# User management steps

Define the user management steps

- Start
- Screen
- Detect duplicated user
- Apply changes
- Custom
- Mail
- Fork
- Join
- End

# Start

# Definition

That is the first step of the workflow. At that step, you could define the fields you want to show when the end users will go to make a request.

# Steps Tabs

## Task details

This process type does not have task details for the start step.

# Fields

In this tab, you could choose what fields the process form will show to the end users. You can choose these fields from all identity attributes, and from the attributes defined for the workflow on the Attributes Tab.

By default, all the identity attributes will be shown, and an additional field called **Action**. You can choose the fields you want to show when the end-users, add new fields, and delete the fields that do not need to generate a task. Also, you can sort the fields, you only need to drag and drop on the Order column.

The Action field is a droplist that will allow end-users to select one of the different options to perform. The available actions, defined by default on the Attributes tab:

- **Add user**: action uses to generate a task to create a new identity.
- **Enable user**: action uses to create a task to enable an identity who is disabled.
- **Modify user**: action uses to create a task to modify identity attributes.
- **Disable user**: action uses to create a task to disable identity.

To enable, modify or disable an identity, you need to add a field with the name **userSelector**, defined on the Attributes tab. That field will be available, to end-users,  to select an existing identity when selecting one of that options. When you select an identity, Soffid will show all the attributes defined on the form to the end user.

For each field, you may indicate if it is a readOnly field, and you may add a Validation script and Visibility script. The validation script allows you to define rules, the field has to comply with these rules. The visibility script allows you to define the rules to show or hide a field.

## Validation examples

```
if (value == null || value.equals(""))
  throw new Exception("The userName is mandatory");
else
  return true;
```

It is also allowed in the following manner:

```
if (value == null || value.equals(""))
  return ("The userName is mandatory");
else
  return true;
```

Validate that a certain field is not repeated:

```
userList = serviceLocator.getUserService().findUserByJsonQuery("attributes.field_XX eq \"" + value +"\"");
if (!userList.isEmpty() {
  return "the field field_XX is associated to another user";
}
return true;
```

## Visibility example

```
user = serviceLocator.getUserService().getCurrentUser();
if ("admin".equals(user.userName))
  return false;
```

## SCIM filter example

```
userType eq "E"
```

# Triggers

On the trigger tab, you could define different triggers using custom scripts. Those triggers will be launched with the events you will define.

- **onLoad**: you can use that trigger to perform some actions before the execution of the step.
- **on PrepareTransition**: you can use that trigger to perform some actions after the execution of the step and before starting a transition to another step.
- **onChange**: you can use that trigger to perform some actions when the value of the attribute is changed. You could choose the field from a list.

## Example

1. Calculate the email when firstName or lastName changes and depending on the userType:

```
firstName   = (inputFields.get("firstName")!=null) ? inputFields.get("firstName").value : null;
lastName    = (inputFields.get("lastName")!=null) ? inputFields.get("lastName").value : null;
userType    = (inputFields.get("userType")!=null) ? inputFields.get("userType").value : null;

if (firstName!=null && !firstName.trim().isEmpty() &&
    lastName!=null && !lastName.trim().isEmpty() &&
    userType!=null && !userType.trim().isEmpty()) {

  emailAddress = firstName + "." + lastName;
   if ("E".equals(userType)) {
    emailAddress = emailAddress + ".ext@soffid.com";
   } else {
     emailAddress = emailAddress + "@soffid.com";
   }
  inputFields.get("emailAddress").value = emailAddress;


}
```

> You can find more information about StandardUserWindow.java on Github.

2. Load the user data into the form.

```
user = serviceLocator.getUserService().getCurrentUser();
task.getVariables().put("action", "M");
task.getVariables().put("userSelector", user.userName);
workflowWindow.fetchUserAttributes()
```

# Incoming transitions

This process type does not have task details for the start step.

# Outgoing transitions

The Outcoming transition tab displays the next steps where the flow can go from the current step. When you create a process from a template or from scratch default outcoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: current step.
- **Incoming transition**: name of the transition.
- **To**: the next step, where the flow goes.
- **Action**: allows creating a custom script to perform specific actions.

When you create an outcoming transition, Soffid creates the proper incoming transition.

## Example

Check if there are any similar identities:

```
firstName = executionContext.getVariable("firstName");
birthDate = executionContext.getVariable("birthDate");

df = new java.text.SimpleDateFormat("yyyy-MM-dd");
query = "firstName co \""+firstName+"\" and attributes.birthDate sw \""+df.format(birthDate)+"\"";

users = serviceLocator.getUserService().findUserByJsonQuery(query);
if ( !users.isEmpty()) {
  throw new es.caib.bpm.toolkit.exception.UserWorkflowException("Your identity is probably registered. Please, contact your system administrator");
}
```

# Screen

# Description

This step is used to define the custom form that will be used by the users who have to approve or to reject the generated task. To configure that step will be necessary to determine the fields that will be show to the users, and the actions that these users could perform.

# Steps Tabs

## Task details

In this tab you could configure next parameters:

- **Task name**: identified name for the task that will be created when the workflow is requested.
- **Actor(s) expression**: allows you to write an expression to identify the actor depending on the requested role. One can use EL expressions (*) based on role and application attributes. For instance: SOFFID_MANAGER/${primaryGroup}
- **Assignment script**: alternatively, allows you to write a Beanshell script to return the actor depending on the process variables. For instance: return primaryGroup.attributes{"owner"};
- **Approve from email**: checked it to allow you to send a mail for approval of the task.

## Fields

In this tab, you could choose what fields the process form will show to the end users. You can choose these fields from all identity attributes, and from the attributes defined for the workflow on the Attributes Tab. By default, all the identity attributes will be shown. You can choose the fields you want to show, add new fields, and delete the fields that do not need to generate a task. Also, you can sort the fields, you only need to drag and drop on the Order column.

For each field, you may indicate if it is a readOnly field, and you may add a Validation script and Visibility script. The validation script allows you to define rules, the field has to comply with these rules. The visibility script allows you to define the rules to show or hide a field.

## Example

```
if (value == null || value.equals(""))
  return ("The NIF is mandatory");
else
  return true;
```

# Trigger

On the trigger tab, you could define different triggers using custom scripts. Those triggers will be launched with the events you will define.

- **onLoad**: you can use that trigger to perform some actions before the execution of the step.
- **on PrepareTransition**: you can use that trigger to perform some actions after the execution of the step and before starting a transition to another step.
- **onChange**: you can use that trigger to perform some actions when the value of the attribute is changed. You could choose the field from a list.

## Example

1. How to set a value depending on a variable (onLoad).

```
userType = task.getVariables().get("userType");
if ("I".equals(userType))  {
  task.getVariables().put("country", "ES");
}
```

2. Validate a field value (onChange)

```
firstName   = (inputFields.get("firstName")!=null) ? inputFields.get("firstName").value : null;
lastName    = (inputFields.get("lastName")!=null) ? inputFields.get("lastName").value : null;
country    = (inputFields.get("country")!=null) ? inputFields.get("country").value : null;

if (firstName!=null && !firstName.trim().isEmpty() &&
    lastName!=null && !lastName.trim().isEmpty() &&
     country!=null && !country.trim().isEmpty()) {

emailAddress = firstName + "." + lastName;
 if ("ES".equals(country)) {
  emailAddress = emailAddress + ".@soffid.es";
```

```
  } else {
    emailAddress = emailAddress + "@soffid.com";
  }
 inputFields.get("emailAddress").value = emailAddress;
}
```

# Incoming transitions

The Incoming transitions tab displays the previous steps where the flow comes from. When you create a process from a template or from scratch default incoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: the previous step, where the flow comes. Allows you to select where the workflow comes from.
- **Incoming transition**: brief name to identify the transition. That is the name of the action the form will show to the final user.
- **To**: current step.
- **Action**: allows creating a custom script to perform specific actions.

When you create an incoming transition, Soffid creates the proper outcoming transition.

## Example

The incoming script action is the same outgoing script action of the previous step.

```
selector = executionContext.getVariable("userSelector");
user = serviceLocator.getUserService().findUserByUserName(selector);
executionContext.setVariable("testName", user.firstName);
executionContext.setVariable("testOperation", "CHECK");
```

# Outgoing transitions

The Outcoming transition tab displays the next steps where the flow can go from the current step. When you create a process from a template or from scratch default outcoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: current step.
- **Incoming transition**: name of the transition.
- **To**: the next step, where the flow goes.
- **Action**: allows creating a custom script to perform specific actions.

When you create an outcoming transition, Soffid creates the proper incoming transition.

## Example

Update custom attributes defined on metadata

```
userName = executionContext.getVariable("userName");
attributes = serviceLocator.getUserService().findUserAttributes(userName);

newAttributes = new HashMap();
newAttributes.put("country", "FR");

language = attributes.get("language");
if (language == null) {
  language = new LinkedList();
}
language.add("Spanish");
language.add("German");

newAttributes.put ("language", language);

serviceLocator.getUserService().updateUserAttributes(userName, newAttributes);
```

---

** https://es.wikipedia.org/wiki/Expression_Language*

# Detect duplicated user

# Definition

That step is used to define the proper rules to determine the potential conflicts between the identity for who is the request, and the Soffid existing identities. Whit that definition, Soffid will find the potential conflicts, and the end-user could select the best option to solve those (merge or create a new one).

# Steps Tabs

## Tasks details

- **Task name**: identified name for the task that will be created. For instance: Check duplicates for #{firstName} #{lastName}
- **Actor(s) expression**: write an expression to identify the actor depending on the requested role. One can use EL expressions based on role and application attributes. For instance: SOFFID_MANAGER/${primaryGroup}
- **Assignment script**: alternatively, write a Beanshell script to return the actor depending on the process variables. For instance: return primaryGroup.attributes{"owner"};
- **Weight threshold**: in the tab "User queries", you can define many different queries to search for similar users, and each query has a weight. If a user is found in one or more queries, the weight of each one of these queries are added. If the total weight is equal to or greater than the current threshold, the user is considered a user match.

## Fields

In this tab, you could choose what fields the process form will show to the end users. You can choose these fields from all identity attributes, and from the attributes defined for the workflow on the Attributes Tab. By default, all the identity attributes will be shown. You can choose the fields you want to show, add new fields, and delete the fields that do not need to generate a task. Also, you can sort the fields, you only need to drag and drop on the Order column.

For each field, you may indicate if it is a readOnly field, and you may add a Validation script and Visibility script. The validation script allows you to define rules, the field has to comply with these

rules. The visibility script allows you to define the rules to show or hide a field.

# User queries

This tab is only available when one of the below Step types is **Detect duplicated user**.

User queries allow you to customize a SCIM or Text query to detect duplicated users. You may define a weight for each query. If a user is found in one or more queries, the weight of each one of these queries are added. If the total weight is equal to or greater than the current weight threshold (defined on the Task details tab), the user is considered a user match.

## Examples

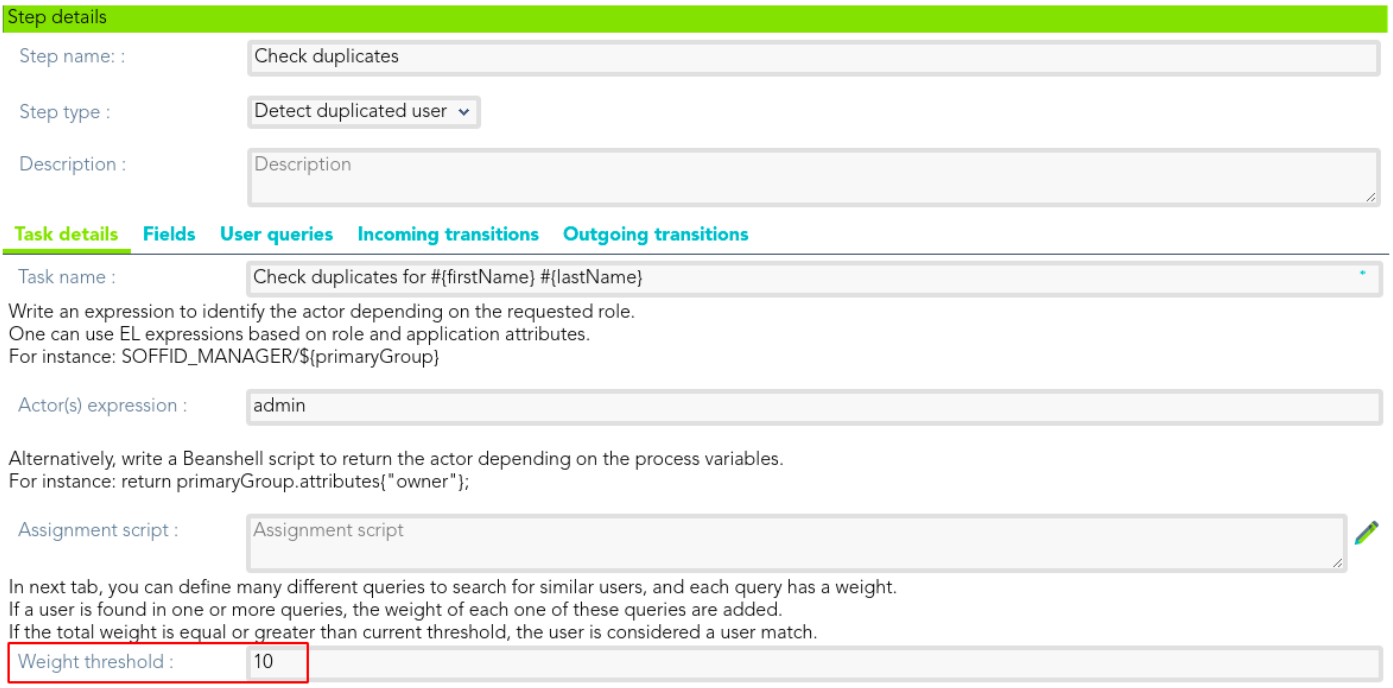
Text Query

```
${lastName}
```

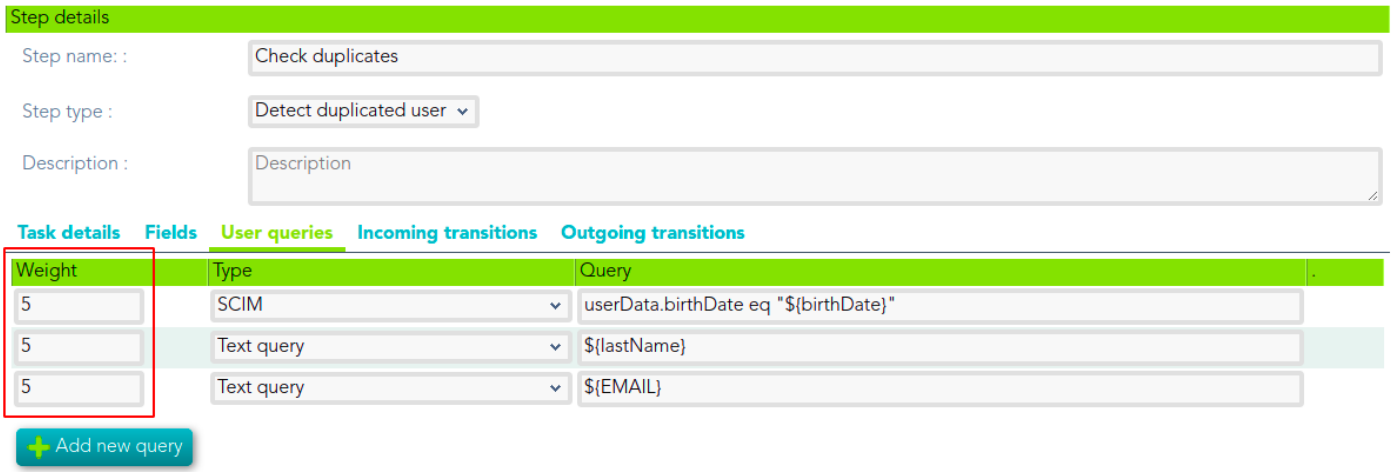
SCIM Query

```
attributes.birthDate eq "${birthDate}"
```

Define the weight threshold on the Task detail tab




Define the weight for each query on the User query tab: A user is considered duplicated when at least two queries are true.




# Incoming transitions

The Incoming transitions tab displays the previous steps where the flow comes from. When you create a process from a template or from scratch default incoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: the previous step, where the flow comes. Allows you to select where the workflow comes from.
- **Incoming transition**: brief name to identify the transition. That is the name of the action the form will show to the final user.
- **To**: current step.
- **Action**: allows creating a custom script to perform specific actions.

When you create an incoming transition, Soffid creates the proper outcoming transition.

## Example

The incoming script action is the same outgoing script action of the previous step.

```
selector = executionContext.getVariable("userSelector");
user = serviceLocator.getUserService().findUserByUserName(selector);
executionContext.setVariable("testName", user.firstName);
executionContext.setVariable("testOperation", "CHECK");
```

# Outgoing transitions

The Outcoming transition tab displays the next steps where the flow can go from the current step. When you create a process from a template or from scratch default outcoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: current step.
- **Incoming transition**: name of the transition.
- **To**: the next step, where the flow goes.
- **Action**: allows creating a custom script to perform specific actions.

When you create an outcoming transition, Soffid creates the proper incoming transition.

## Example

Add comments to the task:

```
executionContext.getToken().addComment("Automatic comments.....");
```

# Apply changes

# Definition

This step is used to apply the identity changes to the Soffid repository.

# Steps Tabs

## Task details

- **Apply users changes**: check it (select the Yes option) to make changes to users on the Soffid repository.
- **Apply entitlements**: check it (select the Yes option) to make changes to permissions on the Soffid repository.

## Incoming transitions

The Incoming transitions tab displays the previous steps where the flow comes from. When you create a process from a template or from scratch default incoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: the previous step, where the flow comes. Allows you to select where the workflow comes from.
- **Incoming transition**: brief name to identify the transition. That is the name of the action the form will show to the final user.
- **To**: current step.
- **Action**: allows creating a custom script to perform specific actions.

When you create an incoming transition, Soffid creates the proper outcoming transition.

### Example

```
requester = executionContext.getVariable("requester");
userR = serviceLocator.getUserService().findUserByUserName(requester);
```

```
if (userR.primaryGroup.equals("admingroup")) {
//TO-DO
} else {
//TO-DO
}
```

# Outgoing transitions

The Outcoming transition tab displays the next steps where the flow can go from the current step. When you create a process from a template or from scratch default outcoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: current step.
- **Incoming transition**: name of the transition.
- **To**: the next step, where the flow goes.
- **Action**: allows creating a custom script to perform specific actions.

When you create an outcoming transition, Soffid creates the proper incoming transition.

## Example

```
userName = executionContext.getVariable("userName");
user = serviceLocator.getUserService().findUserByUserName(userName);
country = user.getAttributes().get("country");
groups = serviceLocator.getGroupService().findUsersGroupByUserName(userName);

if (country.equals("ES")) {
//TO-DO
}
```

# Custom

# Definition

This step is used to define a custom script that will be executed

# Steps Tabs

# Task details

All the process types have the same Task details for the Custom step:

- **Script**: allows you to define a Script this step allows you to add a script to be executed.

## Example

```
comments = executionContext.getToken().getComments();
selector = executionContext.getVariable("userSelector");
if (selector == null || selector.equals("")) {
     return ("The userName is mandatory");
}
user = serviceLocator.getUserService().findUserByUserName(selector);
if (user != null) {
  subject = "Soffid - Notification";
  message = "Automated mail sent .....";

  if (comments != null && !comments.isEmpty()) {
    for (comment : comments) {
      message += comment.message;
    }
  }
  serviceLocator.getUserService().sendHtmlMailToActors(new String[]{user.userName}, subject, message);
}
```

# Incoming transitions

The Incoming transitions tab displays the previous steps where the flow comes from. When you create a process from a template or from scratch default incoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: the previous step, where the flow comes. Allows you to select where the workflow comes from.
- **Incoming transition**: brief name to identify the transition. That is the name of the action the form will show to the final user.
- **To**: current step.
- **Action**: allows creating a custom script to perform specific actions.

When you create an incoming transition, Soffid creates the proper outcoming transition.

## Example

Scroll through the list of roles and the list of grant hierarchies to execute some actions.

```
userName = executionContext.getVariable("userName");


roleList = serviceLocator.getApplicationService().findRolesByUserName(userName);
for (role:roleList) {
  //TO-DO
}


user = serviceLocator.getUserService().findUserByUserName(userName);
roleGrantList = serviceLocator.getApplicationService().findRoleGrantHierarchyByUser(user.id);
for (roleGrant:roleGrantList) {
  //TO-DO
}
```

# Outgoing transitions

The Outcoming transition tab displays the next steps where the flow can go from the current step. When you create a process from a template or from scratch default outcoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: current step.
- **Incoming transition**: name of the transition.
- **To**: the next step, where the flow goes.
- **Action**: allows creating a custom script to perform specific actions.

When you create an outcoming transition, Soffid creates the proper incoming transition.

## Example

Delete additional attribute

```
userName = executionContext.getVariable("userName");
attribute = serviceLocator.getUserService().findDataByUserAndCode(userName, "country");

if (attribute != null) {
  serviceLocator.getAdditionalDataService().delete(attribute);
}
```

# Mail

# Definition

This step allows you to configure the necessary parameters to send an email when the flow reaches this point. That mail will be an informative mail, and the receptor could not perform any action from the mail.

To send mail, you will need to configure mail server parameters. You can visit the Soffid parameters page for more information.

# Steps Tabs

## Task details

When you select the Mail Step type, you could configure the mail information to send and the recipients of that information. To send a mail from Soffid Console is needed to have a mail server configuration.

- **Identities(s):** User, group, role, or email which is the recipient.
- **Email address(es):** Set one or more valid email addresses.
- **Subject:** Subject of the mail.
- **Email message:** Message of the mail.

## Incoming transitions

The Incoming transitions tab displays the previous steps where the flow comes from. When you create a process from a template or from scratch default incoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: the previous step, where the flow comes. Allows you to select where the workflow comes from.
- **Incoming transition**: brief name to identify the transition. That is the name of the action the form will show to the final user.
- **To**: current step.

- **Action**: allows creating a custom script to perform specific actions.

When you create an incoming transition, Soffid creates the proper outcoming transition.

### Example

Get the selected user, first name, and operation from the previous step:

```
selector = executionContext.getVariable("userSelector");
user = serviceLocator.getUserService().findUserByUserName(selector);
executionContext.setVariable("testName", user.firstName);
executionContext.setVariable("testOperation", "CHECK");
```

# Outgoing transitions

The Outcoming transition tab displays the next steps where the flow can go from the current step. When you create a process from a template or from scratch default outcoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: current step.
- **Incoming transition**: name of the transition.
- **To**: the next step, where the flow goes.
- **Action**: allows creating a custom script to perform specific actions.

When you create an outcoming transition, Soffid creates the proper incoming transition.

### Example

Get the account list associated with a user to perform some actions:

```
userName = executionContext.getVariable("userName");
accountList = serviceLocator.getAccountService().findAccountByJsonQuery("name eq \"" + userName + "\" AND
(type eq \"P\" or type eq \"S\" or type eq \"I\")");
for (account:accountList) {
//TO-DO
}
```

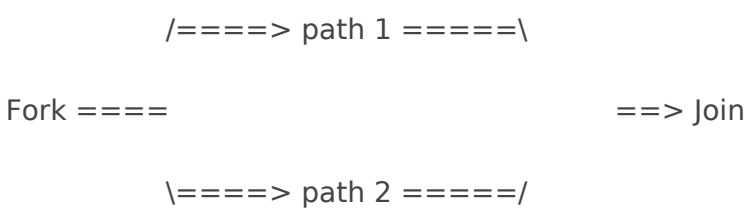
---

** https://es.wikipedia.org/wiki/Expression_Language*

# Fork

# Definition

This step is used to divide the workflow into two or more paths that will run in parallel, allowing multiple activities to run simultaneously.




# Steps Tabs

## Task details

This process type does not have task details for the fork step.

## Incoming transitions

The Incoming transitions tab displays the previous steps where the flow comes from. When you create a process from a template or from scratch default incoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: the previous step, where the flow comes. Allows you to select where the workflow comes from.
- **Incoming transition**: brief name to identify the transition. That is the name of the action the form will show to the final user.
- **To**: current step.
- **Action**: allows creating a custom script to perform specific actions.

When you create an incoming transition, Soffid creates the proper outcoming transition.

### Example

Update custom attributes defined on metadata

```
userName = executionContext.getVariable("userName");
attributes = serviceLocator.getUserService().findUserAttributes(userName);

newAttributes = new HashMap();
newAttributes.put("country", "FR");

language = attributes.get("language");
if (language == null) {
  language = new LinkedList();
}
language.add("Spanish");
language.add("German");

newAttributes.put ("language", language);

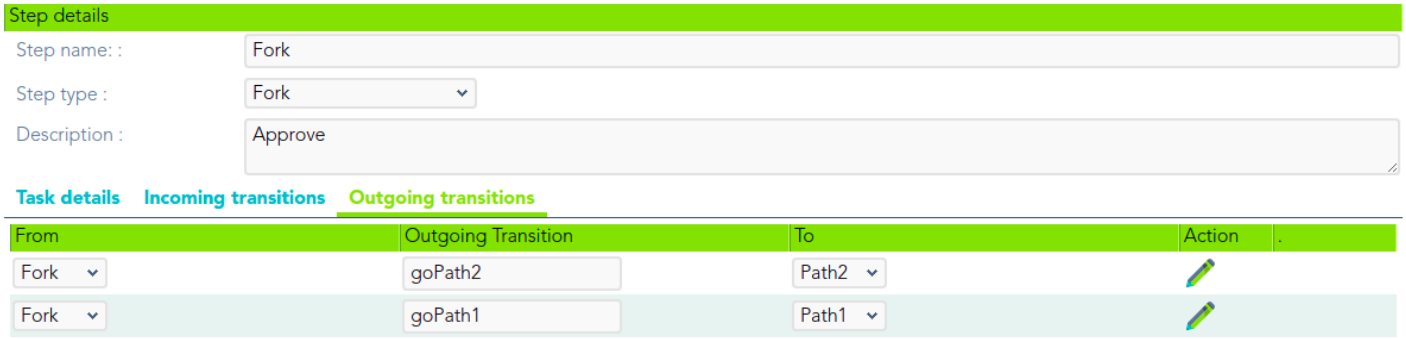
serviceLocator.getUserService().updateUserAttributes(userName, newAttributes);
```

# Outgoing transitions

The Outcoming transition tab displays the next steps where the flow can go from the current step. When you create a process from a template or from scratch default outcoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: current step.
- **Outgoing transition**: name of the transition. It is a required field, you must comply it to the workflow run properly.
- **To**: the next step, where the flow goes.
- **Action**: allows creating a custom script to perform specific actions.

When you create an outcoming transition, Soffid creates the proper incoming transition.




## Example

Scroll through the list of roles and the list of grant hierarchies to execute some actions.

```
userName = executionContext.getVariable("userName");


roleList = serviceLocator.getApplicationService().findRolesByUserName(userName);
for (role:roleList) {
  //TO-DO
}

user = serviceLocator.getUserService().findUserByUserName(userName);
roleGrantList = serviceLocator.getApplicationService().findRoleGrantHierarchyByUser(user.id);
for (roleGrant:roleGrantList) {
  //TO-DO
}
```

# Join

# Definition

This step is used to combine two or more parallel paths into one path.

# Steps Tabs

## Task details

This process type does not have task details for the fork step.

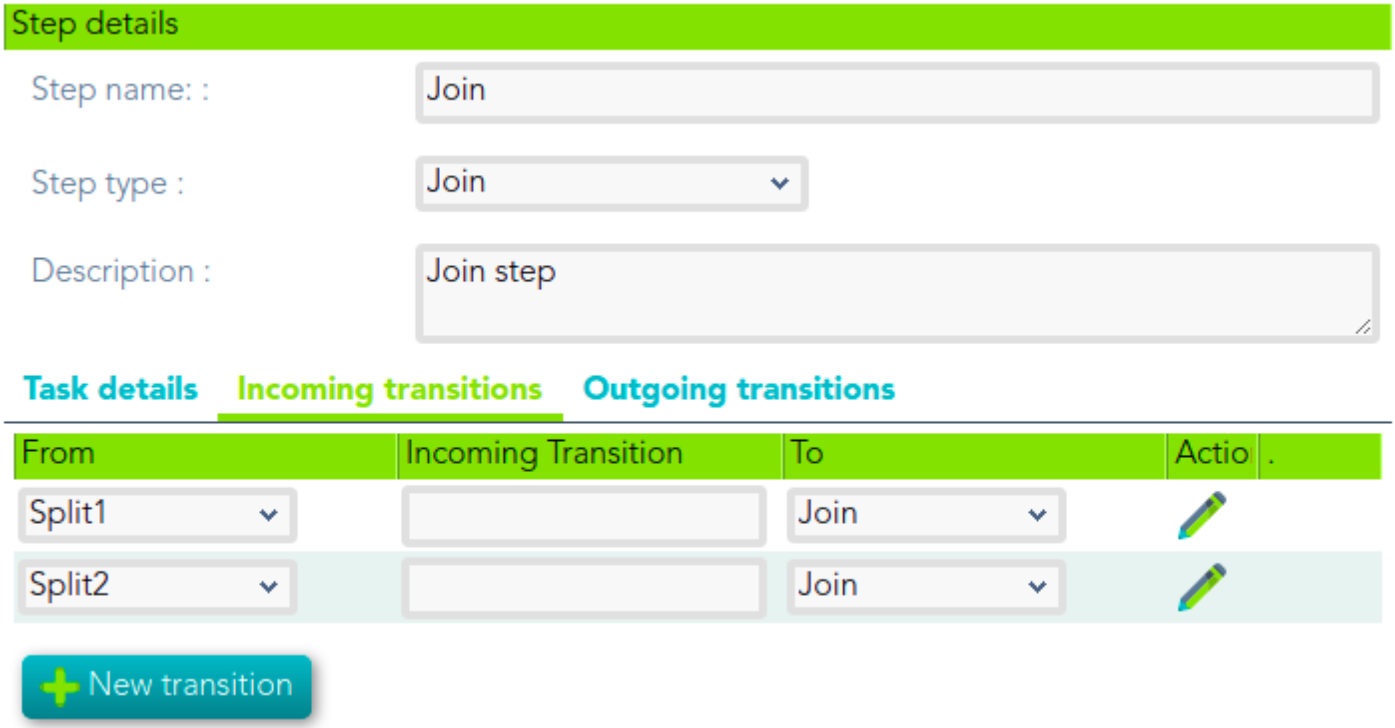
## Incoming transitions

The Incoming transitions tab displays the previous steps where the flow comes from. When you create a process from a template or from scratch default incoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

To join some paths will be mandatory to add the incoming transitions from those forks.

- **From**: the previous step, where the flow comes. Allows you to select where the workflow comes from.
- **Incoming transition**: brief name to identify the transition. That is the name of the action the form will show to the final user.
- **To**: current step.
- **Action**: allows creating a custom script to perform specific actions.

When you create an incoming transition, Soffid creates the proper outcoming transition.




## Example

Delete additional attribute:

```
userName = executionContext.getVariable("userName");
attribute = serviceLocator.getUserService().findDataByUserAndCode(userName, "country");

if (attribute != null) {
  serviceLocator.getAdditionalDataService().delete(attribute);
}
```

# Outgoing transitions

The Outcoming transition tab displays the next steps where the flow can go from the current step. When you create a process from a template or from scratch default outcoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: current step.
- **Incoming transition**: name of the transition.
- **To**: the next step, where the flow goes.
- **Action**: allows creating a custom script to perform specific actions.

When you create an outcoming transition, Soffid creates the proper incoming transition.

## Example

Scroll through the list of roles to execute some actions.

```
userName = executionContext.getVariable("userName");


roleList = serviceLocator.getApplicationService().findRolesByUserName(userName);
for (role:roleList) {
  //TO-DO
}
```

# End

# Description

The end step finalizes the process. It is the last step of the workflow.

# Steps Tabs

## Task details

This process type does not have task details for the start step.

## Incoming transitions

The Incoming transitions tab displays the previous steps where the flow comes from. When you create a process from a template or from scratch default incoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: the previous step, where the flow comes. Allows you to select where the workflow comes from.
- **Incoming transition**: brief name to identify the transition. That is the name of the action the form will show to the final user.
- **To**: current step.
- **Action**: allows creating a custom script to perform specific actions.

When you create an incoming transition, Soffid creates the proper outcoming transition.

### Example

If the user country is Spain, it will perform an action for each role.

```
userName = executionContext.getVariable("userName");
user = serviceLocator.getUserService().findUserByUserName(userName);
country = user.getAttributes().get("country");
```

```
if (country != null && country.equals("ES")) {
	roleList = serviceLocator.getApplicationService().findRolesByUserName(userName);
	for (role : roleList) {
		//TO-DO
}
```

# Outgoing transitions

This step does not have outgoing transitions. It is the last step of the workflow.