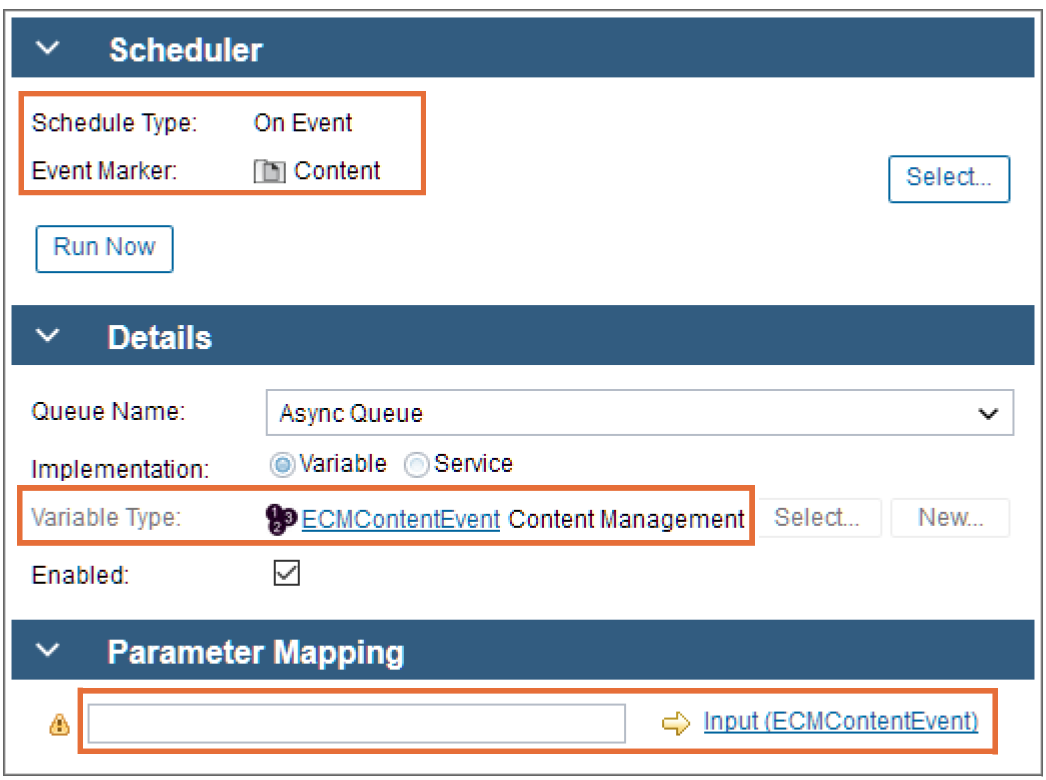
Open the **Data Mapping** properties menu. The `tw.local.contentEvent` variable is already mapped for you.




__ g. Open the **Implementation** properties menu and click the **Hiring Request Document Start** link for the undercover agent to open the UCA.




The Undercover Agent is already configured for you. Under the Scheduler section, you see that the Event Marker is **Content**, which makes it available to implement a Content Start Event. Under the Details section, you see the same business object that is used in the service flow. The Undercover Agent requires an input variable of type `ECMContentEvent`, and this variable is sent to the process that triggers the Undercover Agent.




The warning symbol is present to indicate that no mapping is implemented. With no mapping implemented, the default mapping takes place.

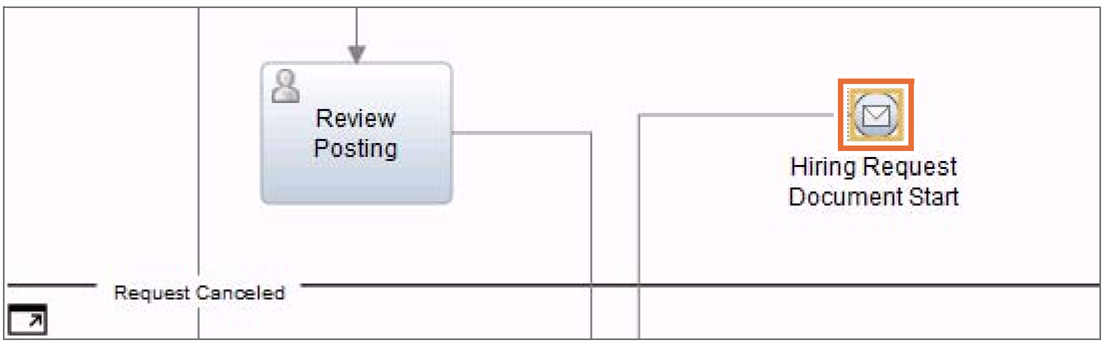
__ h. Always change the event message to better identify UCA events that occur in the logs. You should always change the event message identifiers from the default hash to a relevant event label when creating undercover agents. At the bottom, in the Event section, change the Event Message for the UCA to `HiringRequestDocumentStart`.


__ i. Save your work.

__ 5. Configure the start event

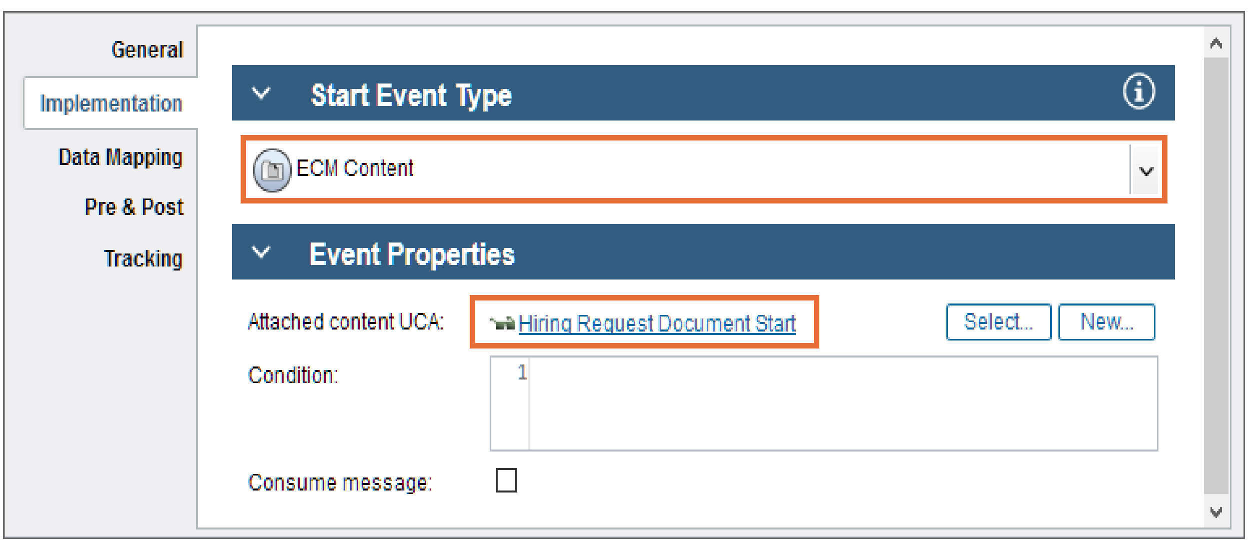
__ a. Return to the **Hiring Request Process** and open the **Definition** tab.

__ b. Select the **Hiring Request Document Start** event.

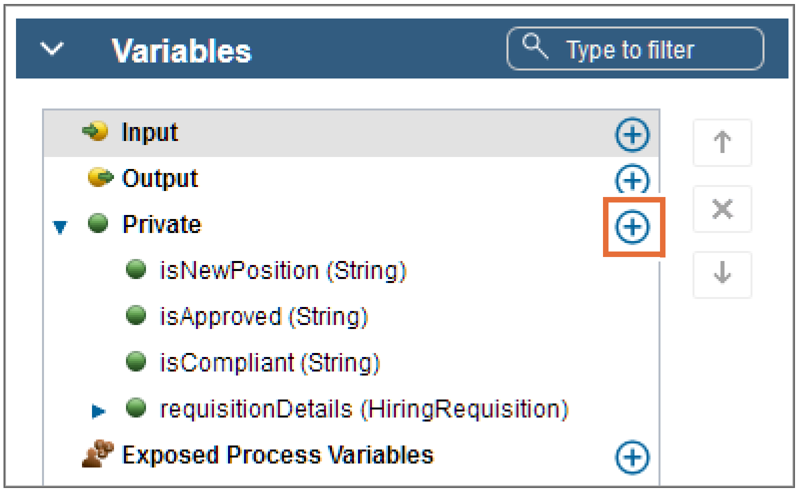



__ c. Click the **Implementation** properties menu. Change the Start Event Type to **ECM Content**.

__ d. Select the **Hiring Request Document Start** for the attached undercover agent.

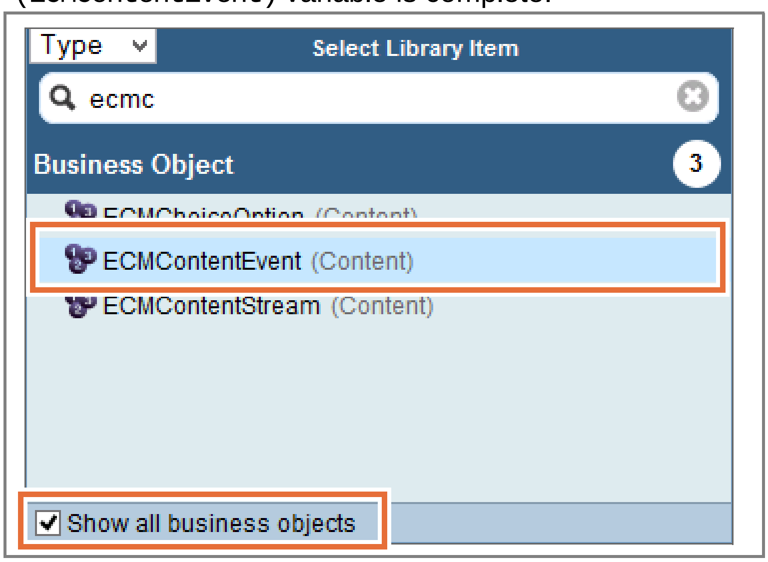


__ e. Click the **Variables** tab. Click the **plus sign (+)** to create a private variable.



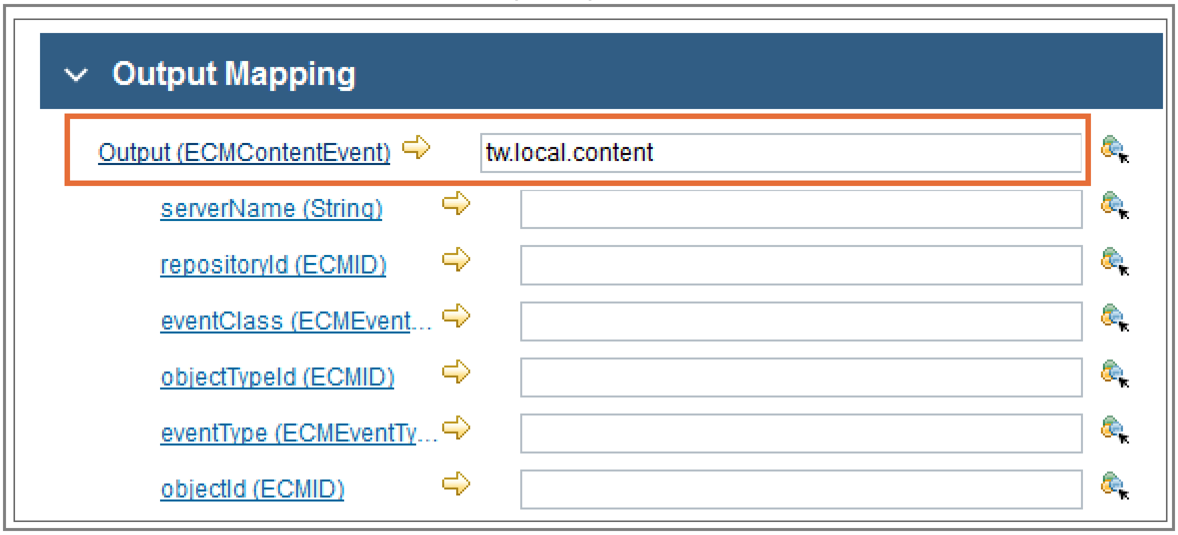

__ f. Name the variable `content`. Next to the Variable type option, click **Select**.

__ g. Select the **Show all business objects** checkbox, then select **ECMContentEvent.** The `content (ECMContentEvent)` variable is complete.




__ h. Return to the **Definition** tab.

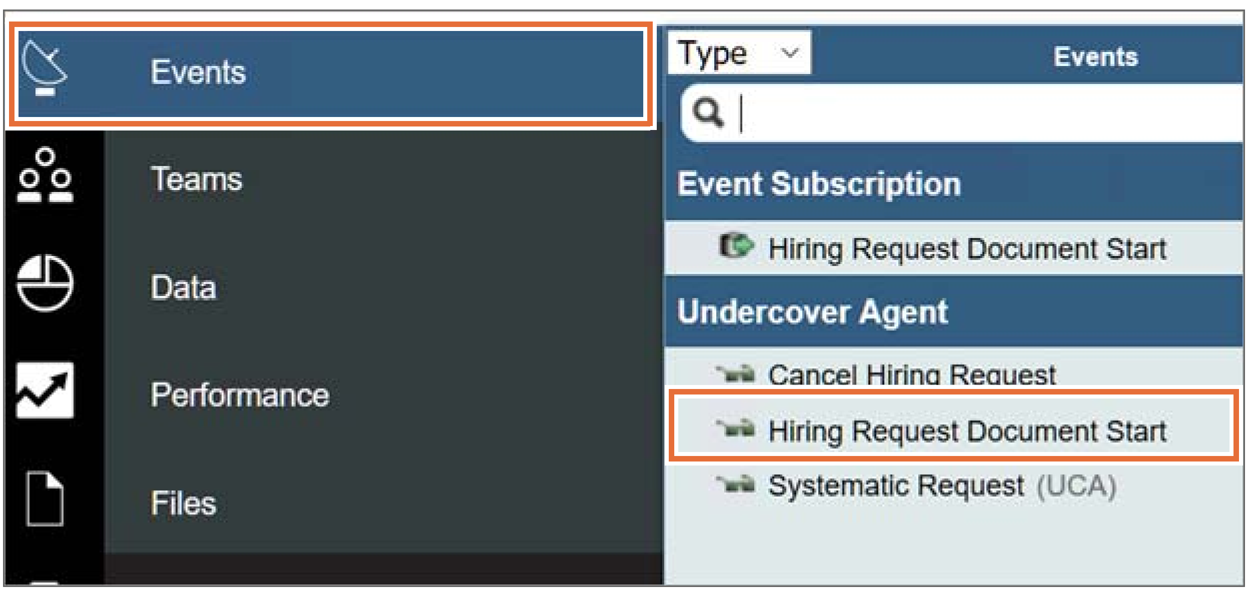
__ i. Open the **Hiring Request Document Start** event and go to the **Data Mapping** properties menu. Map the output of the start event to the `tw.local.content` private variable. There is no need to map the parameters.



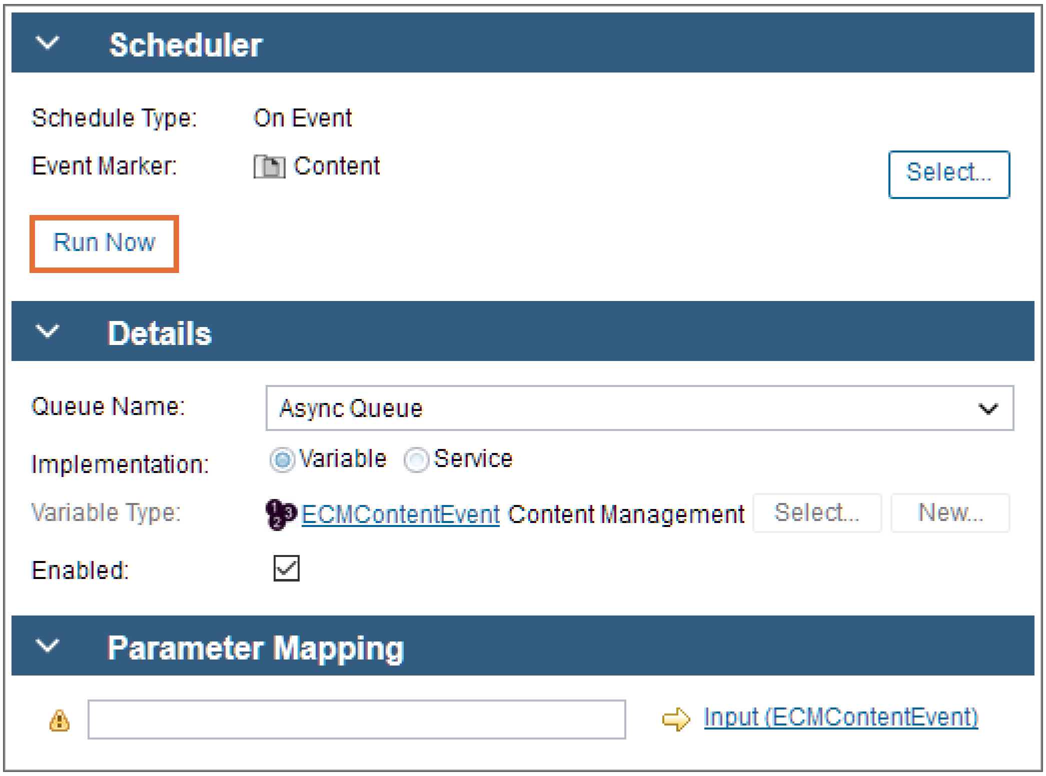

__ j. Save your work.

__ 6. Test your solution by running the Undercover Agent.

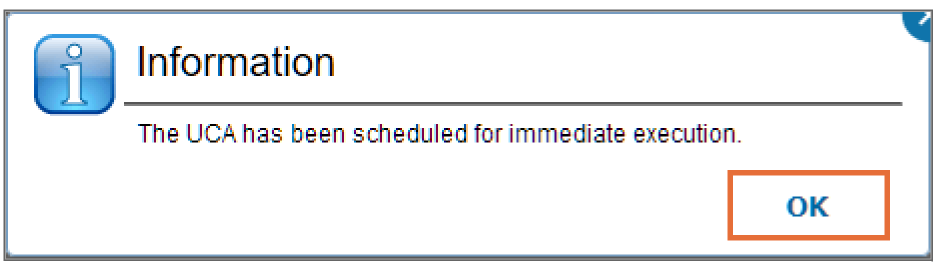
__ a. Return to the Undercover Agent by clicking **Events > Hiring Request Document Start** UCA in the Process Library.



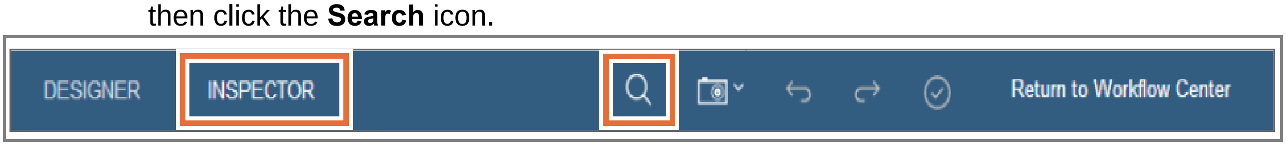
__ b. Click **Run Now**.




__ c. Click the **OK** button on the Information window when it confirms that the UCA is scheduled for immediate execution.



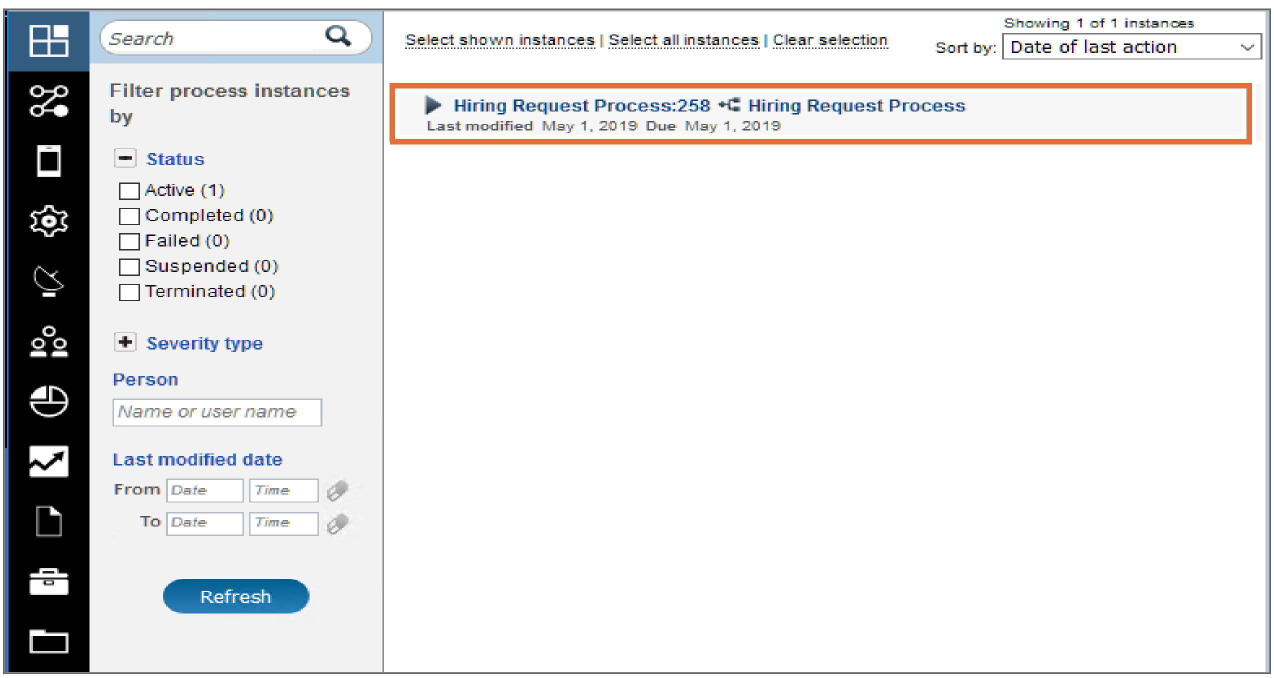

__ d. Verify the creation of a Hiring Request Process instance by clicking the **Inspector** tab, then click the **Search** icon.




Clicking the Search icon starts the Process Inspector.

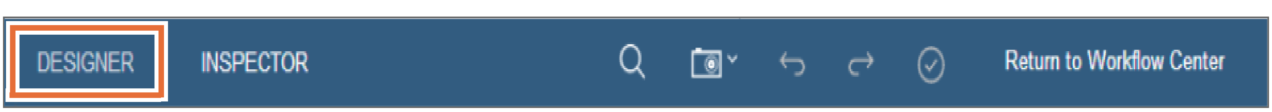
__ e. If the instance you created is not shown, click the **Search** button.


You created an instance of the process, but you didn't upload a document to create an instance of the process. You create the client-side human service to upload your document next. Depending on the number of process instances, the screen capture might not match.

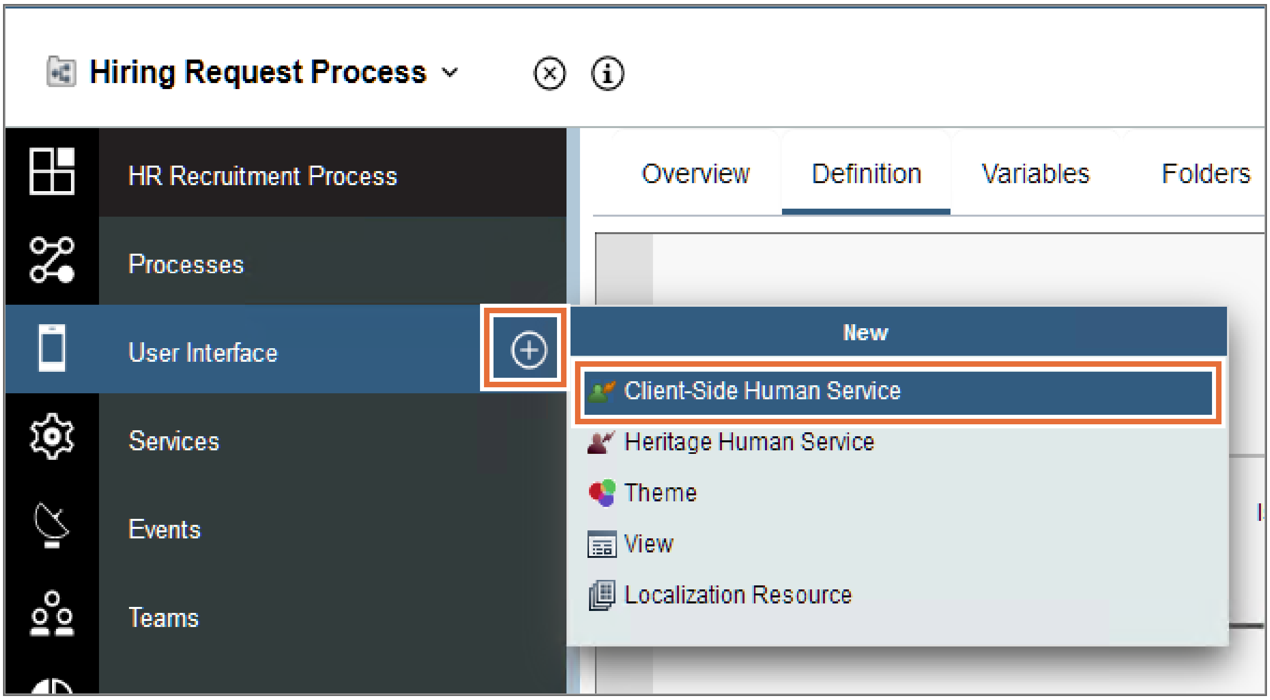



__ 7. Create a client-side human service to trigger the UCA

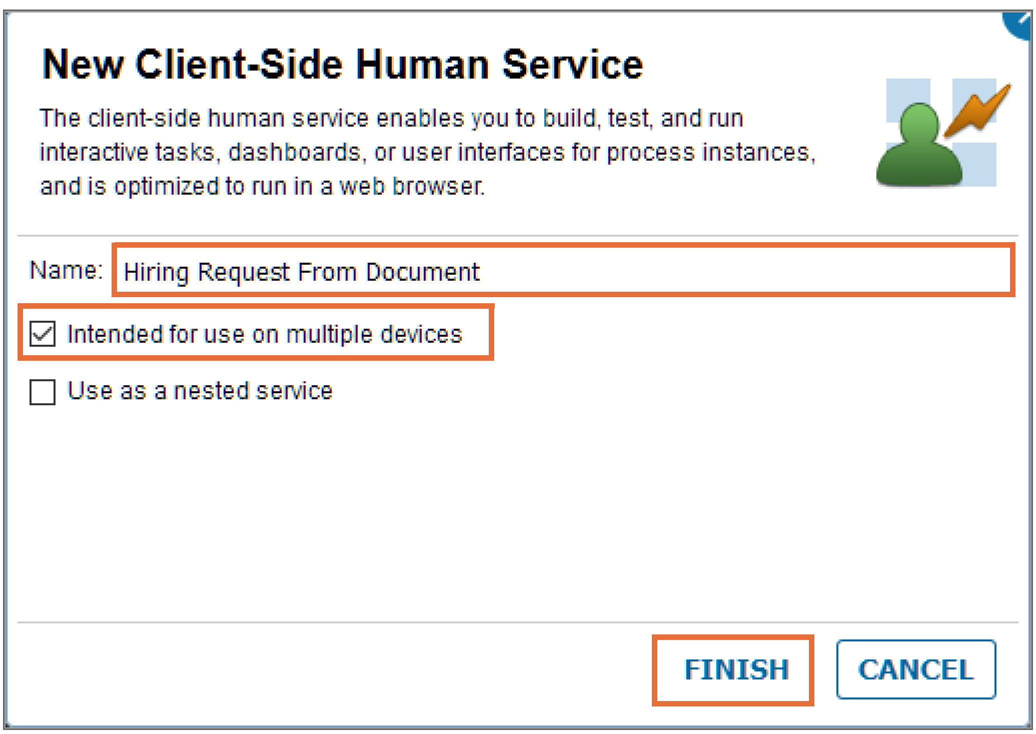
__ a. Switch back to the **Designer** tab.



__ b. In the IBM Process Designer library, click the **plus sign (+)** next to User Interface, and then click **Client-Side Human Service**.

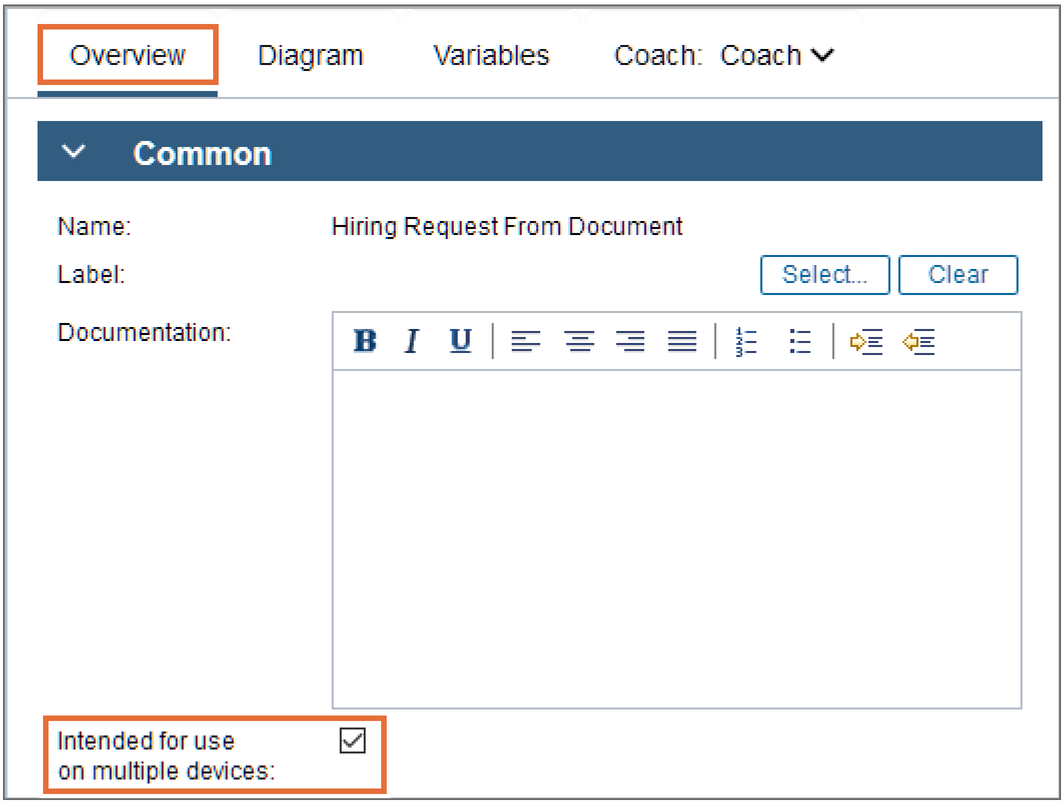



__ c. Name the service `Hiring Request From Document`. Select the **Intended for use on multiple devices** checkbox and click **FINISH**.



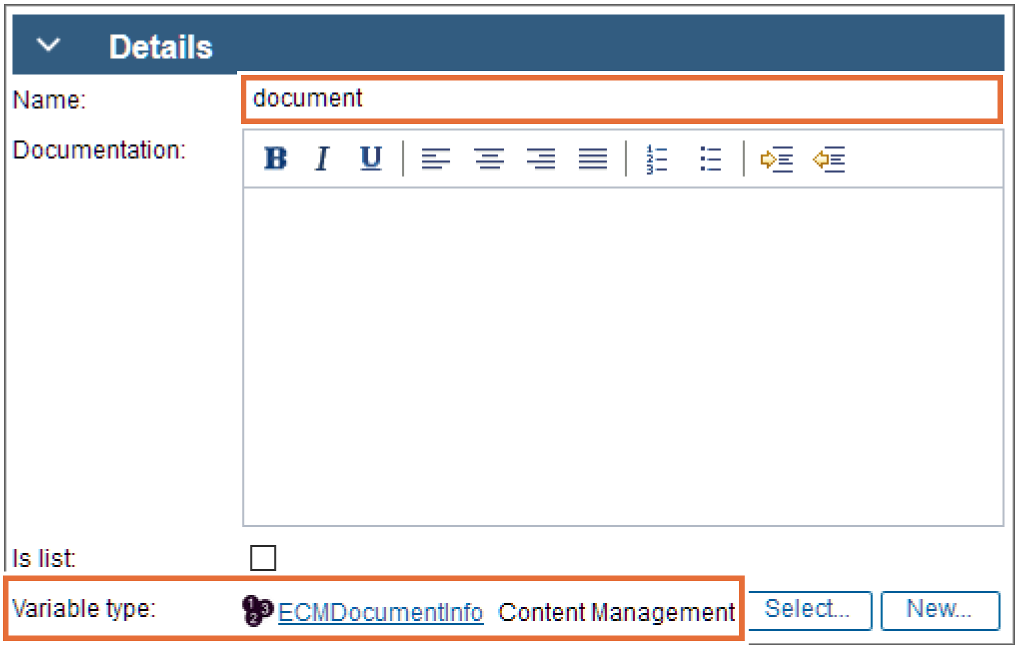
**Information**

If you forget to select the checkbox during the initial creation, or if you decide later to change to a responsive coach, you can change the service. You can mark any Client-side Human Service as a responsive coach service in the **Overview** tab of the service. Select the **Intended for use on multiple devices** checkbox in the Common section. By selecting **Intended for use on multiple devices** checkbox, you are enabling the responsive capabilities on the coach.



__ d. Click the **Variables** tab.

__ e. Create a private variable and name it `document (ECMDocumentInfo)`.

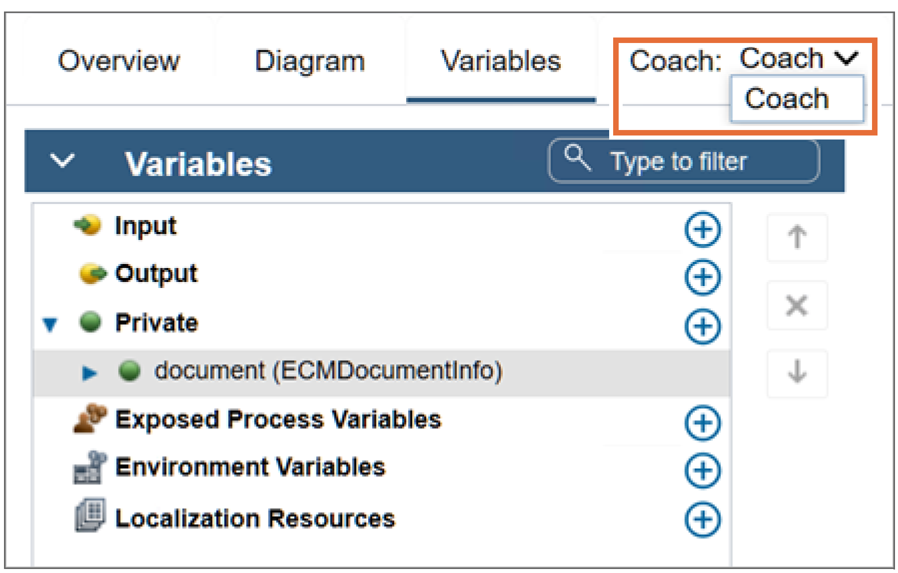



Be sure to choose the `ECMDocumentInfo` variable type and not the `ECMDocument` variable type.

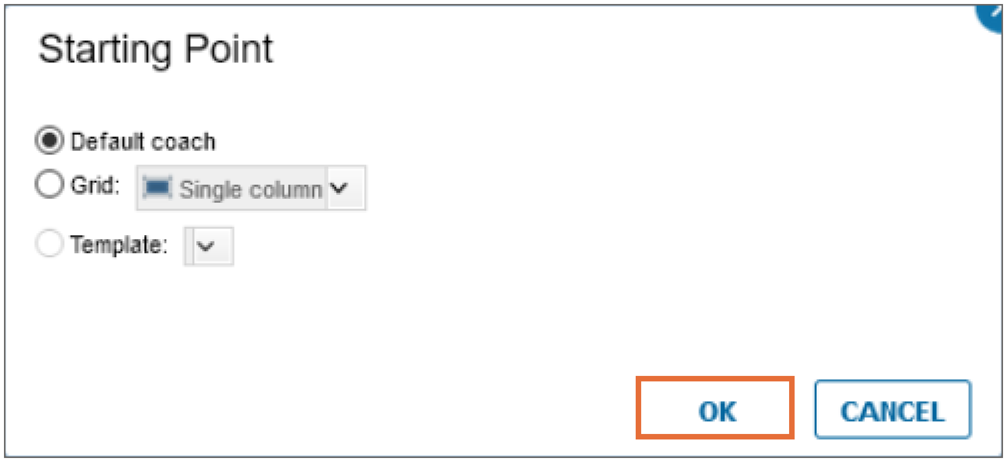
__ f. Save your work.

__ 8. Create the coach

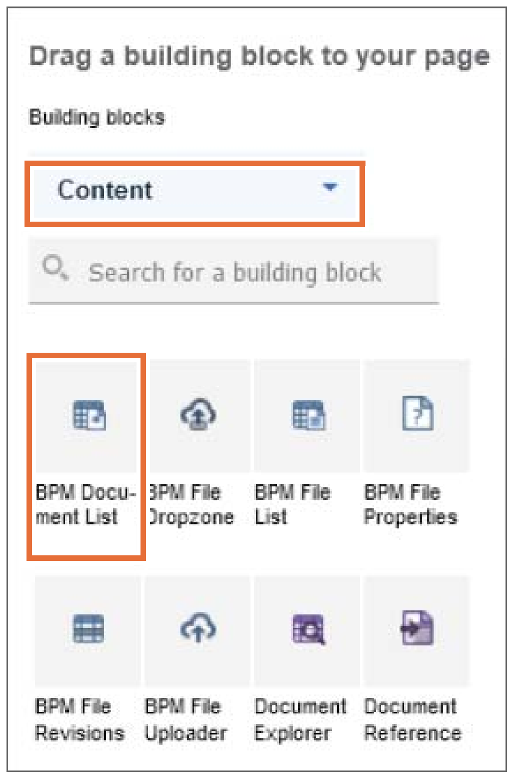
__ a. Click the down arrow next to Coach and select **Coach**.



__ b. In the **Starting Point** window, ensure that the **Default coach** is selected, and click **OK**.



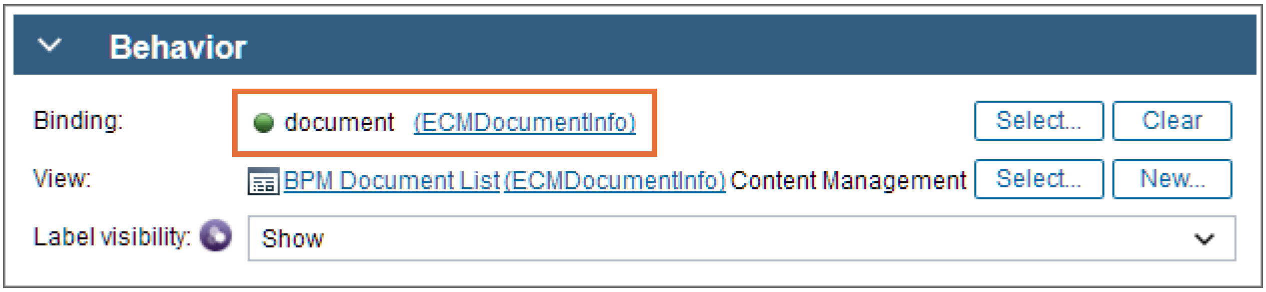

__ c. Select **Content** in the **Building blocks** panel and select **BPM Document List**.




When you added the Content Management Toolkit as a dependency, the palette was updated to include controls from the Content Management Toolkit.

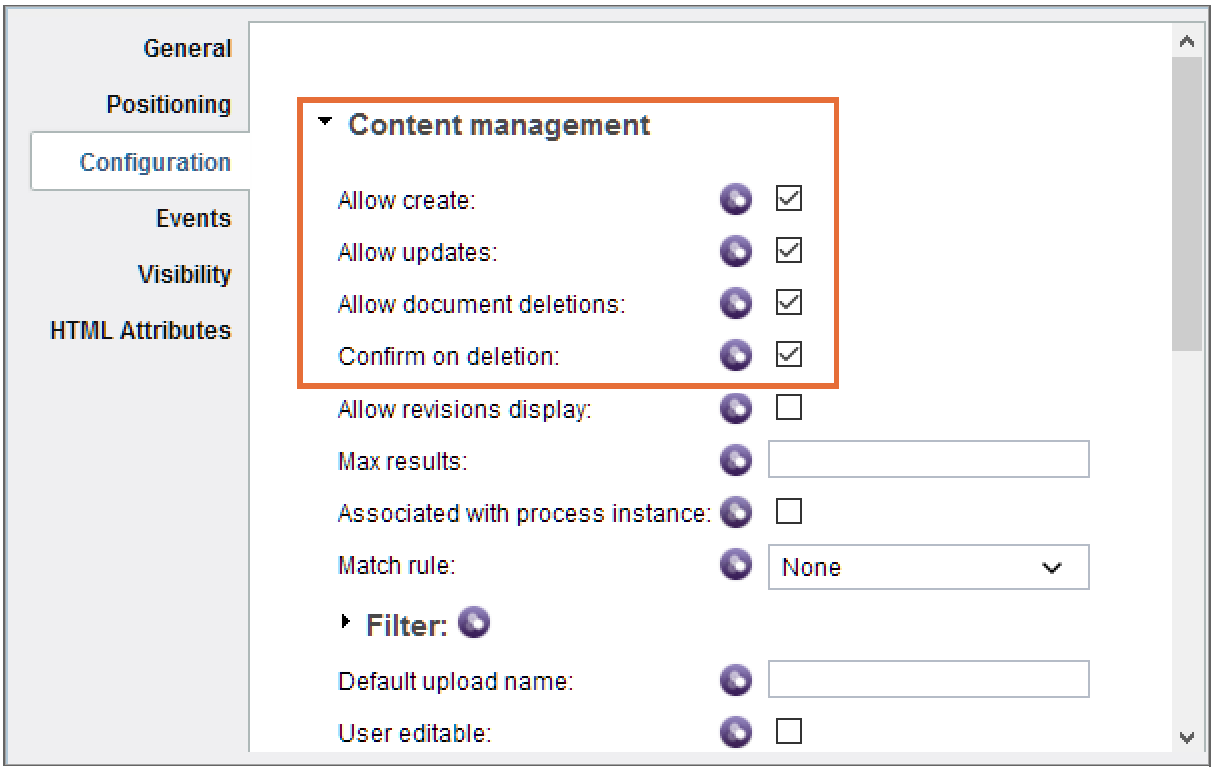
__ d. While holding down the left mouse button, drag the control above the **OK** button. Let go of the mouse button when the top border displays a gold highlight.

__ e. In the **General > Behavior** properties section, verify that the binding is set to **document (ECMDocumentInfo)**.




__ f. Click the **Configuration** properties menu.

__ g. Expand the Content management section and Select the **Allow create**, **Allow updates**, **Allow document deletions,** and **Confirm on deletion** check boxes.



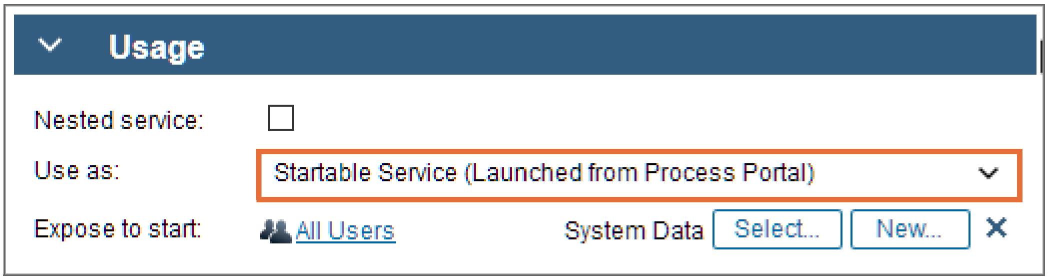

__ h. Save your work.

__ 9. Configure the coach as a Startable Service.

Since the **Hiring Request From Document** client-side human service is not part of the process, there is no need to link it to an activity in the process. In order to allow Process Portal users to start the service, you configure it as a Startable Service.

__ a. Select the **Overview** tab for the **Hiring Request From Document** client-side human service.

__ b. Under the Usage section, select **Startable Service (Launched from Process Portal)** for how it is to be used.




**All Users** is automatically added as the **Expose to start** selection. This allows all users of the Process Portal to run the Startable Service.

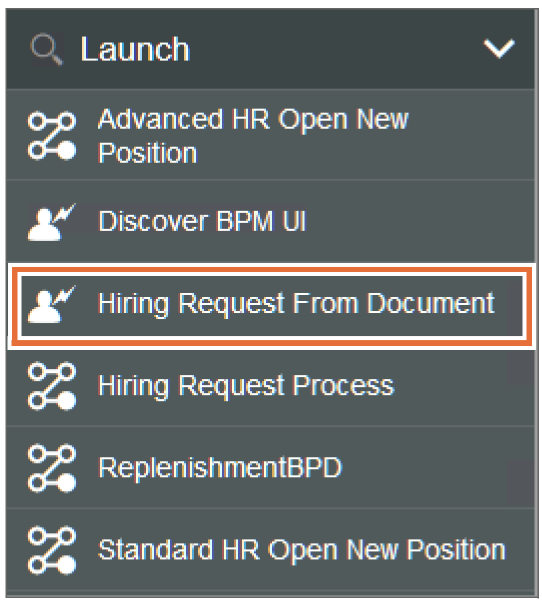
__ c. Save your work.

__ 10. Upload a document by using the coach.

__ a. Open Process Portal in another tab. The Hiring Request From Document is displayed under the Launch menu.

If asked for credentials, enter the following: **Username**: `author1` **Password**: `passw0rd`

__ b. Click the **Hiring Request From Document** startable service in the Launch menu.



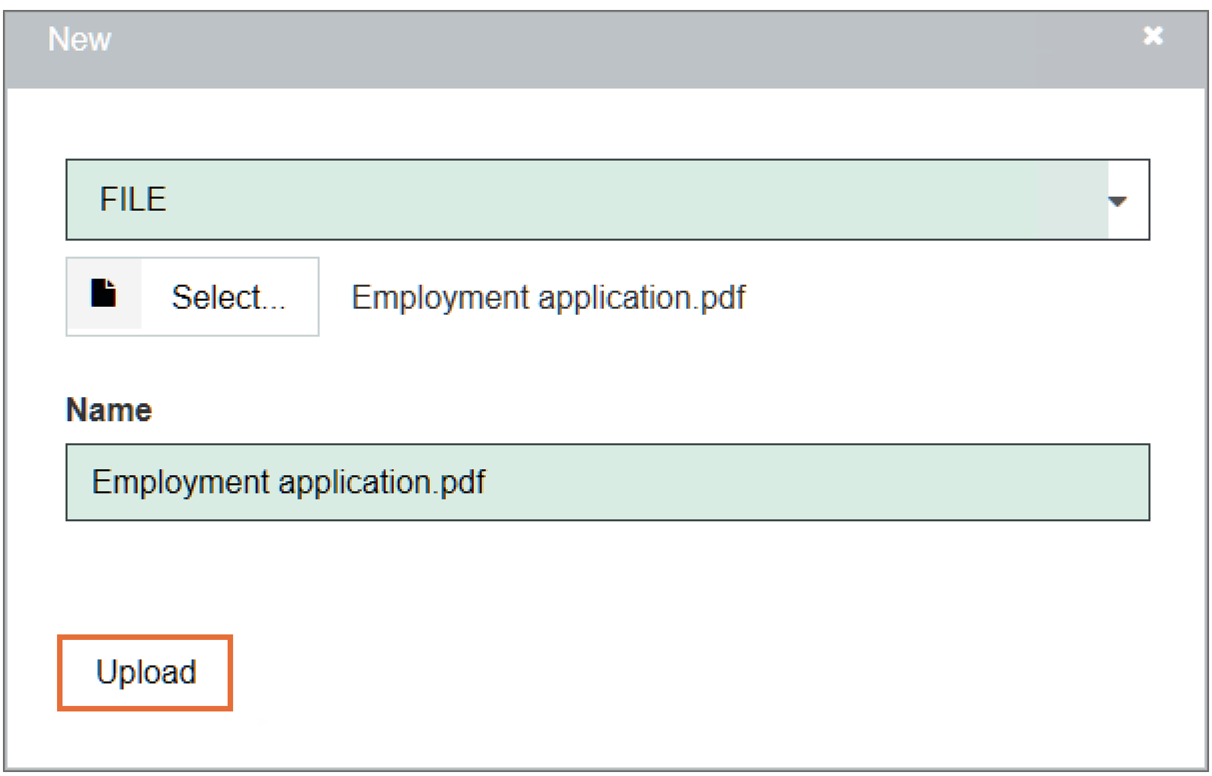
__ c. In the Hiring Request From Document coach, click **Add**.


__ d. In the dialog box, click **Select** and navigate to the following file:

`C:\labfiles\Support files\Ex14\Employment application.pdf`

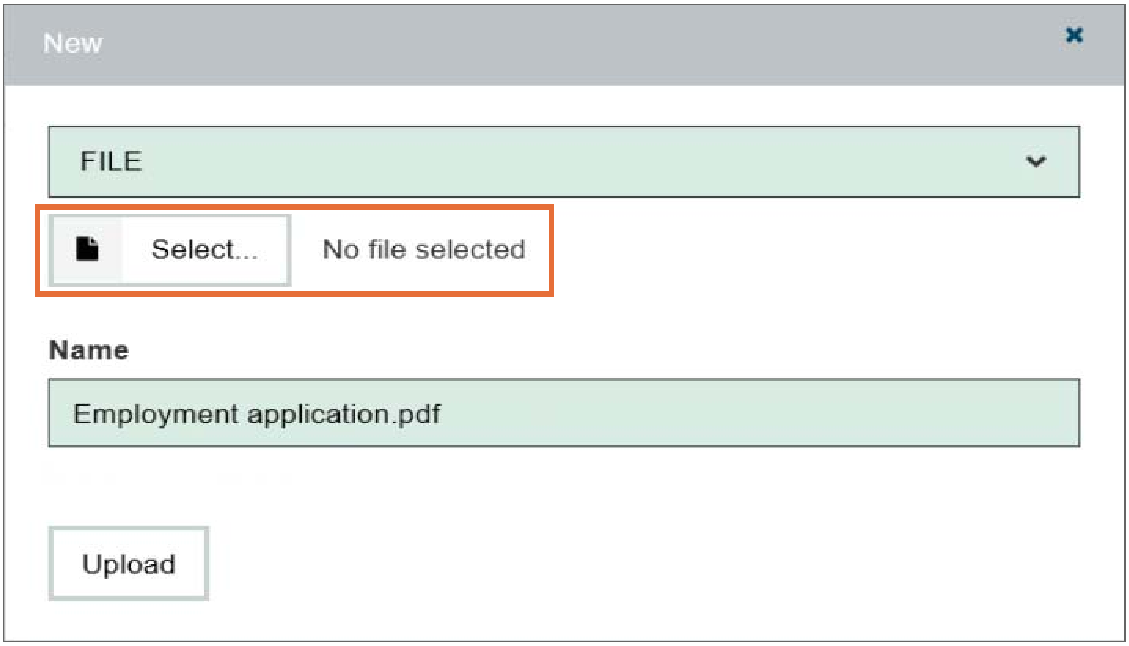
__ e. Click **Open**.

__ f. Click **Upload**.

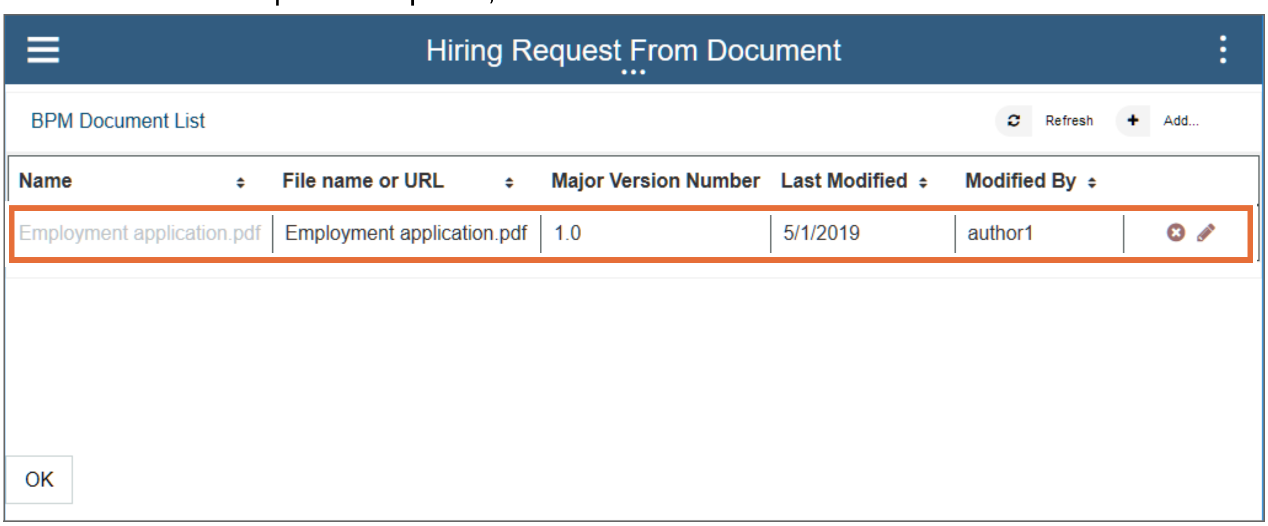


**Troubleshooting**

If the initial upload fails, verify there is a file selected. If no file displays as selected, close the window and try the operation again.



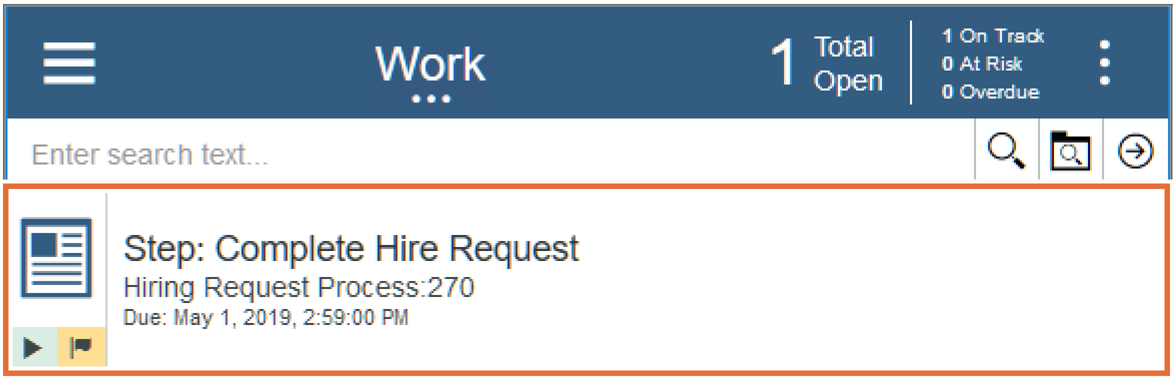

After the upload completes, the document list is refreshed.




__ g. You see a notice in the lower right of the screen that you received a new task.

__ 11. Verify the document start event works

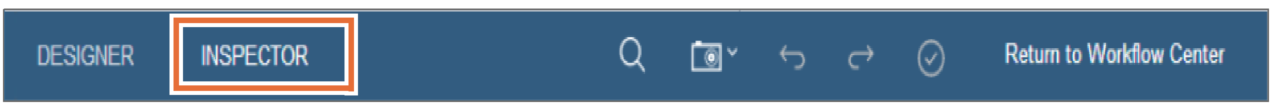
__ a. Click **OK** to close the service. The Work page is displayed with a new work item.



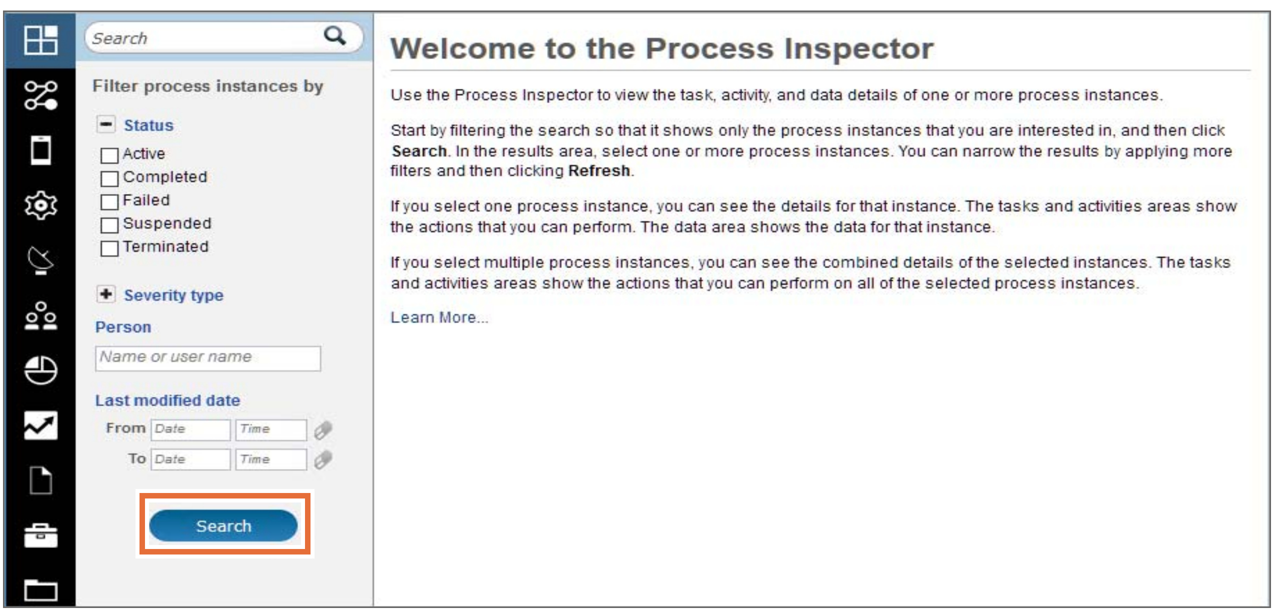

**Troubleshooting**

You might have other work items left from other exercises. If the work item is not listed, press F5 to refresh the page. If it is still not listed, restart the server.

__ b. Return to IBM Process Designer and click the **Inspector** view.

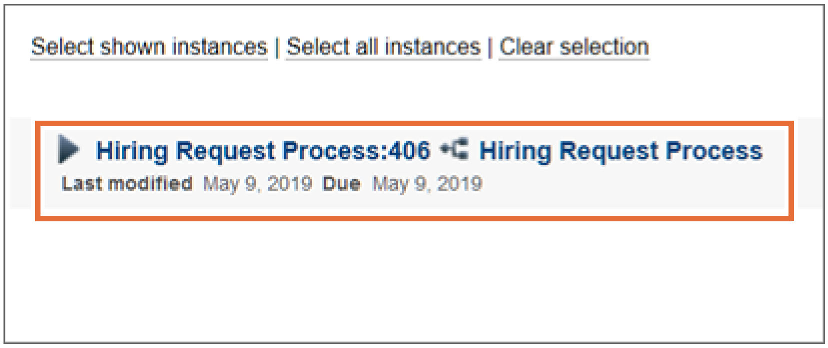



__ c. If the instance you created is not shown, click **Search**. If there are process instances in the list, click **Refresh**.



__ d. Click the newly created **Hiring Request Process** instance that got added to the instance page. The latest instance is the one with the highest instance ID.

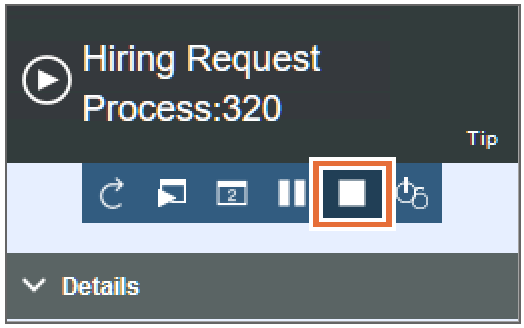



__ e. Scroll down in the right pane and expand the **Data** section. Expand the `content` variable. The `content` variable is populated, and the `objectId` is shown.

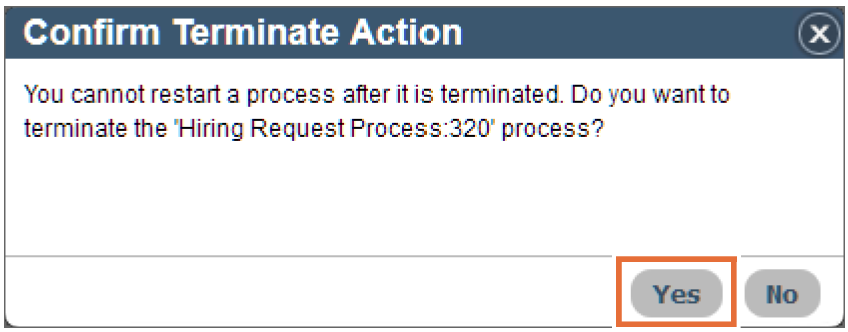

The creation of the document in the internal document repository triggers the creation of the instance. If you created this functionality as part of a production development effort, the next step inside the process is to associate the document with the instance. You would then create the step when the instance is complete and persist the document to an ECM server.

__ 12. Clean up your environment

__ a. Stop the process by clicking the **Terminate** icon.

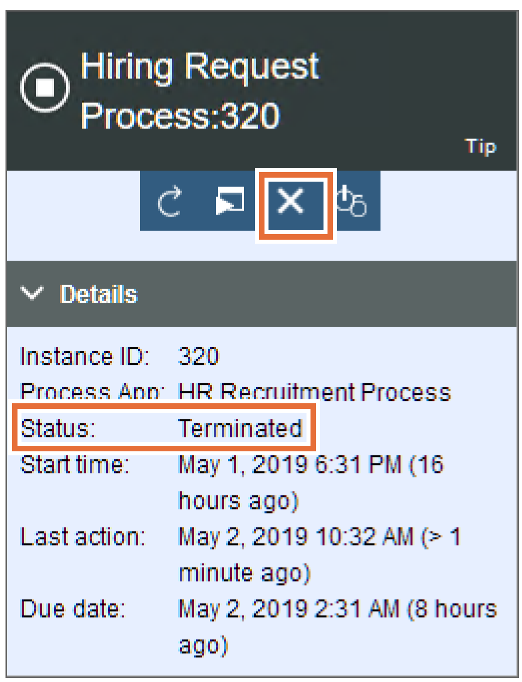



__ b. Click **Yes** when presented with the confirmation dialog.




__ c. The process is now in a **Terminated** status.

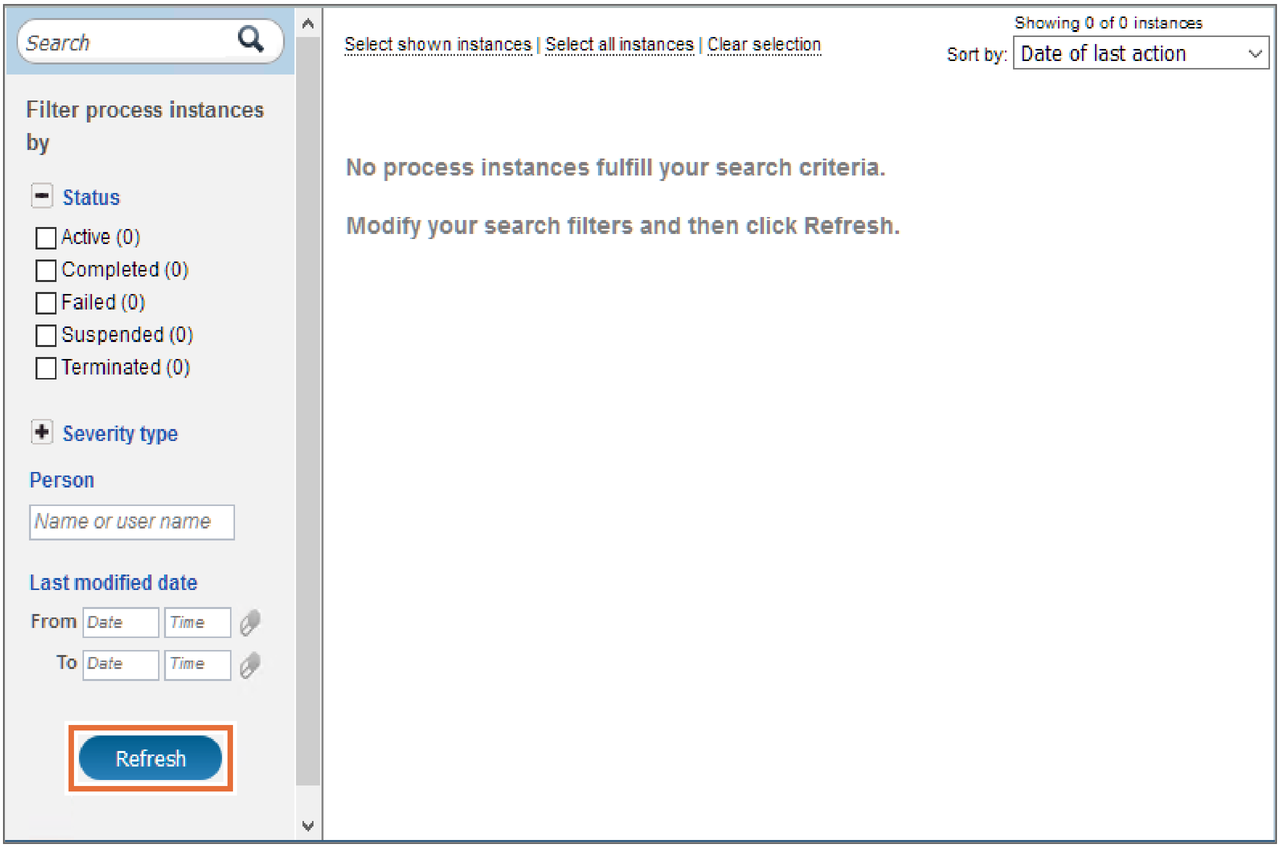
__ d. Click the **Delete** icon to delete the process instance.




__ e. Click **Yes** when presented with the confirmation dialog.

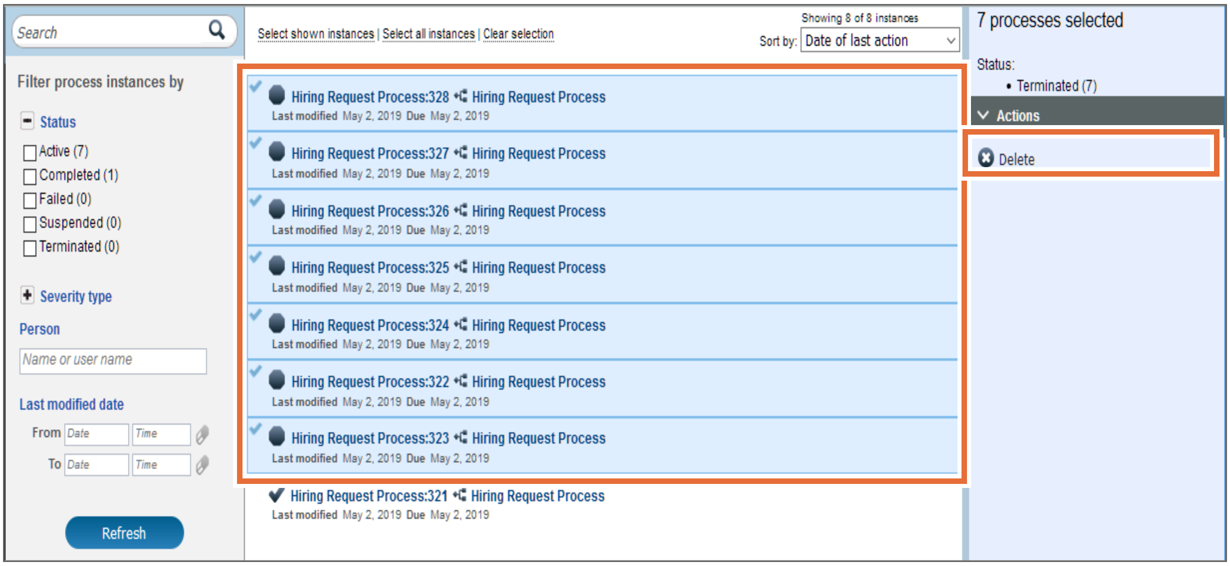
It is important to clean up the environment to maintain performance service levels. Leaving running processes in the environment that are unnecessary can degrade performance and cause unnecessary clutter.

__ f. Click **Refresh** to verify that the process instance is no longer listed.



**Information**

If you have extra process instances as a result of testing, you can use the Process Inspector to terminate and delete these instances. You can highlight instances that are in the same state and perform batch operations that appear to the right.




## *Part 2: Configure the Document Viewer to use the Target Object Store (TOS)*

Up to this point, you configured an out-of-the box use of the local BPM document store. However, Business Automation Workflow also has a Target Object Store. The Target Object Store can be used to share documents between Case solutions and Process applications.

Management would like to see how content can be shared between a Case solution and a BPM solution. Specifically, management would like to see how a Case solution can trigger the new HR structured process solution and share documents between the two solutions by using one content repository.

As part of this exercise, you create a simple, bare-bones Case solution that starts the Hiring Request Process from the HR Recruitment Process BPM solution. To do this, you add the Hiring Request Process as an activity to the Case solution.

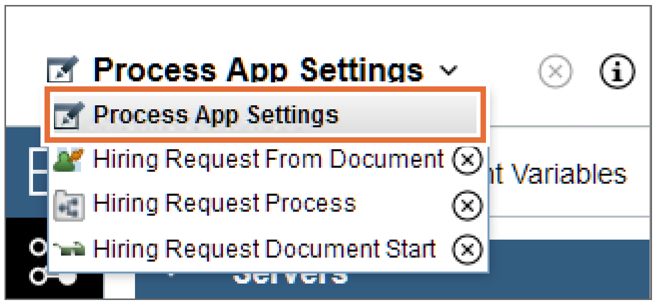
In this section, you also define an ECM server that points to the Target Object Store (TOS). Then, you reuse what you built in the Hiring Request From Document coach and configure the view to display the TOS document store instead of the BPM document store. Then, you test the integration between the Case solution and the BPM solution by adding a new case and document to the Case.

After the document is added, you verify that the Hiring Request Process was started and that the newly configured view displays the new document in the document list.

__ 1. Define the new ECM server to use the Target Object Store.

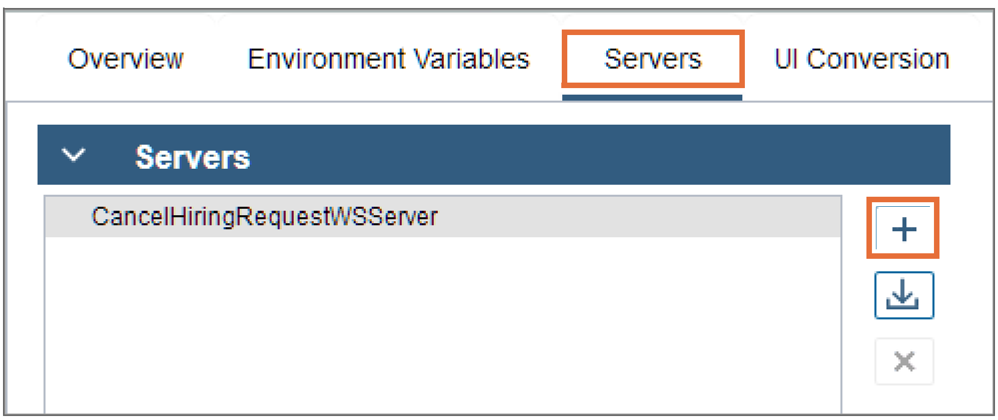
__ a. Return to the **Designer** tab.


__ b. Open the **Process App Settings** by selecting it from the drop-down list.



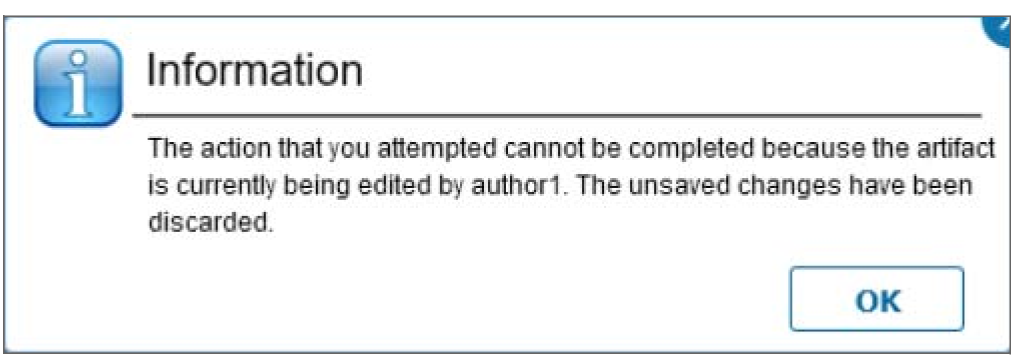

__ c. Select the **Servers** tab.

__ d. Click **Add a new server**.




**Troubleshooting**

If you see the following Information window, click **OK** and close the browser and log in again to Workflow Center.



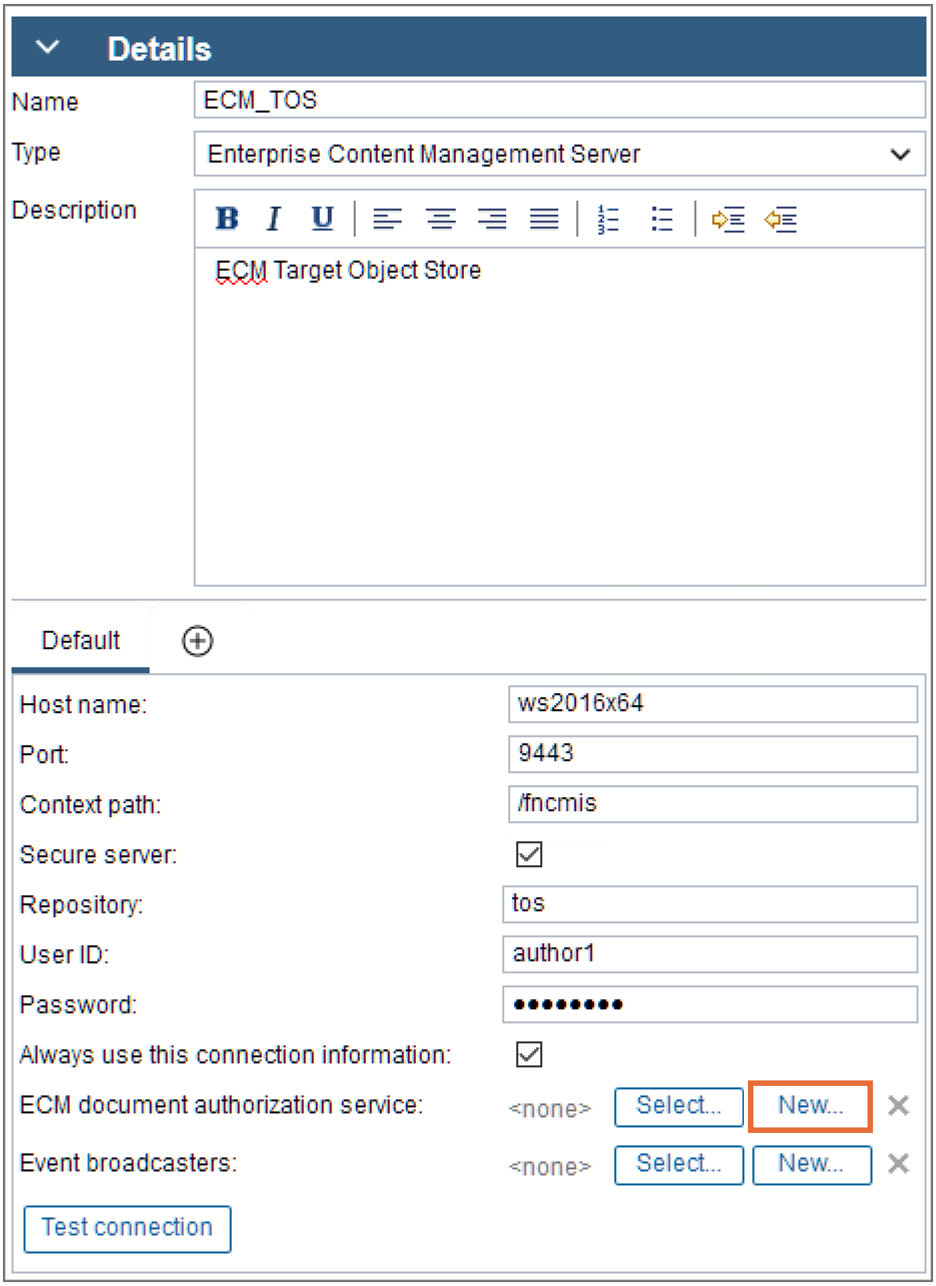
__ e. Configure the new server with the following information:

| Field | Value |
|---|---|
| Name | `ECM_TOS` |
| Type | `Enterprise Content Management Server` |
| Description | `ECM Target Object Store` |
| Hostname | `ws2016x64` |
| Port | `9443` |
| Context path | `/fncmis` |
| Secure server | **checked** |
| Repository | `tos` |
| User ID | `author1` |
| Password | `passw0rd` |
| Always use this connection information | **checked** |

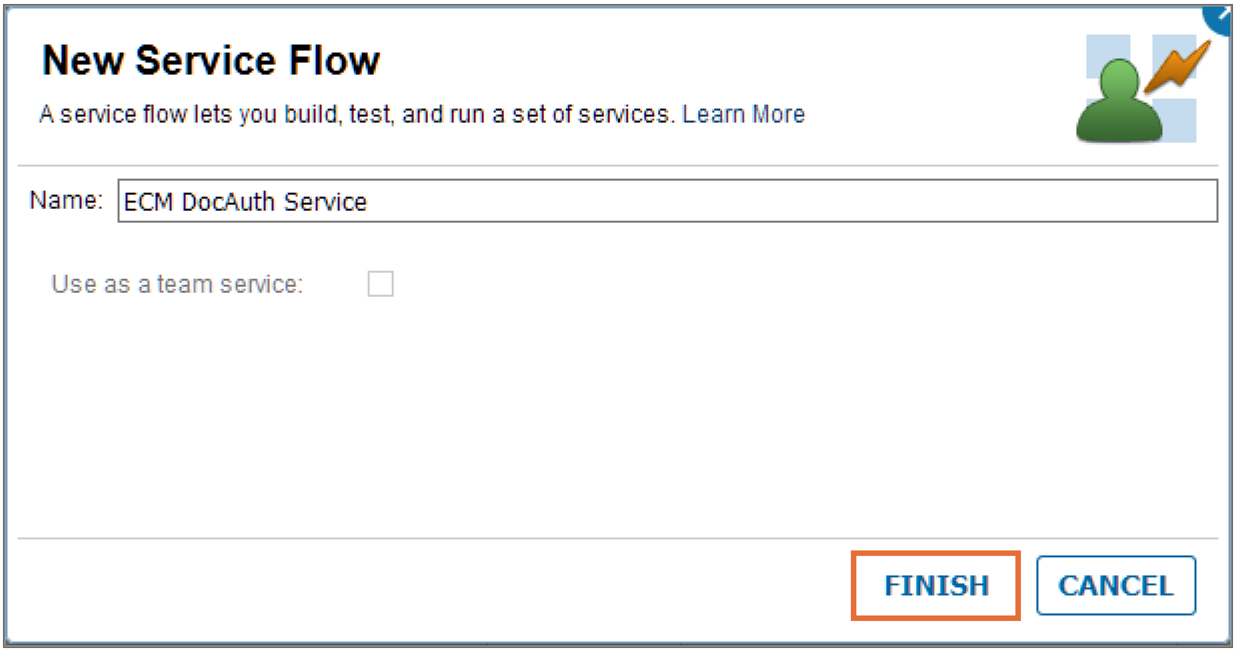
__ f. Save your work.

__ 2. Create the ECM document authorization service.

__ a. Click **New** next to ECM document authorization service.



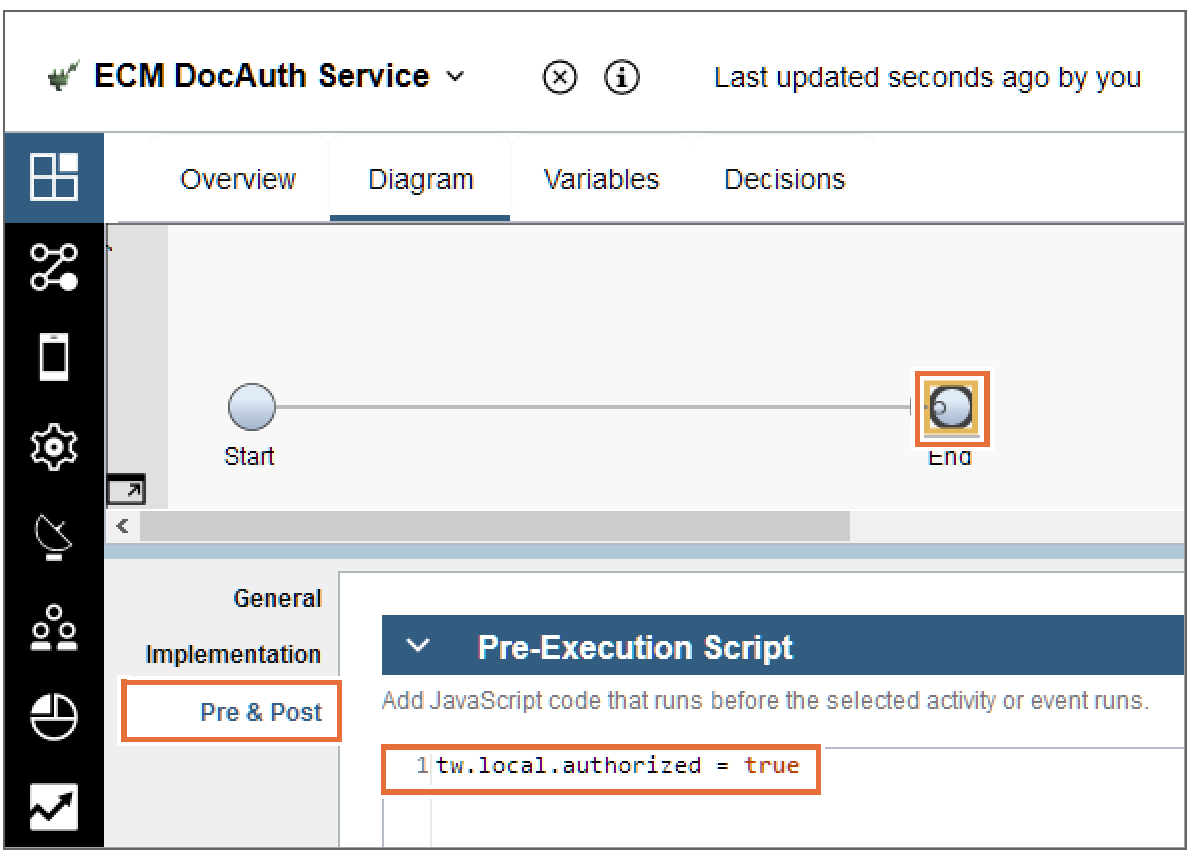
__ b. Name the new Service Flow `ECM DocAuth Service` and click **FINISH**.




__ c. Select the **End** event.

__ d. Select the **Pre & Post** properties menu below.

__ e. Enter the following for the Pre-Execution Script: `tw.local.authorized = true`




__ f. Save your work.

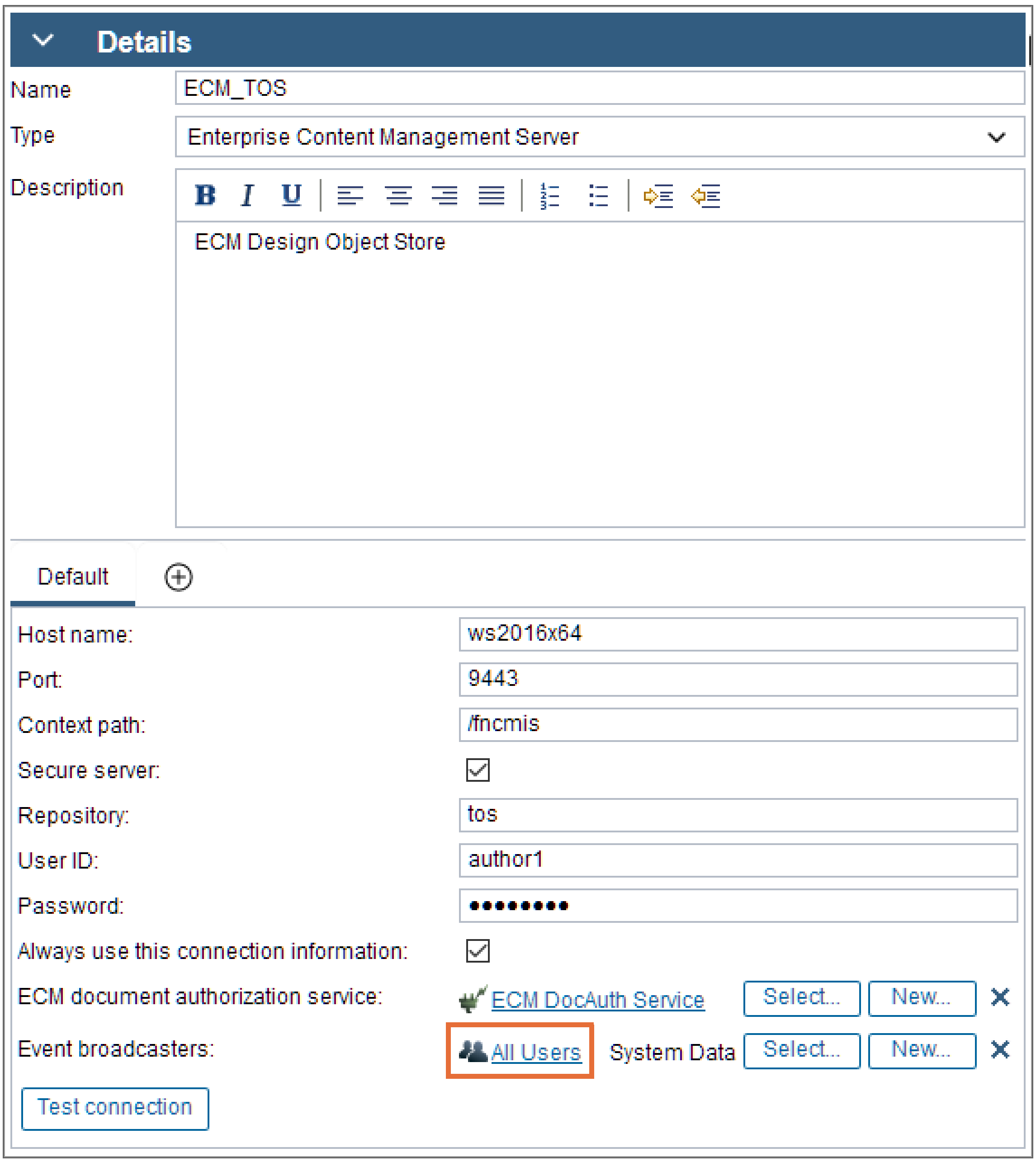
__ g. Close the **ECM DocAuth Service** window.


__ h. Return to the **ECM_TOS** configuration.

__ 3. Configure the new ECM server to broadcast events to all users and test the connection.

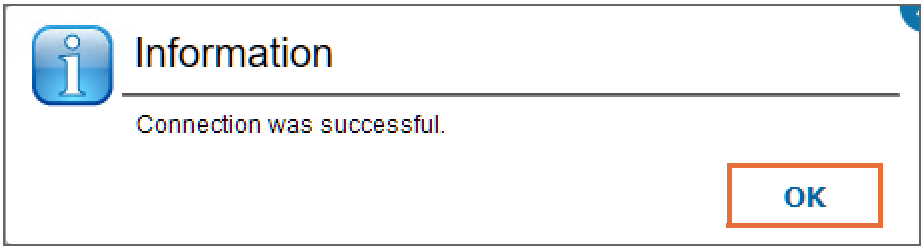
__ a. Click **Select** next to Event broadcasters and select **All Users**.

__ b. Verify your configuration and save your work.




__ c. Click **Test connection** to verify connection to the object store.

__ d. Click **OK** on the Information dialog.

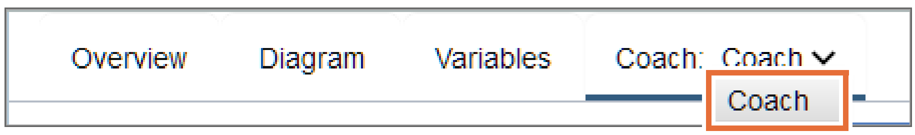



Now that the server is configured, you can return to the Startable Service coach to configure it to use the new ECM_TOS server.

__ 4. Create the ECM list View

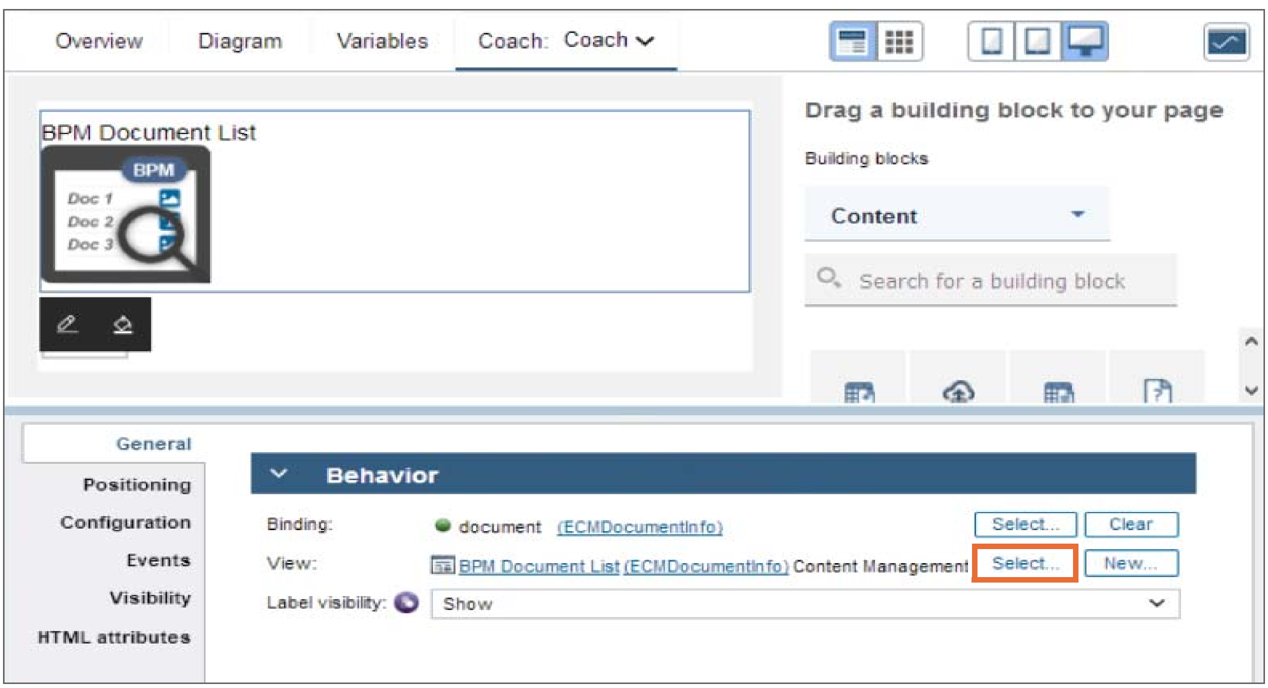
__ a. Return to the **Hiring Request From Document** client-side human service.

__ b. Click the Coach downward arrow and select Coach.

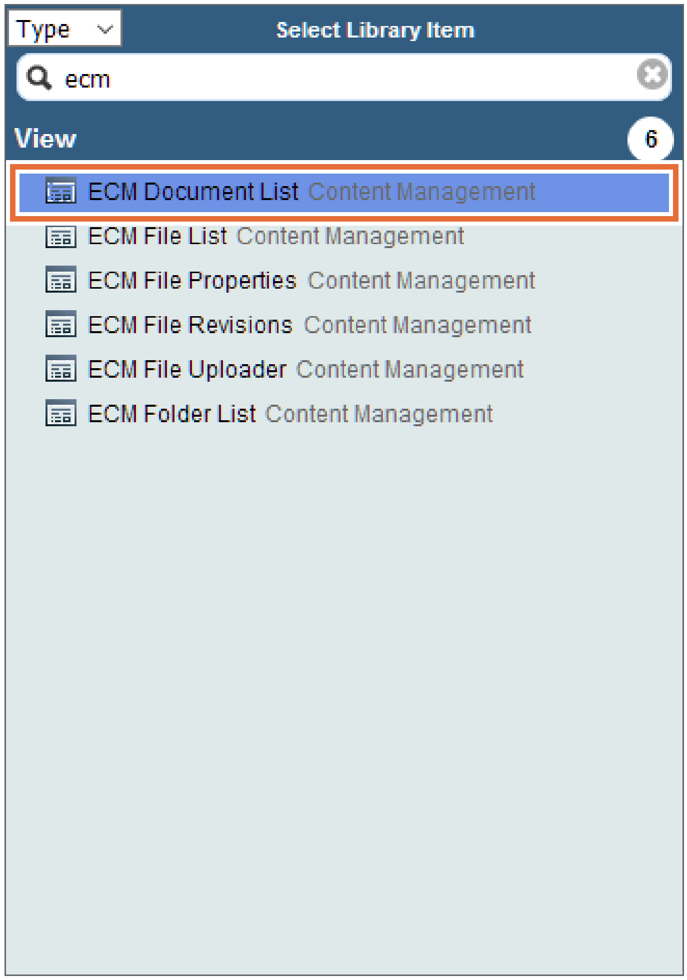



__ c. Select the **BPM Document List**.

__ d. Under the **Behavior** section in the **General** properties tab, click **Select** next to View.



__ e. Enter `ecm` to filter the selection. Select the **ECM Document List** view.




Because the ECM Document List uses the same content object (ECMDocumentInfo), you can leave the binding as is.

__ f. Rename the view to **ECM Document List**.

__ g. Click the Configuration properties tab and expand **Content management advanced**.

__ h. Enter `ECM_TOS` for the ECM server configuration name.


__ i. Save your work.

## Part 3: Build Case solution

Although this course does not cover Case Management development, it is important to understand how the integration works between a Case solution and a BPM solution.

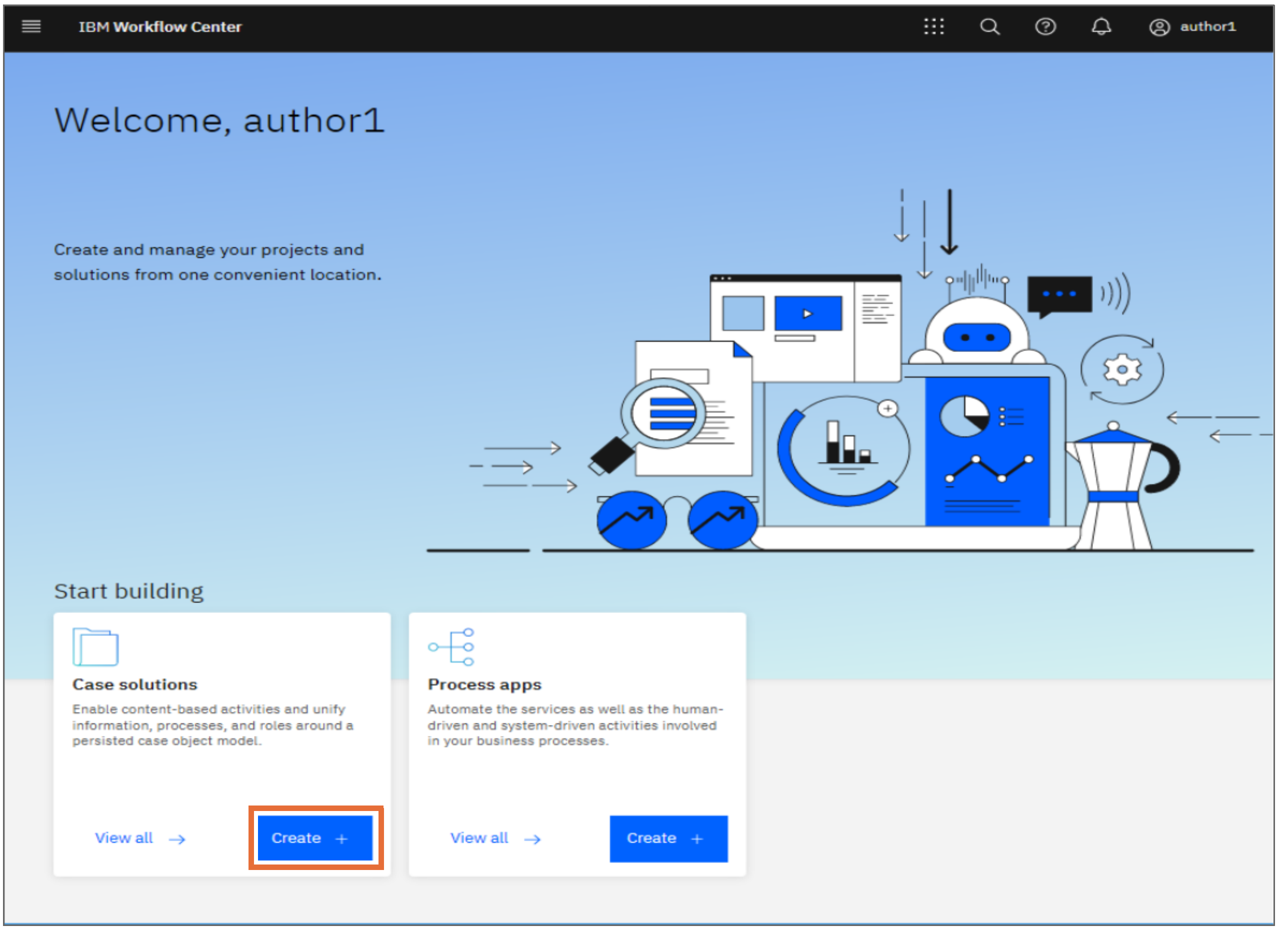
In this section, you build a simple, bare-bones Case solution by using the Workflow Center and Case Builder. You only create the necessary components to run the solution and test integration between the Case and the BPM solution.

__ 1. Create a new case solution

__ a. Open a new tab in the browser and access the Workflow Center.

If prompted, log in to the Workflow Center as `author1` / `passw0rd`

__ b. Click **Create** on the Case solutions tile to create a new Case solution.



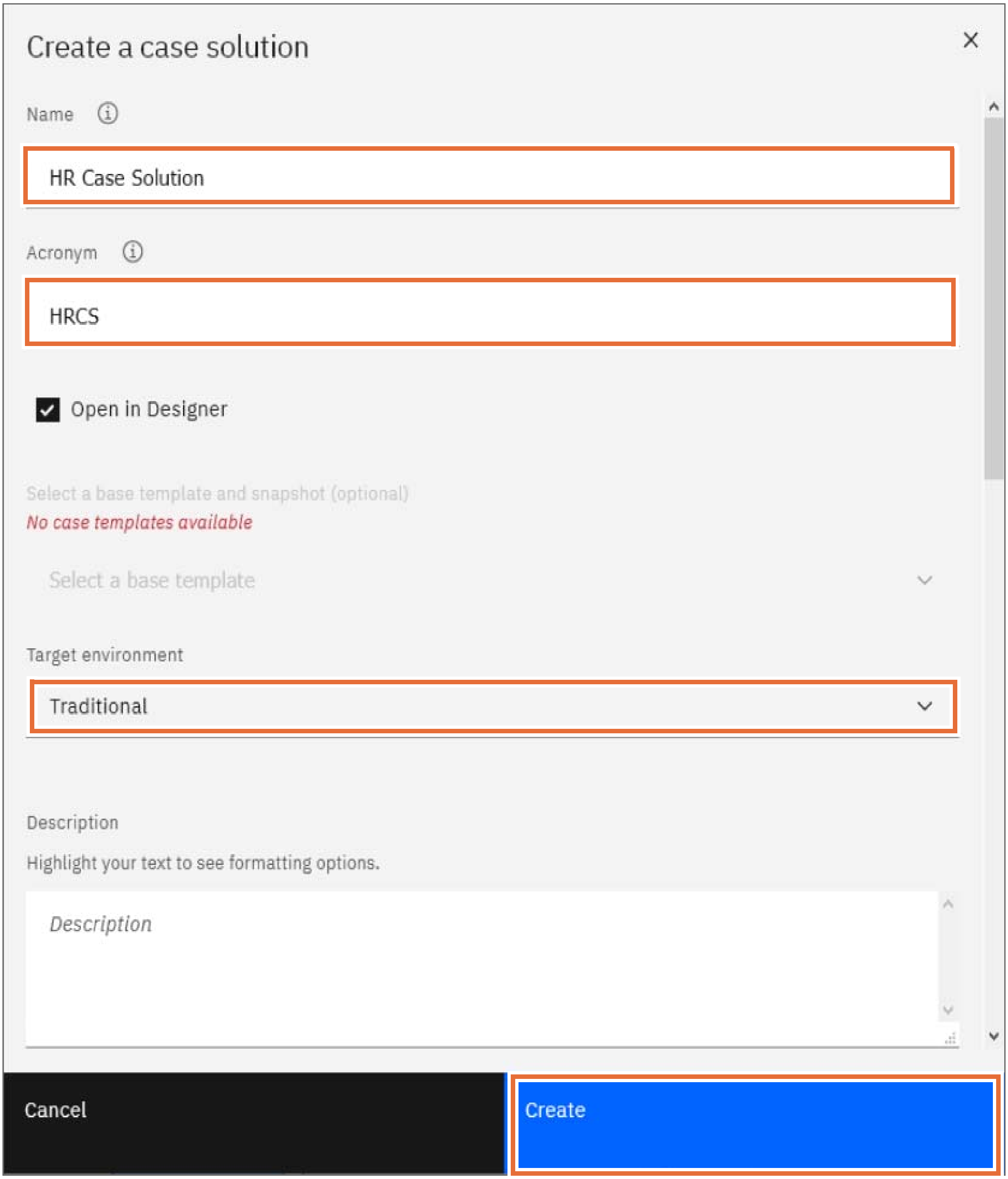

__ c. Name the solution: `HR Case Solution`

__ d. Leave the acronym as the default.

__ e. Make sure that the checkbox is selected for **Open in Designer**

__ f. Choose **Traditional** for Target environment

__ g. Click **Create**.



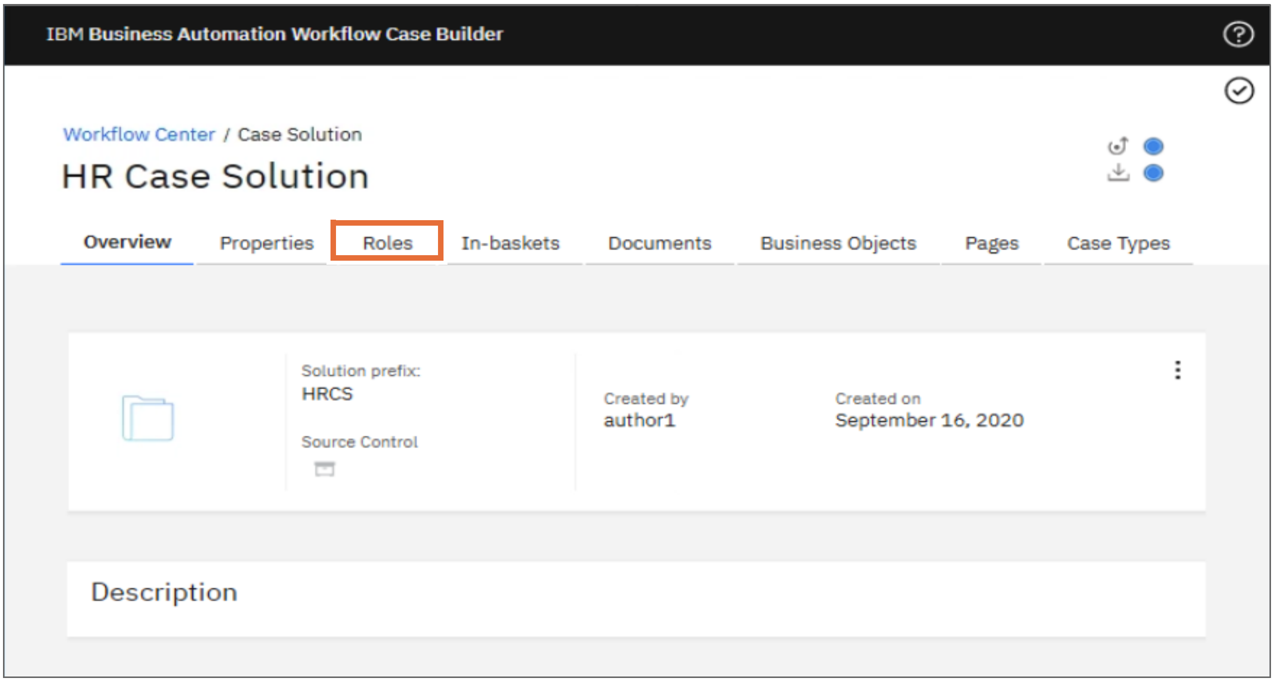

The solution is opened in Workflow Center.

**Hint**

If you do not see all the options, make sure you are using the **Advanced** View Mode in the user preferences of the Workflow Center.

__ 2. Add Role

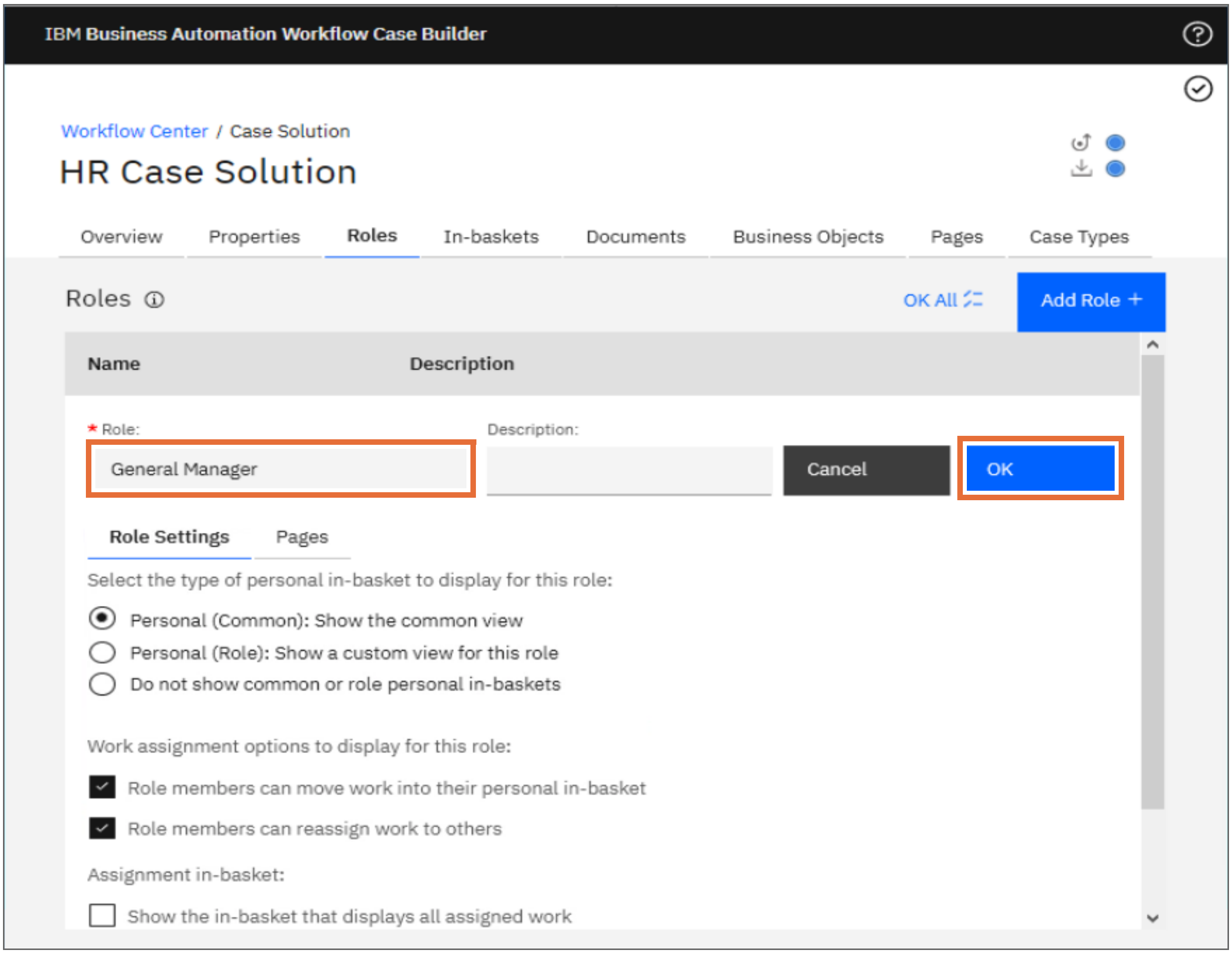
__ a. Click the **Roles** tab.




__ b. Click **Add Role**.

__ c. Enter `General Manager` for the Role.

__ d. Click **OK**.




__ 3. Add Document Class

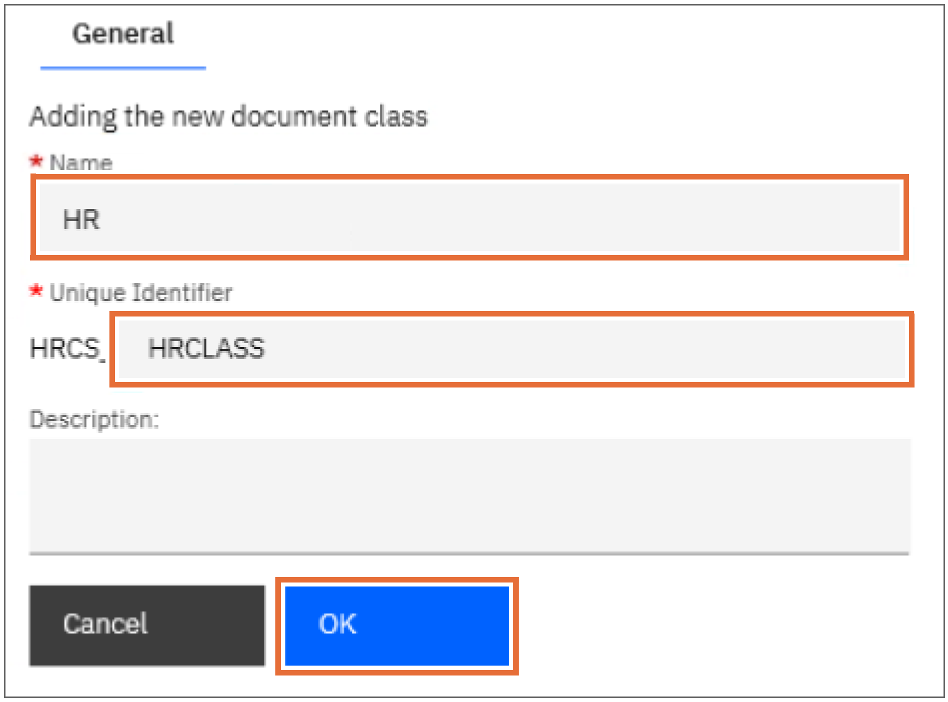
__ a. Click the **Documents** tab.

__ b. Click **Add Document Class** and select **New**.

__ c. Enter HR for Name.

__ d. Add HRCLASS to the Unique Identifier

__ e. Click **OK**.




The document class is displayed to the left.

__ 4. Add Case Type

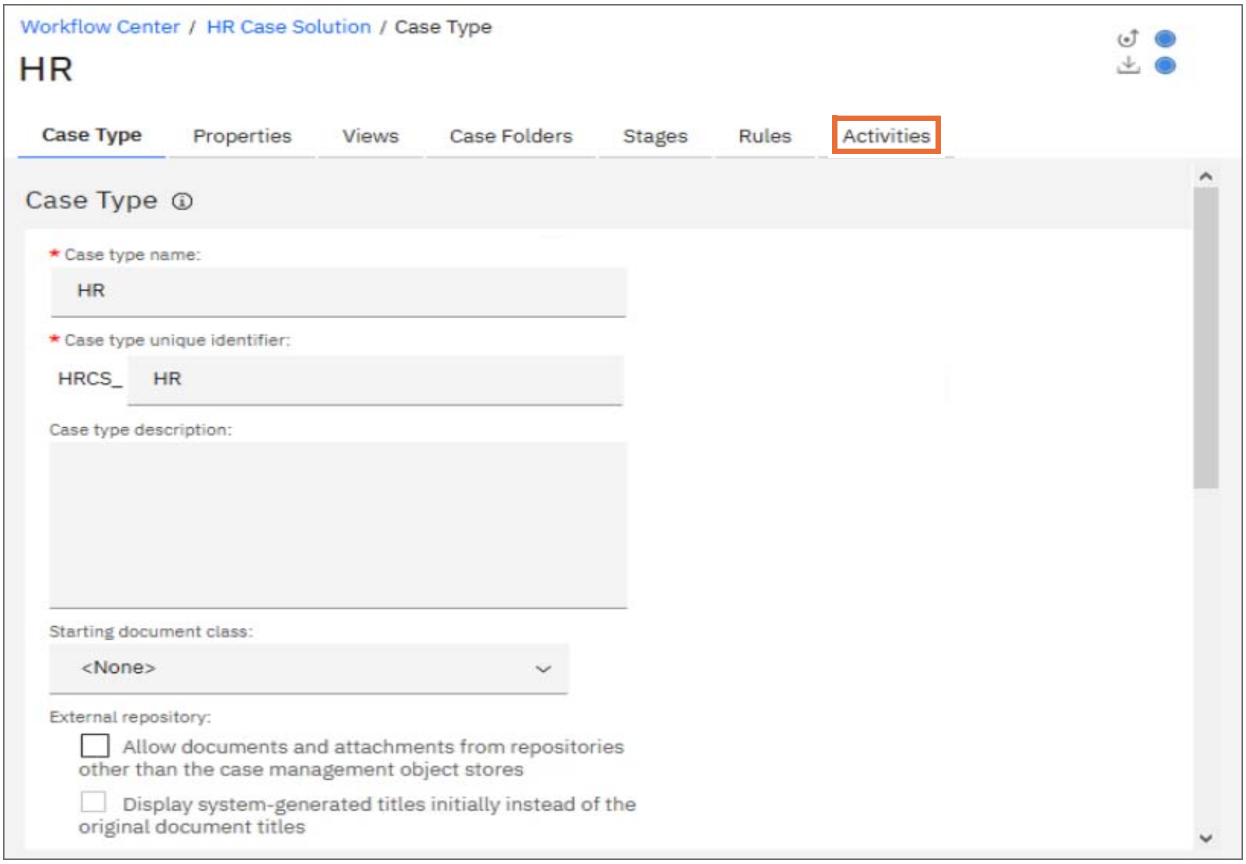
__ a. Click the **Case Types** tab

__ b. Click **Add Case Type**.

The Case is opened in a new editor

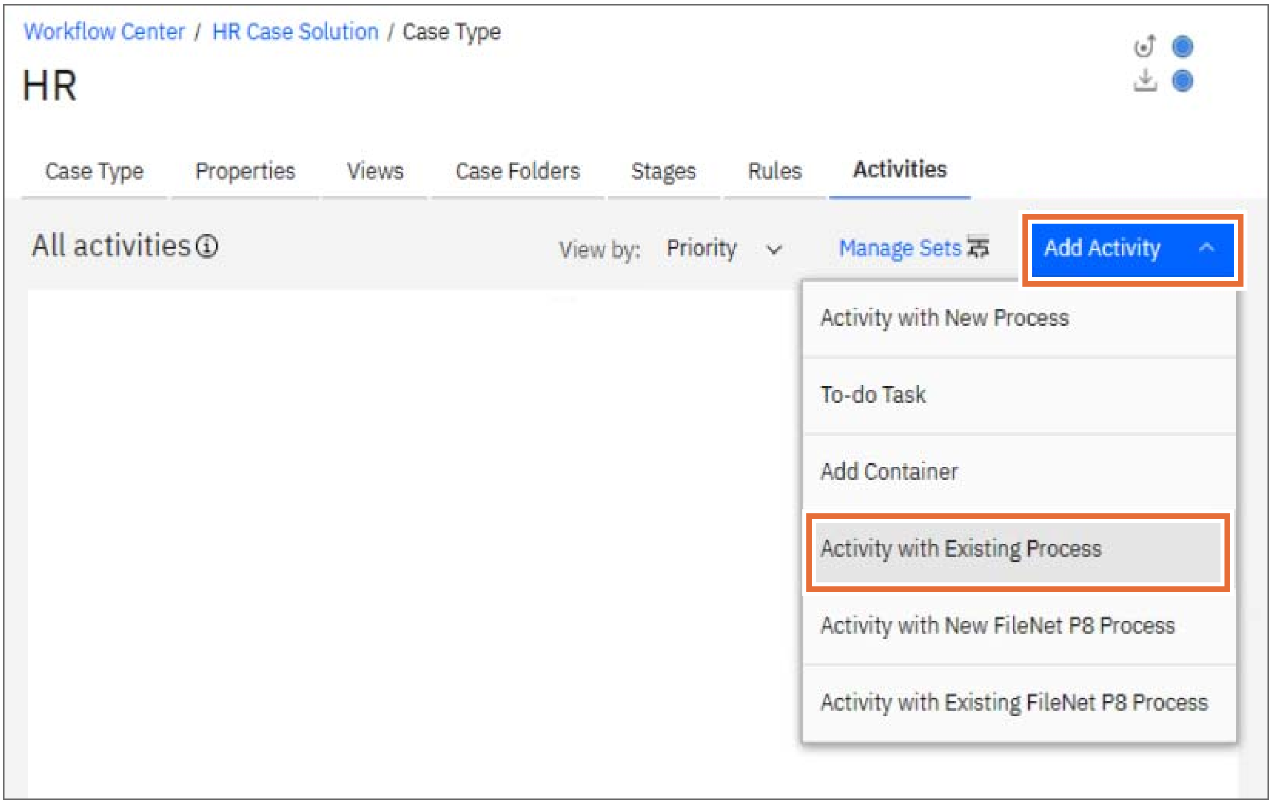
__ c. Enter `HR` for the Case type name.

__ d. Click **Activities**.



__ 5. Add Activity

__ a. Click **Add Activity** and select **Activity with Existing Process**

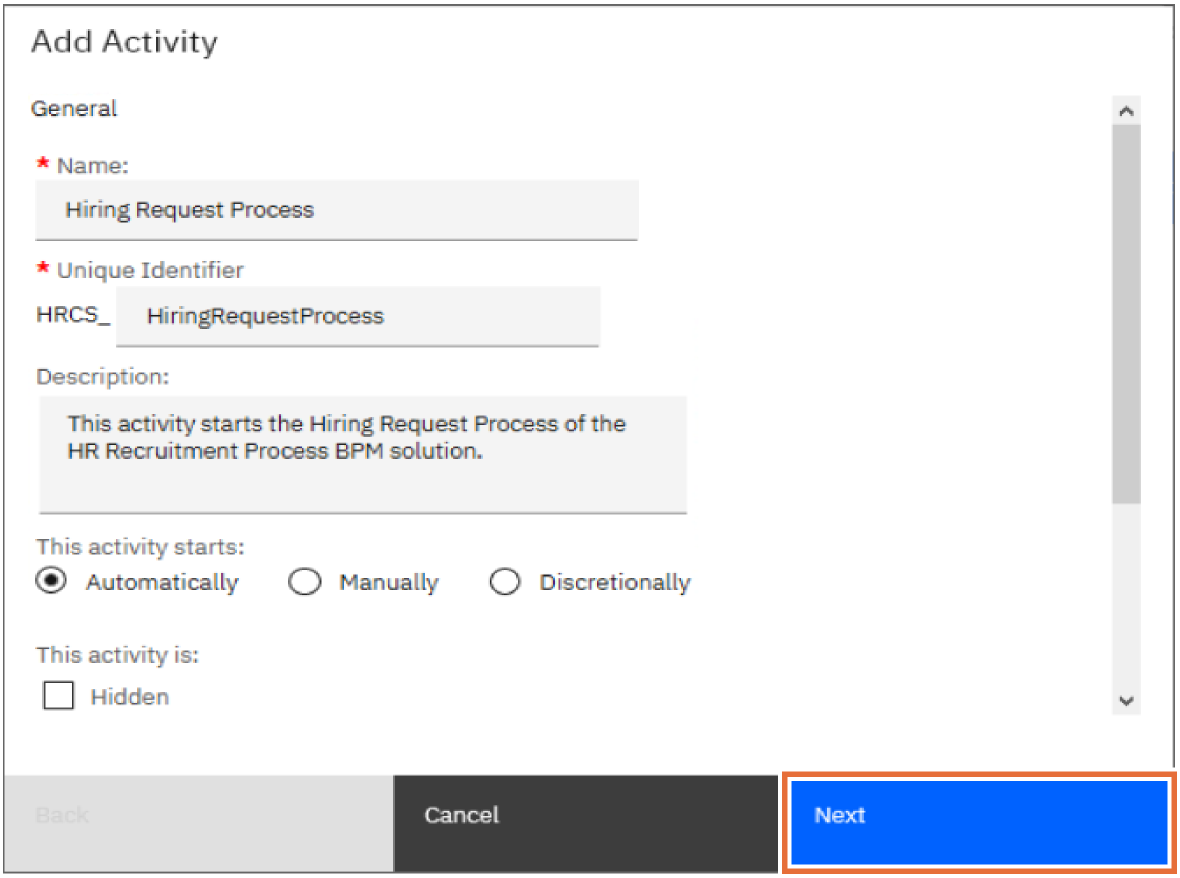



__ b. Enter `Hiring Request Process` as the **Name**.

__ c. Enter the following for description:

`This activity starts the Hiring Request Process of the HR Recruitment Process BPM solution.`

__ d. Click **Next**.

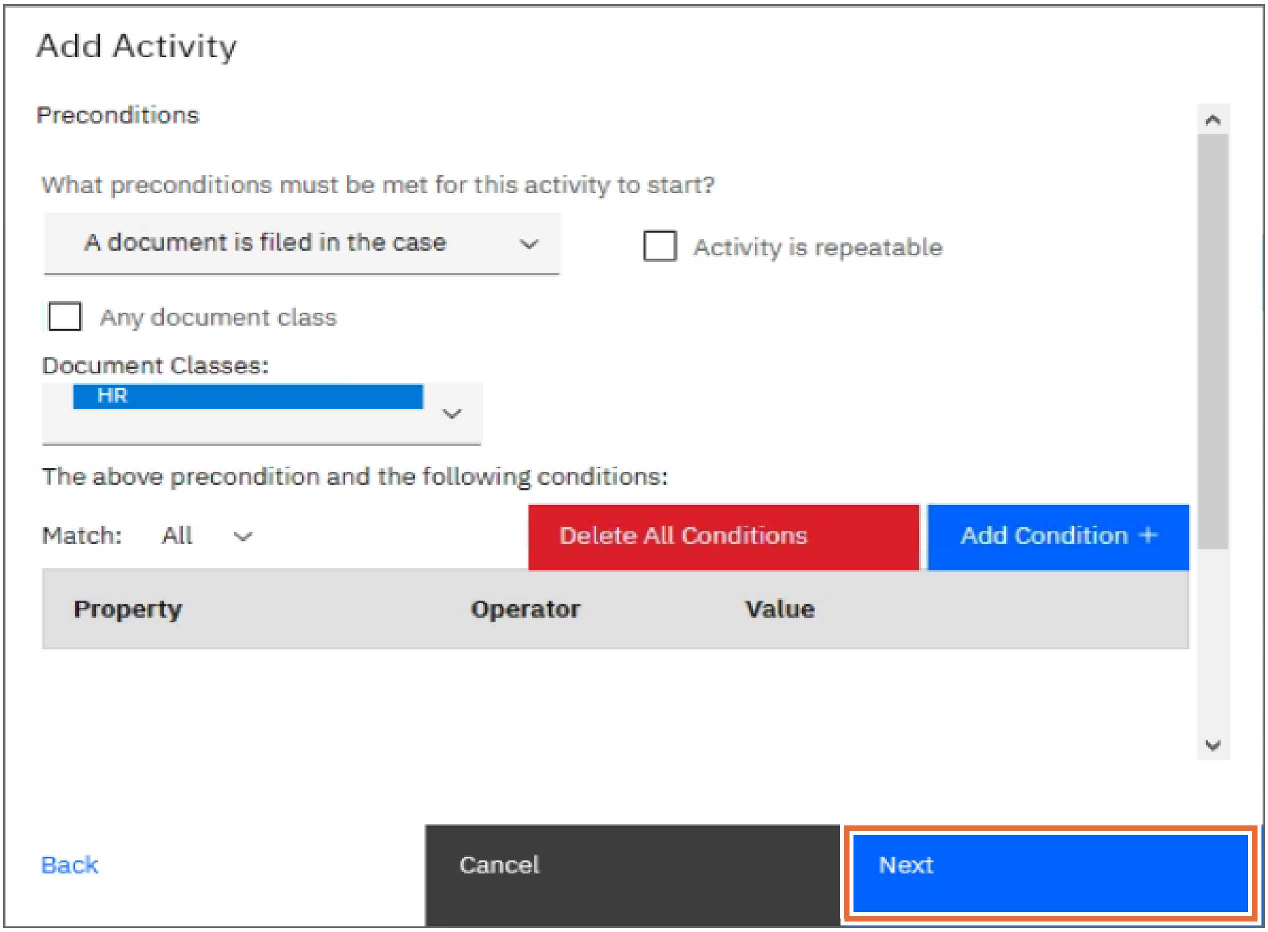



__ e. Click the Preconditions drop down menu and select **A document is filed in the case**.

__ f. Click the **Any document class** checkbox to clear it.

__ g. Verify that the **HR** Document Class is selected.

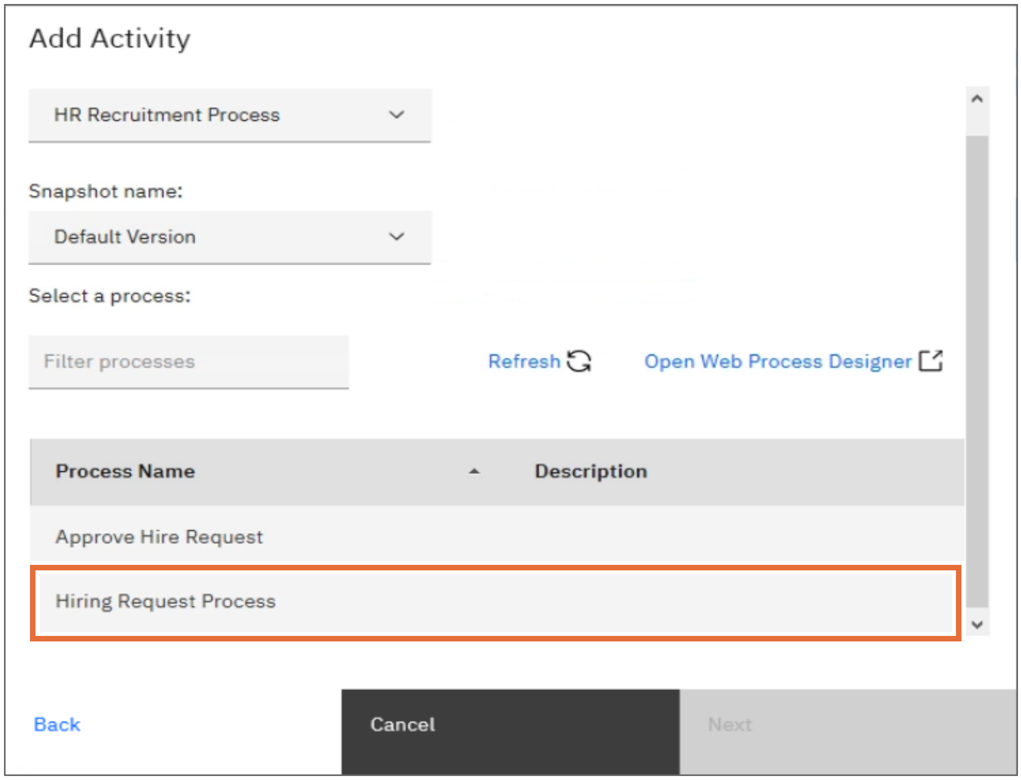
__ h. Click **Next**.




__ i. Select **HR Recruitment Process** for the Workflow Project name

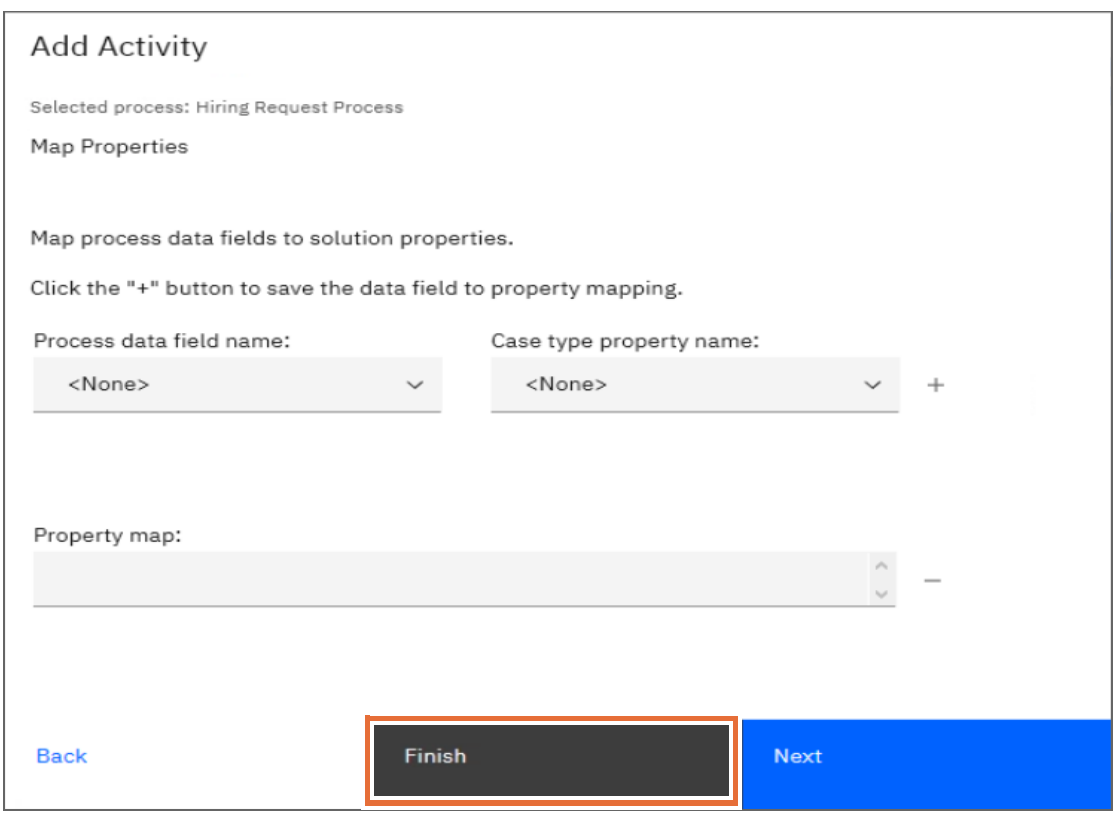
__ j. Leave **Default Version** for the Snapshot name

__ k. Select **Hiring Request Process** under the list of processes populated below.



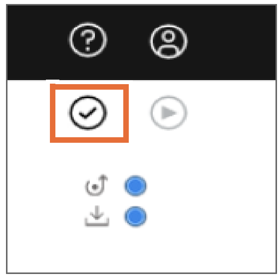

__ l. Click **Next**.

__ m. Because there are no properties in the Case, there is no need to map any properties in the next screen. Click **Finish**.



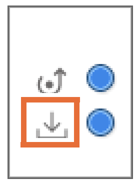
__ n. The activity is added to the Case as an optional activity.

__ o. Click **Save** in the upper right corner.

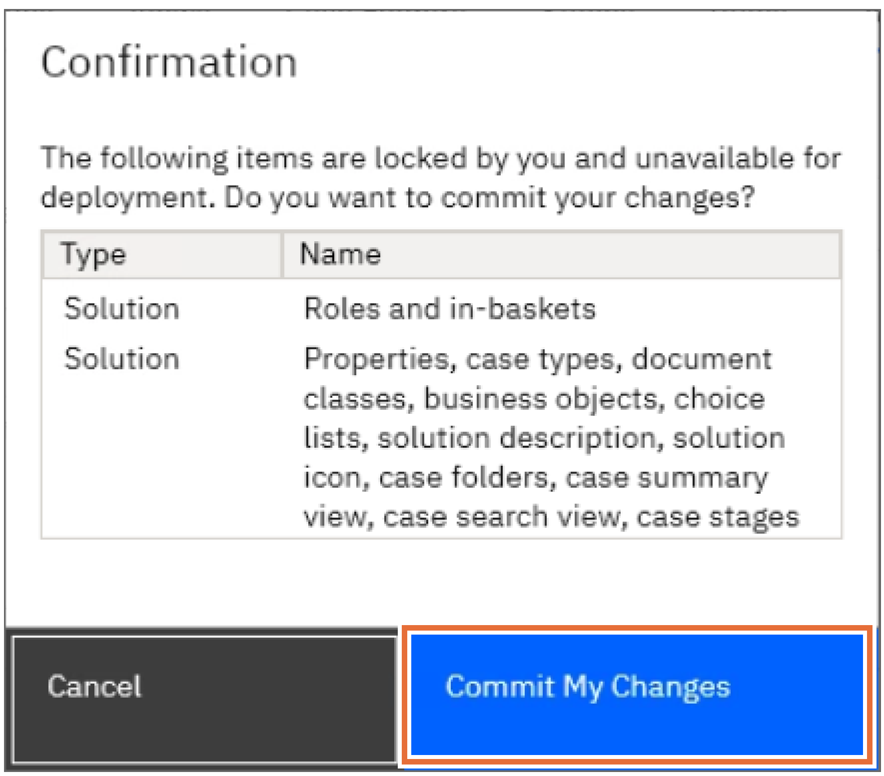


__ 6. Deploy the Case solution.

__ a. Click **Commit** in the upper right corner.

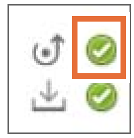


__ b. Click **Commit My Changes** in the Confirmation dialog box. The commit icon turns to a green check mark.



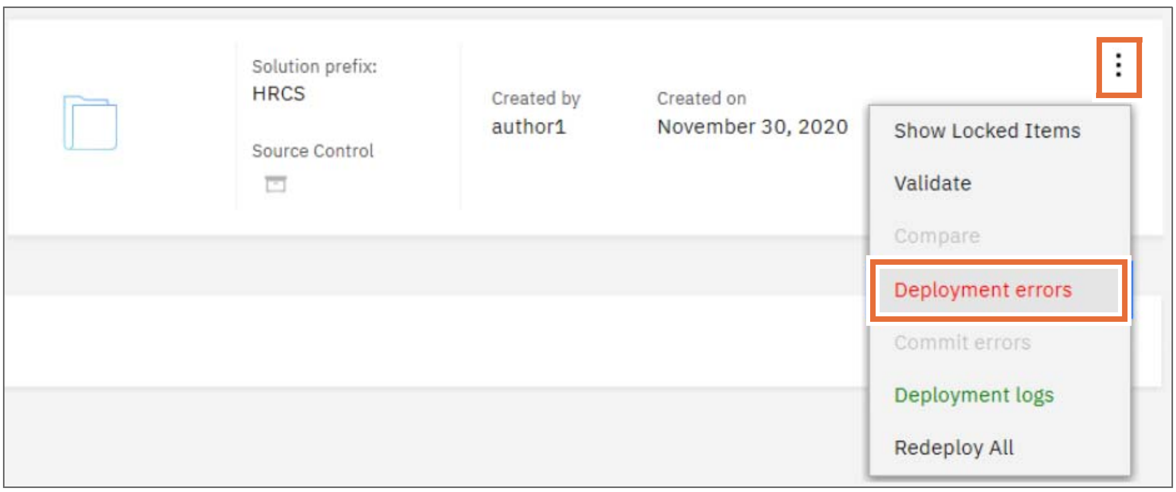
__ c. Click **Deploy** in the upper right corner.

After successful deployment, the deployment icon changes to a green check mark.

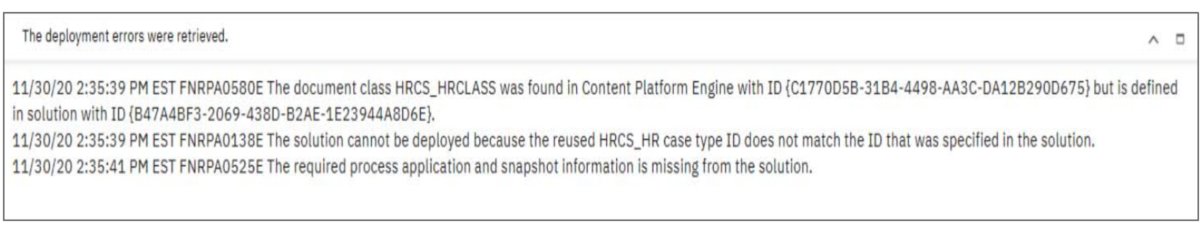


## Troubleshooting

If the deployment fails, you can select the Overview tab, then click the three dots to the right and select **Deployment errors** to view the errors in the deployment.




The deployment errors are listed below.



The deployment errors were retrieved.

11/30/20 2:35:39 PM EST FNRPA0580E The document class HRCS_HRCLASS was found in Content Platform Engine with ID {C1770D5B-31B4-4498-AA3C-DA12B290D675} but is defined in solution with ID {B47A4BF3-2069-438D-B2AE-1E23944A8D6E}.
11/30/20 2:35:39 PM EST FNRPA0138E The solution cannot be deployed because the reused HRCS_HR case type ID does not match the ID that was specified in the solution.
11/30/20 2:35:41 PM EST FNRPA0525E The required process application and snapshot information is missing from the solution.

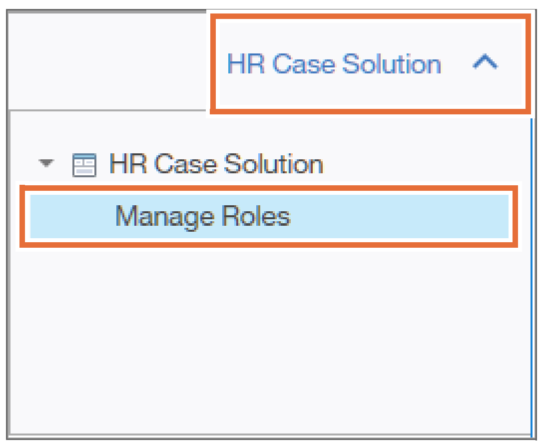
## Part 4: Test your solution

In this section, you test the solution, demonstrating the document list view now displays content from the Target Object Store (TOS) and that the Hiring Request process starts when a document is added to the HR Case solution.

__ 1. Create a new case.

__ a. Click **Start** to run the Case solution. The solution opens in a new window.

__ b. Before adding a case to the solution, you need to add yourself as a user to the General Manager role. Click **HR Case Solution** in the upper right, expand HR Case Solution, and click **Manage Roles**.

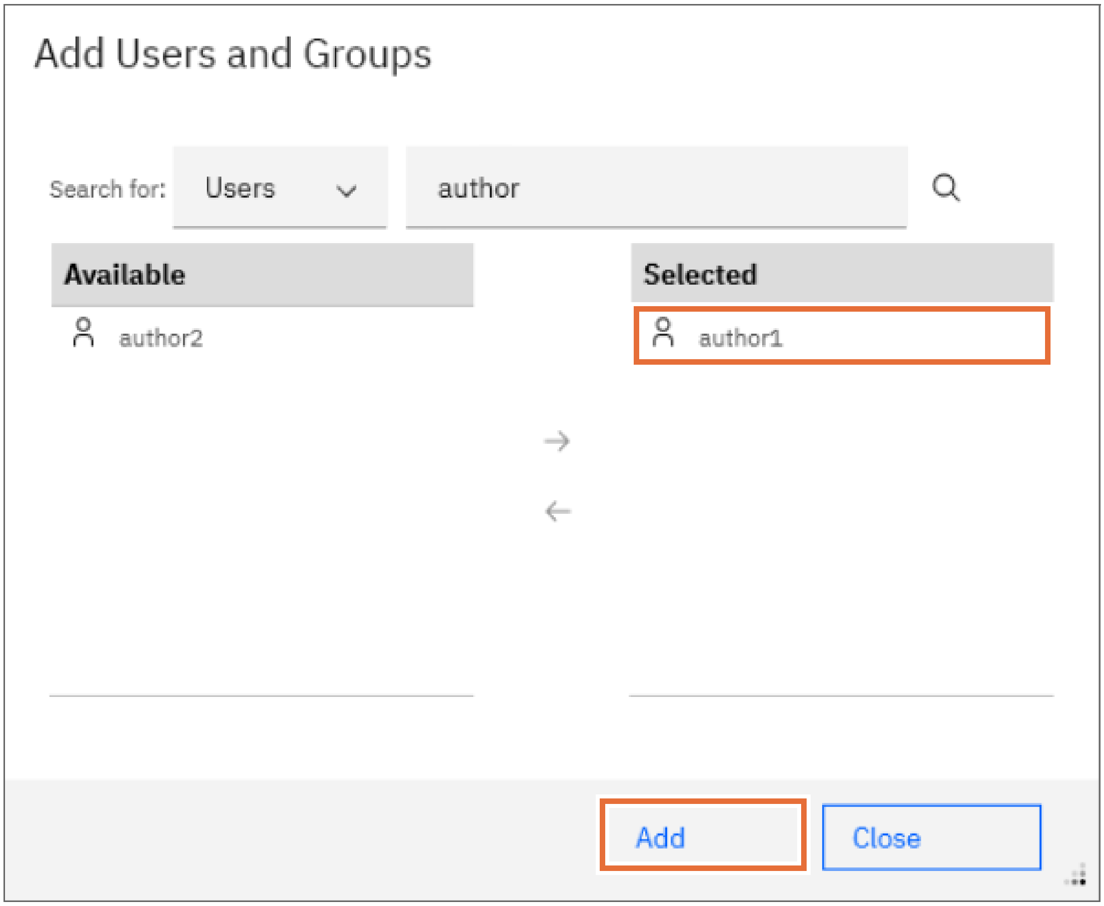



__ c. Click **Add Users and Groups**

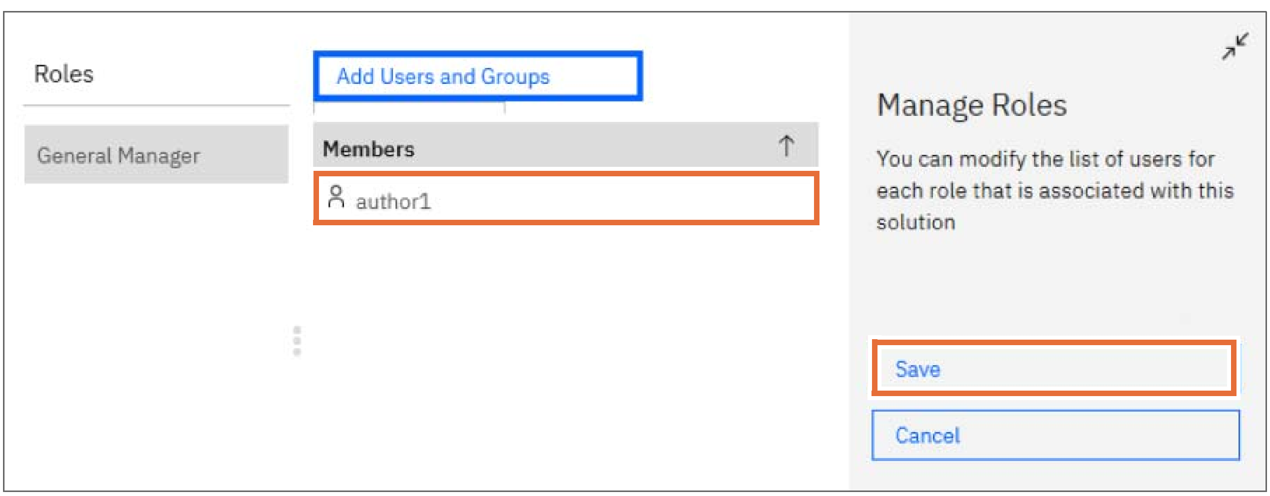
__ d. Enter `author` for the search criteria and click the **Search** icon.

__ e. Select **author1** and click **Add** (right arrow).

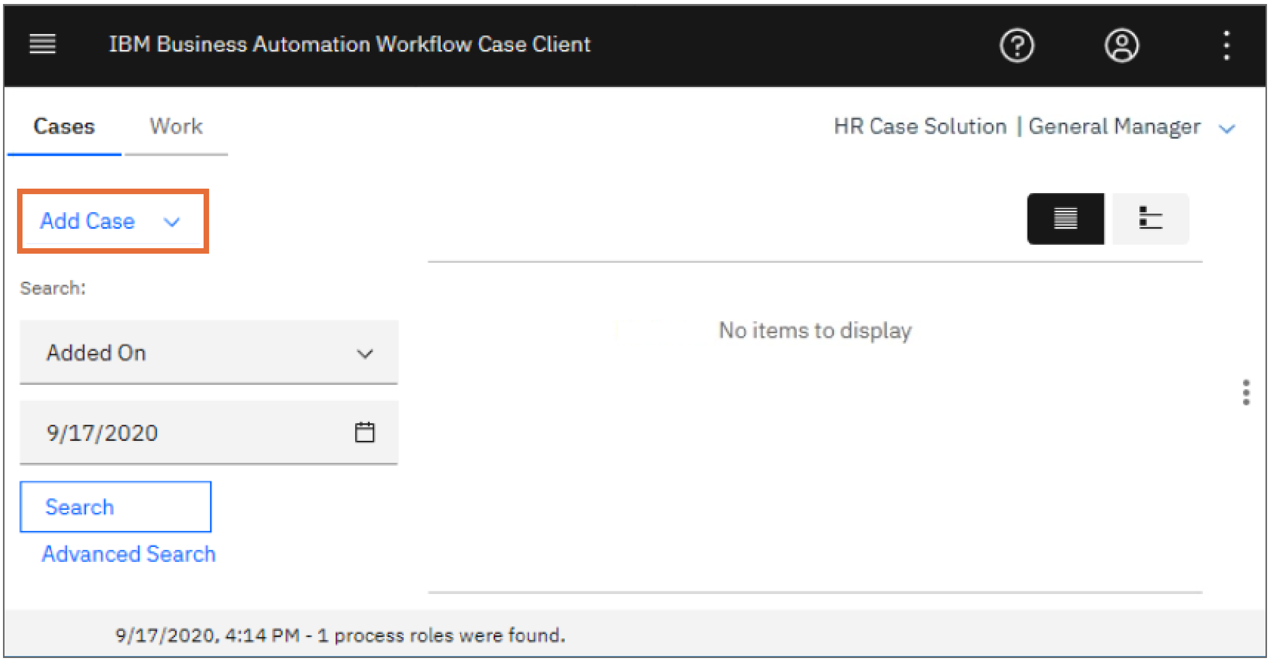
__ f. Once `author1` is in the Selected column, click **Add**.



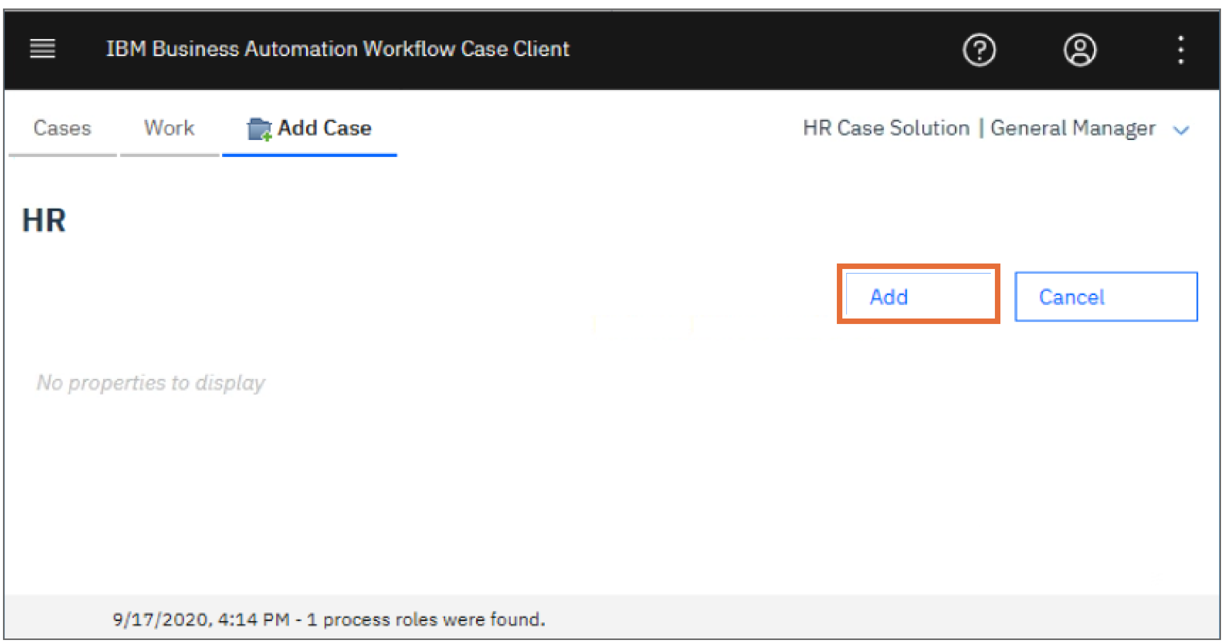

__ g. Verify `author1` is added as a user to the General Manager role and click **Save**.



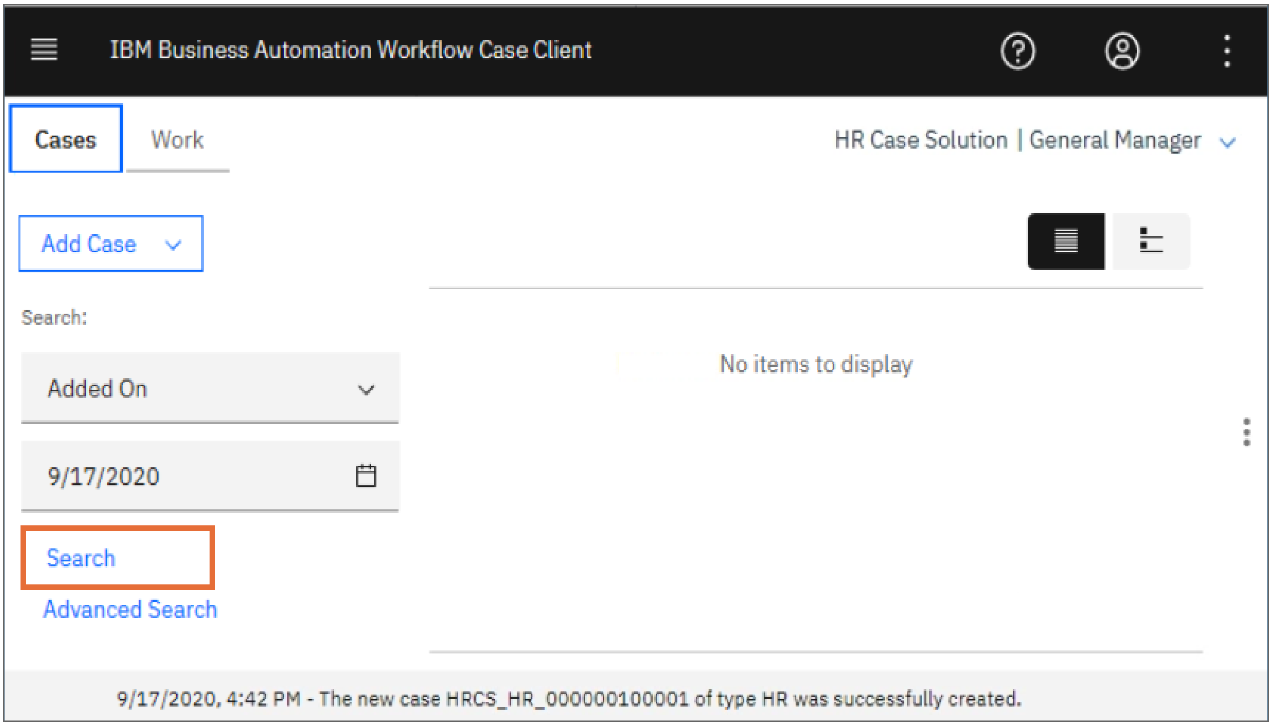
__ h. Click **Add Case**, then select **HR**.




__ i. Click **Add**.

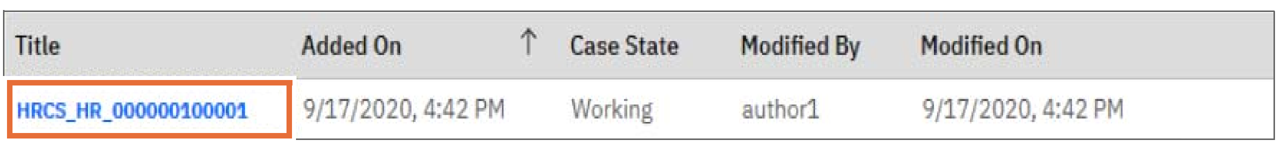


__ j. A message that states the new case was successfully created is displayed in the bottom status bar. Click **Search** to list the new case.

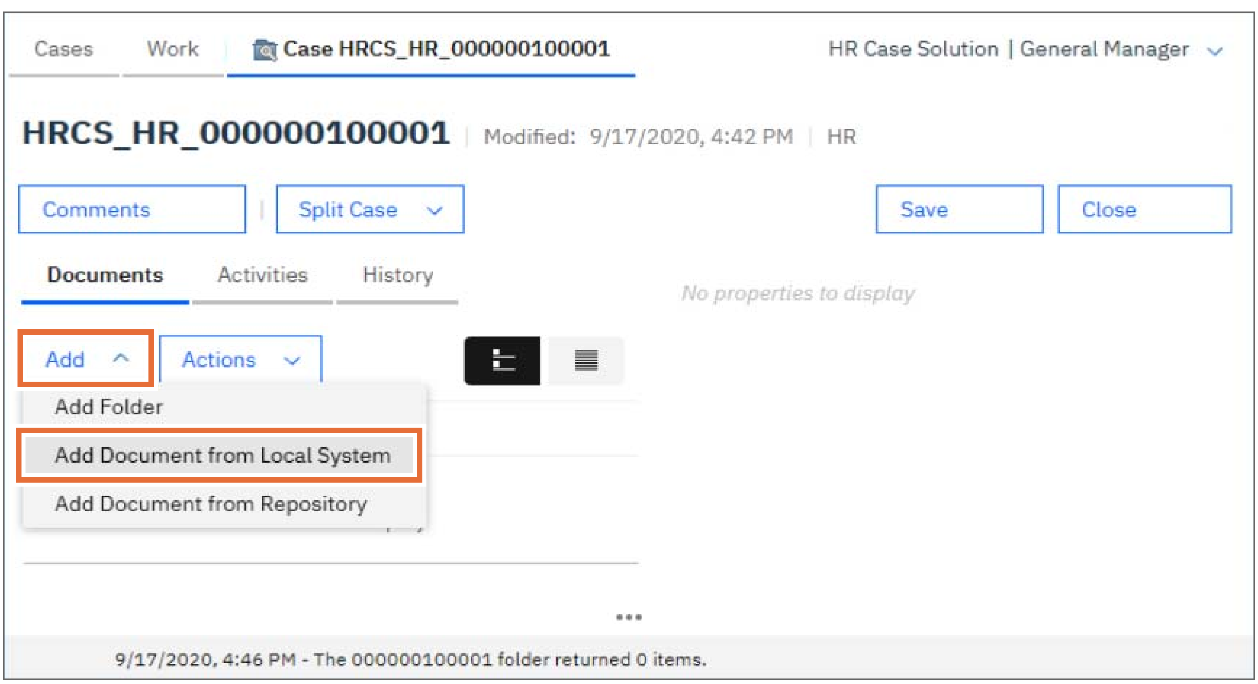



__ 2. Add a document to the case.

__ a. Click the title link for the newly added case. The title link for your case might be different.

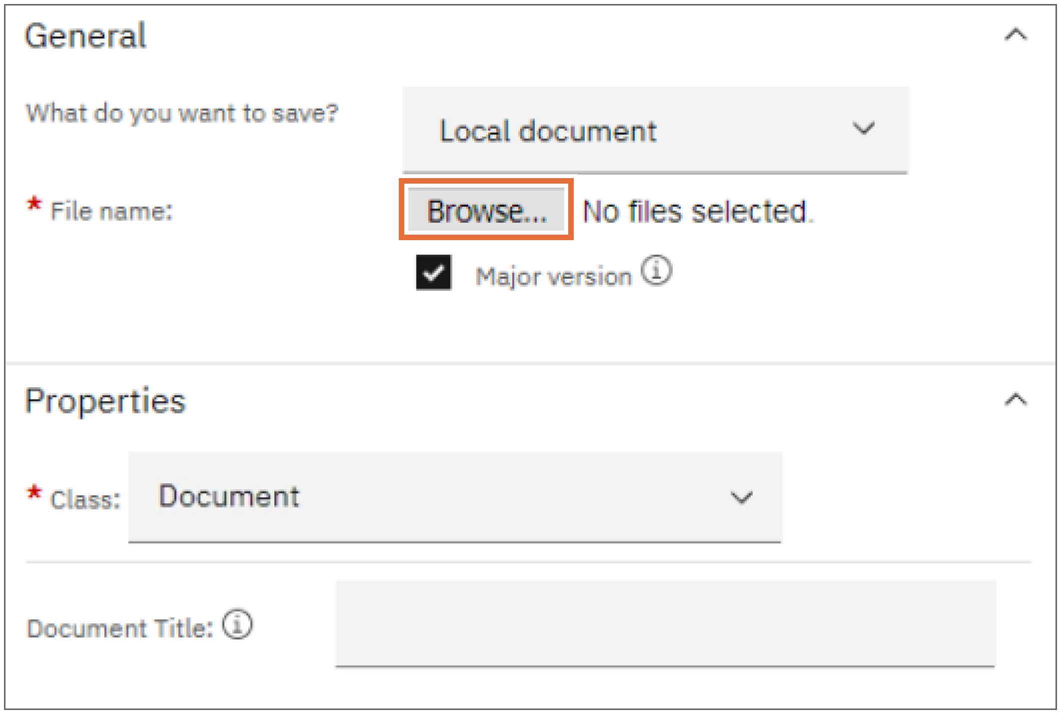



__ b. Click **Add** and select **Add document from Local System**.



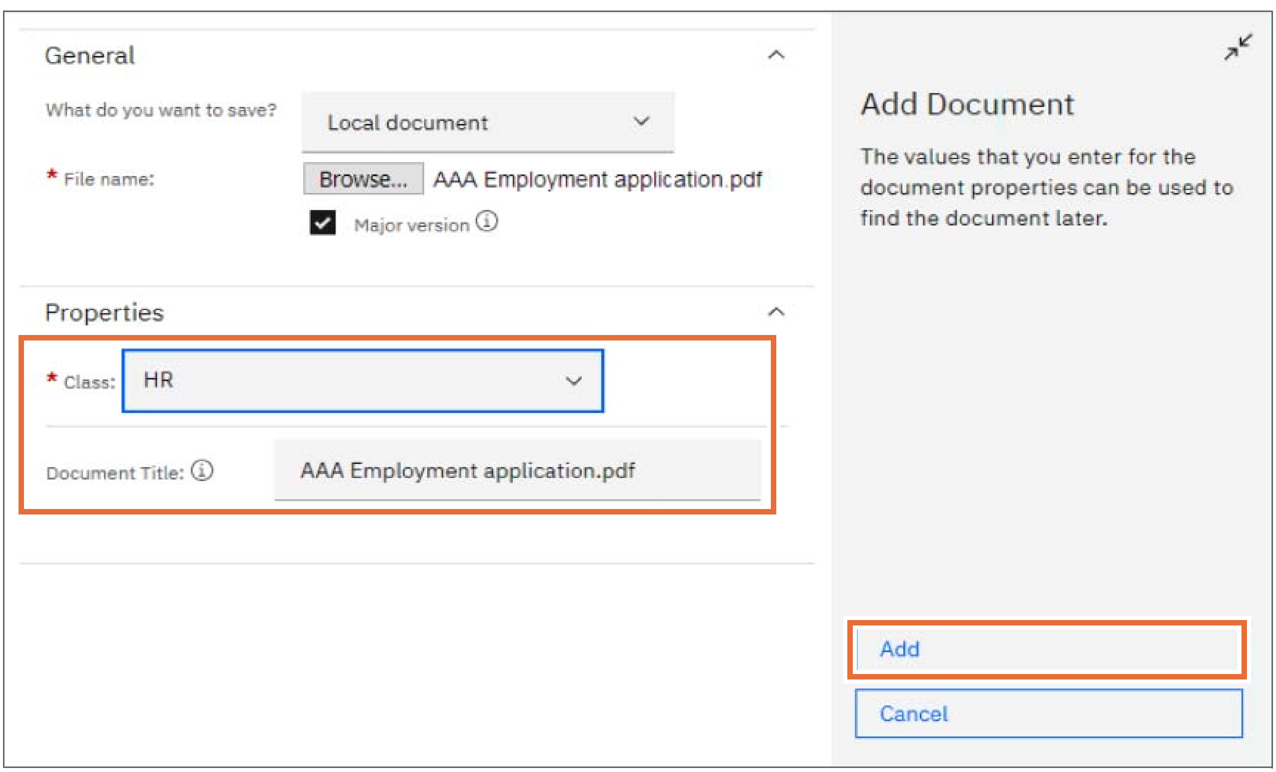
__ c. In the Add Document window, click the **Browse** button, and navigate to:

`C:\labfiles\Support files\Ex14\AAA Employment application.pdf`




__ d. Select HR for the document class by clicking the **Class** drop down list, selecting HR, then clicking **OK**.

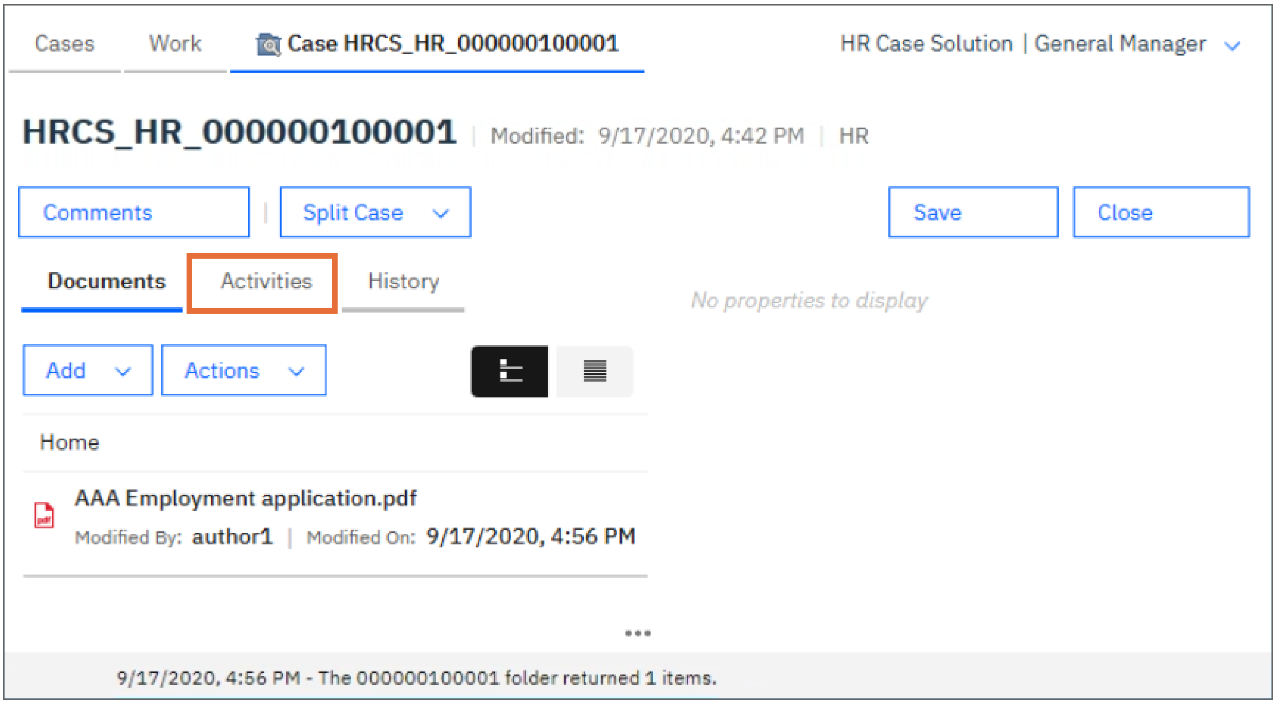
__ e. Verify that the correct class and document are selected. Then, click **Add** to add the document to the case.



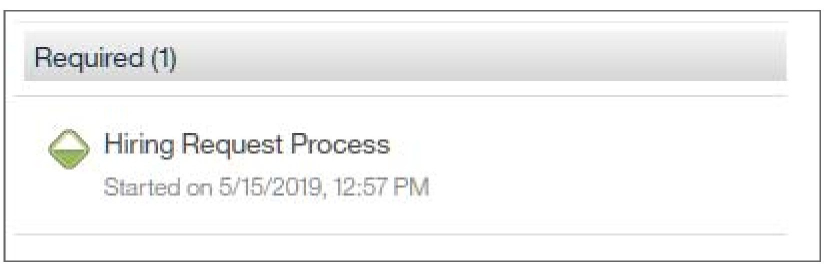
__ f. The document is listed as part of the case.

__ 3. Validate the status of the started activity.

__ a. Click the **Activities** tab.

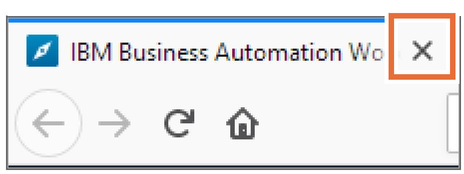



__ b. The case reports that the Hiring Request process is started.




You have added a case to the existing HR Case solution and added a document to the new case. This event started the Hiring Request Process.

__ c. Close the case window by clicking the **X** in the tab.




__ 4. Verify that the Hiring Request Process is started in Process Portal and that the document is listed in the ECM Document List view.

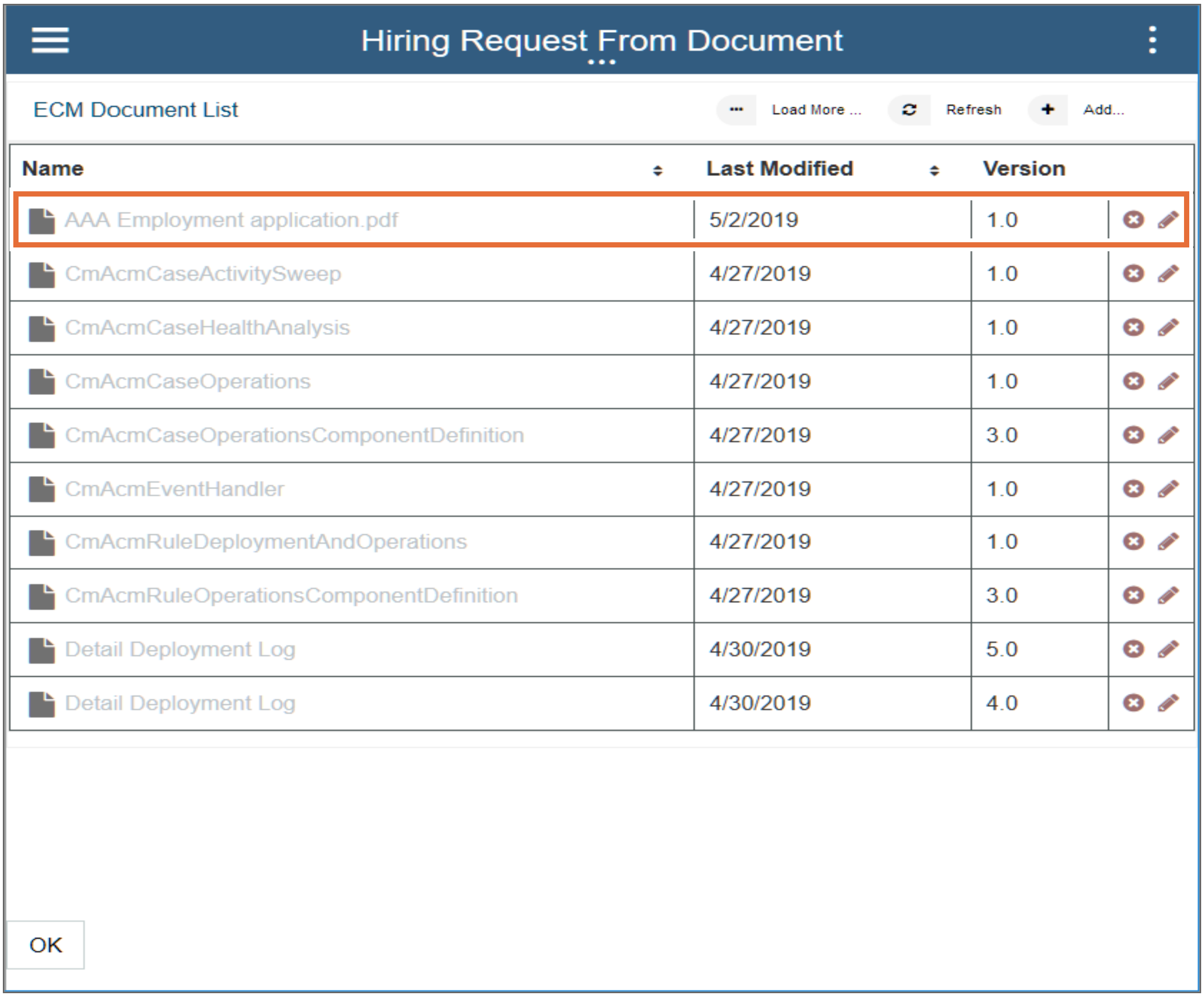
__ a. Open Process Portal.

If asked for credentials, enter the following: **Username**: `author1` **Password**: `passw0rd`

__ b. Verify the Hiring Request Process started and that the **Step: Submit Hiring Request** is in your work queue.

__ c. Click the **Hiring Request From Document** startable service.

The page might take a minute to load. The AAA Employment application.pdf document appears in the list. Your document list might look different.




**Troubleshooting**

If the AA A Employment application.pdf document does not display in the list, you can click **Load More...** to display more documents. You can then sort the documents by name.

__ d. Click **OK** to close the service.

You have completed this exercise.

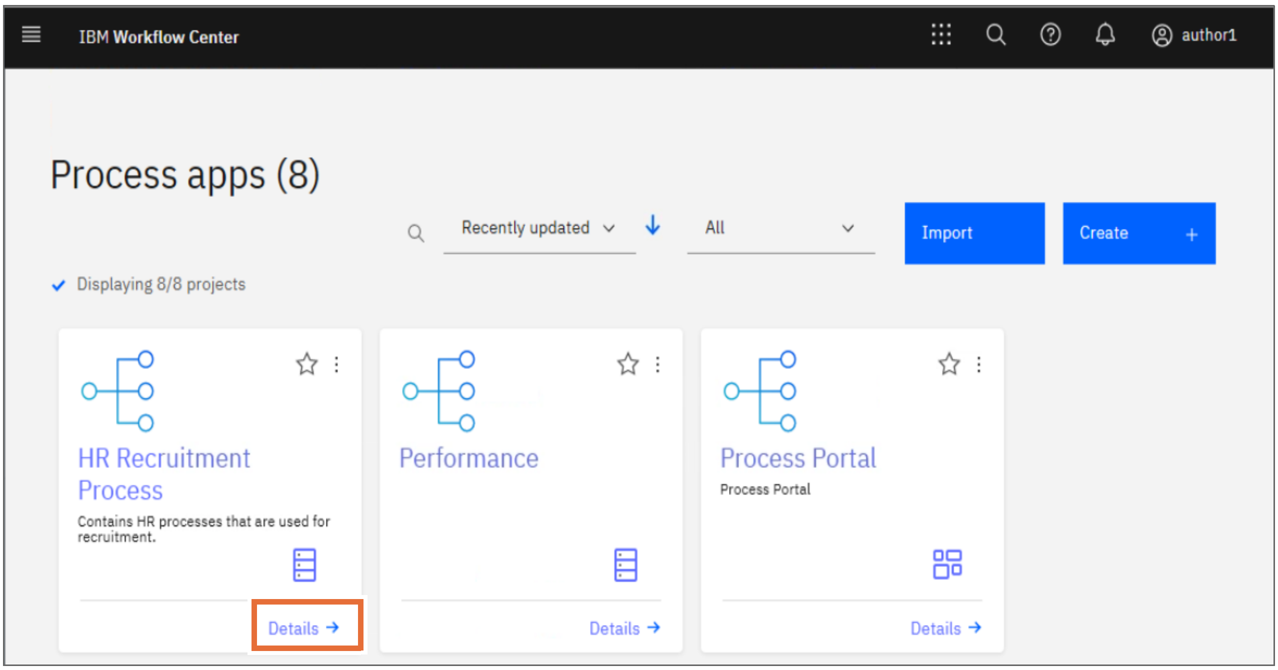
## *Part 5: Export the process application*

Now that this course is complete, you are ready to export the process application. This export saves your application as a `twx` file in a folder for future use. If you wish to keep your solution, you can export it following this procedure and upload it to a web-based file sharing program such as Box.

__ 1. Create a snapshot

__ a. Return to the Workflow Center and click the **Process apps** tile.

__ b. Click **Details** on the HR Recruitment Process tile.

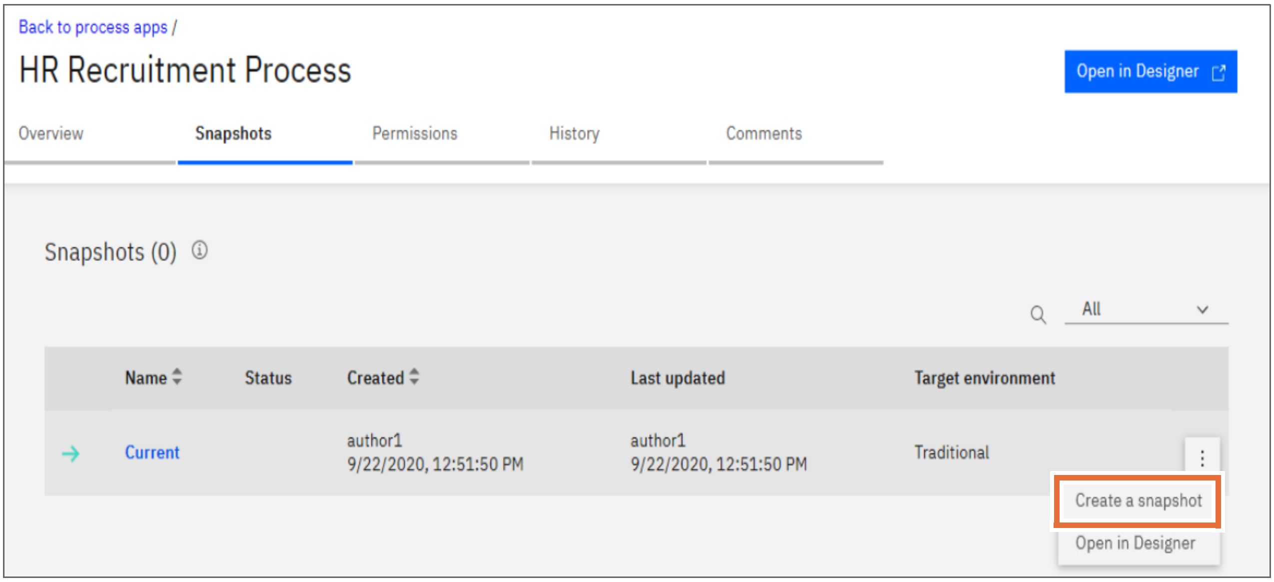



__ c. Select the **Snapshots** tab.

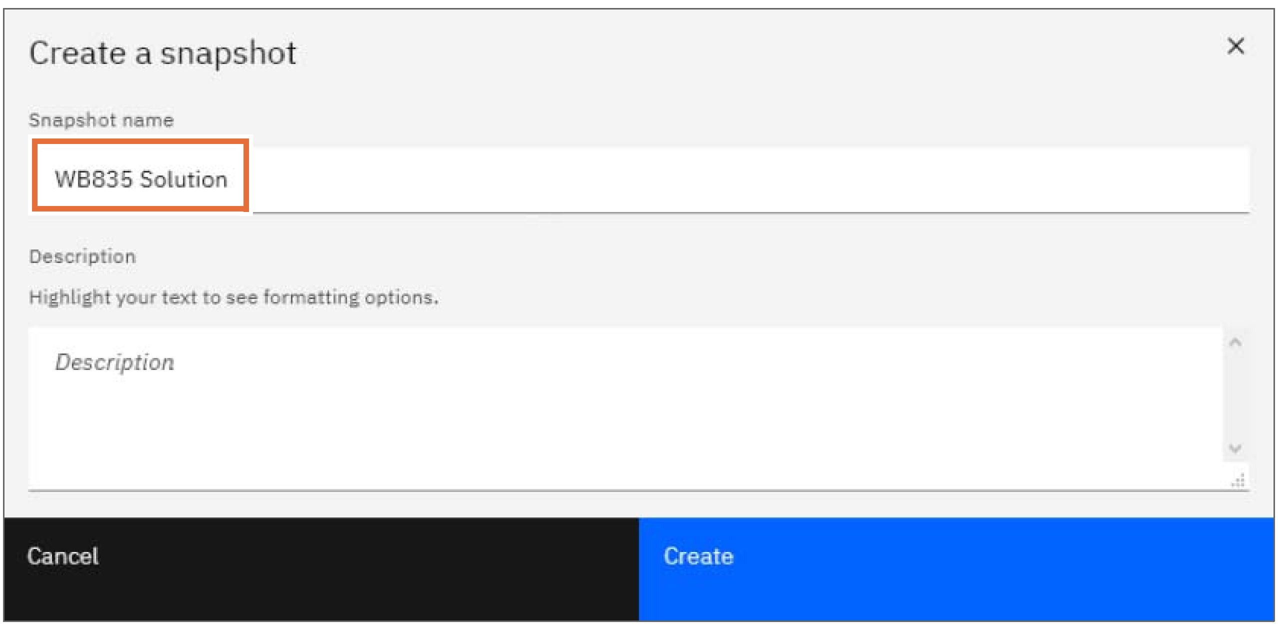
__ d. Open the list of Options by clicking the three dots on right side of the **Current** tip.

The tip is the current working version of a project.

__ e. Select **Create a snapshot**



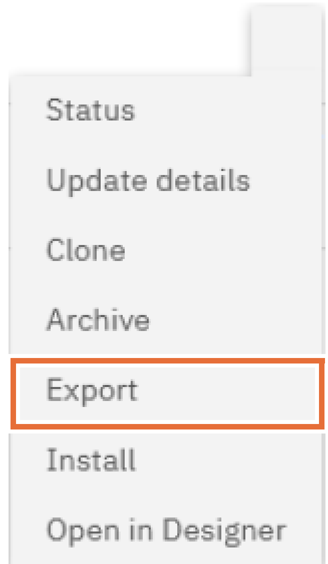

__ f. Enter `WB835 Solution` for the name.




__ g. Click **Create**.

__ 2. Export the application. Exporting the application also exports the toolkit that it depends on. You can only export snapshots.

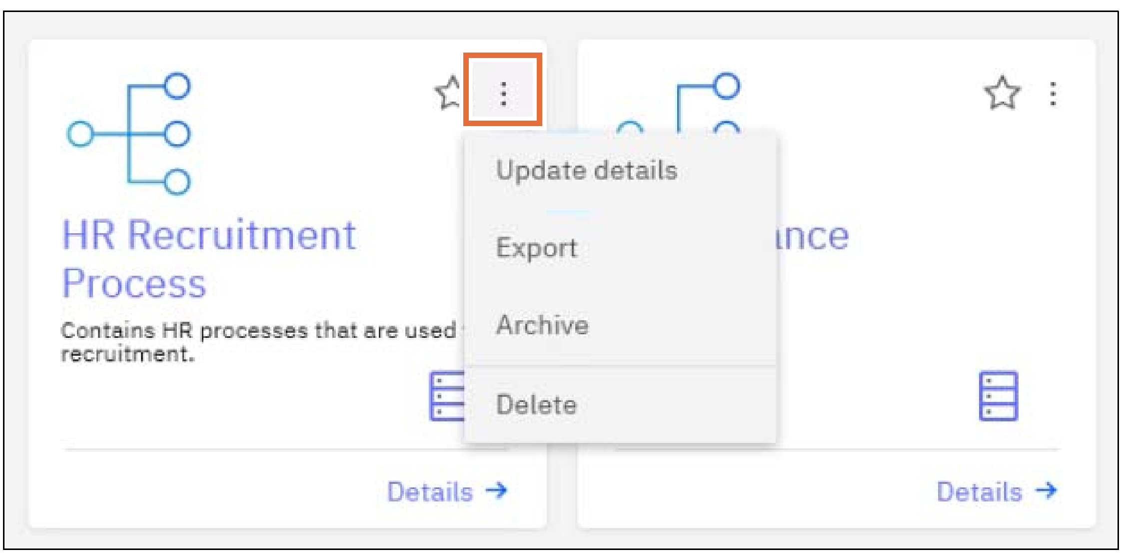
__ a. Open the list of Options for the snapshot and select **Export**




---

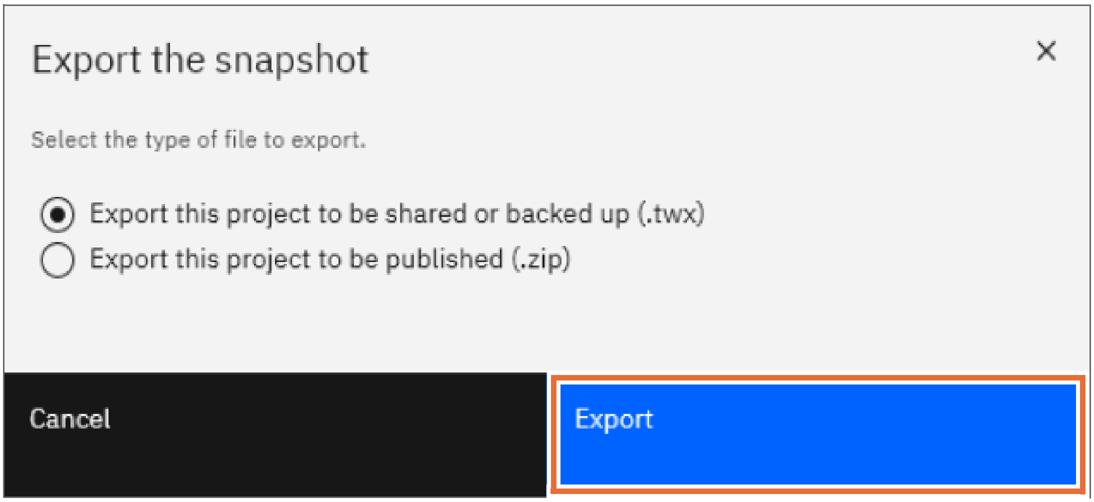
**Hint**

You can also export, archive, and delete process applications directly from the **Process apps** page by accessing the list of options on the tile.




---

__ b. In the **Export the snapshot** window, click **Export**. It may take a minute or two to generate the `.twx` file.




__ c. Click **OK** to in the dialog box to save the file. The application is saved to the default folder, which is the Downloads directory.

__ d. Verify that the application is listed in the downloads directory.

This concludes the solution export.

In the first part of the exercise, you modified an existing process to set up a document start event. You created the event subscription and verified the user-generated settings. You then created a coach and tested the event subscription. You proved that the document was sent as part of the start event output variable.

You then created an ECM server to point to the Target Object Store and reconfigured the BPM Document List view to show the new object store. As part of building a simple Case solution, you attached the Hiring Request Process from the BPM solution as an activity in the Case. Then, you tested the document viewer and integration by running the Case and adding a document to the Case. To verify the integration, you viewed the Submit Hiring Request task in the Work page. To verify the document viewer, you displayed the new document in the Document List view.

You have completed all exercises in this course.

## End of exercise

# Appendix A. Solutions

In this section, instructions for importing solution files is provided. The following solution files are located at `c:\solutions`.

| Solution file |
|---|
| HR_Recruitment_Process - Exercise_1_solution.twx |
| HR_Recruitment_Process - Exercise_2_solution.twx |
| HR_Recruitment_Process - Exercise_3_solution.twx |
| HR_Recruitment_Process - Exercise_4_solution.twx |
| HR_Recruitment_Process - Exercise_5_solution.twx |
| HR_Recruitment_Process - Exercise_6_solution.twx |
| HR_Recruitment_Process - Exercise_7_solution.twx |
| HR_Recruitment_Process - Exercise_8_solution.twx |
| HR_Recruitment_Process - Exercise_9_solution.twx |
| HR_Recruitment_Process - Exercise_10_solution.twx |
| HR_Recruitment_Process - Exercise_11_solution.twx |
| HR_Recruitment_Process - Exercise_12_solution.twx |
| HR_Recruitment_Process - Exercise_13_solution.twx |
| HR_Recruitment_Process - WB835_Solution.twx |
| HR_Case_Solution - WB835_Solution |

There is a working solution for each exercise. The solution files follow the naming convention used in Exercise 1 when exporting your process application. You can import solution files to:

- **Compare your work against a known working solution** to troubleshoot an issue in your current exercise.

  To compare your current work against a known solution, import the solution file labeled with the exercise you are working on.

- **Continue working the exercises in the course** after having an issue in a previous exercise.

  If you are unable to troubleshoot an issue, you can continue the next exercise by importing the solution for the current exercise.

- **Skip to an exercise of interest**.

  If you want to skip ahead to an exercise, you can import the solution file of the exercise previous to that exercise. For instance, if you want to skip to Exercise 10, import the solution file: `HR_Recruitment_Process - Exercise_9_solution.twx`. After importing the file, you can follow the instructions for exercise 10.

Be sure to archive, export, and delete your current working solution beforehand to prevent file conflicts. You can reimport your solution after archiving and deleting the imported solution file. The remaining sections provide instructions for these three scenarios.

## Part 1: Export current process application

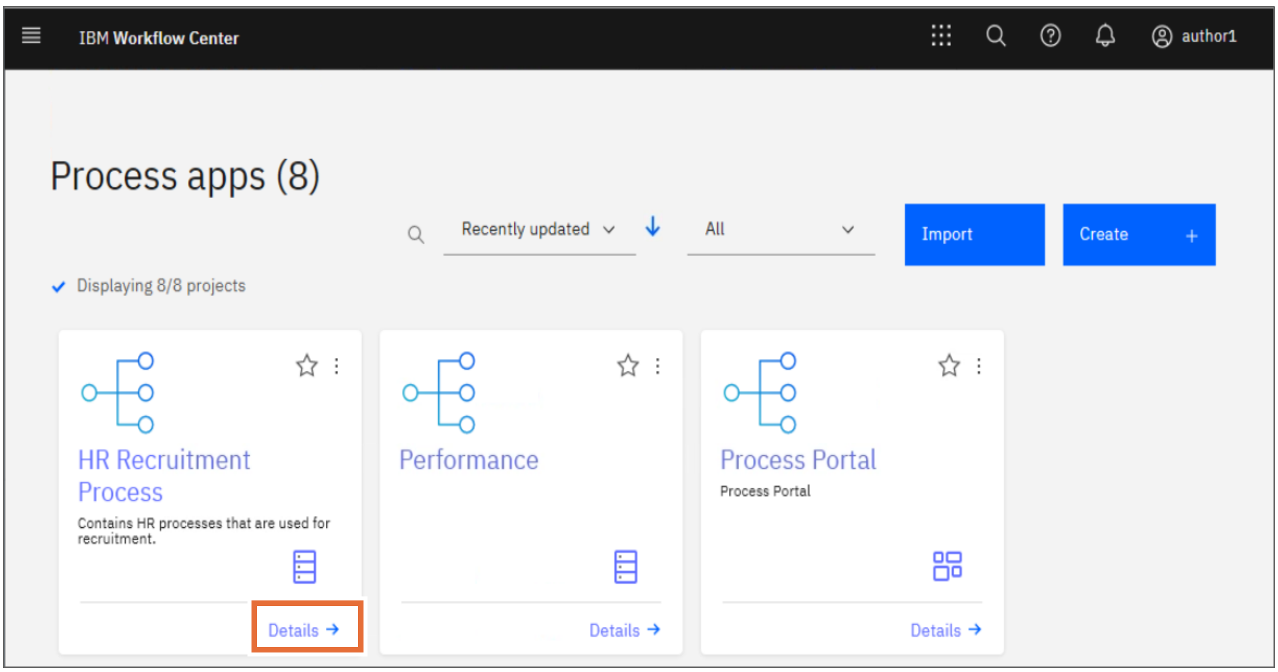
Before performing any of the subsequent sections, you need to export and delete your current process application to prevent file conflicts. You performed this action in exercise 1. The same steps are here for your reference. Be sure to perform these before importing another solution file.

__ 1. Create a snapshot

Only process applications with snapshots can be exported. If you have an existing snapshot you can export, you can skip creating the snapshot.

__ a. Access the Workflow Center and click the **Process apps** tile.

__ b. Click **Details** on the HR Recruitment Process tile.

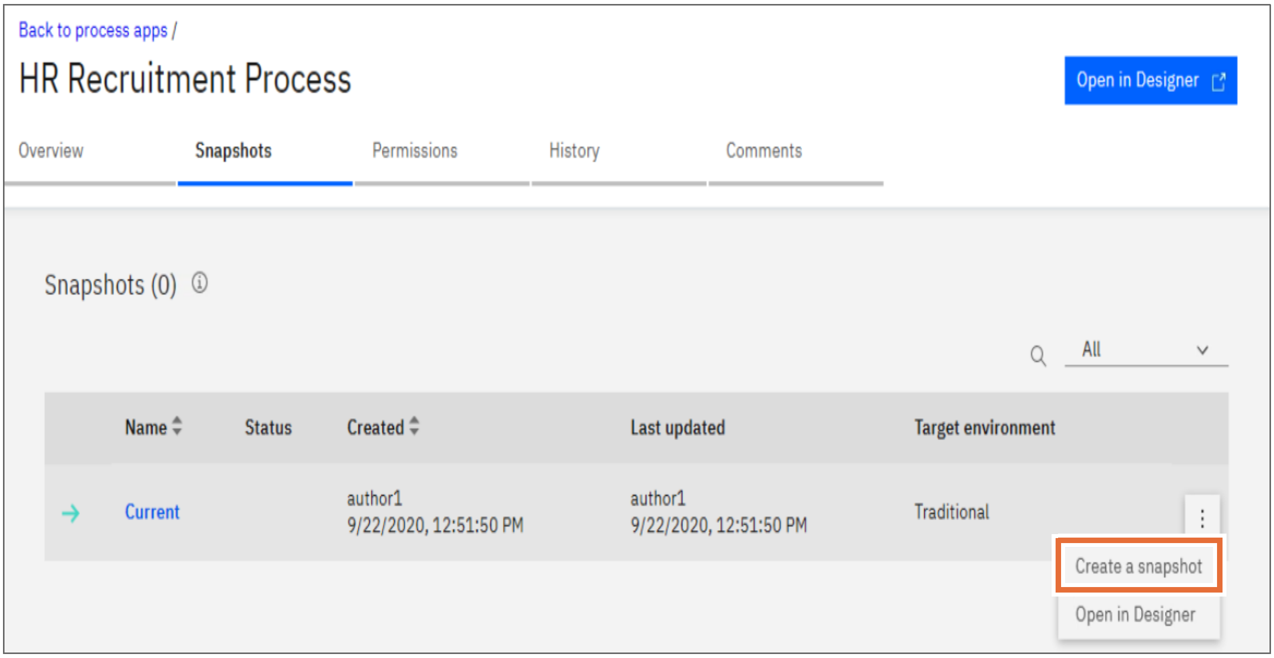



__ c. Select the **Snapshots** tab.

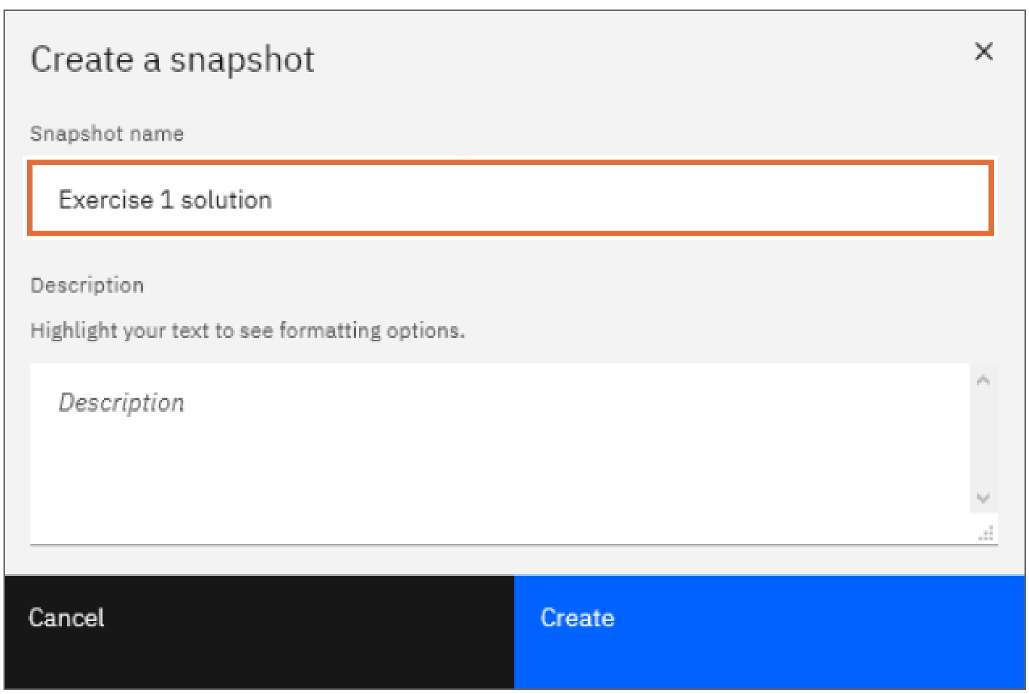
__ d. Open the list of Options by clicking the three dots on right side of the **Current** tip.

The tip is the current working version of a project.

__ e. Select **Create a snapshot**



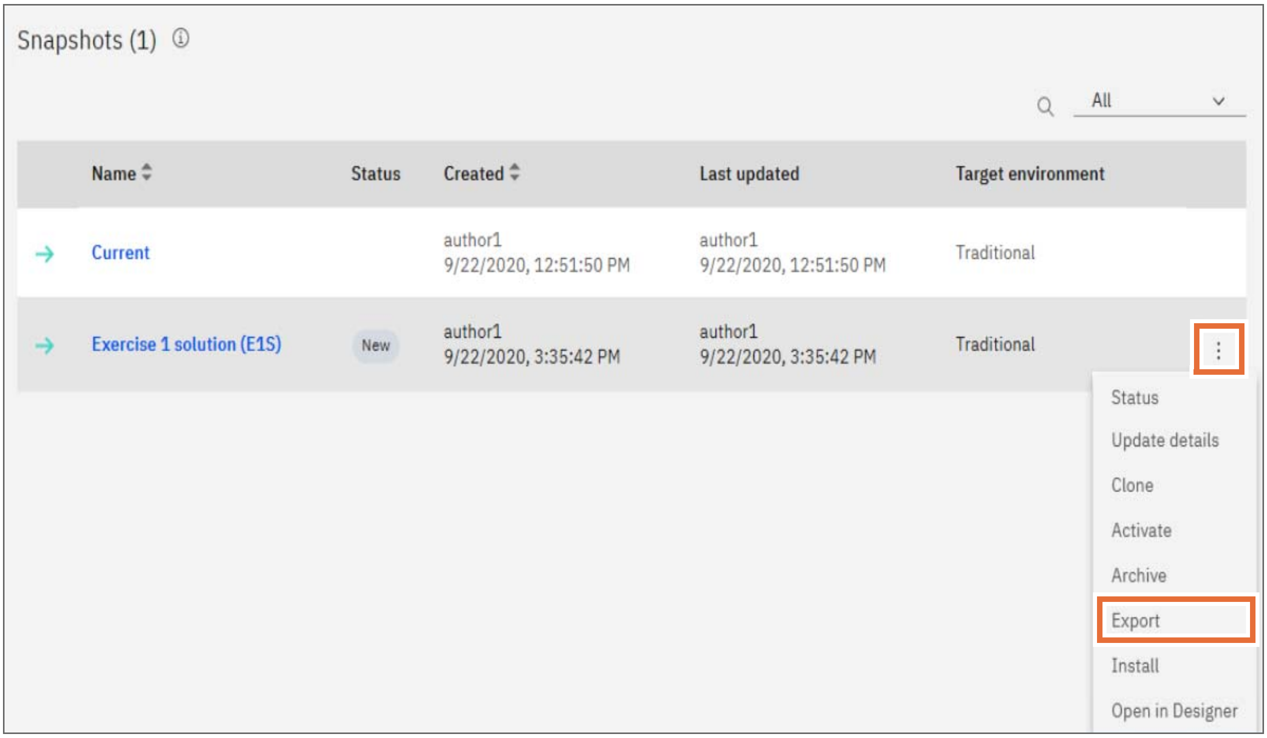

__ f. Enter the name of the exercise you are working on such as `Exercise 1 solution` for the name.



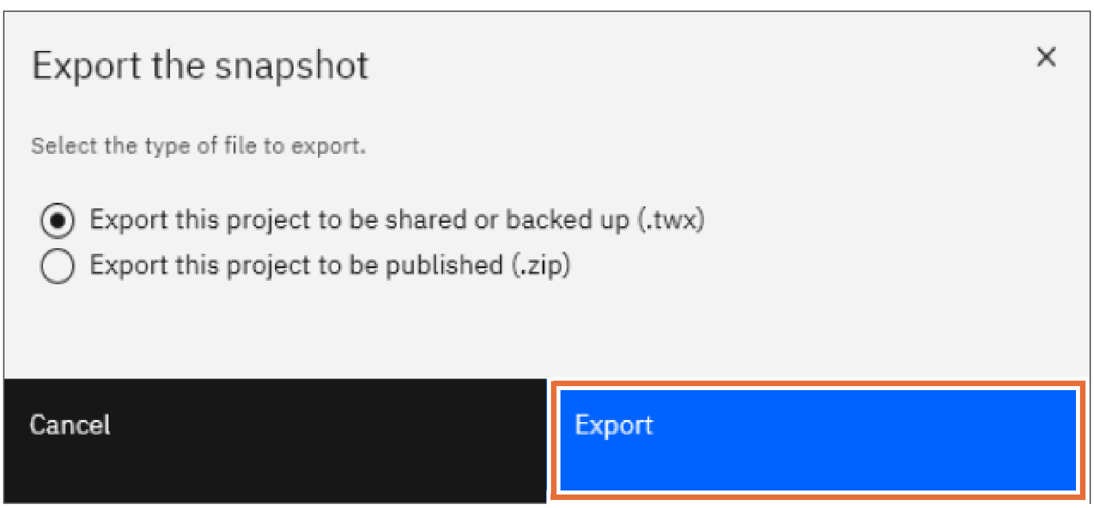

__ g. Click **Create**.

__ 2. Export the application. Exporting the application also exports the toolkit that it depends on. You can only export snapshots.

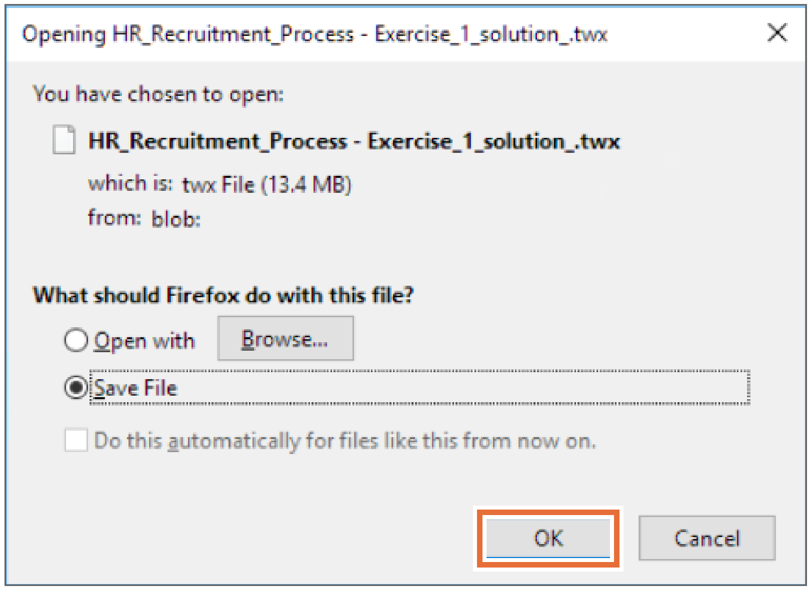
__ a. Open the list of Options for the snapshot and select **Export**




__ b. In the **Export the snapshot** window, click **Export**. It may take a minute or two to generate the `.twx` file.



__ c. Click **OK** to save the file. The application is saved to the default folder, which is the Downloads directory.




__ d. Verify that the application is listed in the downloads directory.

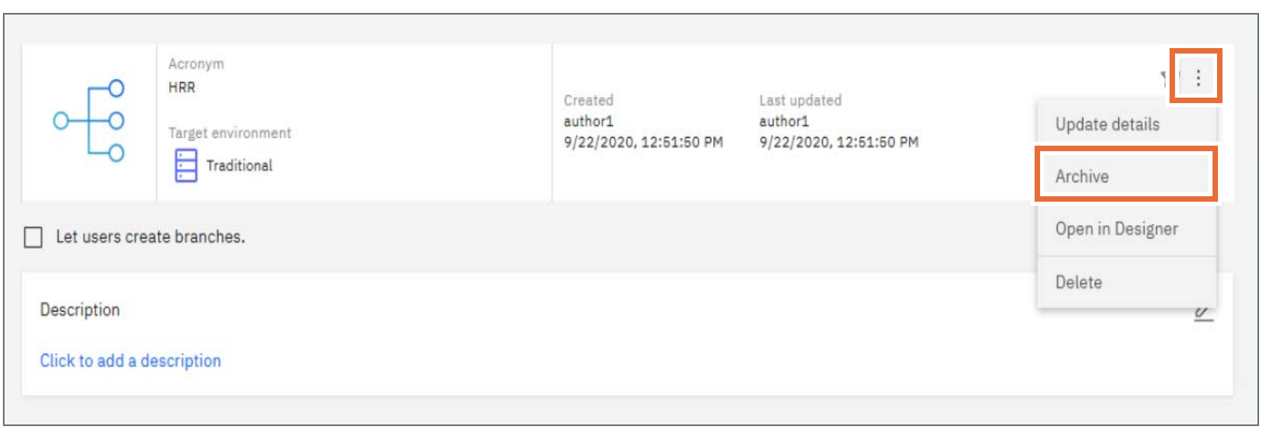
__ 3. Archive the process application and delete it. Although you don't have to archive an application before deleting it, it is a good practice. Archiving an application also archives the toolkit that it depends on.

**Important**

Don't archive a toolkit that other projects depend on nor use archived toolkits when creating toolkit dependencies in Process Designer. Archived items are not part of the active library.

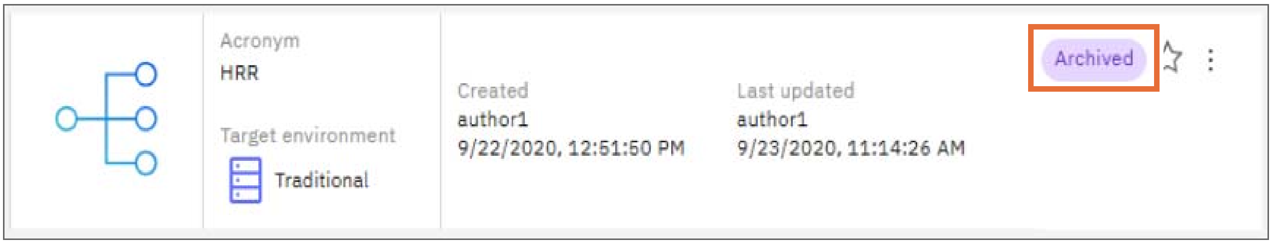
__ a. Return to the **Overview** tab.

__ b. Open the list of Options for the application and select **Archive**.

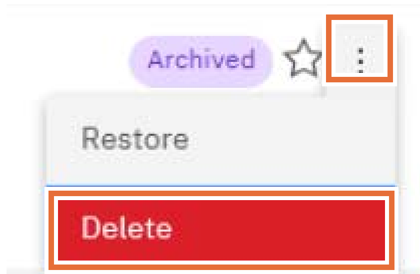



__ c. In the window that opens, click **Archive** to confirm the action. Clicking **Archive** does not delete the application; it removes it from the available **Process Apps** and makes it ready for deletion.

An **Archived** tag appears in the application Overview tile.




__ d. Open the list of Options for the application and select **Delete**.




__ e. In the window that opens, click **Delete** to confirm the deletion. The process application is now deleted.

You are returned to the list of Process apps.

__ f. Verify that **HR Recruitment Process** is no longer listed.

You can now import an existing solution file.

## *Part 2: Perform troubleshooting*

- **Compare your work against a known working solution**

  After exporting your current working solution, archiving, and then deleting it from the repository, you can import an existing working solution.

  If you are having issues in your current exercise, import the solution file corresponding to that exercise as listed in the table. Review the working solution to determine the cause of your issue.

  After you determine the cause of the issue, you can export and delete the solution and re-import your working solution to apply the fix.

- **Continue working the exercises in the course**

  After exporting your current working solution, archiving, and then deleting it from the repository, you can import an existing working solution.

  After importing the working solution for the exercise you cannot troubleshoot, you can continue with the next exercise using the imported solution file. You can continue to develop using this file for the remainder of the course.

- **Skip to an exercise of interest**

  If you are just starting the course and would like to skip to an exercise. You can import the solution file for the exercise prior to the one you would like to perform. You can continue to develop using this file for the remainder of the course. For instance, if you want to skip to Exercise 10, import the solution file: `HR_Recruitment_Process - Exercise_9_solution.twx`. After importing the file, you can follow the instructions for exercise 10.

## *Part 3: Import a solution file*

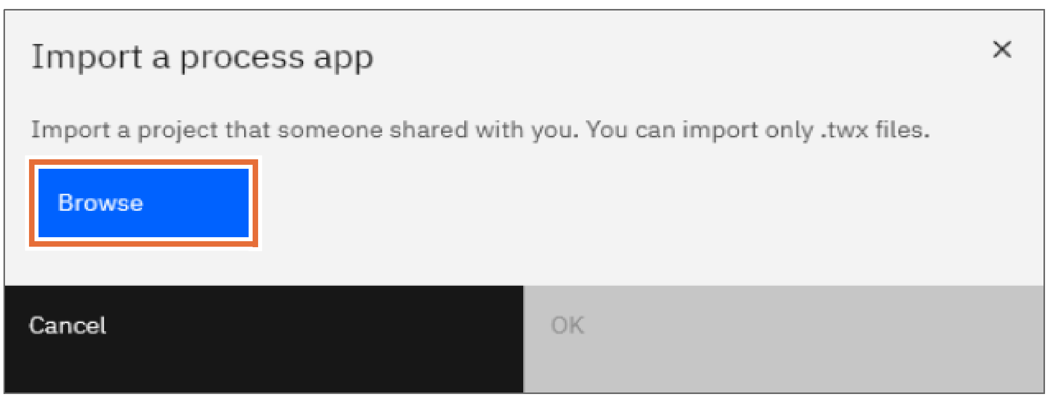
In this section, you import a process application solution for a specific exercise for troubleshooting purposes. This import also helps in making this course modular as you can then work on any exercise in any order. Although, for a good learning experience, it is highly recommended to do the exercises in the order they are provided.

__ 1. Import a process application.

__ a. Access the Workflow Center and click the **Process apps** tile.

__ b. On the list of Process apps, click **Import**.

__ c. Click **Browse**.



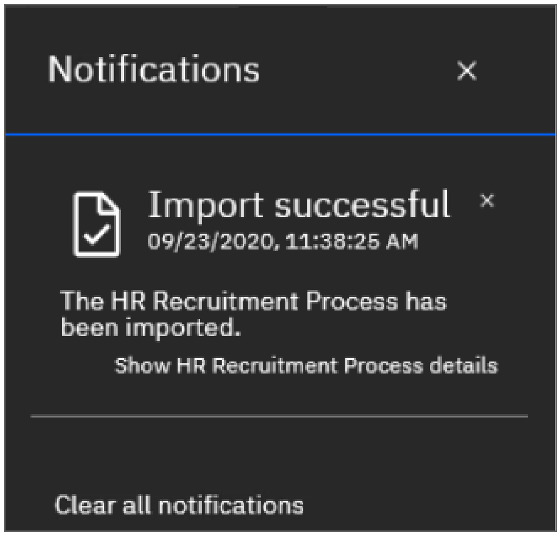

__ d. Navigate to the `c:\solutions` folder and select the file for the exercise you would like to import:

`HR_Recruitment_Process - Exercise_1_solution.twx`

__ e. Click **Open**.

__ f. Click **OK**.

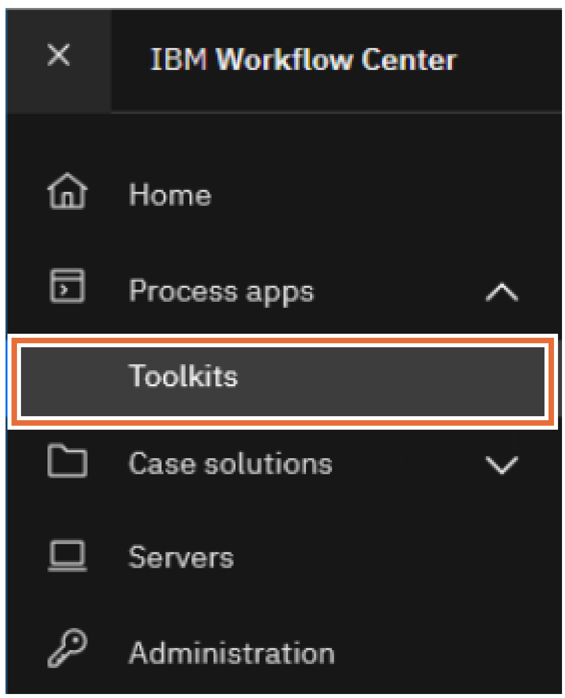
__ g. The Notifications window displays the message



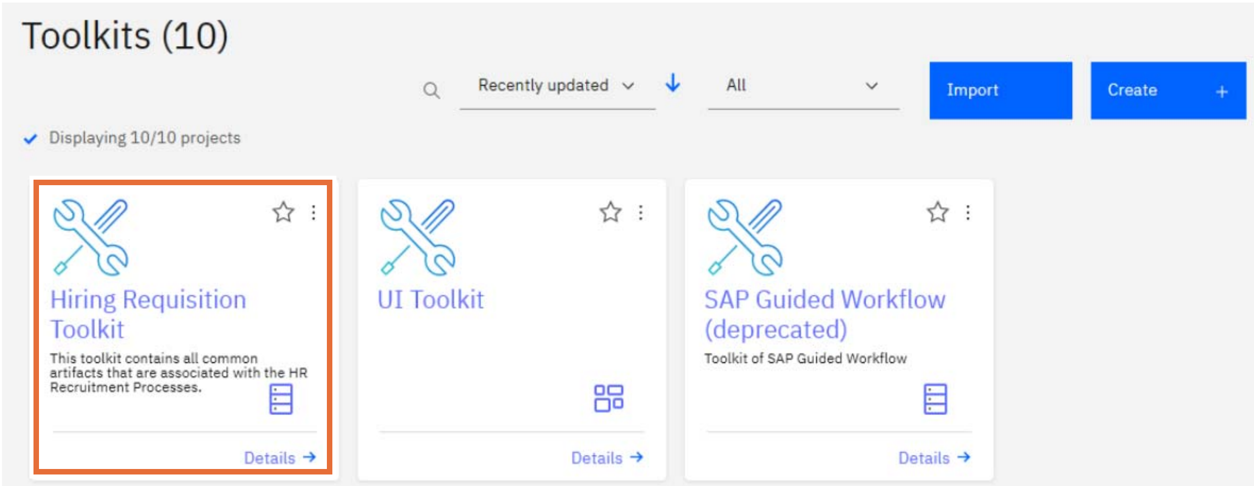

__ h. Close the **Notifications** window by clicking the **X** in the upper right corner.

__ i. Verify that the **HR Recruitment Process** application is listed.

__ j. Open the list of Toolkits.



__ k. If the solution file contains a Toolkit, verify the Toolkit is listed.




__ l. Click **Details** on the **Hiring Requisition Toolkit** tile.

__ m. Click **Let users update the toolkit** to allow changes to be made.


Toolkit updates can be managed using this option.

## End of Appendix

IBM Training