HP Service Manager

Software Version: 9.41

For the supported Windows® and UNIX® operating systems

Process Designer Tailoring Best Practices Guide (Codeless Mode)

Document Release Date: September 2015

Software Release Date: September 2015

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe® is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

UNIX® is a registered trademark of The Open Group.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

For a complete list of open source and third party acknowledgements, visit the HP Software Support Online web site and search for the product manual called HP Service Manager Open Source and Third Party License Agreements.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document, is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://hpln.hp.com/. This site requires that you register for an HP Passport and to sign in. To register for an HP Passport ID, click **Log in** on the HP Live Network site or click **New users - please register** on the HP Passport login page.

Support

Visit the HP Software Support site at: https://softwaresupport.hp.com.

This website provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support website to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and to sign in. Many also require a support contract. To register for an HP Passport ID, click **Register** on the HP Support site or click **Create an Account** on the HP Passport login page.

To find more information about access levels, go to: https://softwaresupport.hp.com/web/softwaresupport/access-levels.

HPSW Solutions Catalog accesses the HPSW Integrations and Solutions Catalog portal website. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this website is https://softwaresupport.hp.com/group/softwaresupport/search-result/-/facetsearch/document/KM01702710.

Contents

Best practices for configuring rule sets

This section describes the best practices and recommendations for configuring some of the rule sets. Rules sets can enable Service Manager to perform tasks automatically. For example, Service Manager can automatically assign a record to the most applicable group based on a preset rule set.

Configure assignment rules

Service Manager uses assignment rules to automatically distribute records, such as tasks, to the most appropriate groups or individuals for processing. You can configure the assignment rules with different conditions. For example, you can set a rule so that a record is automatically assigned to a group that handles the associated service, or you can set a rule so that a record is automatically assigned to an individual who has the lightest workload.

The following sections provide detailed information about the best practices of configuring assignment rules:

- Configure assignment rules for groups
- Configure assignment rules for individuals

Configure assignment rules for groups

You can configure group assignment rules so that a record can be automatically assigned to a specific group or a group that handles the associated service. The assignment rules can be configured either by using the Assignment form or by using JavaScript.

Assign to a group that handles the associated service

In Service Manager, a record is always associated with a service, so you can usually assign the record to the Config Admin group or the Support Groups of the service.

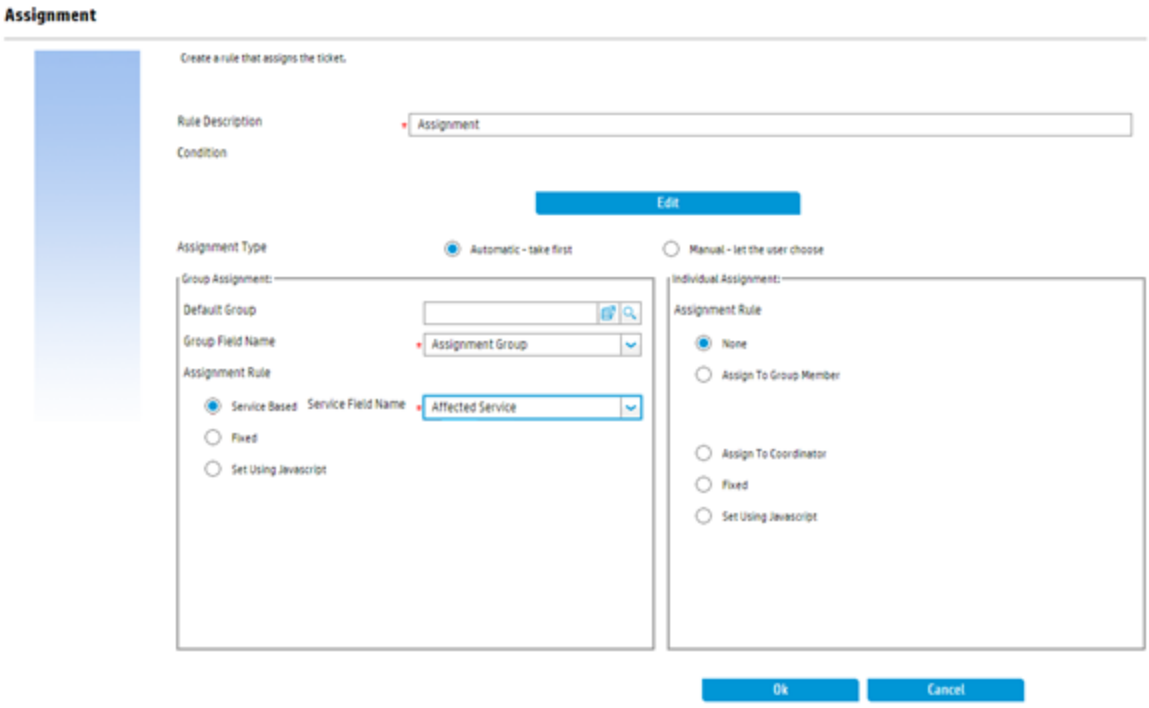
Service Manager follows the following logic to decide the assignment group:

- If the service has only the Config Admin group, and no Support Groups are set, the Config Admin group of the service is used.

- If the service has both the Config Admin group and the Support Groups, then the assignment depends on the assignment type you specify for the rule:
  - If the assignment type is **Automatic - take first**, the Config Admin group of the service is used.
  - If the assignment type is **Manual - let the user choose**, a list that combines the Config Admin group and the Support Groups of the service is shown to the operator for selection.

Note: If this rule is triggered by the system, the assignment type **Automatic - take first** is used even when you have selected **Manual - let the user choose** as the assignment type. This means that the Config Admin group of the system is used.

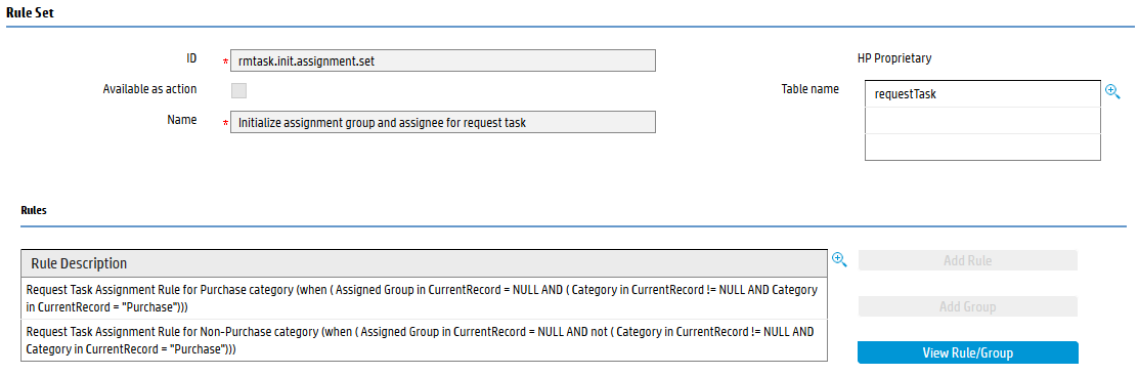
The configuration interface is as follows:



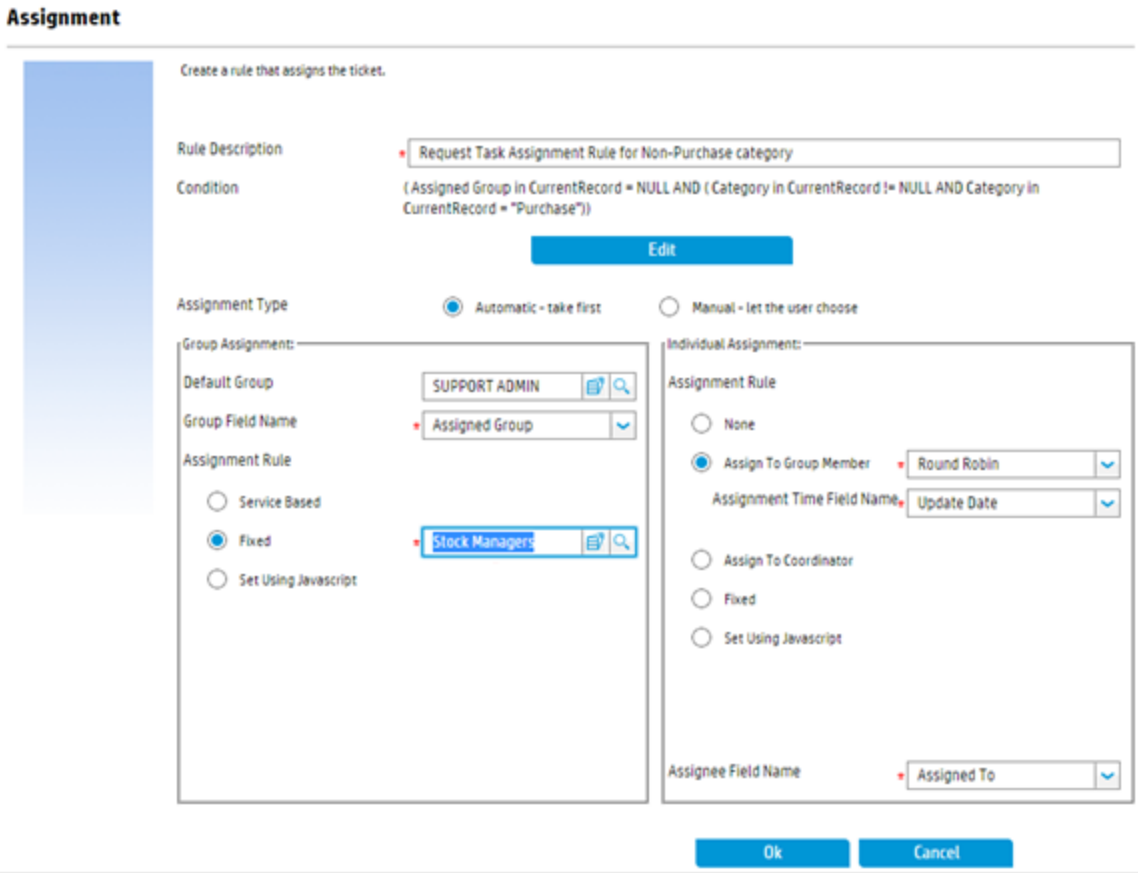
Assign to a specific group

You can use the **Fixed** assignment rule to enable Service Manager to automatically assign a record to a specific group.

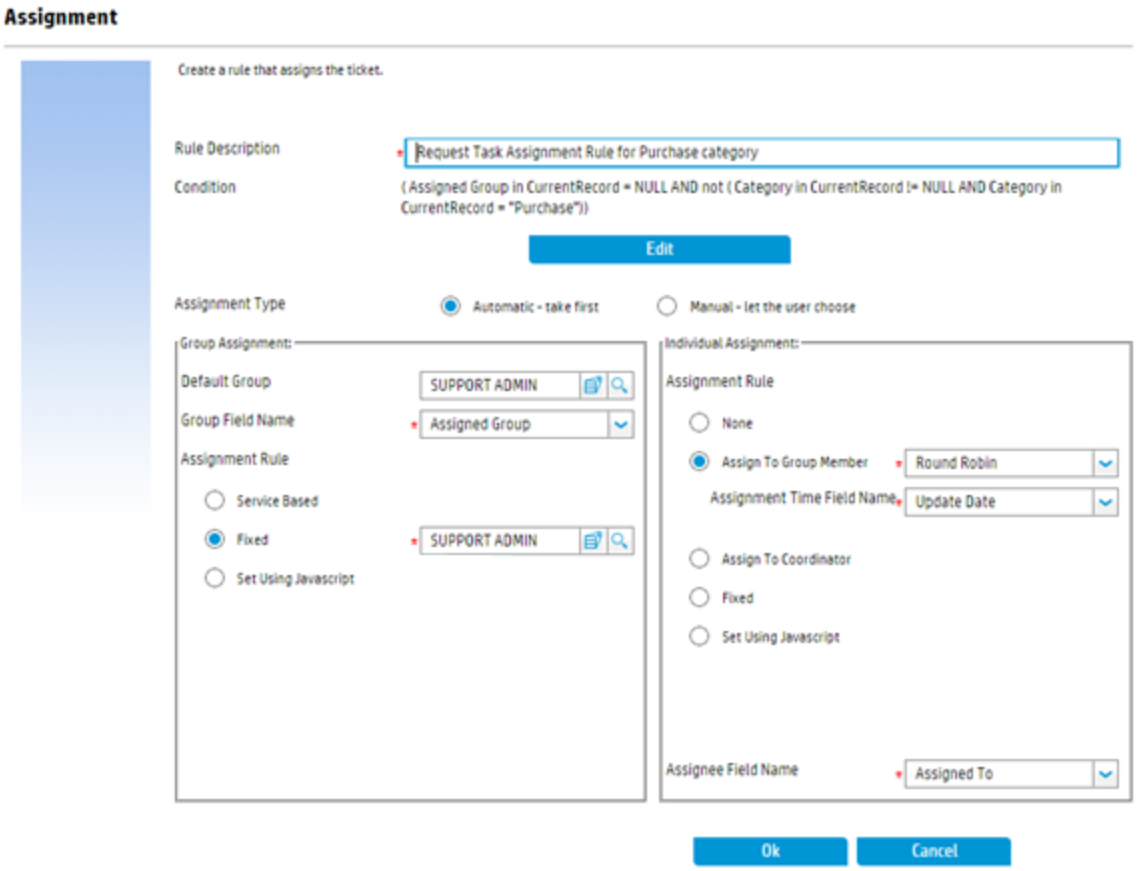
The conditions of this rule allow you to make group assignment based on locations or categories. For example, in the out-of-box configuration, the request task assignment rule is based on categories. Two assignment rules are configured in the rule set “rmtask.init.assignment.set”: One is for the Purchase category, and the other is for other categories, as shown below:



For the Purchase category, the system assigns to the group “Stock Managers”:



For the other categories, the system assigns to the group “SUPPORT ADMIN”:



Use JavaScript to configure assignment rules for groups

In addition to using the configuration forms to configure assignment rules for groups, you have the flexibility to use JavaScript to implement your assignment rules. In JavaScript, you can use the `groupValue` variable to specify an assignment group or a combination of groups.

Configure assignment rules for individuals

You can configure assignment rules for individuals so that a record can be automatically assigned to a specific individual for processing. The assignment rules can be configured either by using the Assignment form or by using JavaScript.

You can configure the assignment rules so that a record can be automatically assigned to the following individuals:

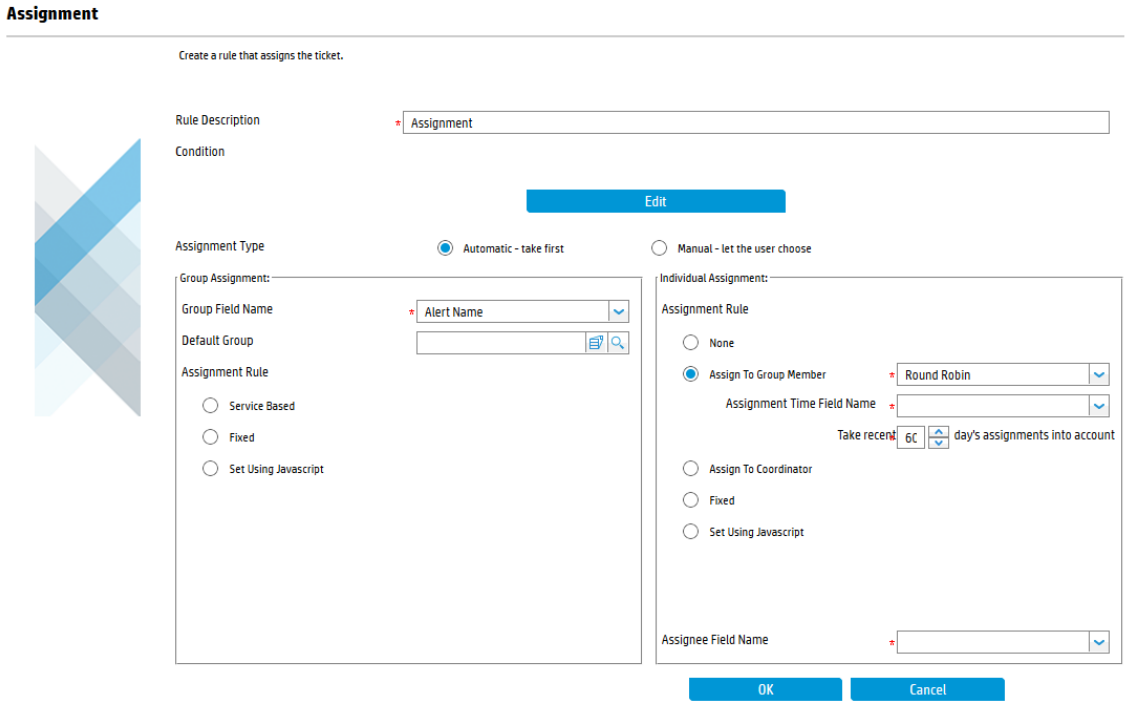
- None: Do not assign to any individual
- A member of the assigned group: The member can be decided in a round robin manner or based on the workload of the group members
- Coordinator of the assigned group
- A specific assignee

Assign in a round robin manner

In the round robin manner, the system checks the latest assignment time of the members in a group, and then assigns the record to the member who has the earliest assignment time.

If you have a huge number of records to be assigned in Service Manager, follow these guidelines to avoid possible performance issues:

- **Only take recent assignments into account.** By default, the value of 60 days is set while you add assignment rules in the round robin manner. You can adjust the value based on your needs. If this value is set to 60 days, it means that only assignments within the most recent 60 days are considered; assignments before 60 days are ignored. If the value is set to 0, it means there is no assignment time restriction; all the assignments of the members are considered.



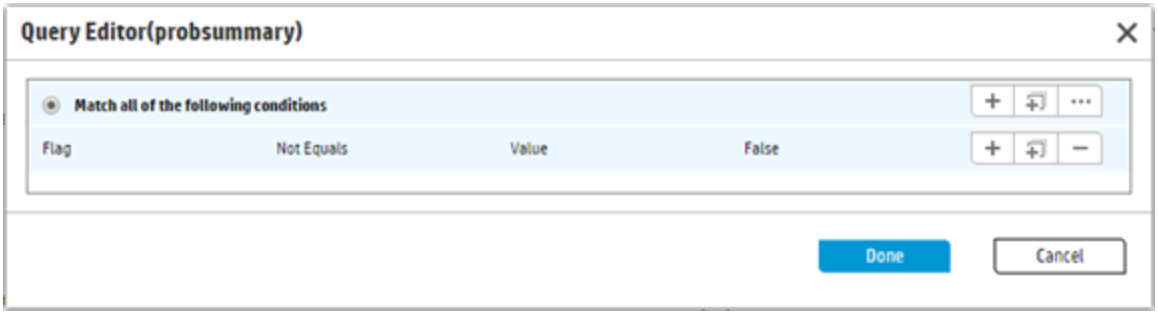
- **Create index on the assignment time field and the assignee field of the corresponding table.** This can avoid full table scans and thus can increase the query speed. At run time, the calculation SQL is as follows:
  `select [assignee field], max([assign time field]) time from [ticket file] where [assign time field] >= [assigned time restriction] and [assignee field] isin [members in group] group by [assignee field] order by time`

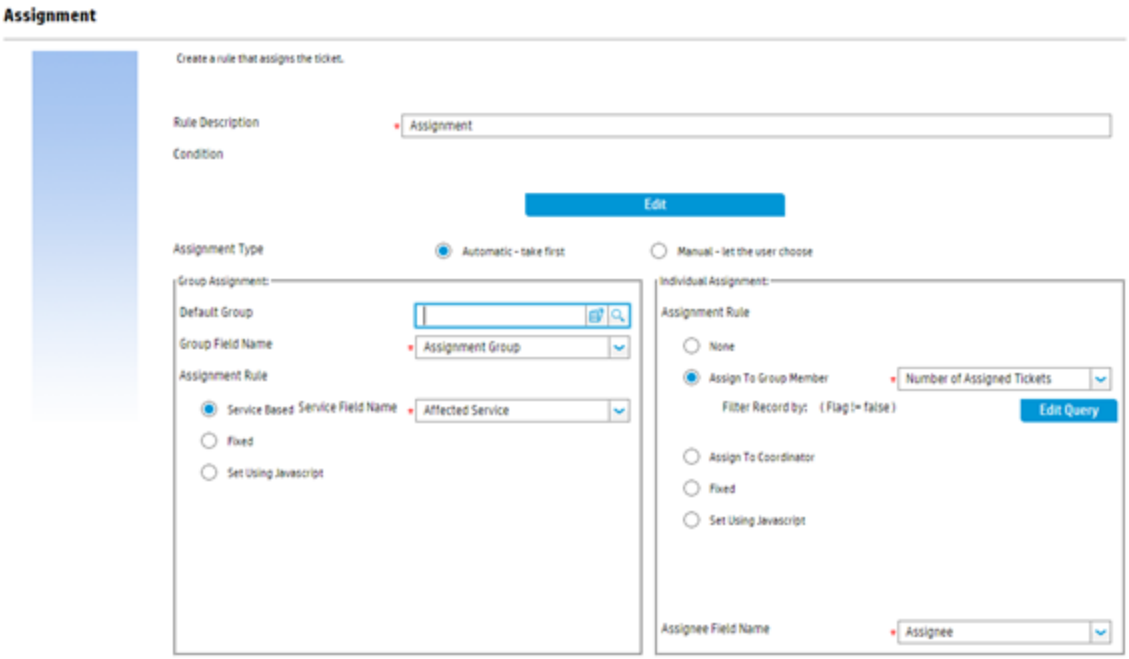
Assign based on workload

You can use the Number of Assigned Tickets assignment type to configure the assignment rules so that a record can be assigned based on people's workload. The system checks the number of working records of all the members in a group and then assigns the record to the member who has least working records.

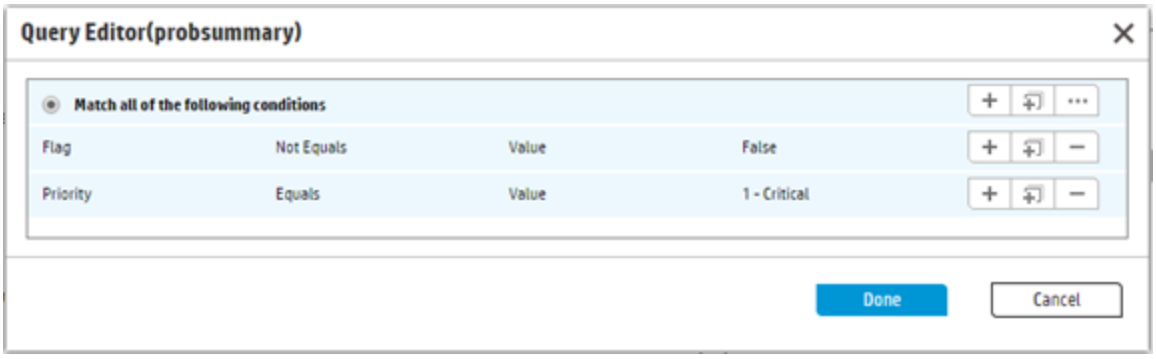
To avoid possible performance issues for this assignment type, follow these guidelines:

- **Only take open records into account.** Usually you need to append the query that filters out records other than the open records. You can use the Query Editor to edit the query string. In the example below, the query string “flag~=false” is appended.





In addition, you can further narrow down the filter result. For example, if you plan to take only the open and critical records into account, you can append to the query string as shown below:



- Create index on the assignee field and the fields of the appended query. This can avoid full table scans and can thus increase the query speed. At run time, the calculation SQL is as follows:
  ```
  select [assignee field], count(*) ncnt from [ticket file] where [assignee field] isin [members in group] and [the appended query string] group by [assignee field] order by ncnt
  ```

Use Javascript to configure assignment rules for individuals

In addition to using the configuration forms to configure assignment rules for individuals, you have the flexibility to use JavaScript to implement your assignment rules. In JavaScript, you can use the `assigneeValue` variable to specify an assignee or a combination of assignees.

Perform automatic operations with Run Action rules

You can use the Run Action rules to enable Service Manager to perform automatic cross table or module operations. For example, you can set the rules so that the solution of an incident can be copied to the interaction from which the incident is escalated.

Service Manager supports three types of the Run Action rules:

- Run Action rules on related records
- Run Action rules on other records
- Run Action rules on the current record

Configure Run Action rules on related records

You can configure the Run Action rules to enable Service Manager to perform automatic operations on related records. The relationship of the related records is usually maintained by the Related Records functionality.

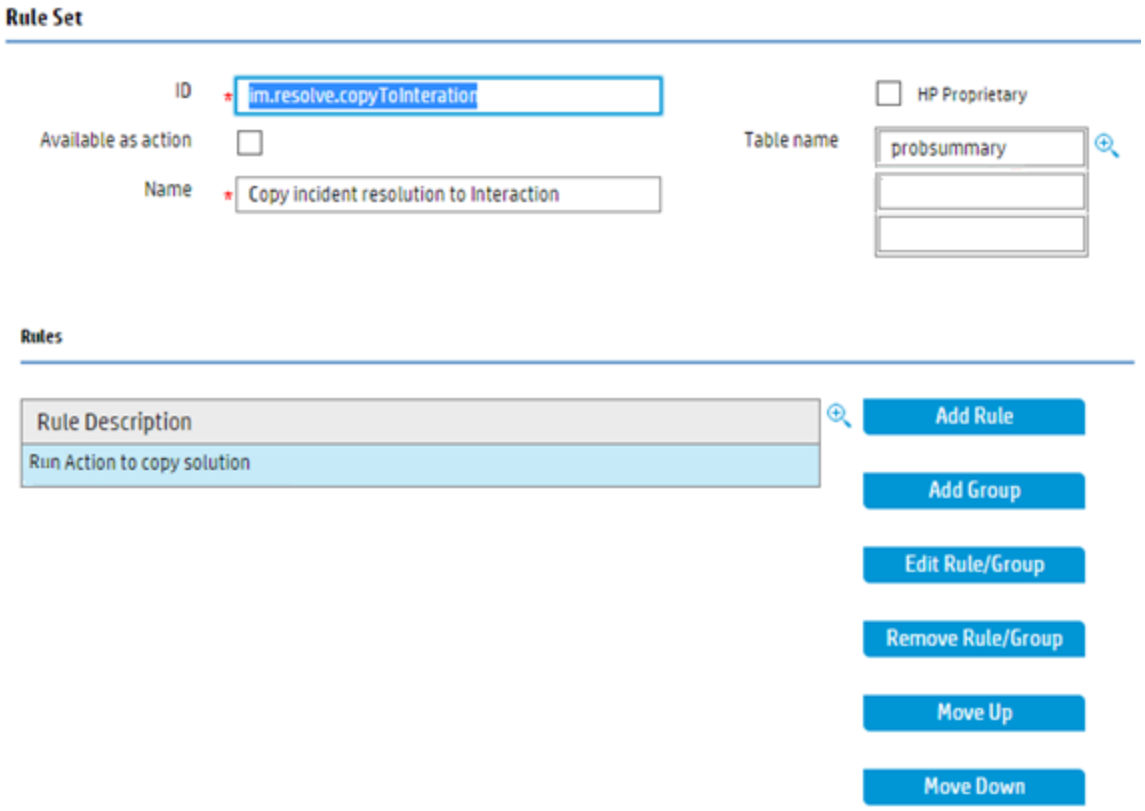
For example, when an incident is resolved, you might want to copy the incident resolution to the related interaction from which the incident is escalated. In this case, you can configure the Run Action rules with these steps:

1. Define a rule set on the source table
2. Define a rule set on the target table

3. Configure the rule set on the source table into an appropriate workflow

Define a rule set on the source table

To copy the incident resolution, you can define a rule set on the source table “probsummary” that stores incident records. The rule set definition form is as follows:



In addition to the above configuration, you also need to perform the following tasks:

- Select **Related Records** for the "Run Action on" field
- Select **Escalate From Interaction** for the "Relation Type" field. This means the target related record is the interaction from which the current incident is escalated.
- Select **sd.copyIncidentSolution** for the "Run Rule Set" field. This rule set is executed against the related interaction.
- Select **Save** for the "Action after Rule Set" field. The rule set is saved for the related interaction.

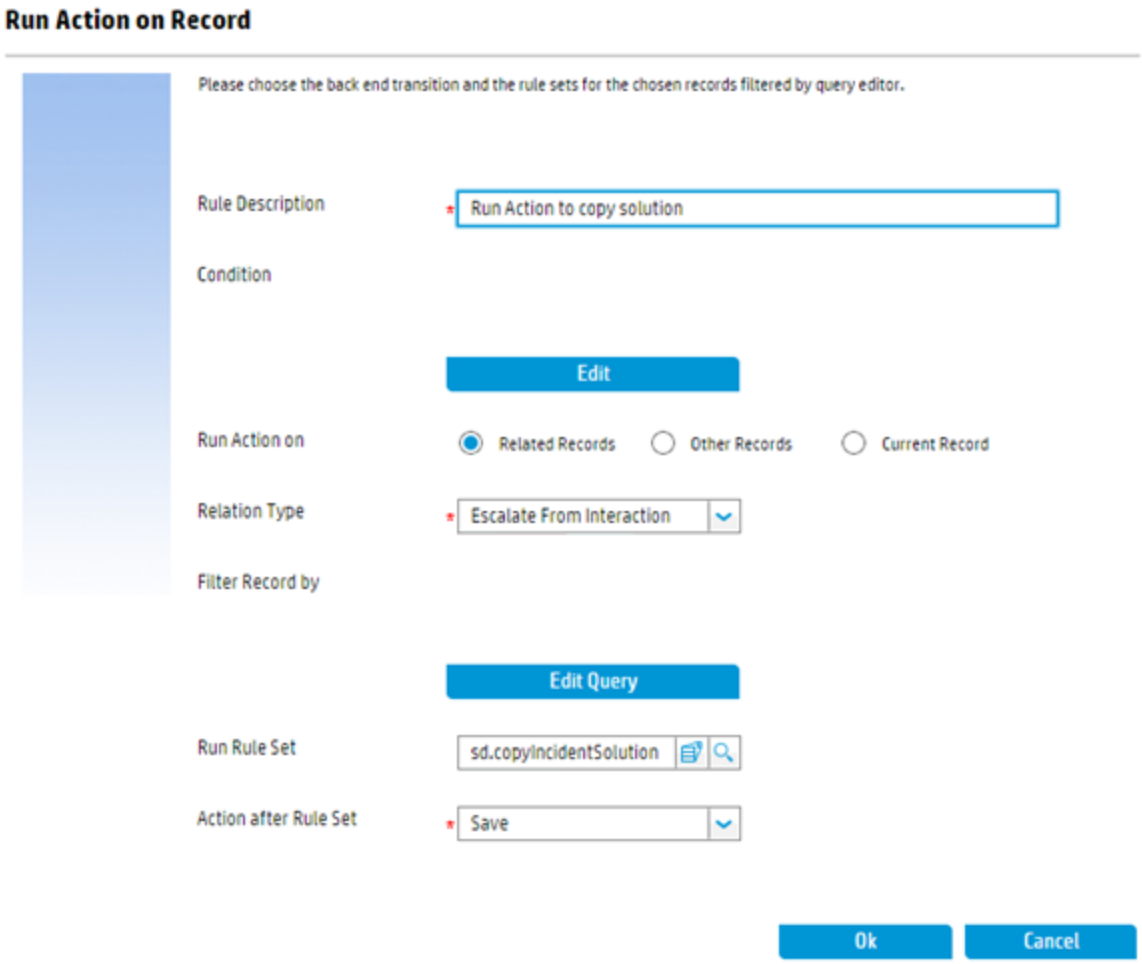
Note: You can select "Do nothing", "save" or one of the back end transitions for this field.

- The back end transition list is retrieved from all the workflows of the current related record file based on your selected relation type. At run time, if some of the workflows do not have your selected back end transition, the corresponding transition is not run.

  For example, suppose you have a Change file "cm3r" and it meets the following conditions:
  - Relation type: You have selected "Caused Changes" as the relation type.
  - Workflows: Based on the selected relation type, there are two workflows for the Change file: "Normal Change" and "Standard Change".
  - Back end transitions: The "Normal Change" workflow has the back end transition "event.close", whereas the "Standard Change" workflow does not.

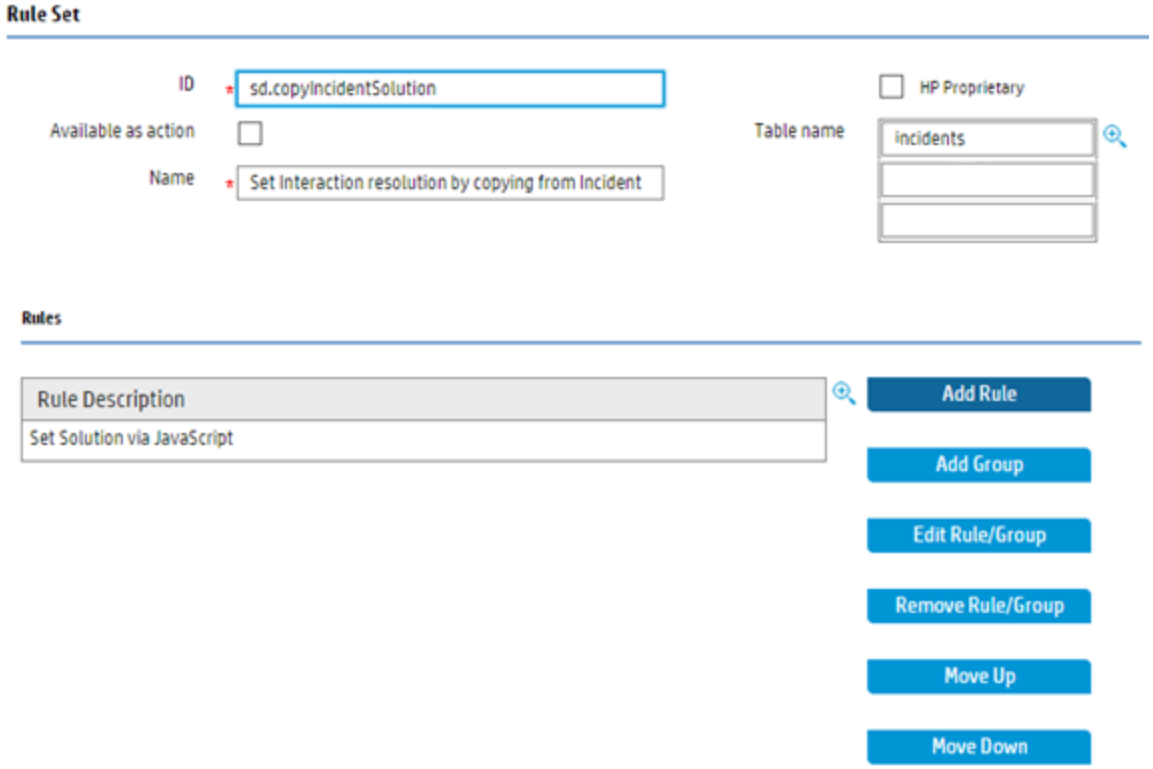
  In this case, if you select the action "Backend Transition: event.close", and if at run time there are two related caused changes: One for "Normal Change" and the other for "Standard Change", the back end transition is only run on the "Normal Change" workflow, not on the "Standard Change" workflow.
- You are suggested to use the same back end transition name in different workflows if these backend transitions follow the same business logic.
- While a back end transition is selected from the list, the interface is updated to indicate to which workflows the transition applies and to which it does not apply.

The form on which you can perform these tasks is shown below:



Define a rule set on the target table

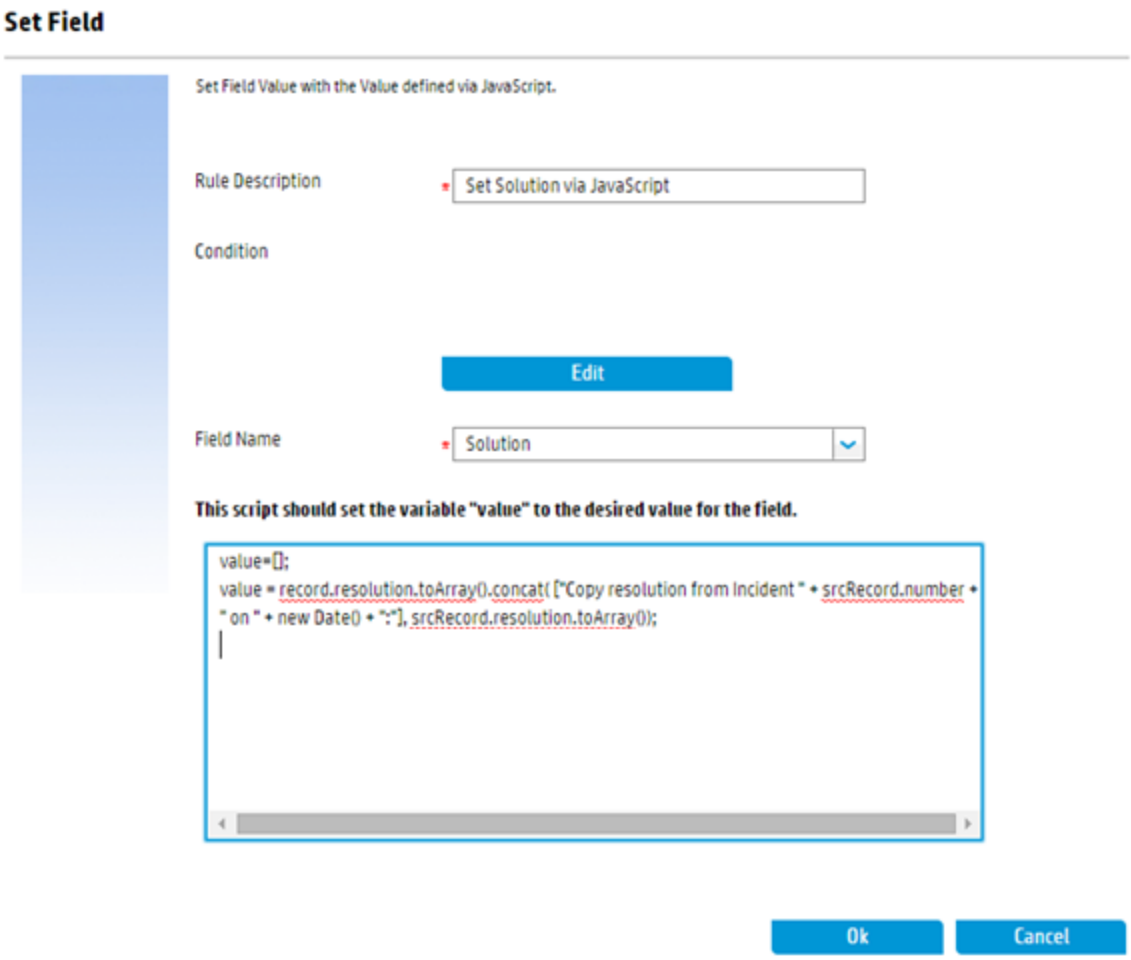
You need to define a rule set on the target table. This rule set is used as the value of the "Run Rule Set" field when you define the rule set on the source table. In this example, you define the rule set “sd.copyIncidentSolution” on the target table “incidents” that stores the interaction records.



You need to configure a Set Field rule in this rule set to append the resolution of the incident to the resolution field of the interaction. To do this, you can use JavaScript, in which you can refer to the current source record as `srcRecord` and the original copy of the source record (before any changes are made by the user) as `oldSrcRecord`. The system only supports invoking the current rule from the Run Action rule at run time. The JavaScript of this example is as follows:

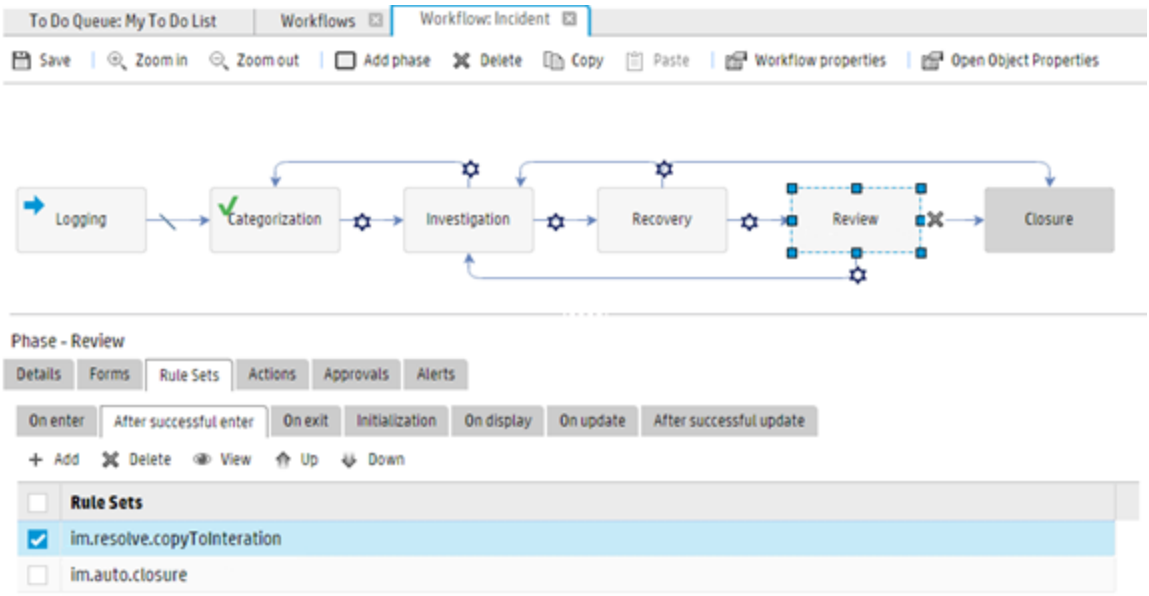
```
value = record.resolution.toArray().concat( ["Copy resolution from Incident " +
srcRecord.number + " on " + new Date() + ":"], srcRecord.resolution.toArray());
```

In JavaScript, the current record and the original copy of the record are also referred as `record` and `oldRecord`. In this example, they are the current interaction record and original interaction record.



Configure the rule set on the source table into an appropriate workflow

In this example, you can configure the rule set into the **After successful enter** event of the Review phase of the Incident workflow. This means only after the incident record is resolved, the rule set is triggered.



Note: The Run Action rule is executed in a sequential manner, which means the configured rule sets and actions of each related record are called one after another. If one of the rule sets and actions fails, for example, one of the related record is locked by another user, the whole execution stops and returns with a failure result. If you want the system to handle the failed execution on related records caused by record locking, you can combine the Run Action rules and the Run Scheduled Action rules. For details, see Combine the Run Action rules and the Run Scheduled Action rules.

Configure Run Action rules on other records

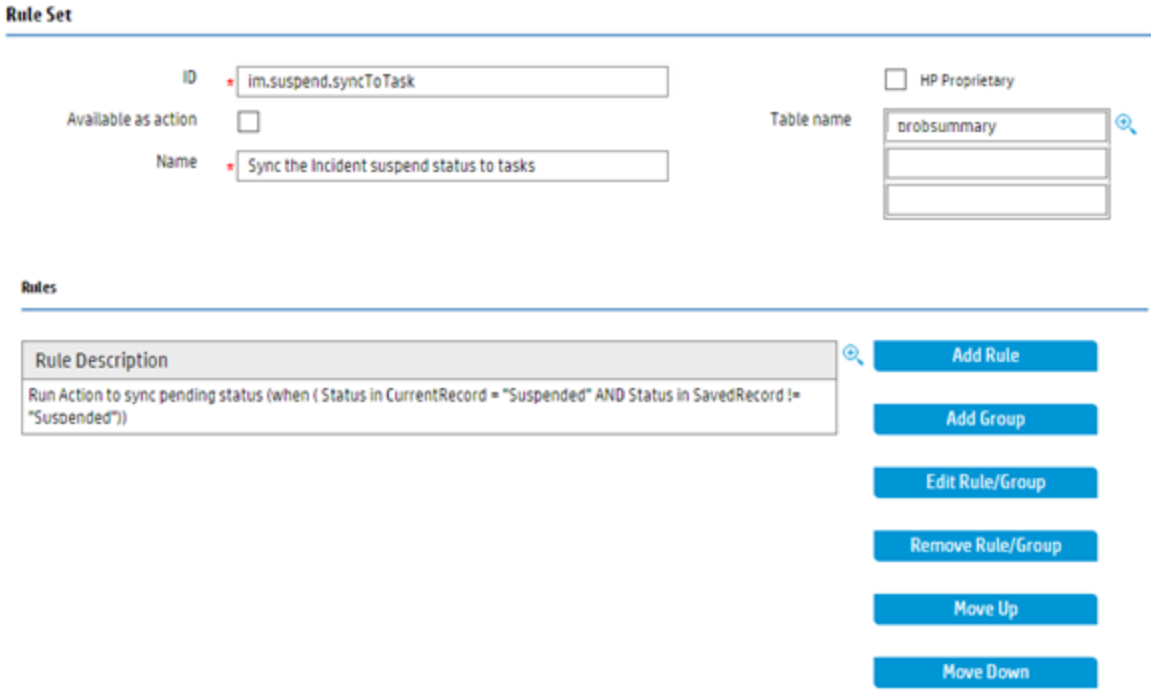
You can configure the Run Action rules to enable Service Manager to perform automatic operations on other records. The relationship of the records is built by manually defined query string. For example, you might want to sync up the status of an incident to its relevant incident tasks, that is to say, if the incident is suspended, the relevant incident tasks should be suspended as well.

To configure Run Action rules on other records, follow these steps:

1. Define a rule set on the source table
2. Define a rule set on the target table
3. Configure the rule set on the source table into an appropriate workflow

Define a rule set on the source table

In this example, you can define the rule set on the source table “probsummary” that stores incident records:



In addition to the above configuration, you also need to configure the Run Action rules in the rule set as follows:

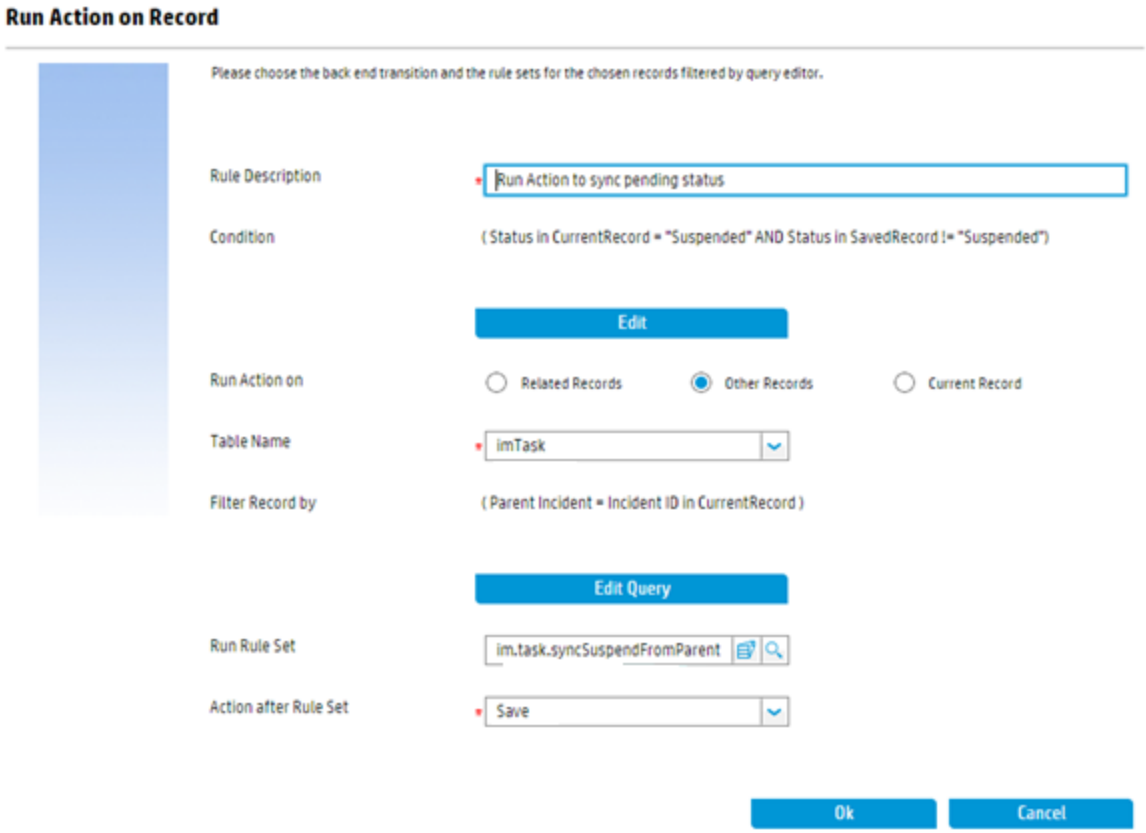
- Select **Other Records** for the "Run Action on" field.
- Select **imTask** for the "Table Name" field. This means the target records are incident tasks.
- Use the Query Editor to configure the Filter Record as shown below. The only incident task that belongs to current incident record is included.

- Select **im.task.syncSuspendFromParent** for the Run Rule Set field. This rule set is executed against the related incident task.
- Select Save for the "Action after Rule Set" field. The save action is executed against the related incident task.

Note: You can select **Do nothing**, **Save**, or one of the back end transitions for this field.

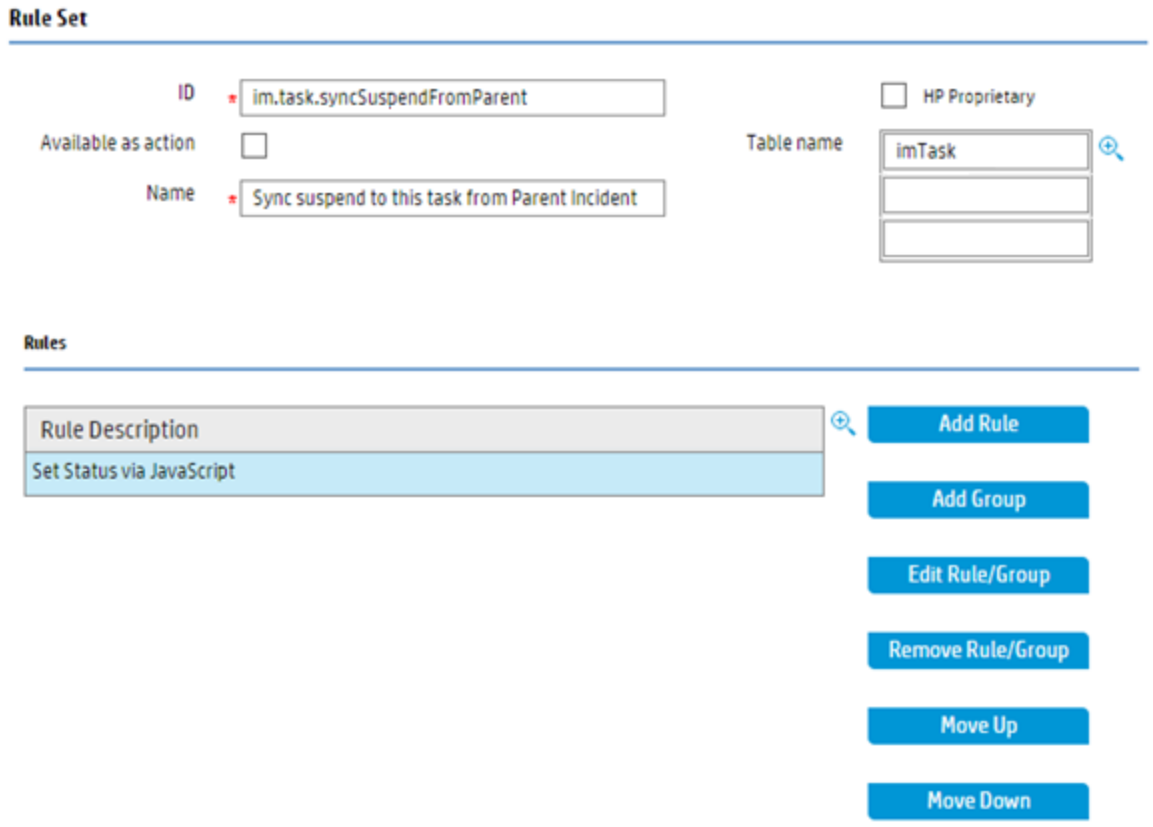
- The back end transition list is retrieved from all the workflows of the selected record file. Similar to Configure Run Action rules on related records, at run time, if some of the other records do not have your selected back end transition, the corresponding transition is not run.
- You are suggested to use the same back end transition name in different workflows if these back end transitions follow the same business logic.
- While a back end transition is selected from the list, the interface is updated to indicate to which workflows the transition applies and to which it does not apply.

The form on which you can perform these tasks is shown below:

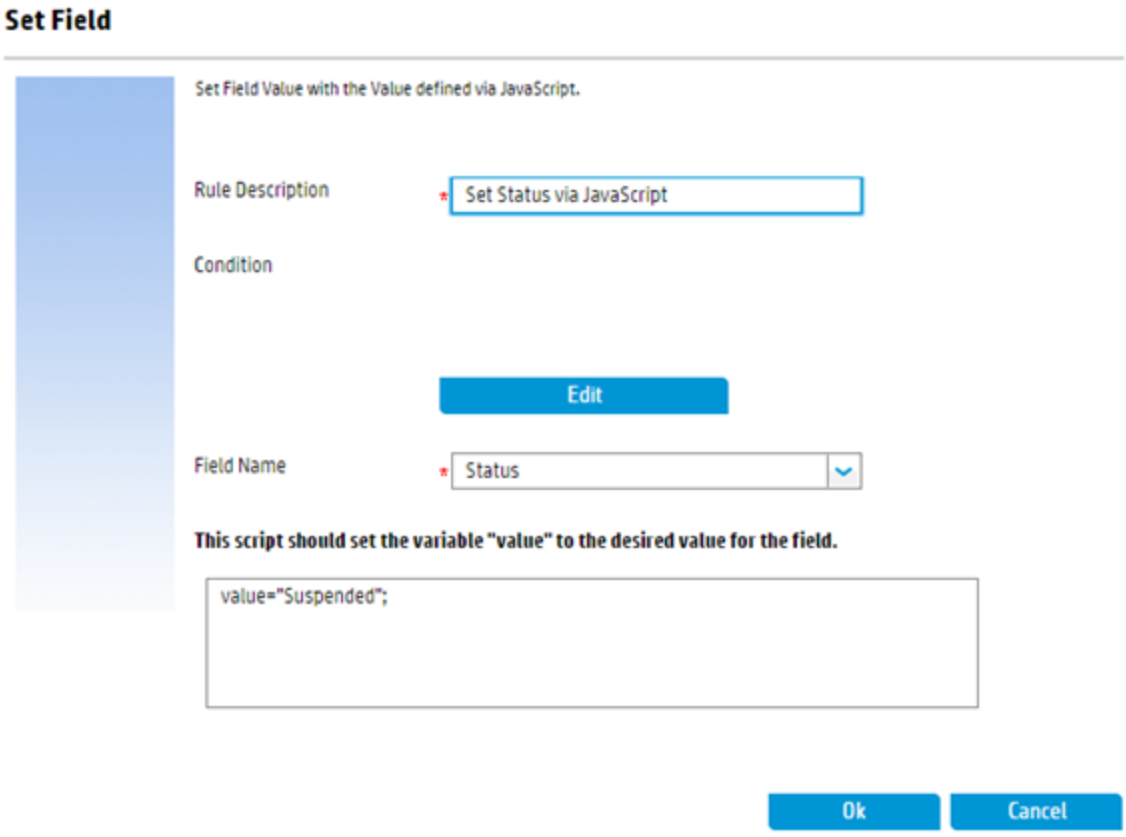


Define a rule set on the target table

You need to define a rule set on the target table. This rule set is used as the value of the "Run Rule Set" field when you define the rule set on the source table. In this example, you define the rule set “im.task.syncSuspendFromParent” on the target table “imTask” that stores the incident task records.



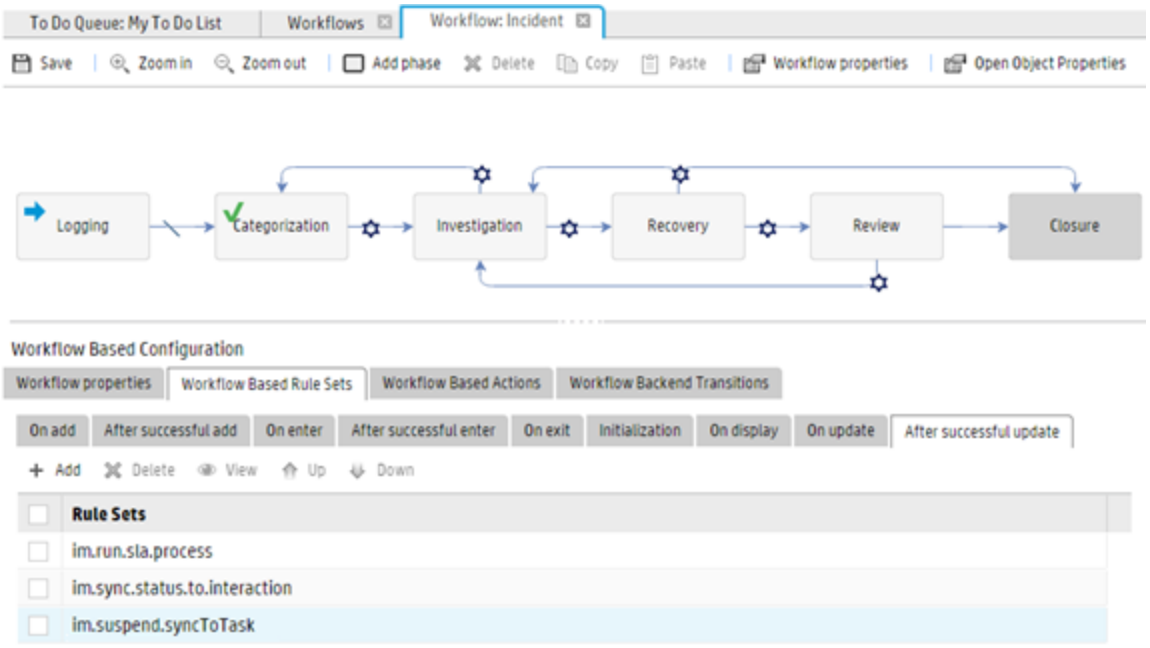
You need to configure a Set Field rule to change the status of the incident task to the value of "Suspended". To do this, you can use JavaScript, in which you can refer to the current source record as `srcRecord` and the original copy of the source record (before any changes are made by the user) as `oldSrcRecord`. The system only supports invoking the current rule from the Run Action rule at run time.



In JavaScript, the current record and the original copy of the record are also referred as `record` and `oldRecord`. In this example, they are the current incident task record and original incident task record.

Configure the rule set on the source table into an appropriate workflow

In this example, you can configure the rule set into the **After successful update** event of the incident workflow, which means when the status of the incident record is changed to “Suspended”, the rule set is triggered.

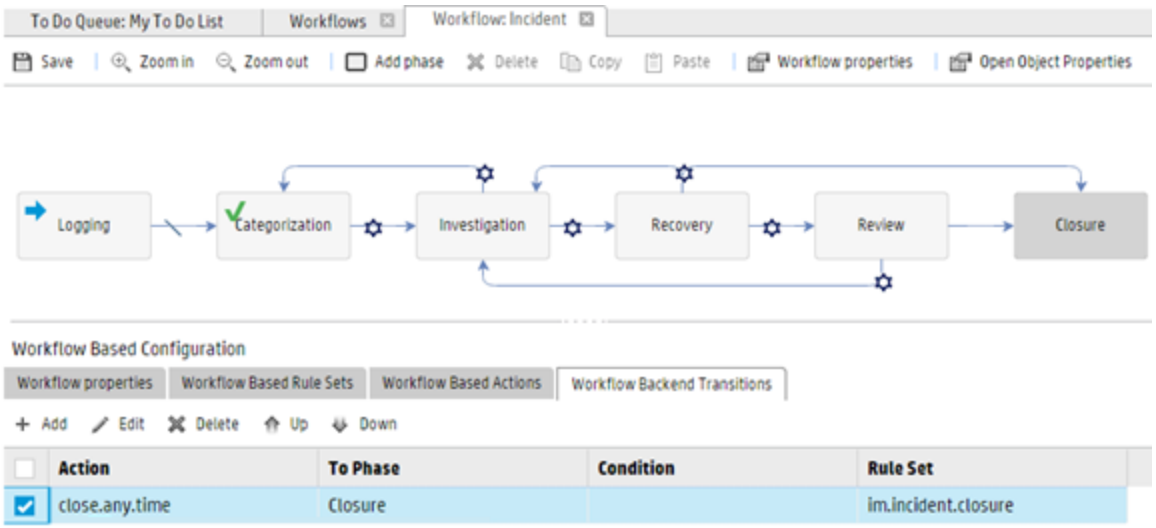


Note: The Run Action rule is executed in a sequential manner, which means the configured rule sets and actions of each related record are called one after another. If one of the rule sets and actions fails, for example, one of the related record is locked by another user, the whole execution stops and returns with a failure result. If you want the system to handle the failed execution on related records caused by record locking, you can combine the Run Action rules and the Run Scheduled Action rules. For details, see Combine the Run Action rules and the Run Scheduled Action rules.

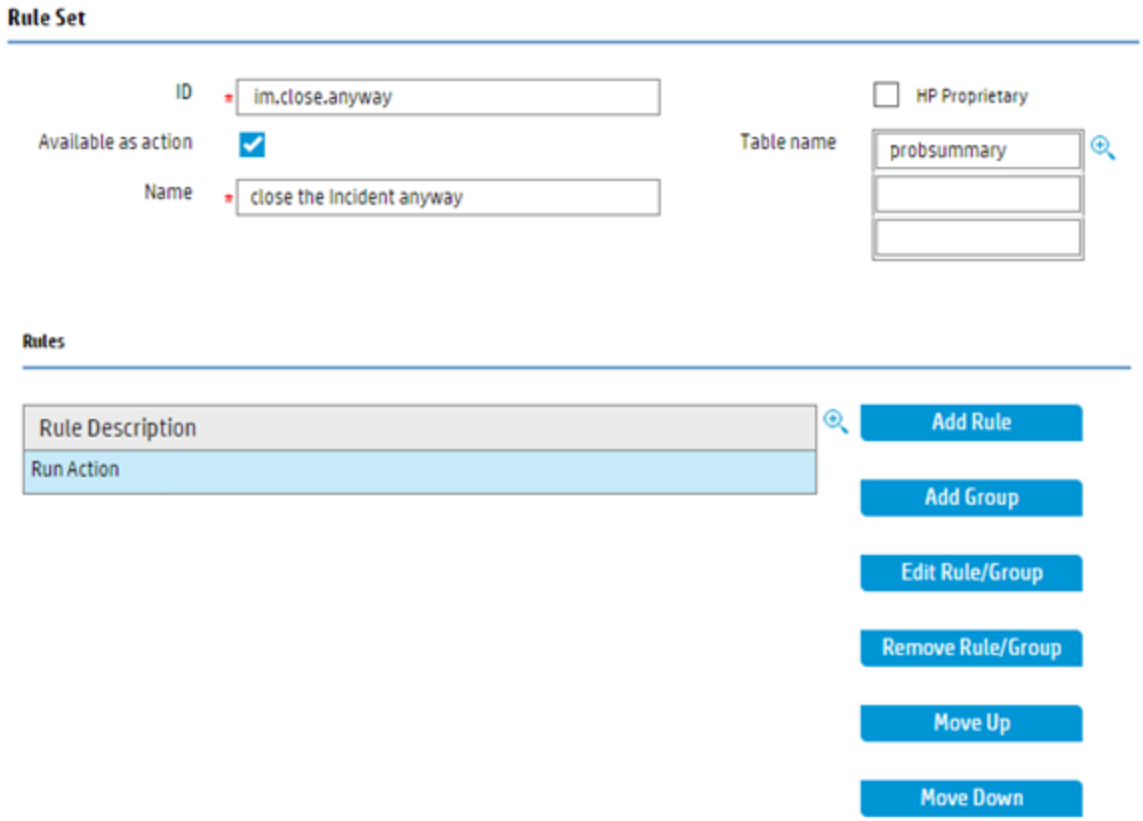
Configure Run Action rules on the current record

You can configure the Run Action rules to enable Service Manager to perform automatic operations on the current record. Usually this rule type can be used to easily expose a back end transition call as an action. To expose a back end transition configured in a workflow, you can use JavaScript or RAD calls as well. However, that approach is not quite user-friendly. By using the Run Action rules, you can easily expose the back end transitions. For details about calling back end transitions, see the Workflow back end transitions section in Configure a workflow.

For example, in the out-of-box incident workflow, there is a back end transition “close.any.time” that can move the incident to the Closure phase whatever current phase is. The back end transition is shown as follows:



To be able to invoke the transition, you need to first add a rule set that contains the Run Action rule, and then mark this rule set as **Available as action**, as shown below:



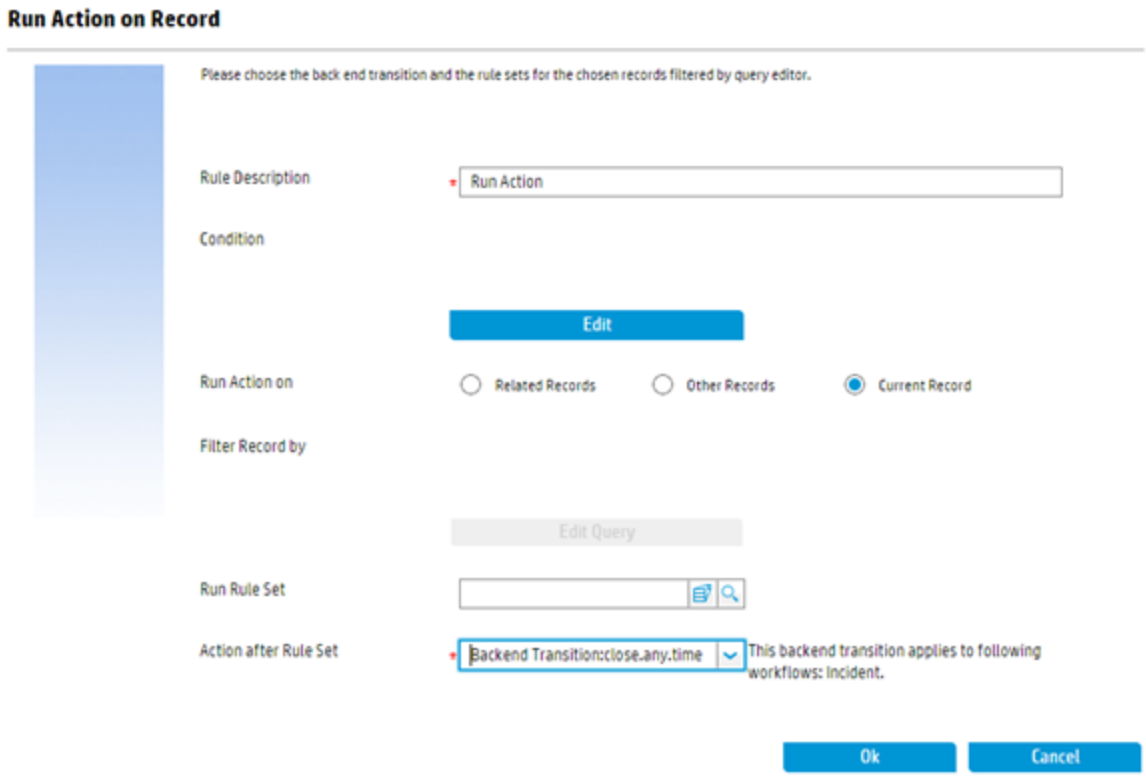
In addition, you also need to configure the Run Action rules in rule set as follows:

- Select **Current Record** for the "Run Action on" field
- Keep the "Run Rule Set" field empty
- Select **Backend Transition:close.any.time** for the "Action after Rule Set" field. The save action is executed against the current incident record.

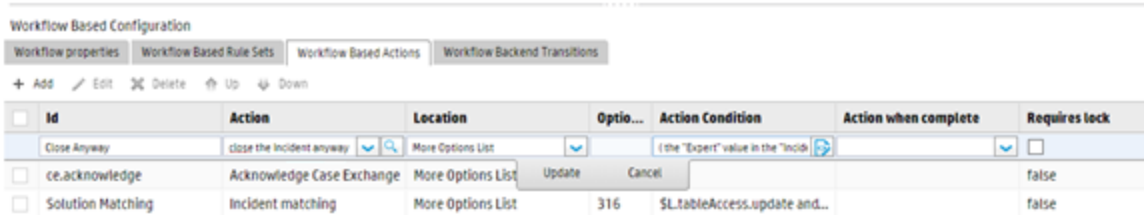
Note: You can select **Do nothing**, **Save**, or one of the back end transitions.

- The back end transition list is retrieved from all the workflows of the selected record file. Similar to Configure Run Action rules on related records, at run time, if some of the other records do not have your selected back end transition, the corresponding transition is not run.
- You are suggested to use the same back end transition name in different workflows if these back end transitions follow the same business logic.
- While a back end transition is selected from the list, the interface is updated to indicate to which workflows the transition applies and to which it does not apply.

The form on which you can perform these tasks is shown below:



Upon completing all the above configurations, you can expose the rule set as an action by adding it to the workflow level actions:

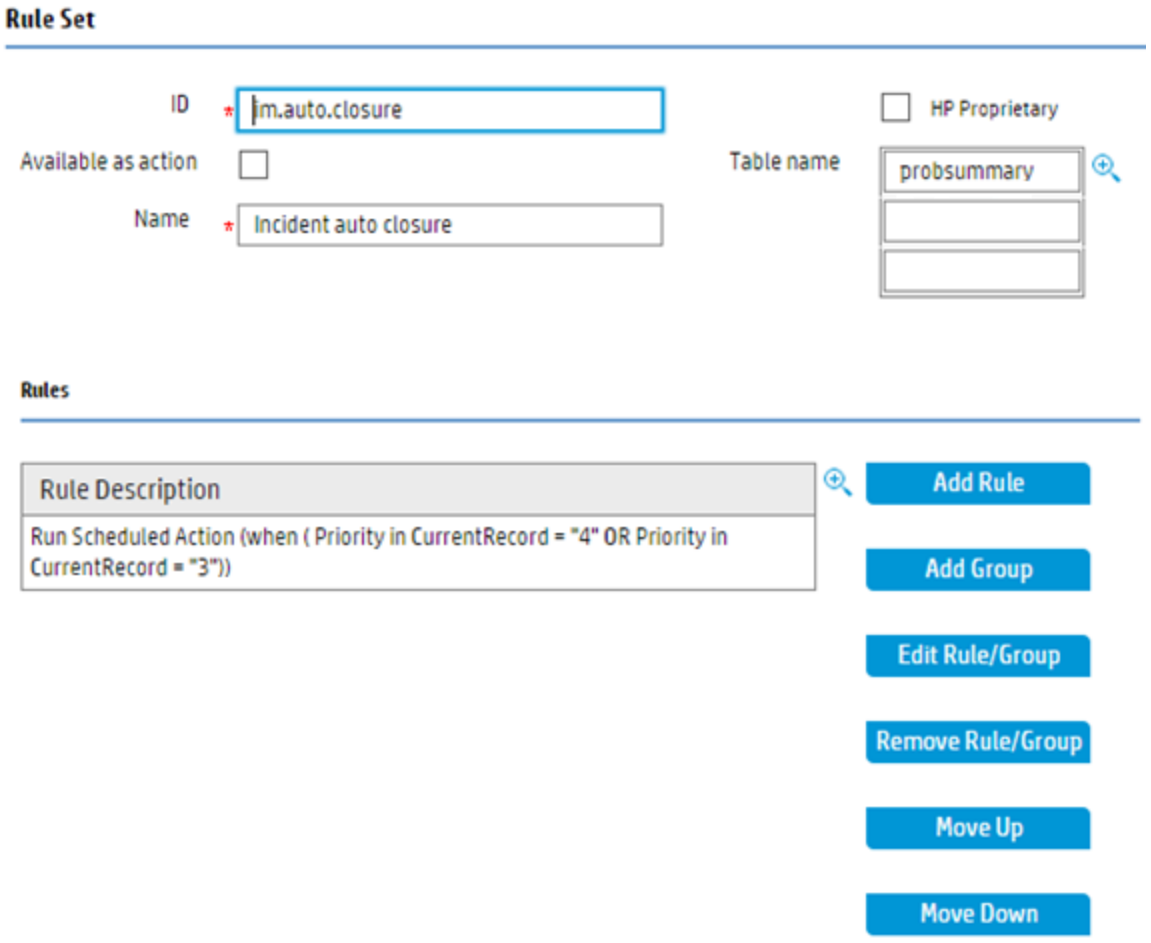


Note: Only expose such actions to specific persons or operations. For example, you can expose it to the person who has the Expert permission on the Incident area by setting the security level with the Action Condition. If you want to make it available only to a Web Service call, set your Action Condition accordingly.

Configure Run Scheduled Action rules

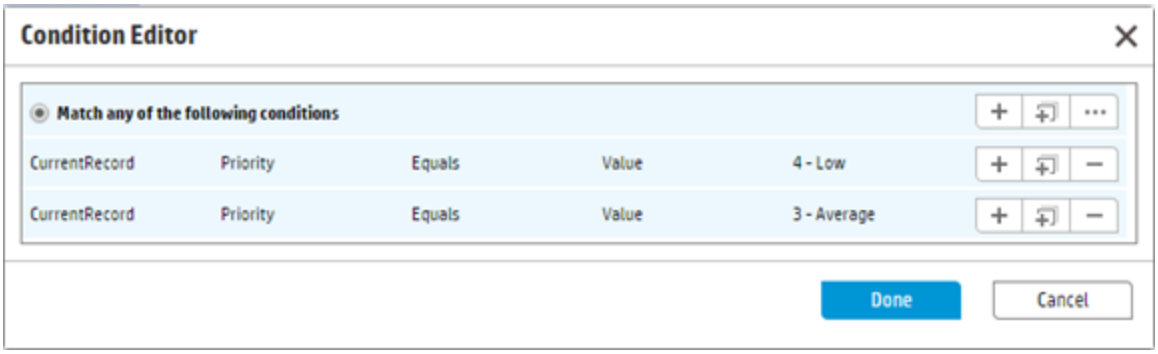
You can configure the Run Scheduled Action rules to enable Service Manager to perform scheduled automatic operations on the current record. This rule can be used in situations like automatic record closure.

For example, you might want the system to automatically close the incidents that were resolved seven days ago. To do that, first define a rule set with the Run Scheduled Action rules. You can define the rule set on the table “probsummary” that stores incident records, as shown below:



In addition to the above configurations, you also need to configure the Run Scheduled Action rules in this rule set as follows:

- Set the conditions as follows. This only takes the low priority incidents into account.



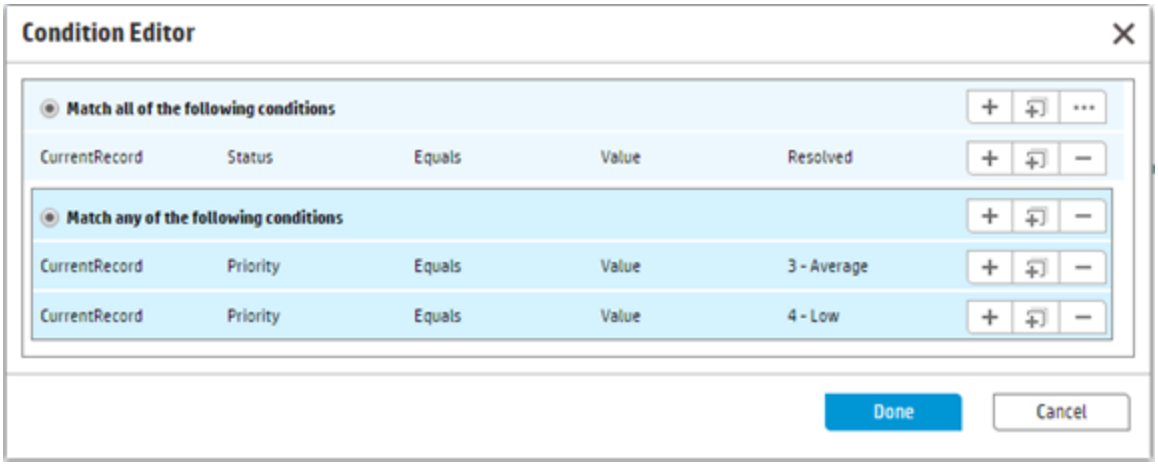
At run time, only when the condition is matched, this rule is executed to generate a scheduled record

with the class of “scheduledAction”. Note that this rule type introduces a new back end scheduler process named “Scheduled Action processor”, which checks for the schedule records and executes them.

- Select **Use field in record + interval** for the Calculation Type field
- Select **Resolved Time** for the Calc Field field

  Note: Make sure the resolve time of the incident is set correctly while the incident is resolved, because the schedule execution time is calculated based on the value of that field.

- Set **7 00:00:00** for the Calc Interval field. This means the scheduled action is executed after seven days.
- Set the Action Condition as follows. The condition is checked when the scheduled time arrives. In this example, it is seven days after the incident is resolved. Only when the status of the incident is still Resolved, and the priority is still low at the time, the action is executed; otherwise the action is ignored.

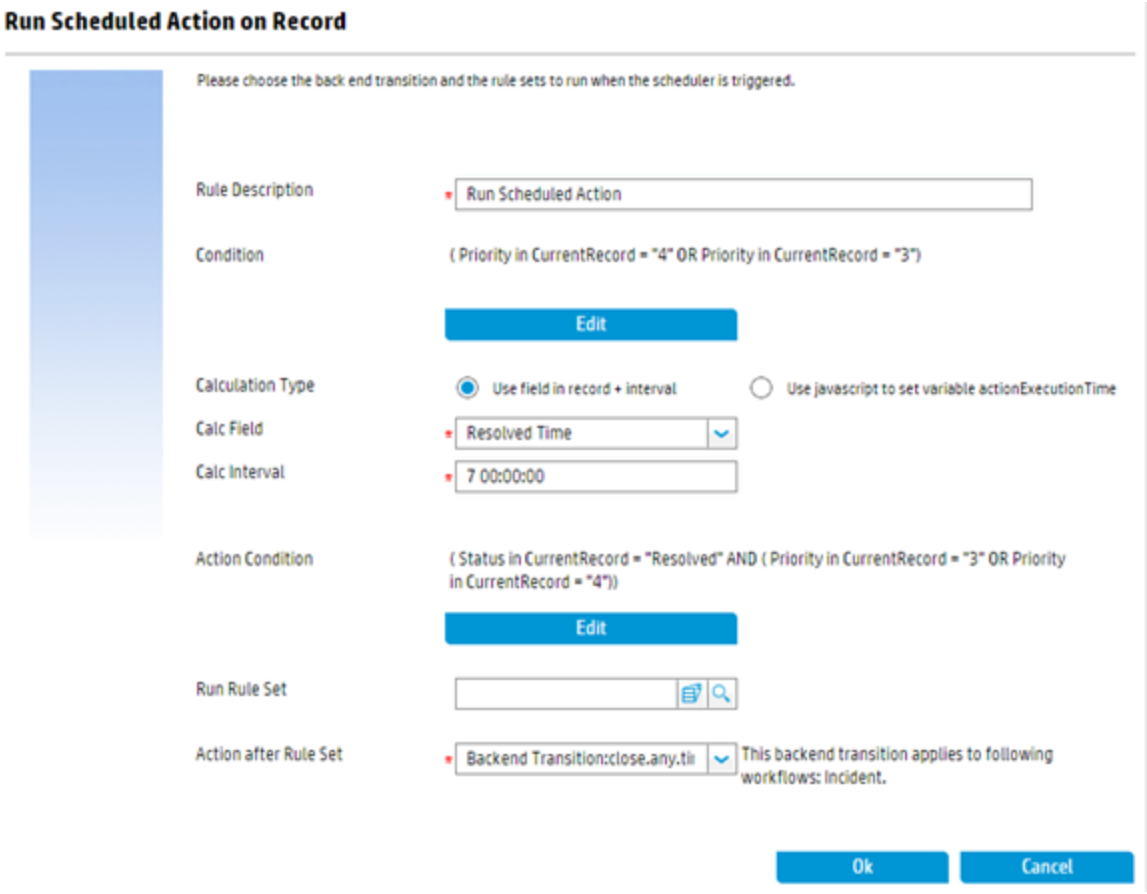


- Keep the Run Rule Set field empty
- Select **Backend Transition:close.any.time** for the "Action after Rule Set" field. The save action is executed against the current record by the back end scheduler process when the scheduled time arrives.

Note: You can select **Do nothing**, **Save**, or one of the back end transitions.

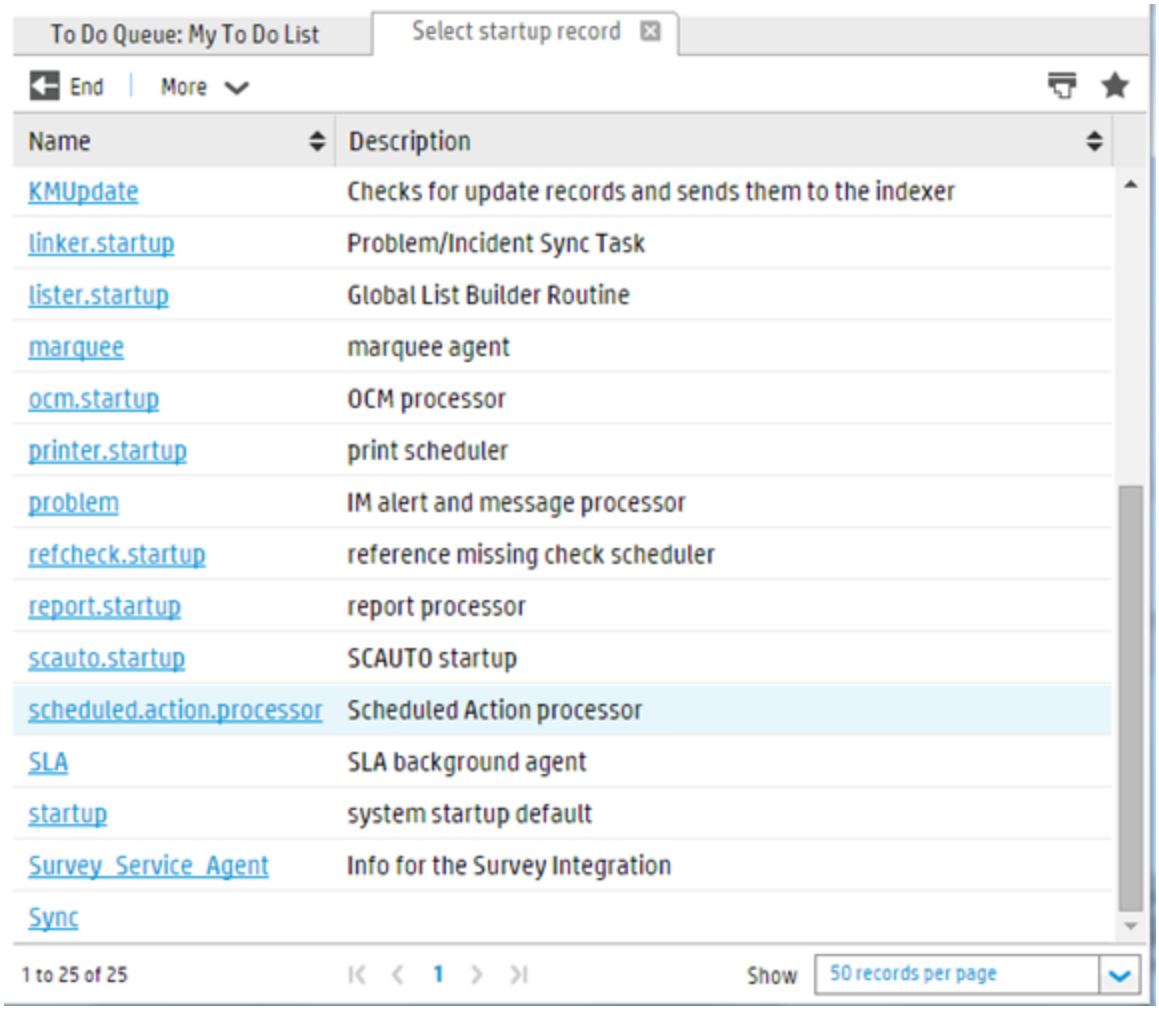
- The back end transition list is retrieved from all the workflows of the selected record file. Similar to Configure Run Action rules on related records, at run time, if some of the other records do not have your selected back end transition, the corresponding transition is not run.
- You are suggested to use the same back end transition name in different workflows if these backend trasitions follow the same business logic.
- While a back end transition is selected from the list, the interface is updated to indicate to which workflows the transition applies and to which it does not apply.

The form on which you can perform these tasks is shown below:



Note: The Run Scheduled Action rule does not support selecting work schedule and time zone in this release.

To make the Run Scheduled Action rules work at run time, make sure the back end processor "Scheduled Action processor" is started, as shown below:

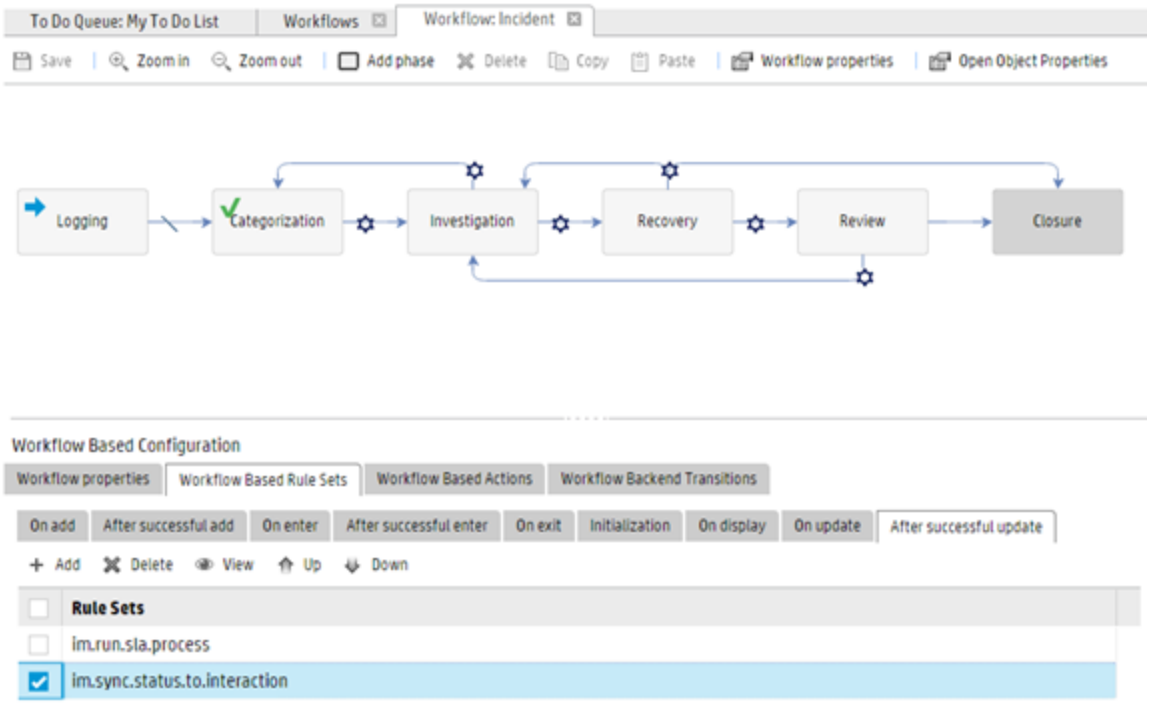


Combine the Run Action rules and the Run Scheduled Action rules

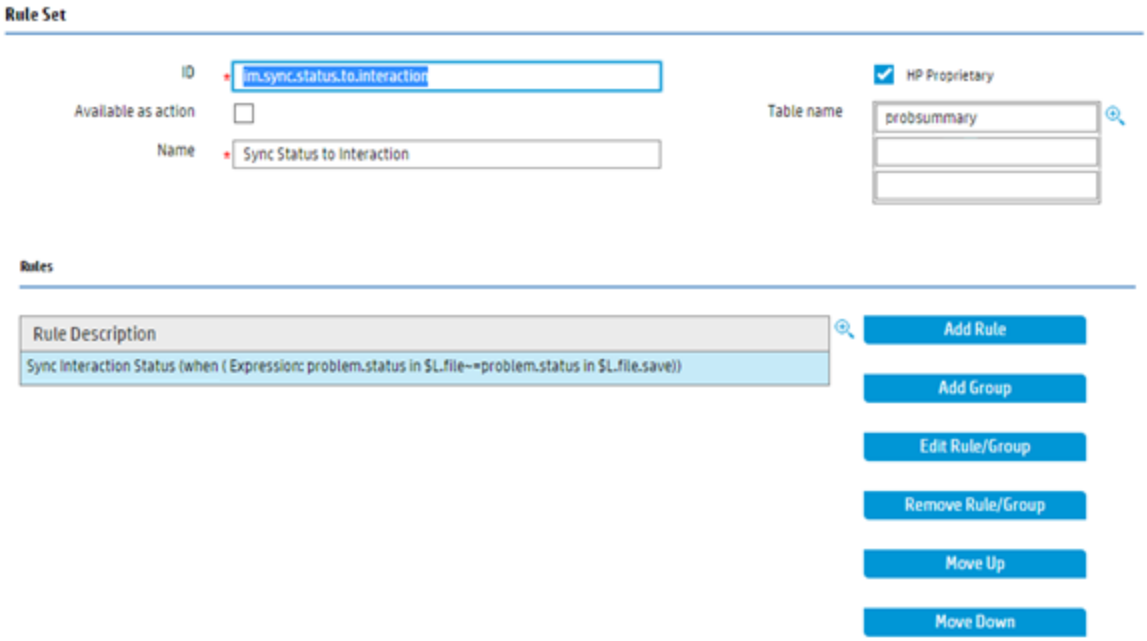
In Service Manager, the Run Action rules are executed in a sequential manner. Take the related records as an example, if one of the rules sets or actions against the related records fails, the whole execution stops and return a failure result. To ensure successful execution, you can combine the Run Action rules

and the Run Scheduled Action rules, so that the execution can be scheduled if it fails due to record locking.

In the out-of-box configuration, when the status of an incident changes, the status of the related interaction also changes accordingly. To achieve this, a rule set “im.sync.status.to.interaction” is configured in the “After successful update” event of the Incident workflow, as shown below:



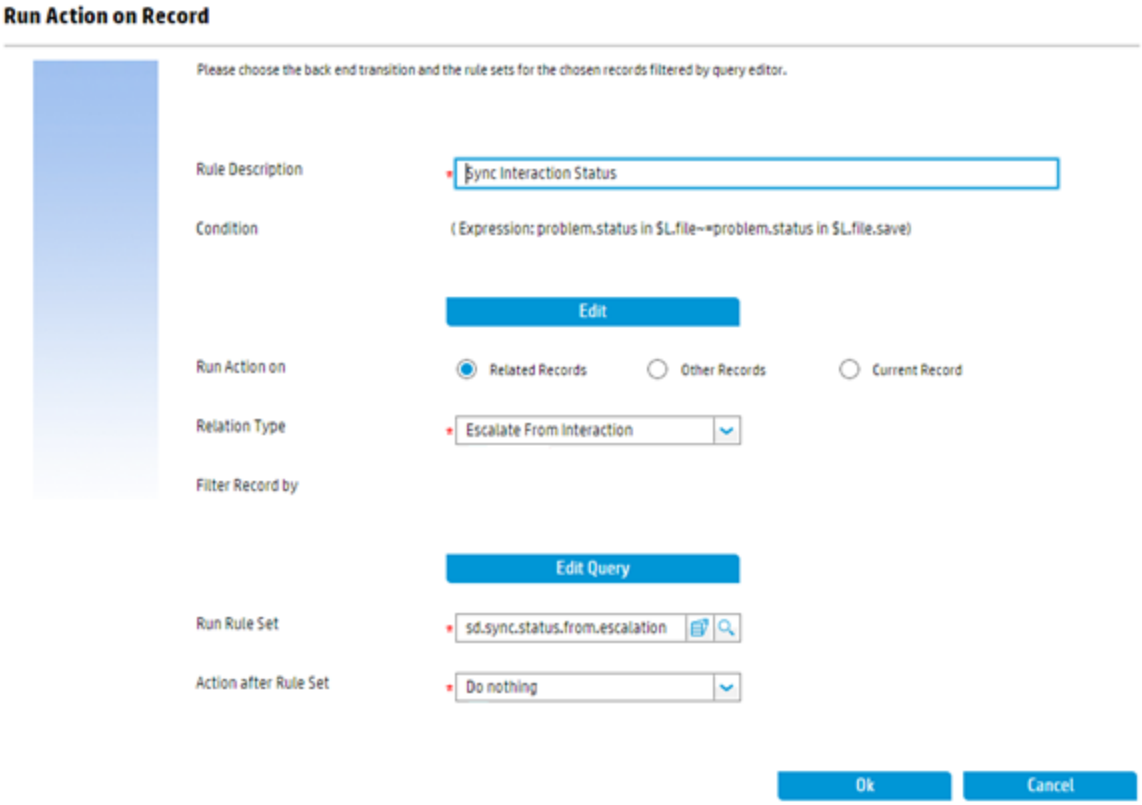
This rule set is configured on the "probsummary" table that stores incident records:



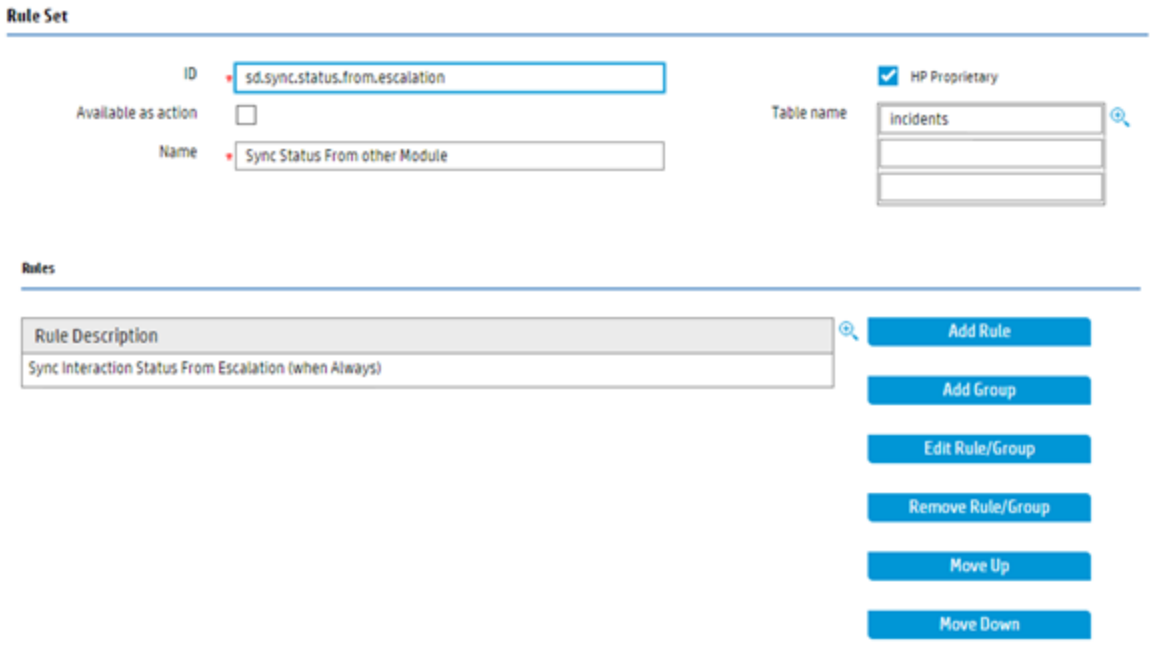
In addition, you also need to configure the Run Action rules in the rule set as follows:

- Name the condition something like "Sync Interaction Status", which makes it easy for you to understand the purpose of the condition
- Select **Related Records** for the "Run Action on" field
- Select **Escalate From Interaction** for the "Relation Type" field. This means the target related record is the interaction from which incident is escalated.
- Select **sd.sync.status.from.escalation** for the "Run Rule Set" field. This rule set is executed against the related interaction. To handle the operation failure on locked interaction, this rule set must contain the Run Scheduled Action rule.
- Select **Do nothing** for the "Action after Rule Set" field. Actions like the save operation on target records are configured in the “sd.sync.status.from.escalation” rule set.

The form on which you can perform these tasks is shown below:



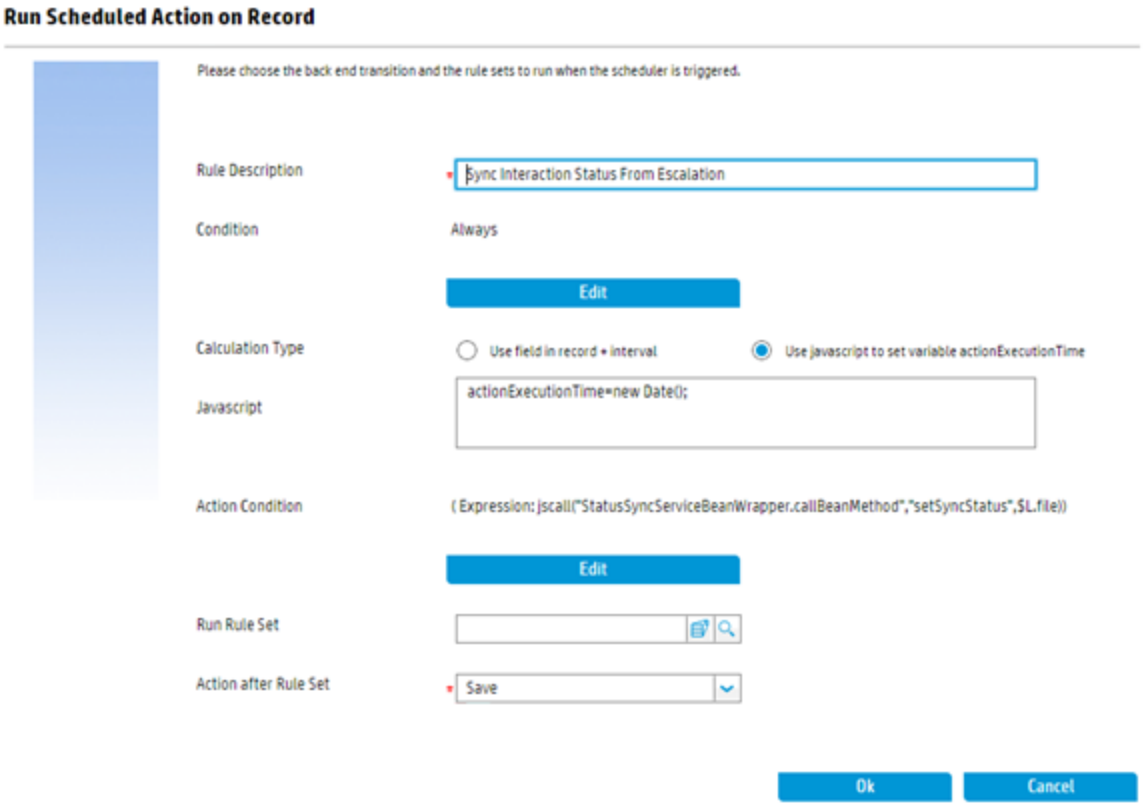
After defining the rule set on the source table, you need to define the rule set on the target table. The rule set on the target table is used as the value of the "Run Rule Set" field when you define the rule set for the source table. In this example, you define the rule set “sd.sync.status.from.escalation” on the target table “incidents” that stores the interaction records:



In this rule set, a Scheduled Action rule is configured to sync up the status of the incident and the status of the interaction. The configuration is as follows:

- Specify the scheduled execution time as the current time by using JavaScript. This ensures the schedule is executed as early as possible, that is, it is executed the next time the back end scheduler "Scheduled Action processor" is activated.
- Select **Escalate From Interaction** as the Relation Type. This means the target related record is the interaction from which the incident is escalated.
- Specify the Action condition to call a JavaScript method `StatusSyncServiceBeanWrapper.callBeanMethod("setSyncStatus",$L.file)`. All the main logic is included in this JavaScript method.
  - If status synchronization is needed and successful, this method returns true, and then the Save action is executed.
  - If status synchronization is not needed, this method returns false, and then the Save action is ignored automatically.
- Keep the Run Rule Set field empty.
- Select **Save** for the "Action after Rule Set"field. While the record is saved, if it is locked, the saving operation fails, and a schedule record is rescheduled instead of being deleted.

Note: Variables, which are referred to as `srcRecord`for the current source record and as `oldSrcRecord` for the original source record in JavaScript, are not supported if the Run Scheduled Action rules are used.



Best practices for configuring task planner

Task Planner enables you to schedule tasks in HP Service Manager modules such as Change Management or Request Management. This section provides information about the best practices for using the Task Planner. For example, you can configure shared information across multiple tasks, or you can use record context to drive workflows.

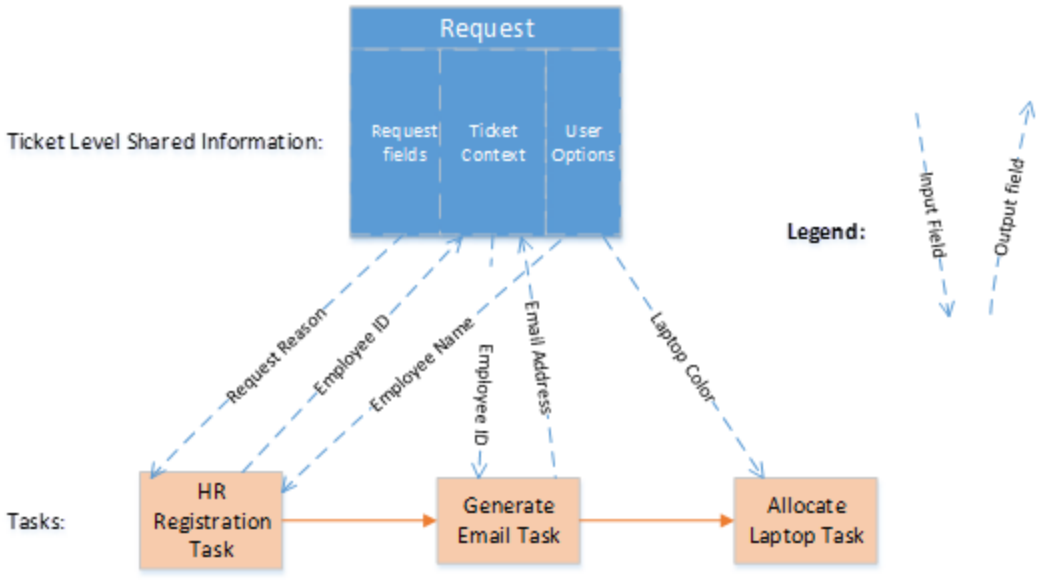
Configure shared information across tasks

You can use Task Planner to configure information shared by multiple tasks.

The following three types of information from a record can be shared by the tasks of a record:

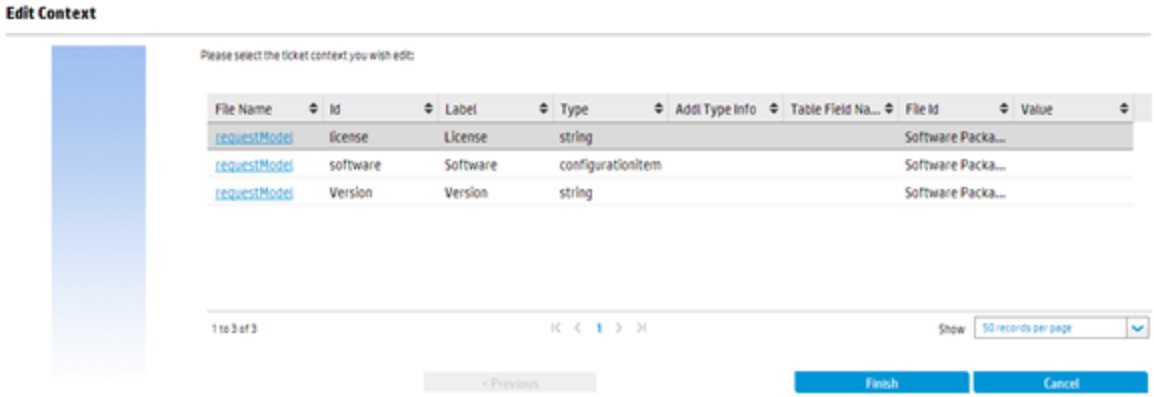
- Record fields, for example, the request fields or the change fields
- User selections, which are input by the end user when they use Service Manager
- Record context, which are fields defined in Task Planner. The values of these fields are populated during task implementation and then shared across the tasks of a record.

The following diagram demonstrates how these three types of information of a Request is shared among its tasks.

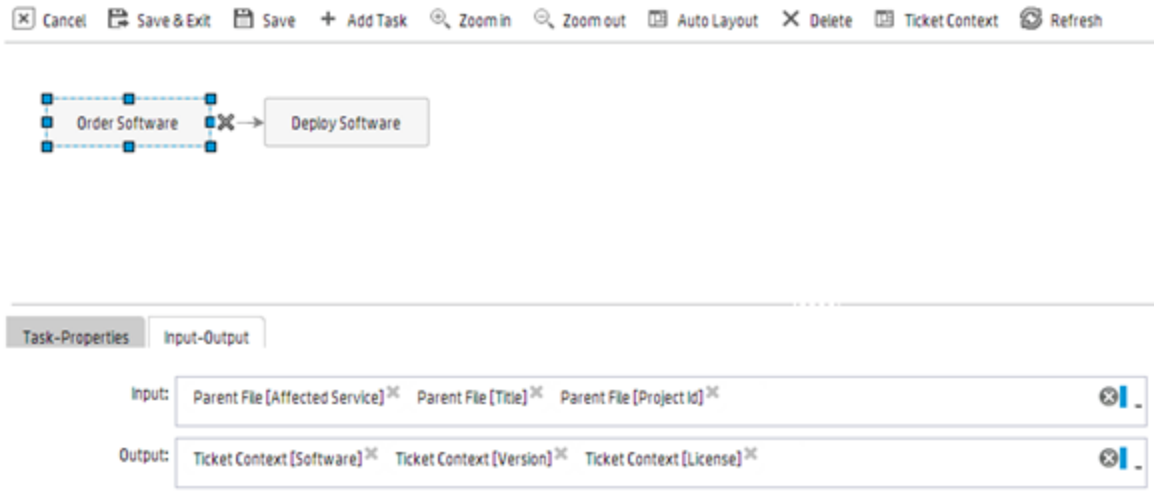


Each task can have its own input and output data. The input fields are used to receive data from other tasks, and can trigger the execution of the current task. The output fields are used to populate shared data to the record during task execution, and the data can then be shared by other tasks.

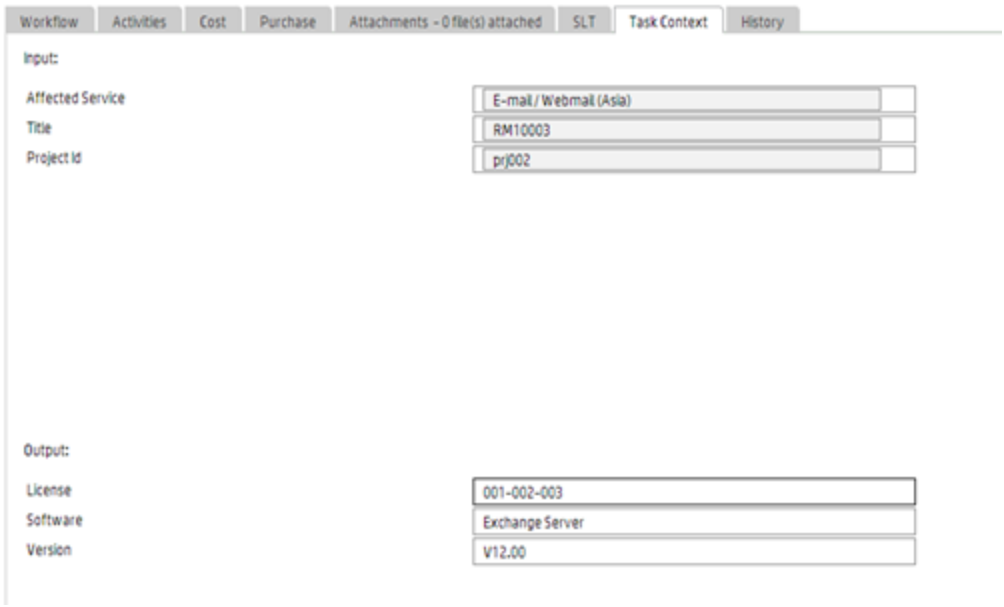
You can launch the Record Context (Ticket Context) definition page from the Task Planner. The page is shown as follows:



On the definition page of the Task Planner, you can configure the input and output fields. The Record Context fields of a record can be used as the input or output fields of a task, and the output fields can be used in subsequent tasks as the input fields.



The following Task Context page shows the run-time result of a task, which contains the input and output fields/values. The input fields are read-only, while the output fields can be edited with appropriate values that can be populated to the next task later.



Use record context fields in Condition Editor

The Condition Editor does not support using the value of a record context field directly as part of the condition, so you have to manually retrieve the value and put it into a Service Manager variable, and then you can use this variable in Condition Editor.

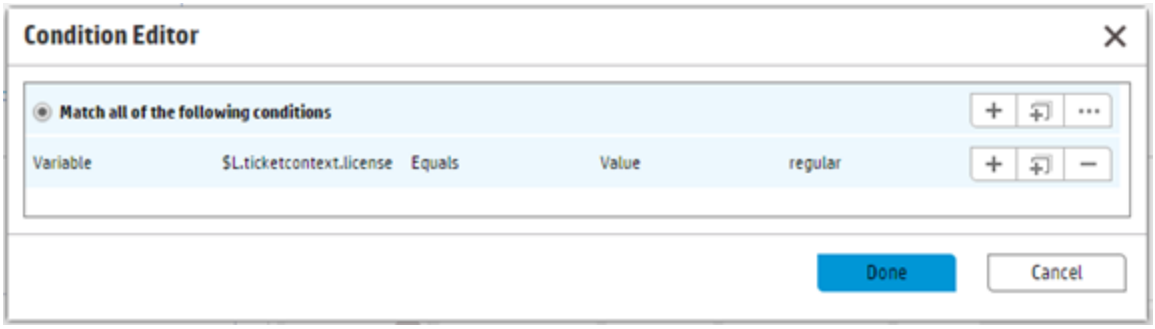
For example, if you want to use the record context field “employeeType” as part of the condition in the Request workflow, you need to execute the following JavaScript code before your condition is evaluated:

vars[“$L.ticketcontext.license”] = lib.c.$("#taskPlannerService").getTicketContextFieldValue("request", record.number, "employeeType");

Or you can use the JavaScript below in the Request Task workflow:

vars[“$L.ticketcontext.license”] = lib.c.$("#taskPlannerService").getTicketContextFieldValue("request", record.parent_request, "employeeType");

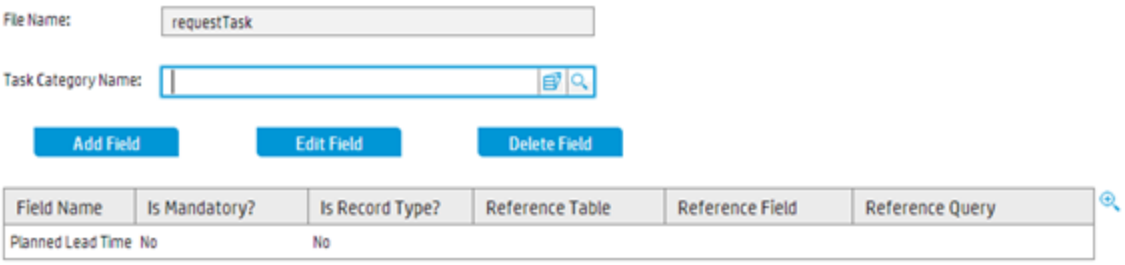
Finally, you can configure the condition to something in the Condition Editor as shown below:



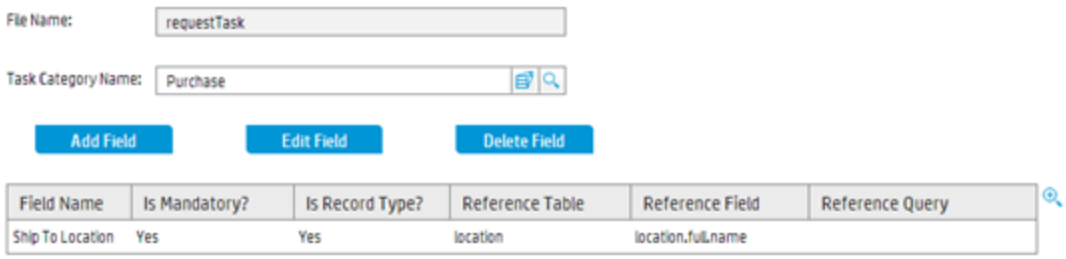
Use predefined additional task fields

When you plan a task in Task Planner, you might want to use fields that are not available in the out-of-box configuration. In this case, you can define new fields for Task Planner at the object level or category level.

For example, in the following screenshot you can see that a field is defined at the object level – if you do not specify any value for the Task Category Name field, the field is defined at the object level, which applies to all category tasks.

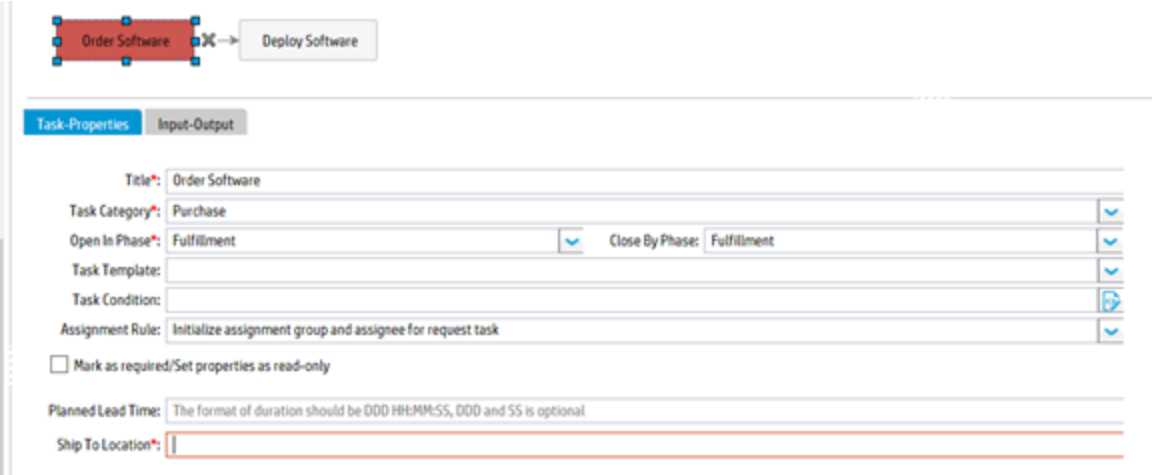


And as shown in the following screen shot, if you specify a value for the Task Category Name field, the field is defined at the category level, which applies only to this task category.



At run time, the predefined fields at both the object level and the category level are merged together and displayed in Task Planner. If a field is defined at both levels, the definition at the category level is used.

After being defined, the predefined additional task fields are displayed at the bottom of the Task-Properties tab. Your entered data is inserted into the task that is created in Task Planner.

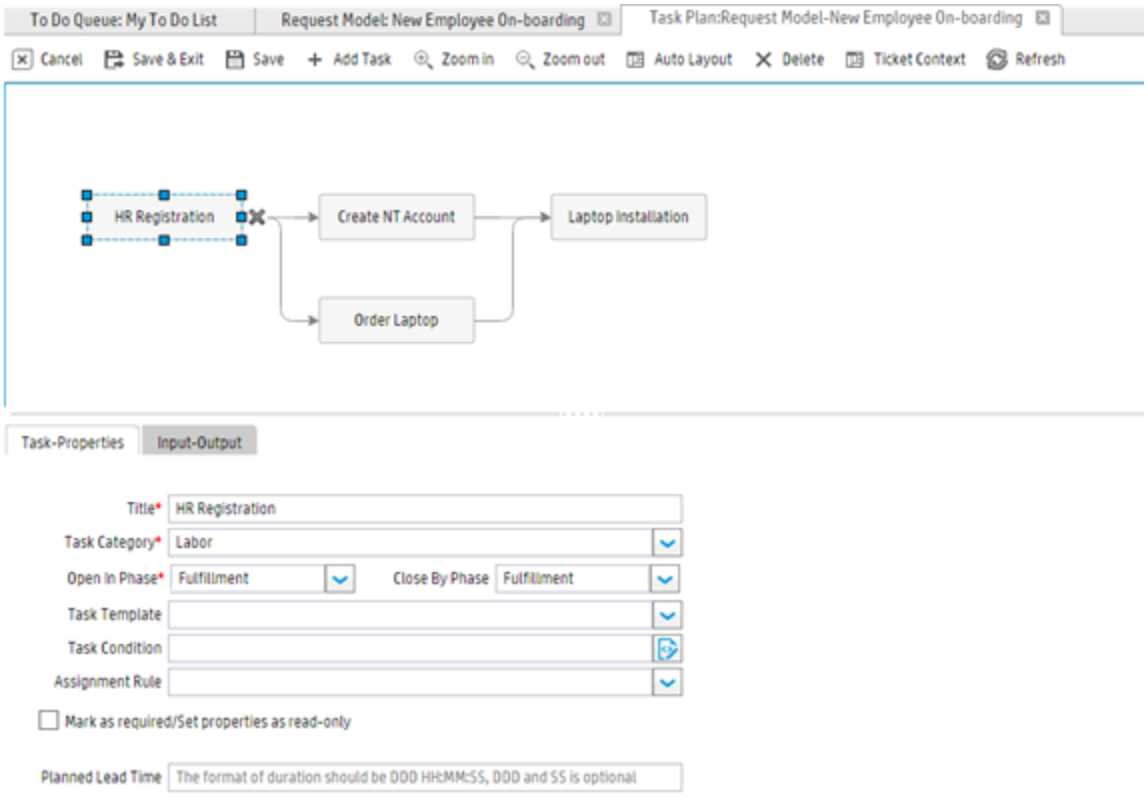


Plan appropriate number of tasks in a same open phase

When you define a task plan, you can specify the “Open in Phase” field of a task, which means this task is created at run time when the workflow of the record moves to the phase. Currently all the tasks you plan with the same “Open in Phase” value are created immediately when the record moves to the corresponding phase.

Too many tasks for a phase may result in decreased performance. For example, if you plan more than 10 tasks, the operator might feel obvious performance downgrade when he moves the record to that phase. To ensure the best performance, it is recommended that no more than 10 tasks are automatically started once a phase is entered.

Note: The performance of generating tasks in Task Planner is not only affected by the number of planned tasks, but also by the business logic you plan for task creation. The more complex and time-consuming the logic is, the more performance downgrade you might encounter.


To Do Queue: My To Do List
Request Model: New Employee On-boarding
Task Plan:Request Model-New Employee On-boarding
Cancel
Save & Exit
Save
Add Task
Zoom in
Zoom out
Auto Layout
Delete
Ticket Context
Refresh
HR Registration
Create NT Account
Laptop Installation
Order Laptop
Task-Properties
Input-Output
Title*
HR Registration
Task Category*
Labor
Open In Phase*
Fulfillment
Close By Phase
Fulfillment
Task Template
Task Condition
Assignment Rule
Mark as required/Set properties as read-only
Planned Lead Time
The format of duration should be DDD HH:MM:SS, DDD and SS is optional

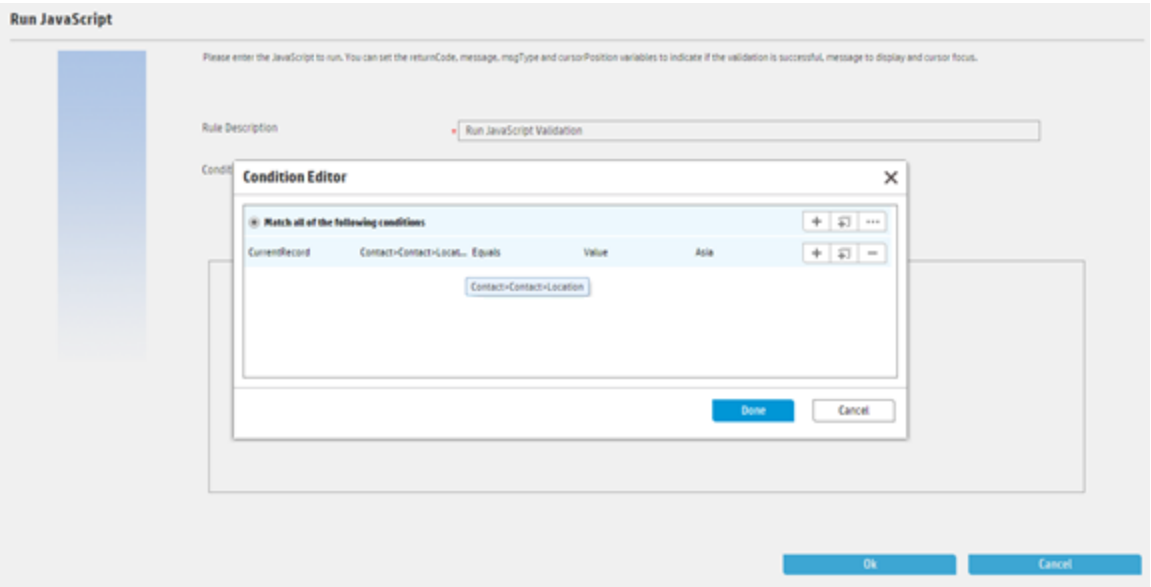
Best practices for using the Condition Editor

The Condition Editor enables you to build a condition without any knowledge of programming languages. Conditions always evaluate to True or False. When a condition evaluates to True, the system runs the rule or applies an action that the condition controls. This section provides information about the best practices of using the Condition Editor.

Use cross-table fields in Condition Editor

Cross-table fields, which are defined by a reference record, are the fields of related tables. The table relationship is maintained by the Relation Manager in Service Manager. You can use cross-table fields in Condition Editor to configure various conditions.

To configure the cross-table fields as part of a condition, you can select the cross-table field names in Condition Editor. However, using cross-table fields might be time-consuming because the system needs to prepare the variable of the reference record at run time. For example, an incident record has the field “Contact”, which refers to the Contact record. You can configure the Rule condition to use the Location of the Contact record directly, as shown below:



At run time, the expression is evaluated to something as follows:

```
location in $L.file.contact.name.contacts.contact.name="Asia"
```

The variable `$L.file.contact.name.contacts.contact.name` is prepared by the rule engine.

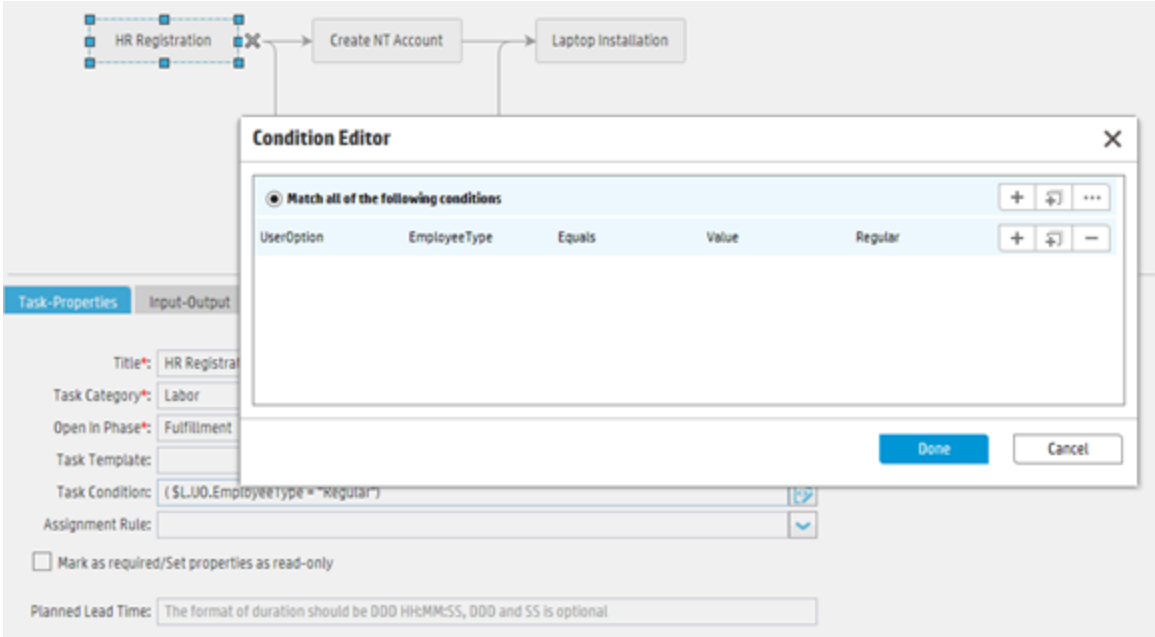
If you use the Condition in your own scenario, make sure to invoke the JavaScript method below at run time before evaluating the condition expression, which automatically prepares the cross-table reference record for you:

```
lib.Workflow.initVarForCondition(conditionXml);
```

Use user option fields in Condition Editor

You can configure the User Option fields as part of a condition by specifying the names and values of the user option fields in Condition Editor.

However, using the user option fields might be time-consuming because the system needs to prepare variables for the user option fields at run time. For example, you configure to use the User Option as part of a task condition in Task Planner as follows:



At run time, the expression is evaluated to something as follows:

```
$L.UO.EmployeeType="Regular"
```

The variable `$L.UO.EmployeeType` is prepared by the system.

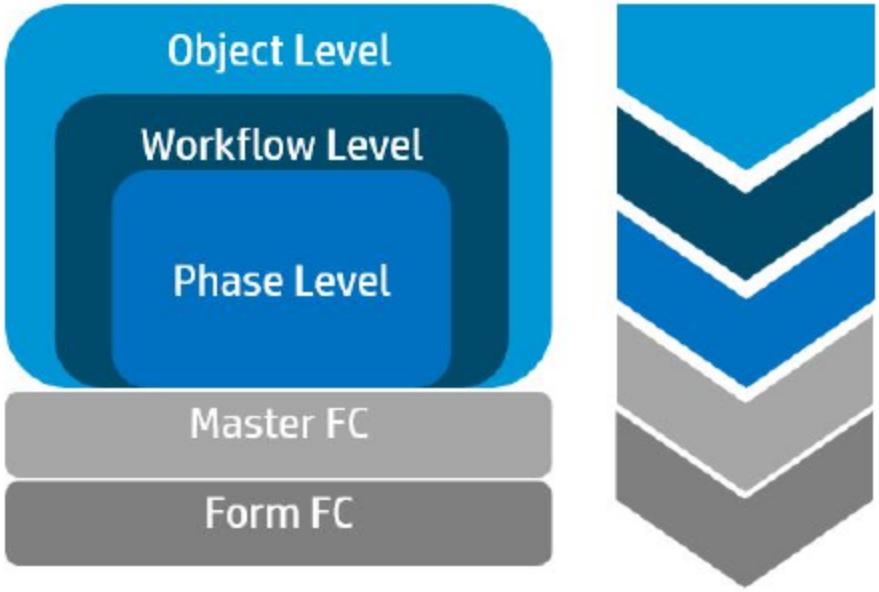
If you use the condition in your own scenario, make sure that you invoke the JavaScript method below at run time before evaluating the condition expression, which automatically prepares the user option variable for you:

```
lib.Workflow.initVarForCondition(conditionXml);
```

Note: The instance data of a user option is stored as a string value in the file userOption, and thus at run time, when the user option value is evaluated, it is also treated as a string value.

Consider rule set execution order

A rule set contains a list of rules that you may run against a record. Rules implement business logic to drive a workflow or a process. In Service Manager, you can define rule sets at the object level, workflow level, or phase level. The rule sets defined at different levels are executed in order. For example, rule sets defined at the object level are executed first. The following diagram illustrates this execution order (from top to bottom):



For each action in Service Manager, such as creating a record, rule sets defined at different levels might be executed. The following table provides a high level view of the execution order of these rule sets:

Action	Execution order of the rule sets (from left to right)
Create a record	On add, On enter, Format control (add), After successful add, After successful enter, Format control (subroutine after add), Initialization, Format control (initial), On display, Format Control (display)
Update a record	On update, Format control (update), After successful update, Format control (subroutine after update), Initialization, Format control (initial), On display, Format control (display)
Search and access a record	Initialization, Format control (initial), On display, Format control (display)

Action	Execution order of the rule sets (from left to right)
Fill, find, screen-redraw	On display, Format control (display)
• Move from one phase to another • Close record (inactive)	On exit (global + old phase), Transition, On enter (global + new phase), Format control (update), Format control (subroutine after update), Initialization, Format control (initial), On display, Format control (display) **Note**: The term "global" means rule sets defined at the object level or the workflow level.

When you define your own rule sets in Service Manager, you might need to consider the execution order of the rule sets, so that you can optimize the definition of your own rule sets accordingly.

The following tables provides a more detailed view of the execution order of the rule sets that are executed for each action. The events and their corresponding rule sets are executed from top to bottom.

Open an existing record

Events	Rule sets
Initialization	• Initialization rule sets at the object level • Initialization rule sets at the workflow level • Initialization rule sets at the phase level • Master format control initialization • Format control initialization for the form of the phase
Display	• On-display rule sets at the object level • On-display rule sets at the workflow level • On-display rule sets at the phase level • Master format control display • Format control display for the form of the phase

Create a new record (launched from document.new)

Events	Rule sets
Initialization	• Initialization rule sets at the object level • Initialization rule sets at the workflow level • Initialization rule sets at the phase level • Master format control initialization • Format control initialization for the form of the phase
Display	• On-display rule sets at the object level • On-display rule sets at the workflow level • On-display rule sets at the phase level • Master format control display • Format control display for the form of the phase

Add a record and stay in the logging phase

Events	Rule sets
On-add	• On-add rule sets at the object level • On-add rule sets at the workflow level • On-enter rule sets at the object level • On-enter rule sets at the workflow level • On-enter rule sets at the phase level • Master format control on-add • Format control on-add for the form of the phase

Events	Rule sets
Add the record to the database **Note:** This is a user action, not an event.	N/A
After-add	• After-successful-add rule sets at the object level • After-successful-add rule sets at the workflow level • After-successful-enter rule sets at the object level • After-successful-enter rule sets at the workflow level • After-successful-enter rule sets at the phase level • Master format control subroutine after-add • Format control subroutine after-add for the form of the phase
Initialization	• Initialization rule sets at the object level • Initialization rule sets at the workflow level • Initialization rule sets at the phase level • Master format control initialization • Format control initialization for the form of the phase
Display	• On-display rule sets at the object level • On-display rule sets at the workflow level • On-display rule sets at the phase level • Master format control display • Format control display for the form of the phase

Add a record and automatically move to next phase

Events	Rule sets
Old phase on-add	• On-add rule sets at the object level • On-add rule sets at the workflow level • On-enter rule sets at the object level • On-enter rule sets at the workflow level • On-enter rule sets at the phase level • Master format control on-add • Format control on-add for the form of the phase
To phase on-enter	• On-exit rule sets at the object level • On-exit rule sets at the workflow level • On-exit rule sets in the "from phase" • Transition rule sets • On-enter rule sets at the object level • On-enter rule sets at the workflow level • On-enter rule sets in the "to phase" **Note:** In this event, if you need to use the current phase of the record in the rule sets or format control, the value of the current.phase field in the "current record" is the name of the "to phase," and the value of the current.phase field in the "saved record" is the name of the "from phase."
Add the record to the database **Note:** This is a user action, not an event.	N/A
From phase after-add	• After-successful-add rule sets at the

Events	Rule sets
	object level • After-successful-add rule sets at the workflow level • After-successful-enter rule sets at the object level • After-successful-enter rule sets at the workflow level • After-successful-enter rule sets at the phase level • Master format control subroutine after-add • Phase form's format control subroutine after-add
To phase post	• After-successful-enter rule sets at the object level • After-successful-enter rule sets at the workflow level • After-successful-enter rule sets at the phase level • Master format control subroutine after-add • Phase form's format control subroutine after-add
To phase initialization	• Initialization rule sets at the object level • Initialization rule sets at the workflow level • Initialization rule sets at the phase level • Master format control initialization • Format control initialization to the form of the phase

Events	Rule sets
To phase display	• On-display rule sets at the object level • On-display rule sets at the workflow level • On-display rule sets at the phase level • Master format control display • Format control display to the form of the phase

Save a record and stay in current phase

Events	Rule sets
On-update	• On-update rule sets at the object level • On-update rule sets at the workflow level • On-update rule sets at the phase level • Master format control on-add • Format control on-add for the form of the phase
Save the record to the database **Note:** This is a user action, not an event.	N/A
After-update	• After-successful-update rule sets at the object level • After-successful- update rule sets at the workflow level • After-successful-update rule sets at the phase level • Master format control subroutine after-update • Format control subroutine after-update for the form of the phase

Events	Rule sets
Initialization	• Initialization rule sets at the object level • Initialization rule sets at the workflow level • Initialization rule sets at the phase level • Master format control initialization • Format control initialization for the form of the phase
Display	• On-display rule sets at the object level • On-display rule sets at the workflow level • On-display rule sets at the phase level • Master format control display • Format control display for the form of the phase

Save a record and automatically move to the next phase

Events	Rule sets
From phase on-update	• On-update rule sets at the object level • On-update rule sets at the workflow level • On-update rule sets at the phase level • Master format control on-update • Format control on-update for the form of the phase
To phase on-enter	• On-exit rule sets at the object level • On-exit rule sets at the workflow level • On-exit rule sets in the "from phase"

Events	Rule sets
	• Transition rule sets • On-enter rule sets at the object level • On-enter rule sets at the workflow level • On-enter rule sets in the "to phase" **Note:** In this event, if you need to use the current phase of the record in the rule sets or format control, the value of the current.phase field in the "current record" is the name of the"to phase", and the value of the current.phase field in the "saved record" is the name of the "from phase".
Update the record to the database **Note:** This is a user action, not an event.	N/A
From phase after-update	• After-successful-update rule sets at the object level • After-successful-update rule sets at the workflow level • After-successful-update rule sets at the phase level • Master format control subroutine after-update • Format control subroutine after-update for the form of the phase
To phase post	• After-successful-enter rule sets at the object level • After-successful-enter rule sets at the workflow level • After-successful-enter rule sets in the "to phase"

Events	Rule sets
	• Master format control subroutine after-update • Format control subroutine after-update from the form of the phase
To phase initialization	• Initialization rule sets at the object level • Initialization rule sets at the workflow level • Initialization rule sets at the phase level • Master format control initialization • Format control initialization to the form of the phase
To phase display	• On-display rule sets at the object level • On-display rule sets at the workflow level • On-display rule sets at the phase level • Master format control display • Format control display to the form of the phase

Manual transition without selecting “Save record prior to executing the transition”

Events	Rule sets
To phase on-enter	• On-exit rule sets at the object level • On-exit rule sets at the workflow level • On-exit rule sets in the "from phase" • Transition rule sets • On-enter rule sets at the object level

Events	Rule sets
	• On-enter rule sets at the workflow level • On-enter rule sets in the "to phase" • Master format control On-update • Format control on-update from the form of the phase **Note:** In this event, if you need to use the current phase of the record in the rule sets or format control, the value of the current.phase field in the "current record" is the name of the"to phase", and the value of the current.phase field in the "saved record" is the name of the "from phase".
Save the record to the database **Note:** This is a user action, not an event.	N/A
To phase after-enter	• After-successful-enter rule sets at the object level • After-successful-enter rule sets at the workflow level • After-successful-enter rule sets in the "to phase" • Master format control subroutine after-update • Format control subroutine after-update from the form of the phase

Manual transition with “Save record prior executing the transition” selected

Events	Rule sets
From phase on-update	• On-update rule sets at the object level • Workflow level on-update rule sets • On-update rule sets in the "from phase" • Master format control On-update • Format control on-update from the form of the phase
Save the record to the database **Note:** This is a user action, not an event.	N/A
From phase after-update	• After-successful-update rule sets at the object level • After-successful-update rule sets at the workflow level • After-successful-update rule sets in the "from phase" • Master format control subroutine after-update • Format control subroutine after-update from the form of the phase
To phase on-enter	• On-exit rule sets at the object level • On-exit rule sets at the workflow level • On-exit rule sets in the "from phase" • Transition rule sets • On-enter rule sets at the object level • On-enter rule sets at the workflow level

Events	Rule sets
	• On-enter rule sets in the "to phase" • Master format control on-update • Format control on-update from the form of the phase **Note:** In this event, if you need to use the current phase of the record in the rule sets or format control, the value of the current.phase field in the "current record" is the name of the"to phase", and the value of the current.phase field in the "saved record" is the name of the "from phase".
Save the record to the database **Note:** This is a user action, not an event.	N/A
To phase after-enter	• After-successful-enter rule sets at the object level • After-successful-enter rule sets at the workflow level • After-successful-enter rule sets in the "to phase" • Master format control subroutine after-update • Format control subroutine after-update from the form of the phase

Back end transition

Events	Rule sets
From phase initialization	• Initialization rule sets at the object level

Events	Rule sets
	• Initialization rule sets at the workflow level • Initialization rule sets in the "from phase" • Master format control initialization • Format control initialization from the form of the phase
To phase on-enter	• On-exit rule sets at the object level • On-exit rule sets at the workflow level • On-exit rule sets in the "from phase" • Transition rule sets • On-enter rule sets at the object level • On-enter rule sets at the workflow level • On-enter rule sets in the "to phase" • Master format control On-update • Format control on-update from the form of the phase
Save the record to the database **Note:** This is a user action, not an event.	N/A
To phase after-enter	• After-successful-enter rule sets at the object level • After-successful-enter rule sets at the workflow level • After-successful-enter rule sets in the "to phase" • Master format control subroutine after-update • Format control subroutine after-update from the form of the phase

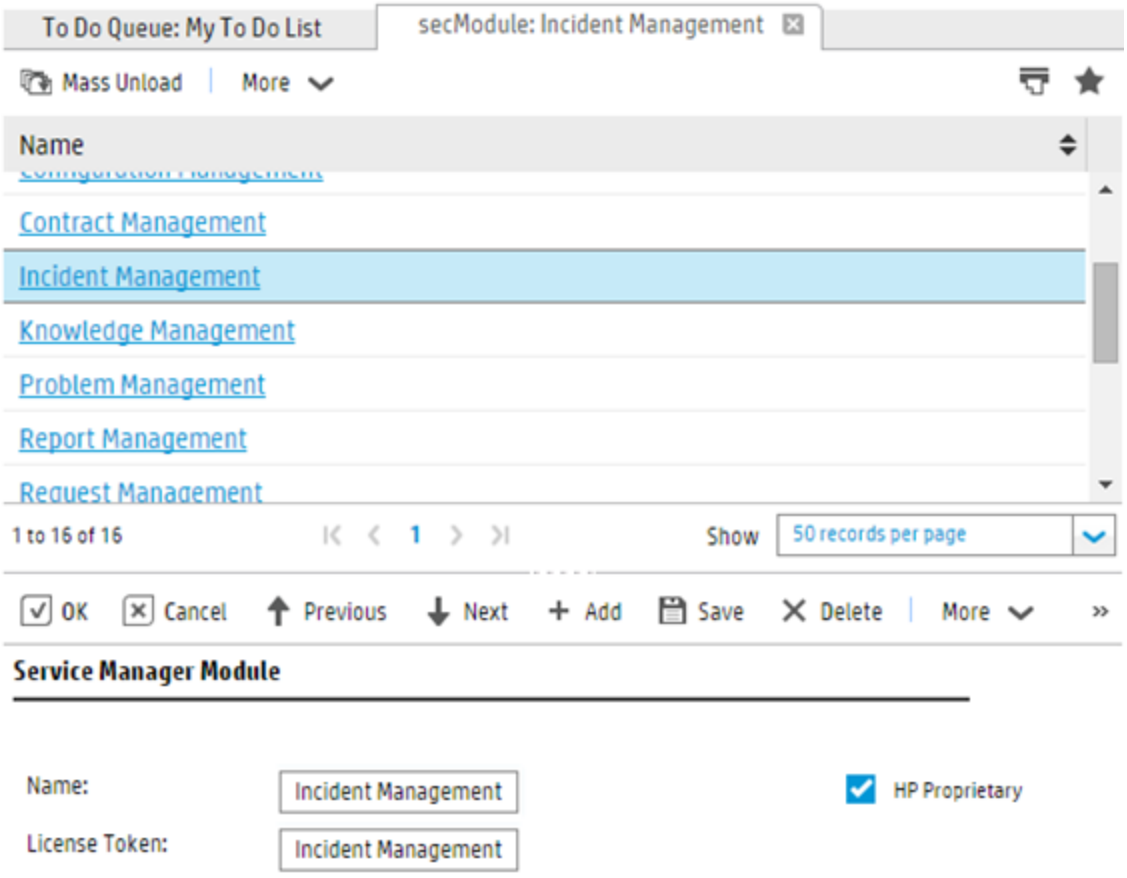
Best practices for tailoring Service Manager Codeless module

This chapter describes the best practices and recommendations for creating Service Manager Codeless module from scratch or for tailoring an out-of-box Service Manager Codeless module.

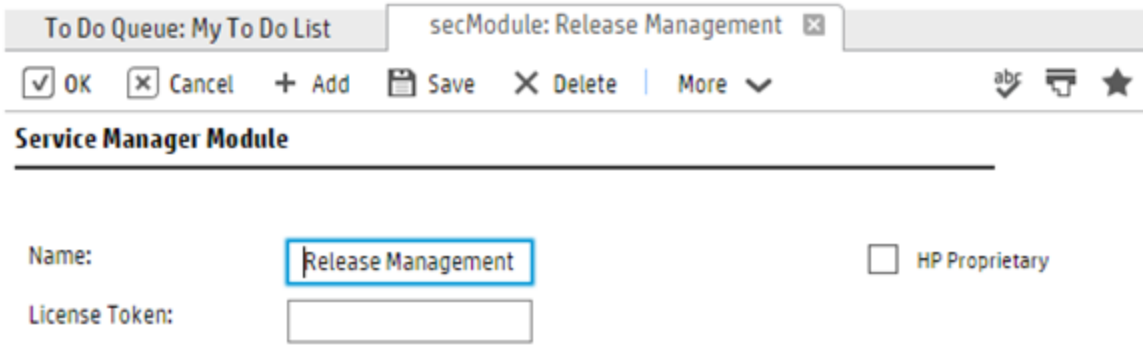
Create a Security module

Note: If you are going to use an existing Service Manager Codeless module rather than adding a new Service Manager Codeless module, ignore this section.

Service Manager does not include a menu entry for the Security module. To access the Security module, run the **db** command in Service Manager, and then select the secModule table to access the Security module.



If you want to add a new Service Manager Codeless module to Service Manager, you must first add a new Security module. Usually, each Service Manager Codeless module maps to a Security module. For example, if you want to add a new Service Manager Codeless Release Management module, you must also add a Release Management security module.



Configure Security Areas for the module

A security area defines a specific functional area within Service Manager, such as Incident, Incident Task, or Incident Management Configuration. Each security area definition includes default rights that are copied to the role whenever a new role is created.

Usually, you must create the following Security Areas for Service Manager Codeless module:

- A Security Area that contains the default security rights and settings for tickets
- A Security Area that contains the default security rights and settings for the module's Management configuration. For example, the security rights for maintaining the Categories, Approval Definitions, and so on
- A Security Area that contains the default security rights and settings for request tasks (only if your Service Manager Codeless module includes tasks)

The default rights and settings in the Security Area are copied to new roles that are created for this area. However, the values of specific settings are inherited only if no value is specified for those settings in the role.

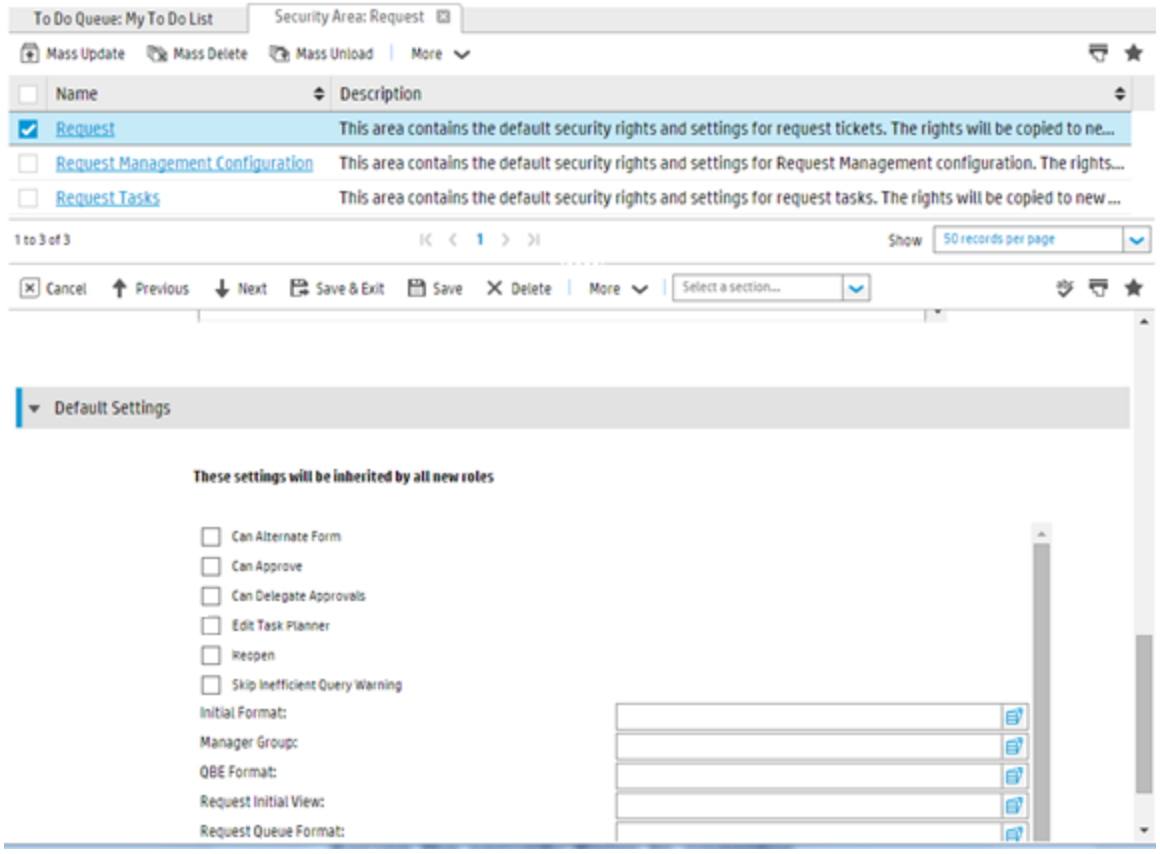
For example, an out-of-box Request module contains the following Security Areas:

- Request
- Request Management Configuration
- Request Tasks

Configure the Security Area settings

If you need to configure additional security rights in addition to the default rights (such as View, New, Update, Delete/Close, and Modify Template) to control a Security Area, you can modify the Security Area settings.

For example, out-of-box Request modules include a security control that determines whether users have the right to edit the task planner for a record. You can add settings such as this to the Request Area.



Configure the Security Roles

Security Roles are groups of rights in Security Areas.

Note: Security Roles replace the profiles in Service Manager Classic modules.

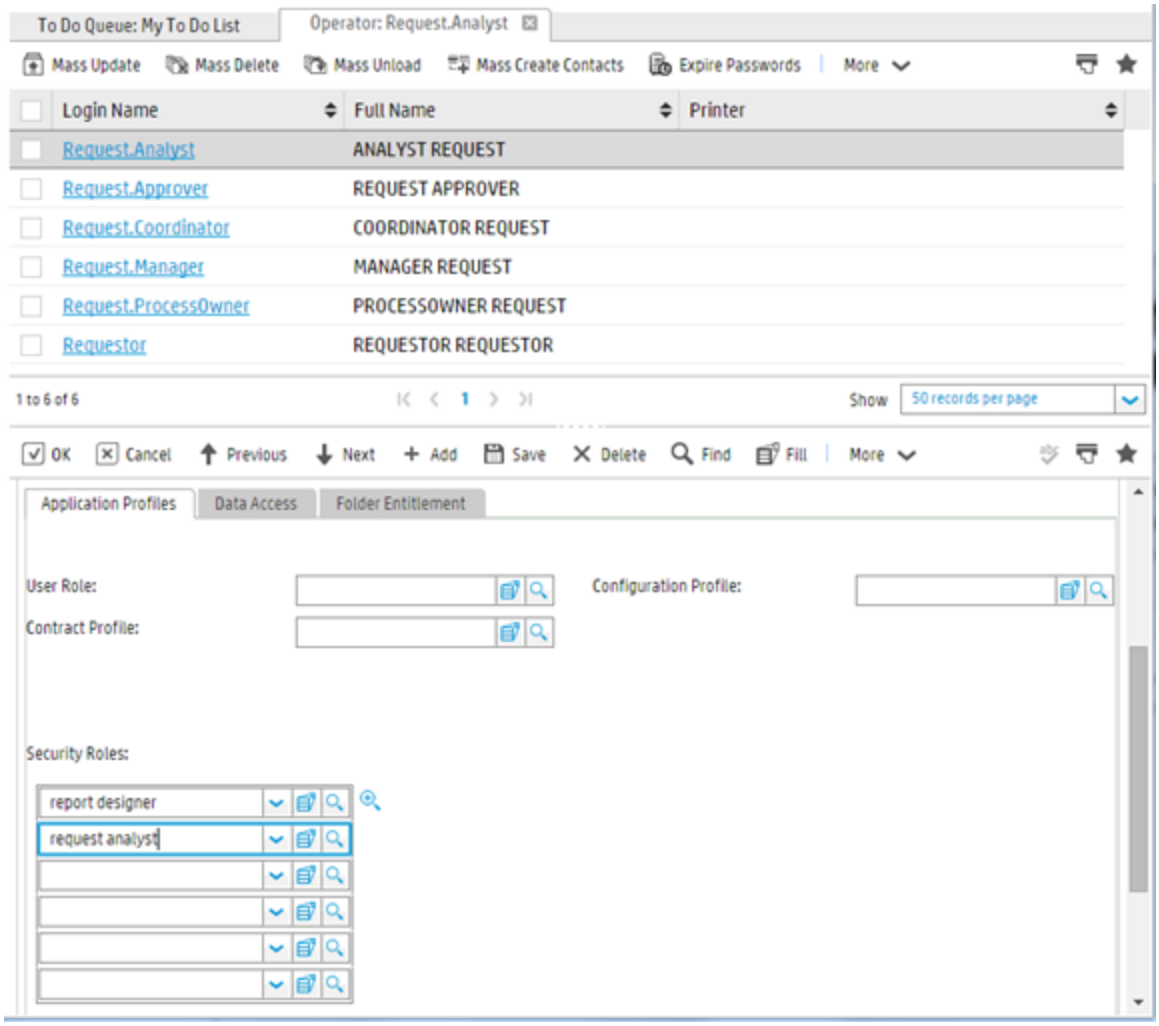
Out-of-box Request modules include the following Security Roles:

- Request Analyst
- Request Approver
- Request Coordinator
- Request Manager
- Request Process Owner
- Requestor

Assign Security Roles to an operator

To grant an operator the rights associated with a specific role, you must assign the Security Role to the operator.

The following screenshot demonstrates the Request module Security Roles that are assigned to Request Analysts in an out-of-box Request module.



Configure the dbdict and the data policy

Process designer framework requires the following fields in dbdict and datadict.

Name	Type	Comments
category	character	If your workflow is based on a category, this field stores the category of the record. Ignore this field if your workflow is based on an object. This field is automatically filled by the Process Designer framework in the following situations: • A record is created by the `document.new` RAD script. • A record's category is changed by the `document.chgCat` RAD script.
current.phase	character	This field indicates the current workflow phase of the record. This field is automatically filled by the Process Designer framework when the workflow phase changes.
record.active	logic	This field indicates whether this record is active or not. Usually, a record is set to inactive after it is closed, canceled, or withdrawn. This field is automatically filled by the Process Designer framework based on the value of the "Records in this phase are active" phase property on the target phase when the workflow phase changes. For more information about the "Records in this phase are active" workflow phase property, see the TODO, ..." section.

Note: If you are already using other field names in your dbdict to achieve the same purposes, add alias fields to them with the above field names.

In the data policy, you must set the Area for the current file. This enables you to use the following variables to check whether a user can access a record. The variables are calculated based on the new, view, update, and delete folder settings from this Area, according to the Security Roles assigned to the current user:

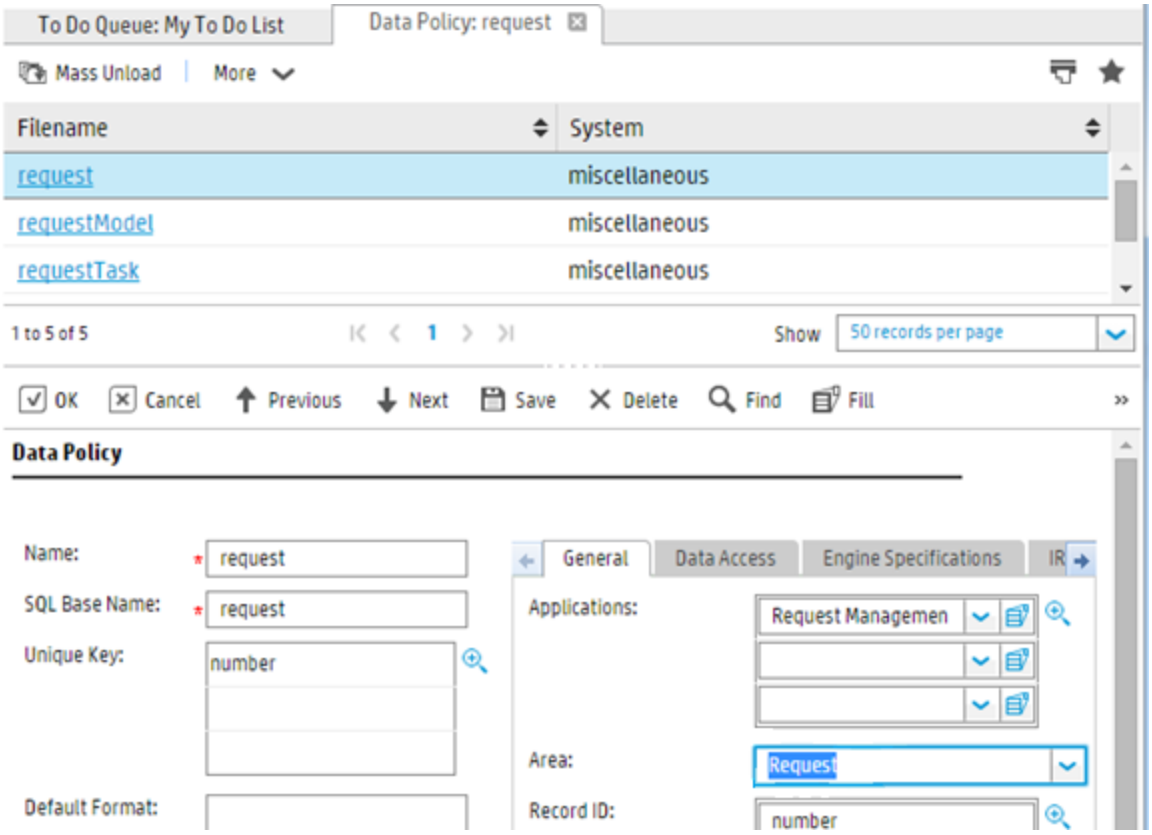
- $L.tableAccess.new
- $L.tableAccess.view
- $L.tableAccess.update
- $L.tableAccess.delete

- $L.tableAccess.expert
- $L.tableAccess.admin

For example, out-of-box Request modules include the following mapping between files and Security Areas:

- Request (the Request record table) maps to the "Request" Security Area

As soon as the area is set, every access against this object is controlled by the security settings of that Security Area.



Enable a Document Engine object to use Process Designer

To enable Document Engine objects to use Process Designer, you must configure the following fields:

- **Profile application**
  Set this field to **secRoleBasedAccess** to replace the profile-based security application with the new

role-based security application.

- **Profile variable**
  Set this field to **$L.env**.

- **Category table name**
  Specify the category table name if your module uses this concept.

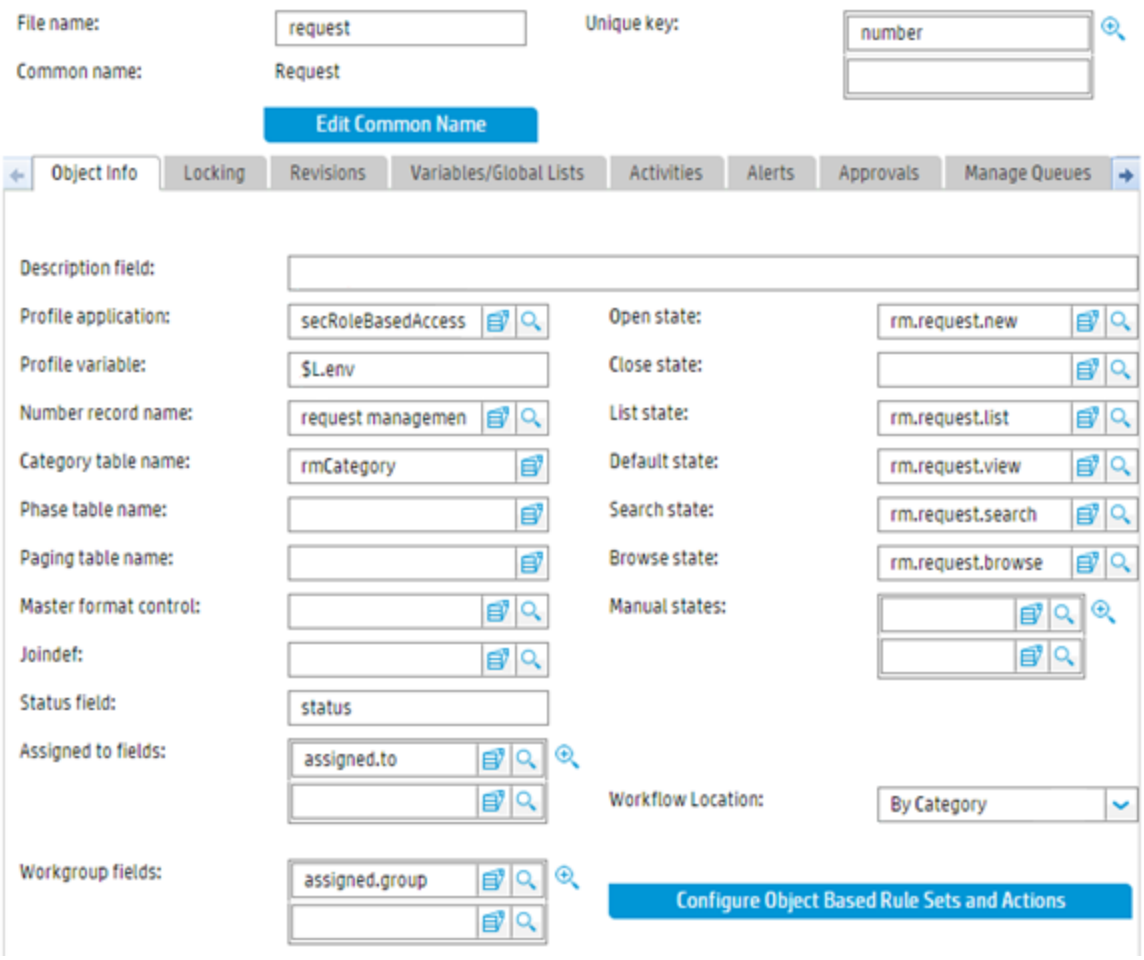
- **Phase table name**
  If you store additional phase information (other than the Workflow Phase) in a table, specify the table name. Otherwise, leave this field empty.

- **Master format control**
  Leave this field empty if you do not use master format control any more.

  Note: Although format control is still supported with the Process Designer framework, we recommend that you convert all business logic into rule sets instead of format control.

- **Workflow Location**
  An object in Service Manager Codeless module is always associated with a workflow or workflows. Select one of the following options to configure the workflow location:
  - By Object (if the object has only one workflow)
  - By Category (if the object has several workflows, and each category is associated with a workflow)

  Note: If you use categories, your category table must contain a field called "workflow" in the dbdict. If the workflow is stored in another field in this category table, the workflow will not work.



Object-based rule sets and actions

Object-based rules and actions help to reduce duplicated definitions. Instead of defining them for each workflow, you can define them at the object level, and they will apply to all workflows that are available for an object.

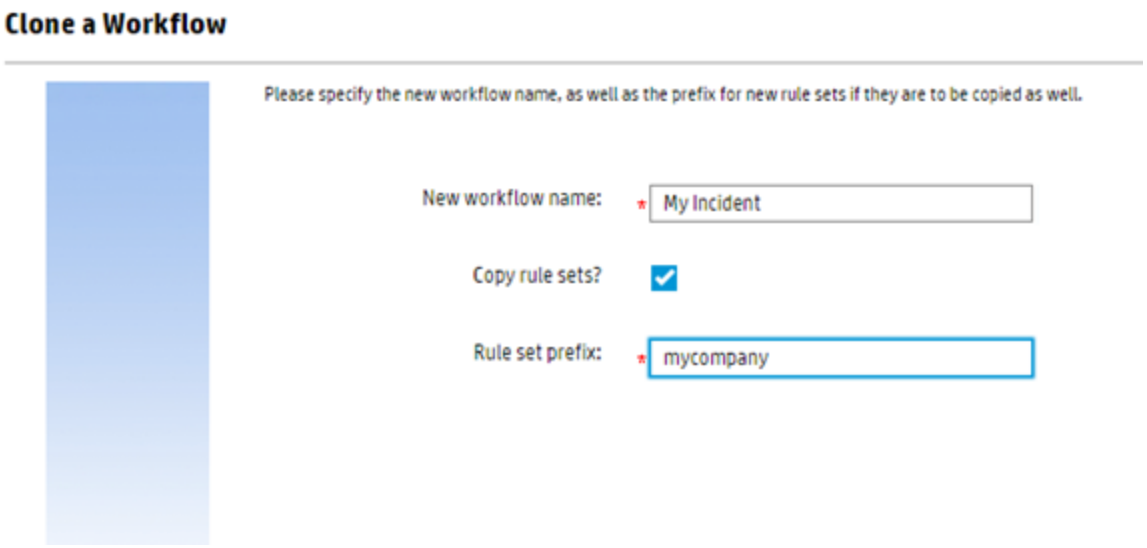
For more information, see "Configure rule sets and actions" on page 81.

Configure a workflow

To create a new workflow, you can create one from scratch or copy an existing workflow.

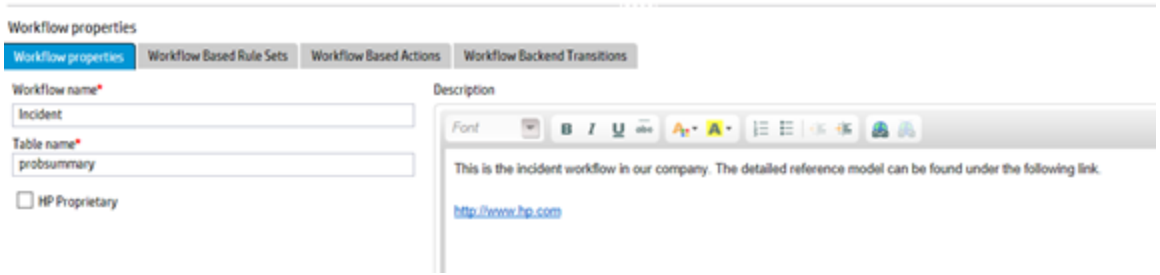
If you copy an existing workflow, you must enter the information that is described in the following table.

Field	Description
New workflow name	The name of the new workflow
Copy rule sets?	Determines whether the rule set that are used by the workflow are also copied. Out-of-box rule sets are marked as "HP Proprietary" and are read-only. However, copied rule sets are editable. • Select this option if you want to customize rule sets. • Do not select this option if you want to use the out-of-box logic. **Note:** We recommend that you use the out-of-box rule sets as far as possible, as this facilitates the upgrade process and enables you to benefit from any future enhancements to out-of-box rule sets.
Rule set prefix	If you copy the rule sets, you must specify a prefix for the copied rule sets. The format new rule set name is <*prefix*>.<*the original rule set name*>

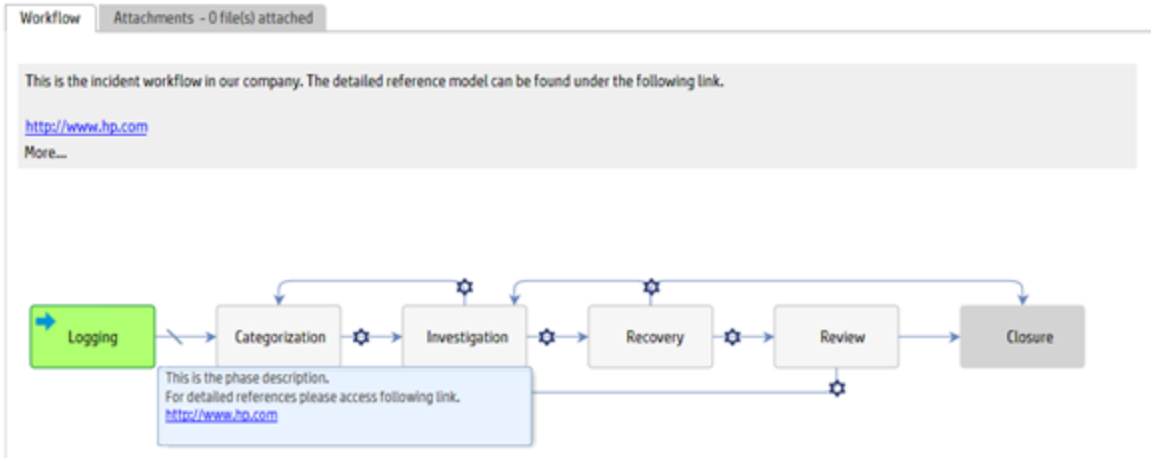


Workflow Properties

We recommend that you add a description of the workflow in the Workflow properties tab of the workflow editor, as this provides guidance to operators when they work on records that follow the workflow.



When a description is entered into this field, that description is displayed when operators hover the mouse over the workflow in the workflow viewer of a record.



Workflow-based rule sets and actions

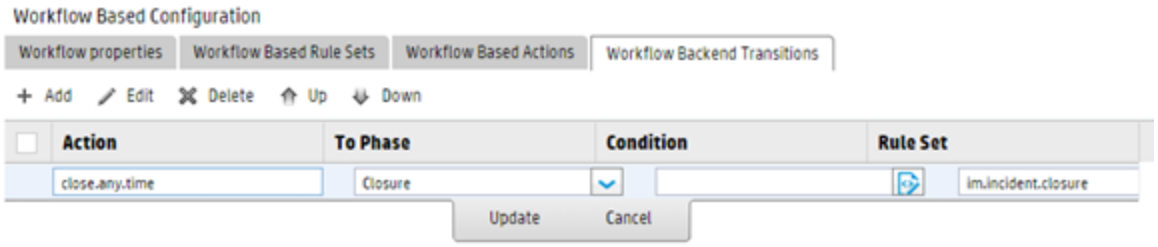
Workflow-based rules and actions help to reduce duplicated definitions. Instead of defining them for each workflow phase, you can define them at the workflow level, and they will apply to all workflow phases.

For more information, see "Configure rule sets and actions" on page 81.

Workflow backend transitions

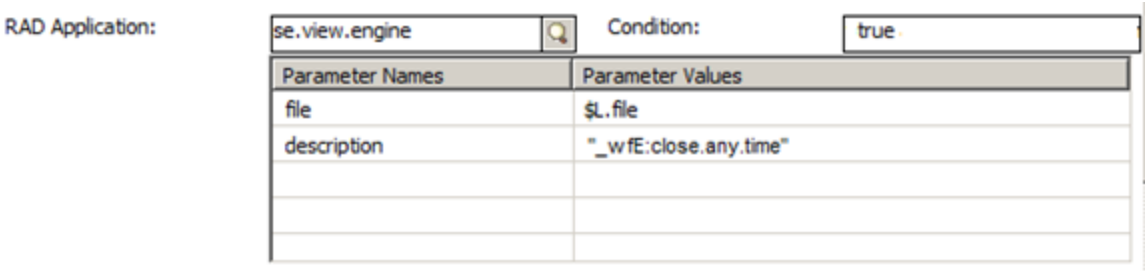
Workflow phases are connected by transitions to move from one phase to another phase, however if you want to move to one phase from whatever current phase is, you can use backend transition to achieve it.

For example, the Incident workflow in out-of-box systems includes a backend transition that moves an Incident record to the Closure phase.



You can invoke backend transitions by using the following methods:

- By using the Run Action rule (for more information, see "Perform automatic operations with Run Action rules" on page 8)
- By using the Run Scheduled Action rule (for more information, see "Configure Run Scheduled Action rules" on page 23)
- By using the `se.view.engine`, RAD application, as demonstrated by the following image.



- By using a JavaScript, as demonstrated by the following image.

```
// with PD(Incident), do backend WF transition close
rc = this.scFile.doAction("_wfE:close.any.time");
```

Configure workflow phases

Workflow phases show the state of a record in the workflow and enforce the business logic that must occur in order for the record to move to the next phase. Transitions are used to move from one phase to another phase.

The workflow editor graphical interface enables you to add a phase to an existing workflow. You can add a phase and configure its properties from scratch,or you can clone an existing phase if you want to create a phase that has similar information (such as same form name, rule sets, or other attributes) to an existing phase.

Phase properties

When you create or configure a phase, you must configure the following properties.

Field	Description
Phase Order	The Phase Order field provides each phase with a numerical order within a workflow. These numbers are used in calculations for Response Service Level Targets (SLTs) and similar metrics, so that the number and timestamps of entries and exits from specific phases can be tracked. For example, a Process Target calculation can determine the time of entry to the final phase, and therefore determine whether a breach has occurred. As a best practice, you should specify your starting phase as 1, and your closing phase as the highest number. We also recommend that these numbers should be roughly sequential from phase to phase. However, some workflows may loop multiple times through a sequence or take divergent paths.
Name	The name of the phase. You cannot modify it after you save current workflow.
Display	The display name of the phase.

name	
Table name	The selected table name during workflow creation. You cannot modify it.
Form Edit condition	If the condition evaluates to true for a user, that user can edit the form. If it does not, the form is read-only.
Records in this phase are active	Select the check box if you want the records in this phase to be active. The "record.active" record field is set to "true" when the record is moved to this phase. Usually, this option is not selected for end phases (such as "Closure," "Cancel," or "Withdraw," and is selected for all other phases.
Make this the first phase	Select this option if you want this phase to be the first phase. Usually you set the "Logging" phase as the first phase.
Make this the default phase	Select this option if you want this phase to be the default phase. **Note:** If the current phase of a record is set to a phase that does not exist in the current workflow, it will be moved to the default phase. This may occur if a phase is removed from a workflow or if data is imported from another source that did not share the same workflow. Or, if you change the category of a record, this record will be moved to the default phase.
Additional Phase Information	Select this option to open the Extended Phase Information page to modify the phase information. **Note:** Only the Change Management module supports this feature. You cannot edit or delete a phase name from the Extended Phase Information page or the cm3rcatphase.main form. Change Management workflows have unique workflow phases but they will share change phases if the workflow phases have the same name. For example, if Workflow 1 and Workflow 2 each have a phase named "Build and Test," they will share the same change phase record.

Phase-based rule sets and actions

Phase-based rules and actions apply only to the current phase. For more information, see

"Configure rule sets and actions" on page 81.

Alerts

You can use alerts to configure phases, however we recommend that you use the Run Scheduled Action rule instead.

For more information about the Run Scheduled Action rule, see "Configure Run Scheduled Action rules" on page 23.

Configure transitions between workflow phases

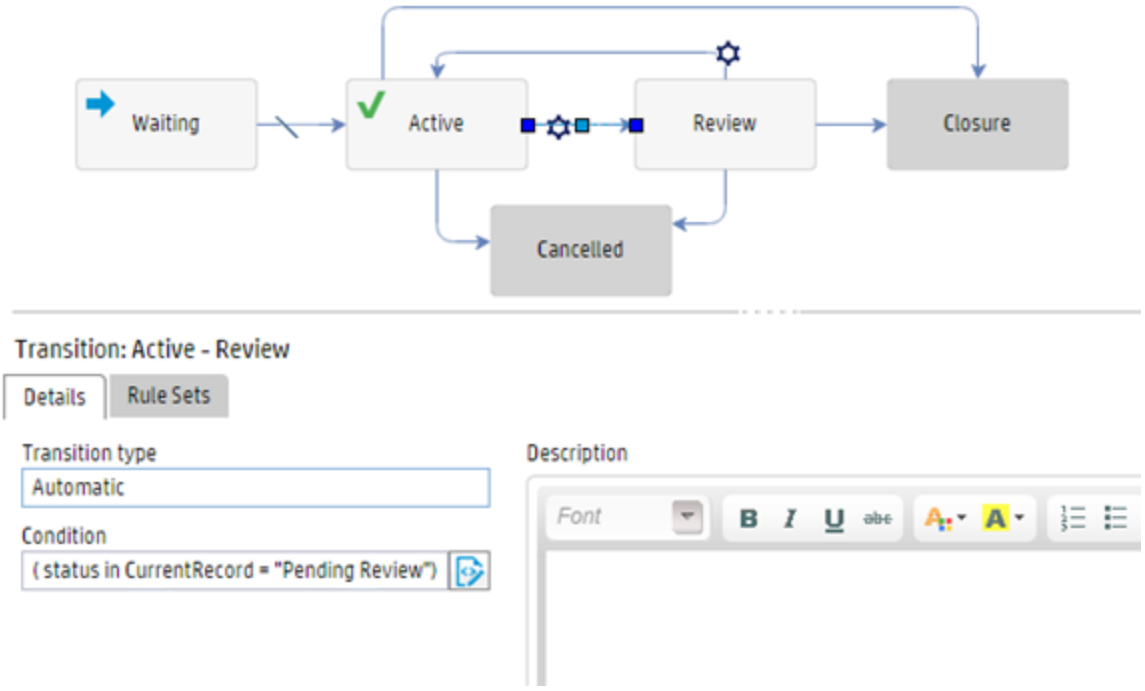
Process Designer workflow transitions occur when a record moves from one phase to another phase. Transitions can happen manually, automatically, or by default.

Automatic transitions

An automatic transition moves the workflow to another phase based on data in the workflow record. The transition occurs when the configured condition is met.

Usually, if your workflow is status driven, you can use automatic transitions and configure the status as the condition of this transition.

In the following example, when the record is in the "Active" phase and its status is set to "Pending Review," the record moves automatically to the "Review" phase when the record is saved.



We recommend that you add descriptions to automatic transitions. These are displayed to the operator to help guide their work.

Manual transitions

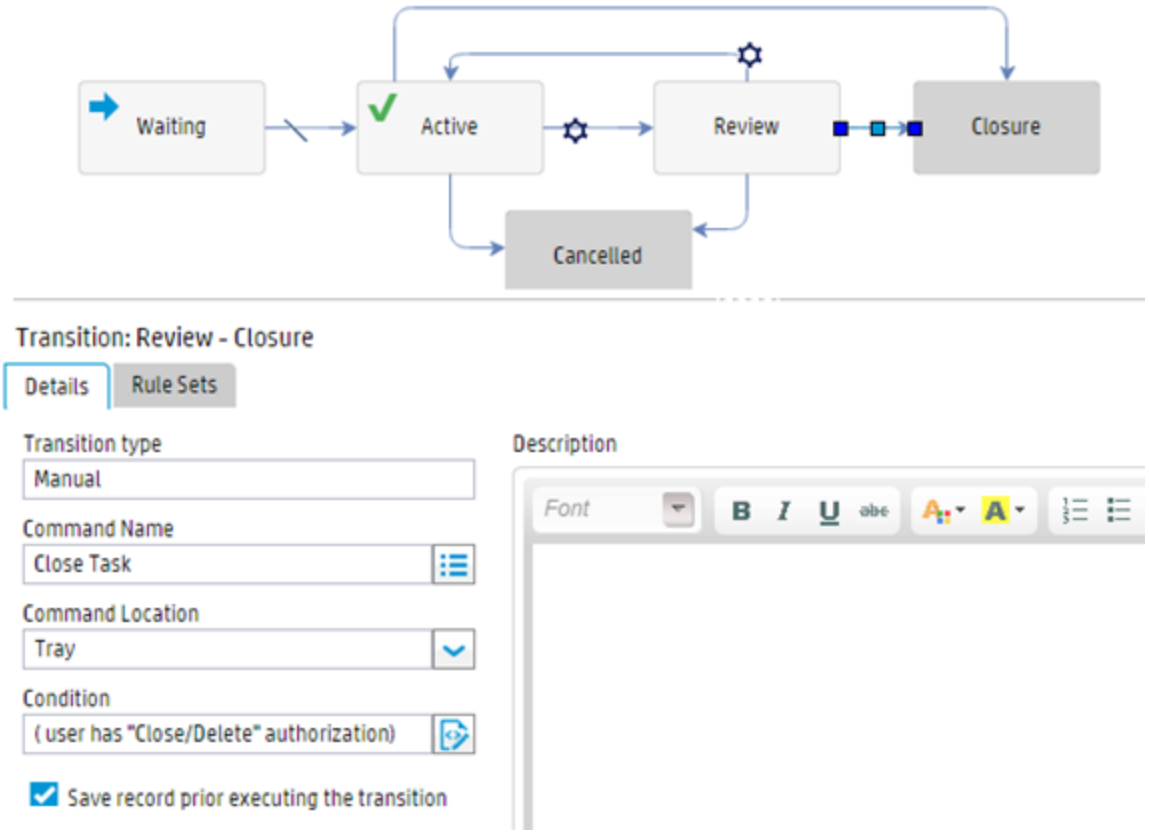
A manual transition requires the operator to perform an action to move a record from one phase to another. This type of transition, in which an operator must press a button or otherwise trigger an action, is a manual transition.

Usually, if your workflow is actions driven, you can use manual transitions so that operators move the workflow phase manually. You can also use manual transitions in a status-driven workflow if you still want to move to a specific phase manually.

We recommend that you add descriptions to manual transitions. These are displayed to the operator to help guide their work.

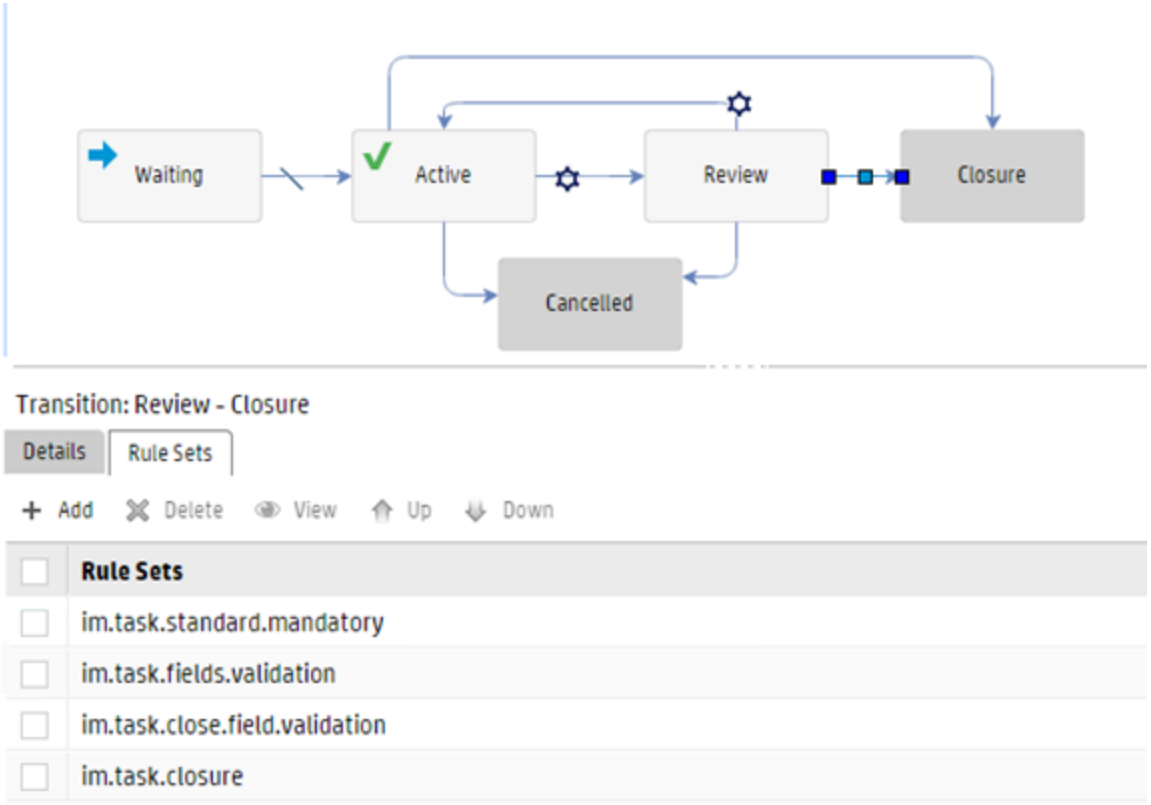
You can configure whether the record is saved before the transition occurs by selecting the "Save record prior executing the transition" option. Usually, if a manual transition needs to trigger the same events as a save operation (for example, to perform the same validation of a save operation), and if you do not want to configure duplicate validation Rule Set on this manual transition, you select this option.

For example, the following image shows a typical manual transition in which the target phase is the "Closure" phase. Therefore, the "Save record prior executing the transition" option is selected.

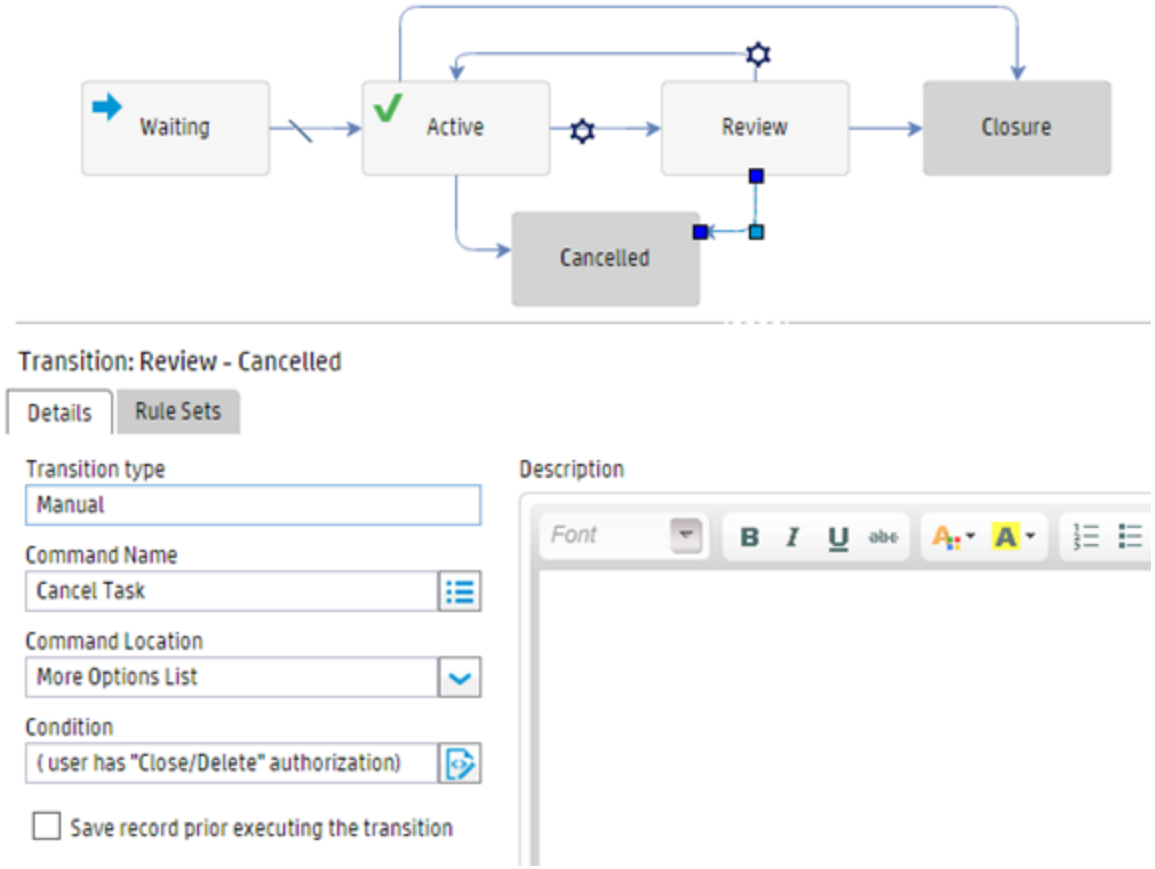


This transition uses the following rule set:

- The "im.task.close.field.validation" rule set is needed specifically for this close operation.
- These common mandatory and validation rules are triggered when the record is saved before the transition occurs.
- Only configure rule sets in transitions if they are absolutely necessary to move from one phase to another. Common mandatory rules and validation rules are already configured for the beginning phase (the "Review" phase in the following image), and do not need to be configured in the transition. We do not recommened that you configure rules that are triggered in the different "events."

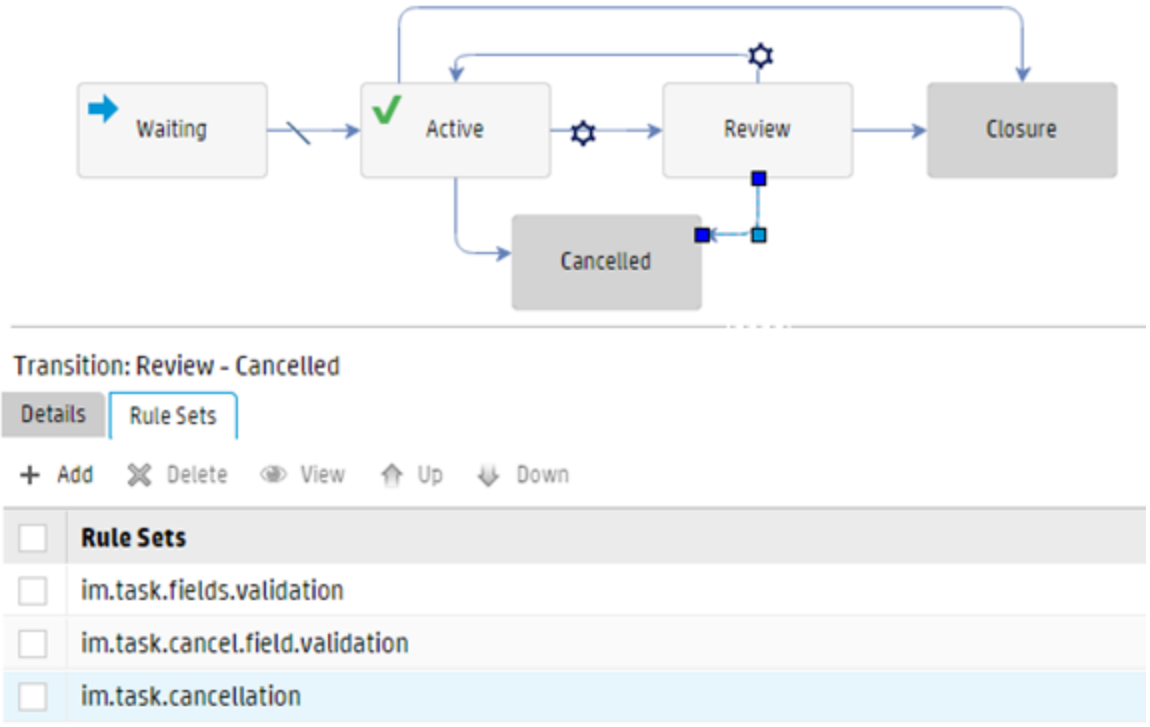


The following image shows an example manual transition in which the target phase is the "Canceled" phase. Therefore, the "Save record prior executing the transition" option is cleared.



This transition uses the following rule sets:

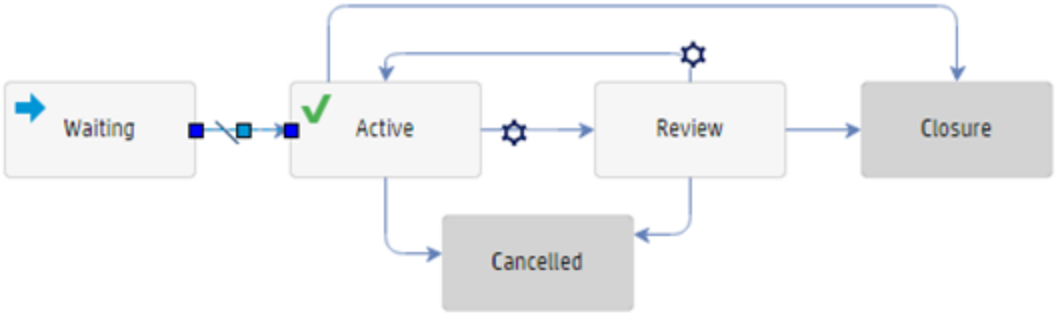
- The only common field validation used is the "im.task.fields.validation" rule set.
- The "im.task.cancel.field.validation" rule set is needed specifically for this close operation.

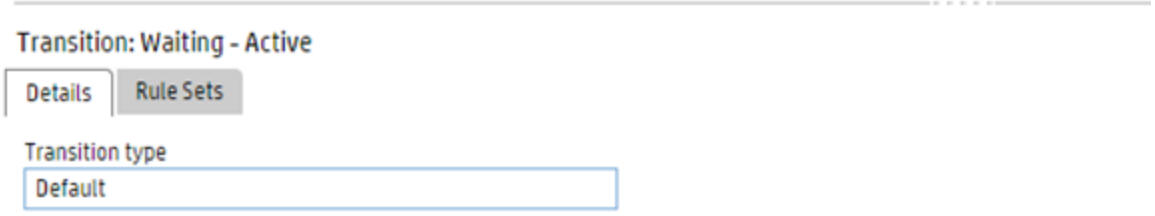


Use Default Transition

A default transition is a special case that moves the workflow automatically only when no other automatic transition conditions are satisfied.

If your "Logging" (or first) and default phases are not the same phase, you can use a default transition between the "Logging" phase and the default phase.



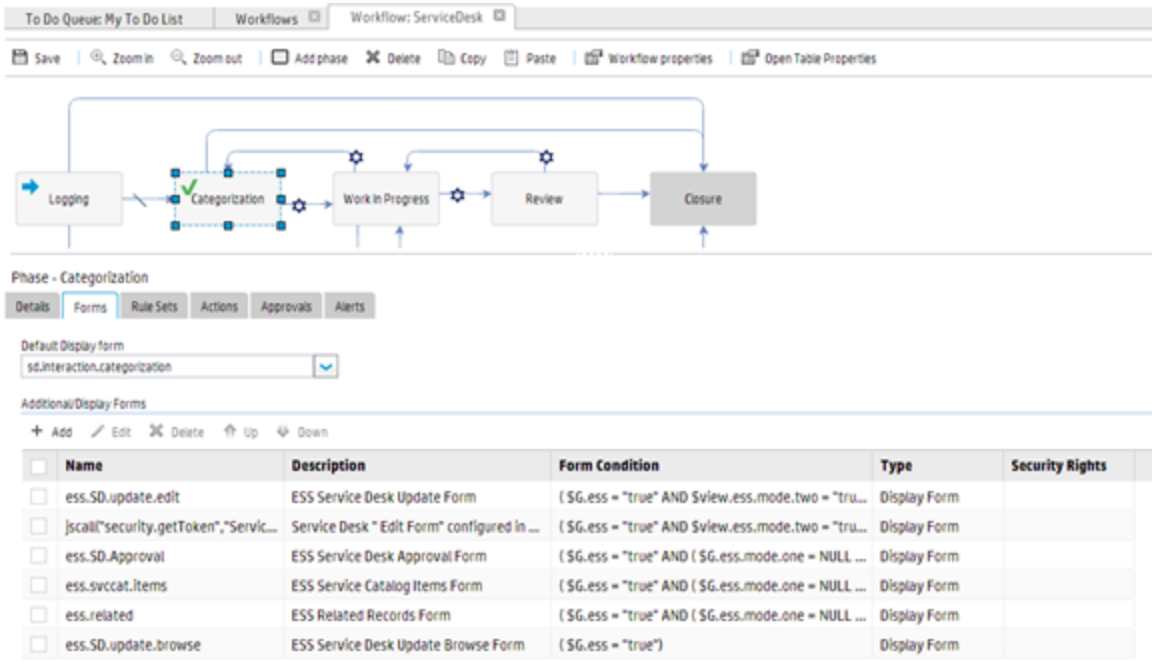


Forms

You can configure workflows so that a specified form is displayed when that a record moves to a specified phase.

You can user the following methods to specify the form that is displayed:

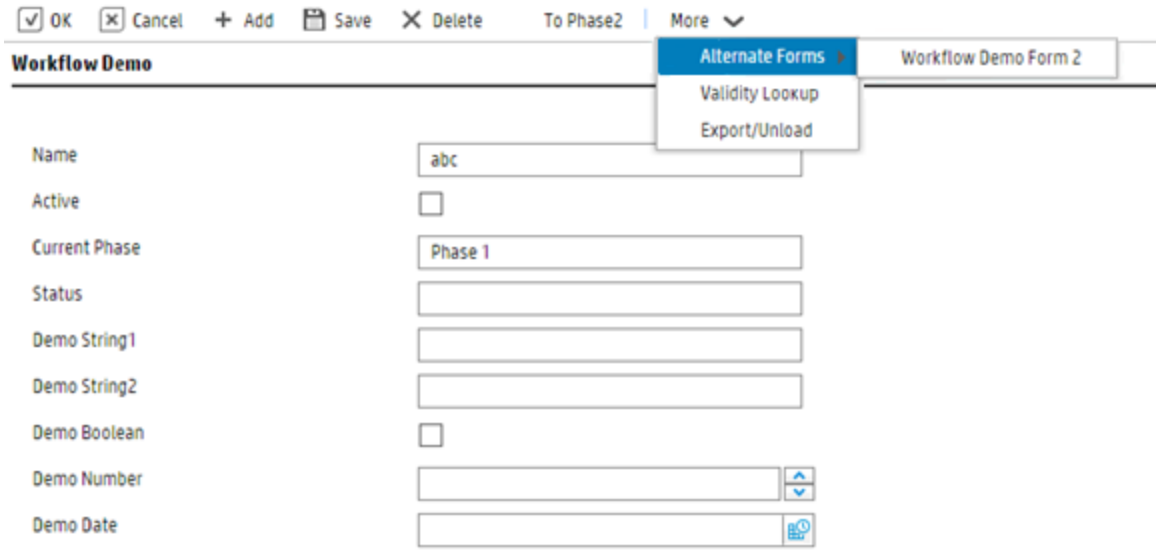
- In the **Forms** tab in the workflow editor, set the form that is displayed by default in the **Default Display form** field. This form is used when no conditional display forms are displayed.
- In the **Forms** tab in the workflow editor, configure forms that are displayed when certain conditions are met by clicking **Add** in the **Conditional/Additional Forms** section, and selecting **Display Form** in the **Type** drop-down list.



Note:

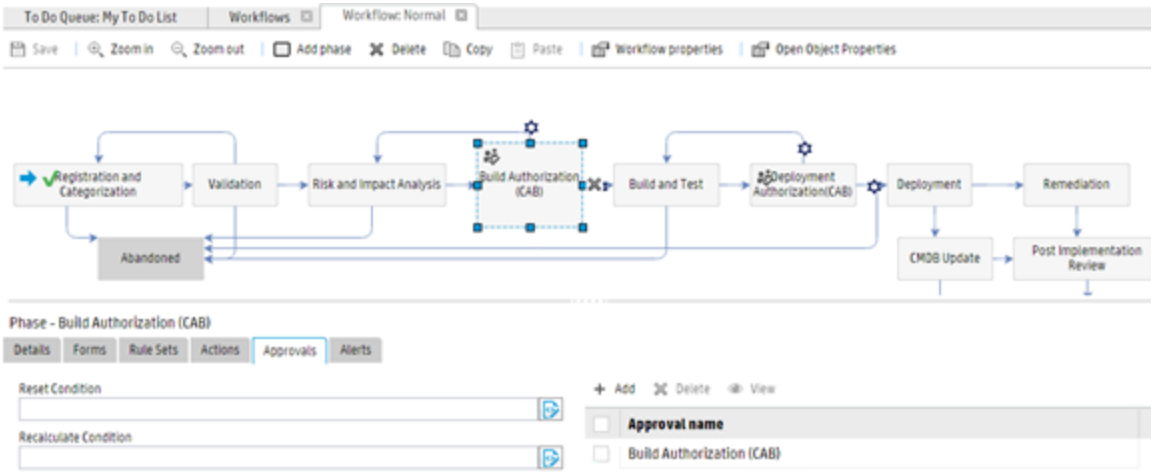
- Display forms that you specify for phases by using this method take precedence over the format setting in State and Display Screen. We recommend that you set the display form in the phase settings in the workflow editor, and that you use different forms for different phases to allow different information to be displayed and captured at various stages of the workflow.
- The name field supports JavaScripts, the `jscall` RAD expression, and RAD variables.

- In the **Forms** tab in the workflow editor, configure additional forms by clicking **Add** in the **Conditional/Additional Forms** section, and selecting **Additional Form** in the **Type** drop-down list. Additional forms are available as alternate forms when users view a record.



Approvals

You can define approvals for a specific phase in the **Approvals** tab of the workflow editor if you require an approval when a record moves to this phase.

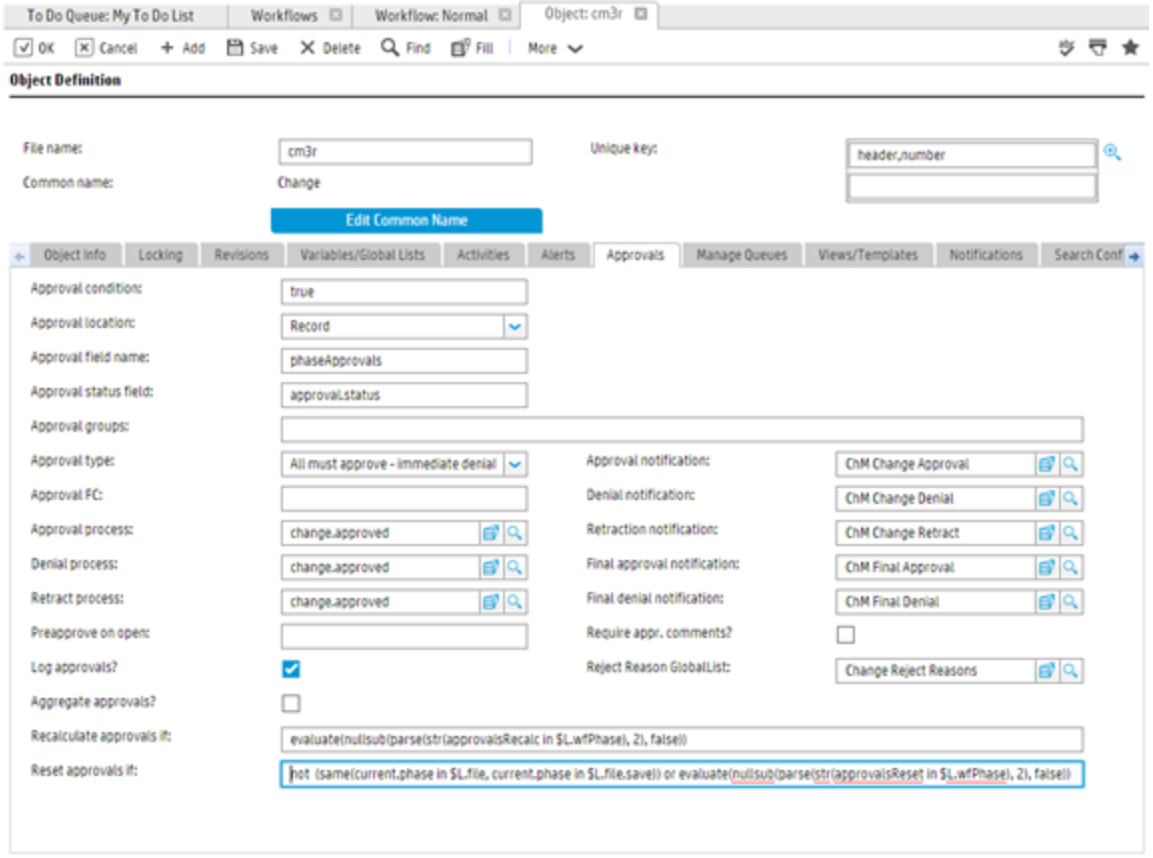


To do this, make the following configurations in the **Approvals** tab of the object definition record:

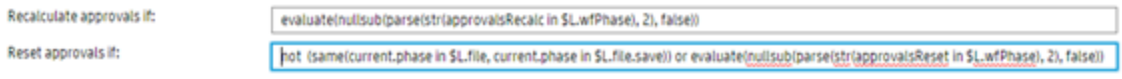
- To determine whether the document engine triggers the approval mechanism, set the value of the **Approval condition** field to "true," or set a specific condition.
- To set the file that contains the approval status, type the file name in the **Approval status** field.

- To set the conditions under which all existing approvals for a record are deleted and regenerated, enter a condition in the **Reset condition** field.
- To set the conditions under which the approvals are recalculated without first deleting the existing approvals, enter a condition in the **Recalculate condition** field.

Note: The reset condition takes priority over the recalculate condition.



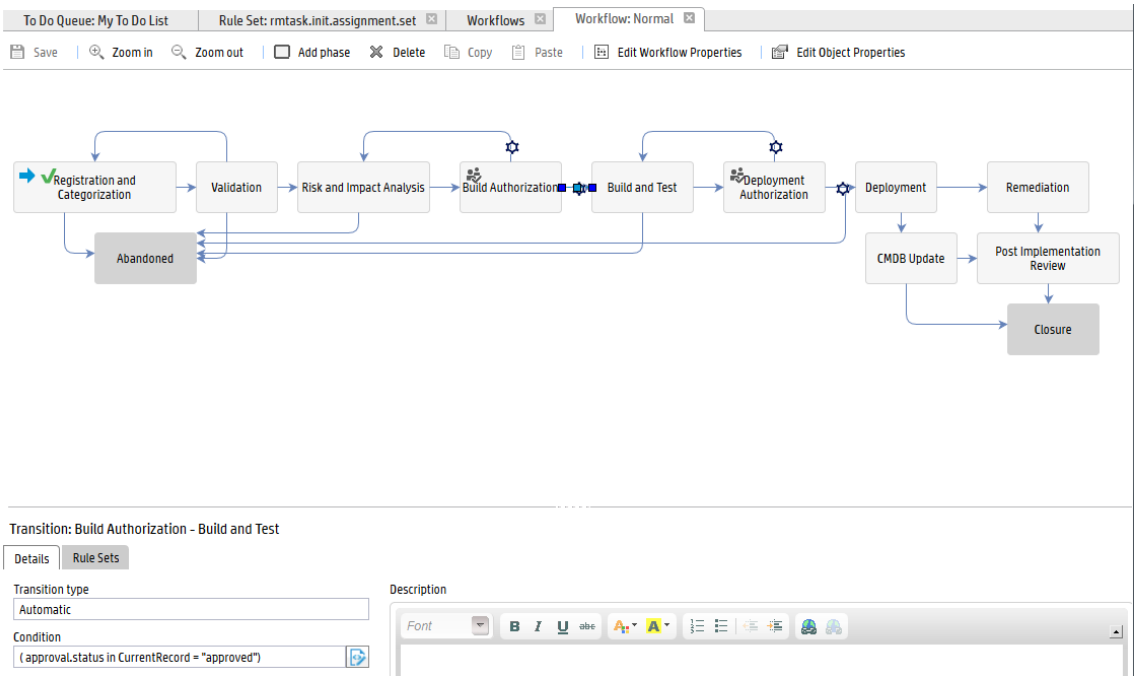
Note: Usually, the recalculate and reset conditions in the object record are configured as follows:



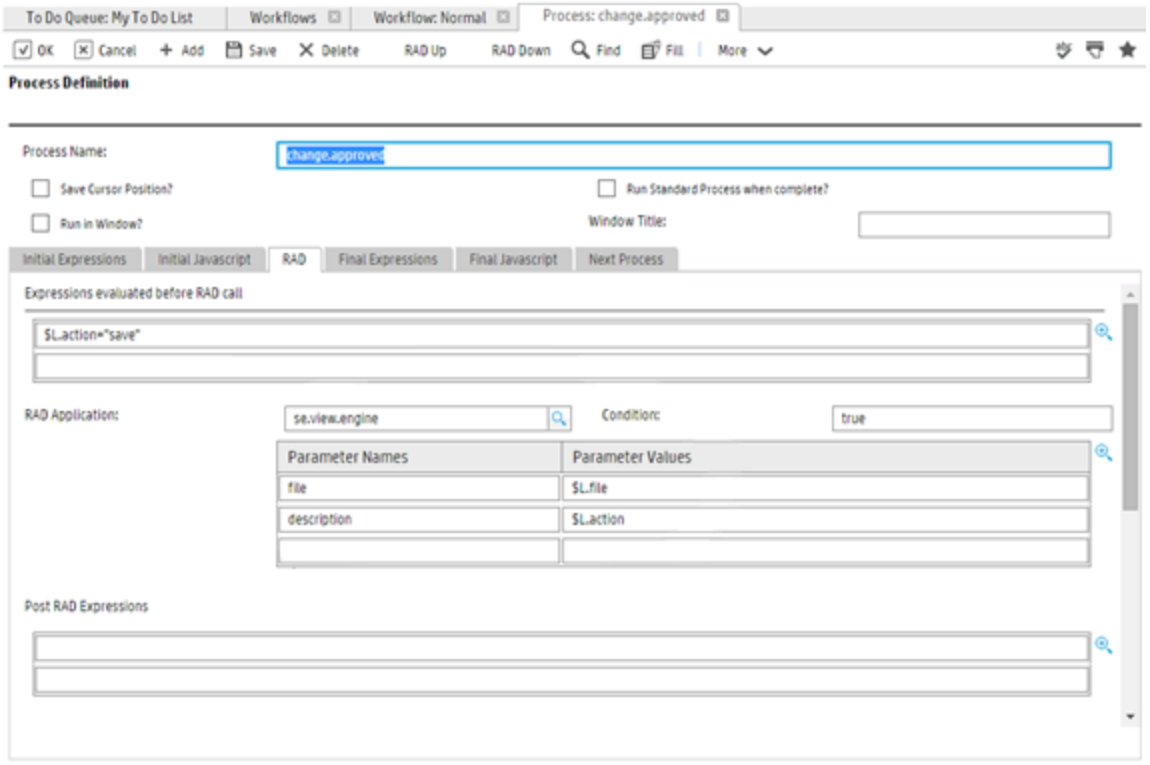
For more information about making further changes to approvals, see the Service Manager Help.

Configure the Approval/Denial process:

Usually, transitions from phases that require approvals to move to the next phase use automatic transitions. For example, you can configure a condition that the **approval.status** field must equal "approved" for an approved record and "denied" for denied record.



In order to trigger the automatic transition after the record is approved or denied, you must configure the approval/denial process to call the `se.view.engine` RAD script together with the "save" action. This triggers the Process Designer record save operation, and then triggers the automatic transition which matches the condition.



Configure rule sets and actions

Rule sets enforce business logic in elements such as phases or transitions, for example checking if the user has filled in the required data or has the proper security level to perform a transition.

Actions perform a task for a phase, and refer to rule sets that are marked as "Available as Action."

Configure rule sets at various levels

You can configure rule sets in workflows at the following levels:

- At the phase level (only applies to the current phase)
- At the workflow level (applies to all phases in the workflow)
- At the object level (applies to all workflows of the object)

If you expect that a rule set will only be triggered at a specific phase, configure the rule set at the phase level. If you expect that a rule set will be triggered in all the phases of a workflow, configure the rule set at the workflow level. If you expect that a rule set will be triggered in all workflows that are associated

with a specific object, configure the workflow at the object level. For example, you can move business logic that you have implemented in triggers to object-based rule sets.

Note: If a call bypasses the Document Engine, the triggers are still invoked, but the rule sets at various levels are not invoked.

Rule set are executed in the following order:

1. Object level
2. Workflow level
3. Phase level

For more information, see "Consider rule set execution order" on page 42.

Configure rule sets at various triggering events

Following is the typical usages:

Triggering event	Triggering action	Typical usage	Typical configured rule type	Phase level	Workflow level	Object level
On add	Runs immediately before the record is added to the database	• Set the record ID if delaying assigning record number • Set default field values if they are empty • Automatically assign to the best team or assignee • Check the mandatory fields and variables for	• Set Field from Number rule • Set Field rule • Assignment rule • Mandatory check: ◦ Set Mandatory Fields rule ◦ Mandatory Variables rule • Validation check:		Yes	Yes

Triggering event	Triggering action	Typical usage	Typical configured rule type	Phase level	Workflow level	Object level
		logging • Check whether the completed fields are valid	◦ Validate Date rule ◦ Validate Text/Number rule ◦ Validate against List rule ◦ Validate against Table rule			
After successful add	Runs immediately after the record is added to the database	• Send email to notify relevant people • Send notification to notify operator with a customized message • Start or stop the elapsed time of the record	• Send HTML Email rule • Send Notifications rule • Start or Stop Clock		Yes	Yes
On enter	Runs when the record tries to move from another phase into this phase	• Check the mandatory fields and variables for Logging • Check whether the completed fields are valid	• Mandatory check: ◦ Set Mandatory Fields rule ◦ Mandatory Variables rule	Yes	Yes	Yes

Triggering event	Triggering action	Typical usage	Typical configured rule type	Phase level	Workflow level	Object level
		• Set default field values	• Validation check: ◦ Validate Date rule ◦ Validate Text/Number rule ◦ Validate against List rule ◦ Validate against Table rule • Set Field rule			
After successful enter	Runs after the record successfully moves from another phase into this phase	• Send email to notify relevant people • Send notification to notify operator with customized message • Start or stop the elapsed time for a situation of the record • Scheduled automatic actions, forexample, automatic	• Send HTML Email rule • Send Notifications rule • Start or Stop Clock • Run Scheduled Action • Run Action	Yes	Yes	Yes

Triggering event	Triggering action	Typical usage	Typical configured rule type	Phase level	Workflow level	Object level
		closure • Cross-module actions				
On exit	Runs when the record tries to move out of this phase			Yes	Yes	Yes
Initialization	Runs once just before the record is displayed to the user	• Set the record ID if not delaying assigning record number • Initialize the list that the current phase uses	• Set Field from Number rule • Run JavaScript rule	Yes	Yes	Yes
On display	Runs each time the record is displayed after a user action	Initialize the variables that are used on the display form	Run JavaScript rule	Yes	Yes	Yes
On update	Runs immediately before the record is updated in the database	• Check the mandatory fields and variables for logging • Check whether the completed fields are valid • Wizard to suspend or resume a process	• Mandatory check: ◦ Set Mandatory Fields rule ◦ Mandatory Variables rule • Validation check: ◦ Validate Date rule	Yes	Yes	Yes

Triggering event	Triggering action	Typical usage	Typical configured rule type	Phase level	Workflow level	Object level
			◦ Validate Text/Number rule ◦ Validate against List rule ◦ Validate against Table rule • Run a Wizard			
After successful update	Runs immediately after the record is updated successfully in the database	• Send email to notify relevant people • Send notification to notify operator with customized message • Start or stop the elapsed time of the record • Scheduled automatic actions, for example, automatic closure • Cross-module actions	• Send HTML Email rule • Send Notifications rule • Start or Stop Clock • Run Scheduled Action • Run Action	Yes	Yes	Yes
Rule sets on Transitions	Runs when the transition	• Check the mandatory	• Mandatory check:			

Triggering event	Triggering action	Typical usage	Typical configured rule type	Phase level	Workflow level	Object level
	occurs	fields and variables for logging • Check whether the completed fields are valid • Wizard to close or cancel the process for manual transition	◦ Set Mandatory Fields rule ◦ Mandatory Variables rule • Validation check: ◦ Validate Date rule ◦ Validate Text/Number rule ◦ Validate against List rule ◦ Validate against Table rule • Run a Wizard			

For more information, see "Consider rule set execution order" on page 42.

Configure actions at various levels

Actions can be applied to individual phases.

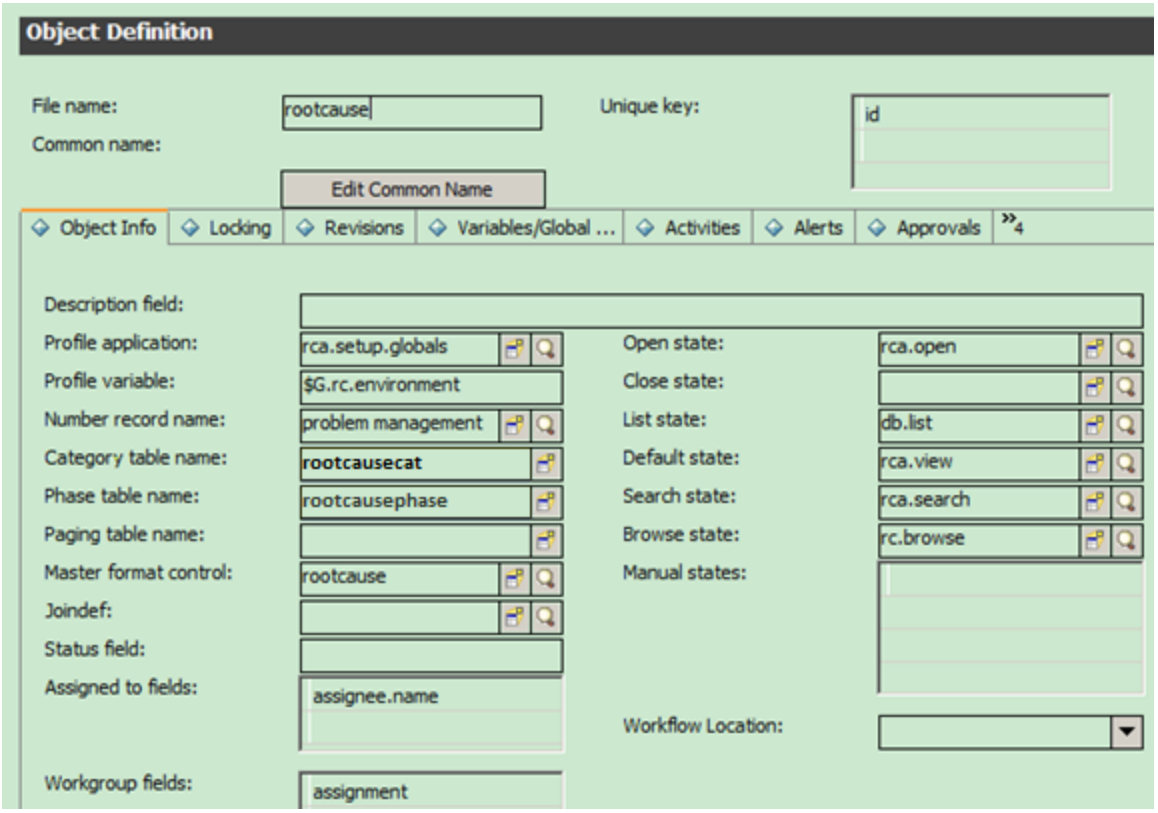
In certain circumstances, you may want to define actions at the workflow level. For example, in cases where a certain button needed to be present in each phase of a workflow. In other cases, you may want to define actions at the object level (for example, if a certain button needs to be present in each phase of all workflows).

Migrate legacy code to Process Designer

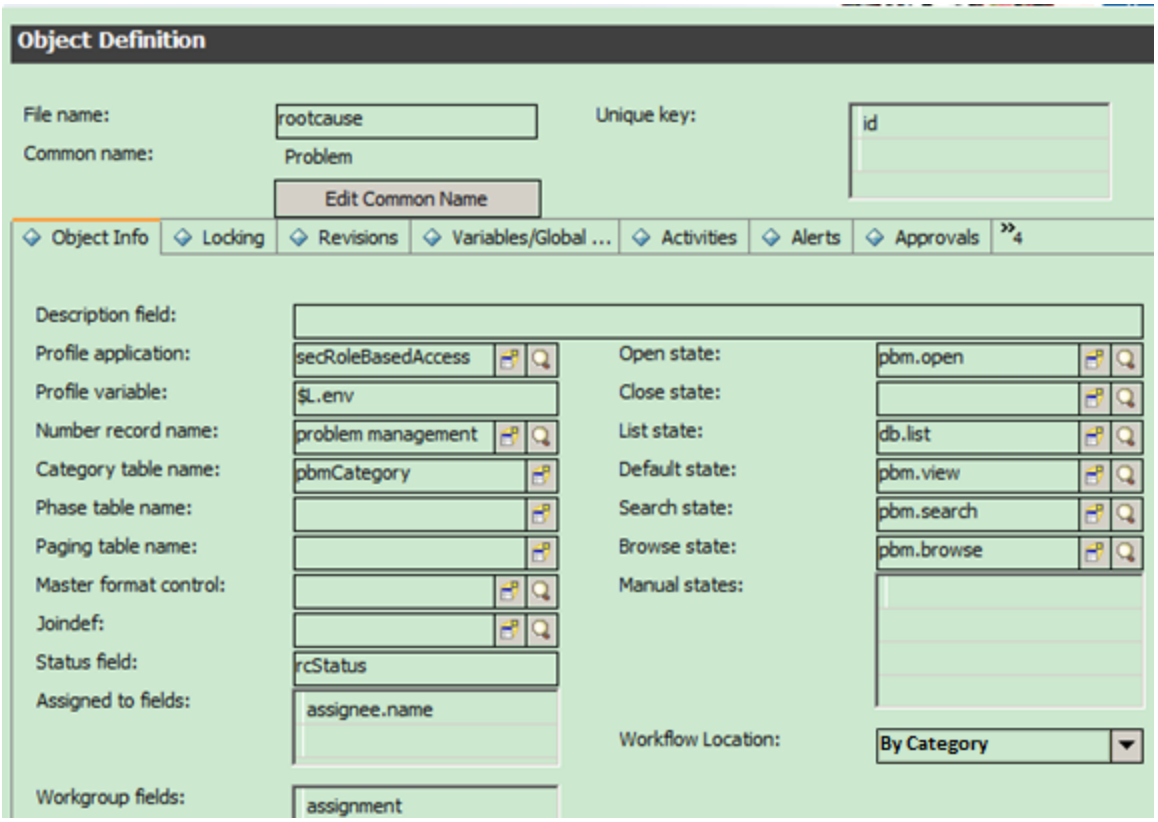
If you are upgrading to Service Manager Codeless from Service Manager Classic, all the legacy coding technologies (format control, display option, and so on) are supported by the Process Designer framework, and you can run your system in a hybrid mode. You can also migrate your legacy technology code to Process Designer technology, which is easier to maintain.

Enable workflow

In a Service Manager Classic module, a workflow is usually configured in the category, and the category table name and phase table name are configured in the object.

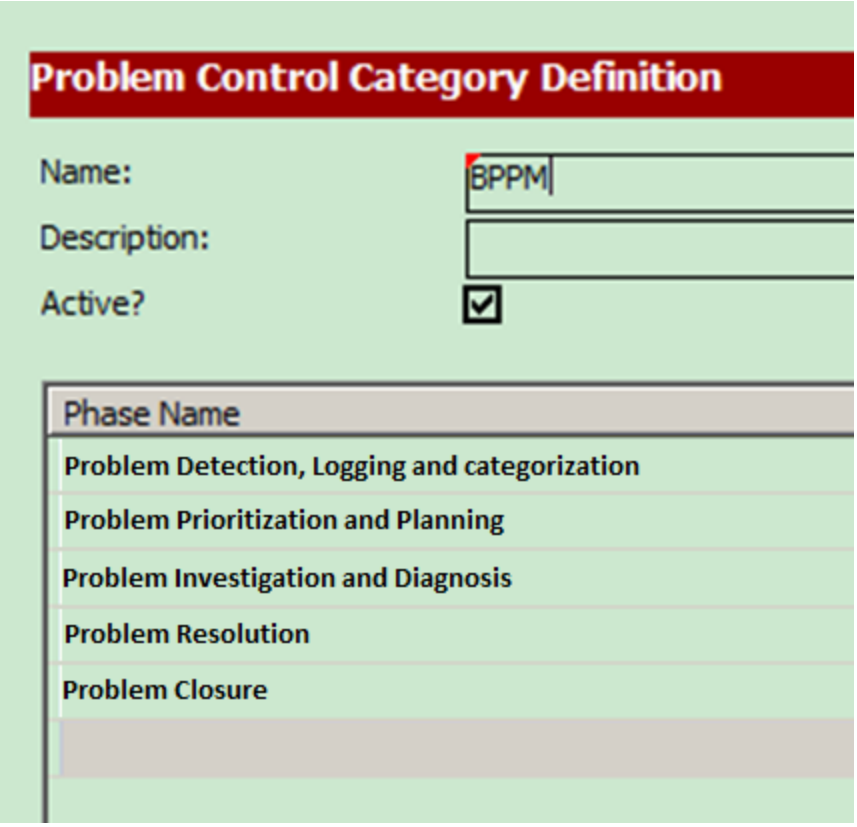


To enable the workflow, you must configure the workflow location in the object definition. If you set the location to "By Category," the workflow defined in the category record is used. If you set the location to "In Object," you must set the workflow in the object definition.

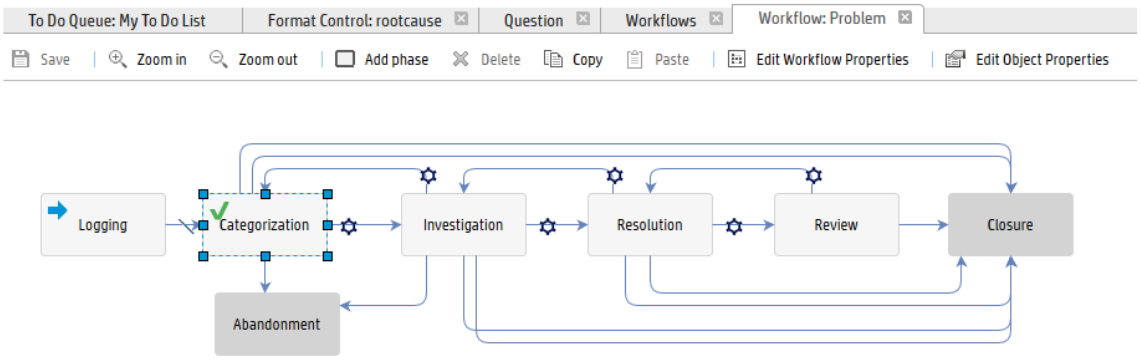


Configure the workflow

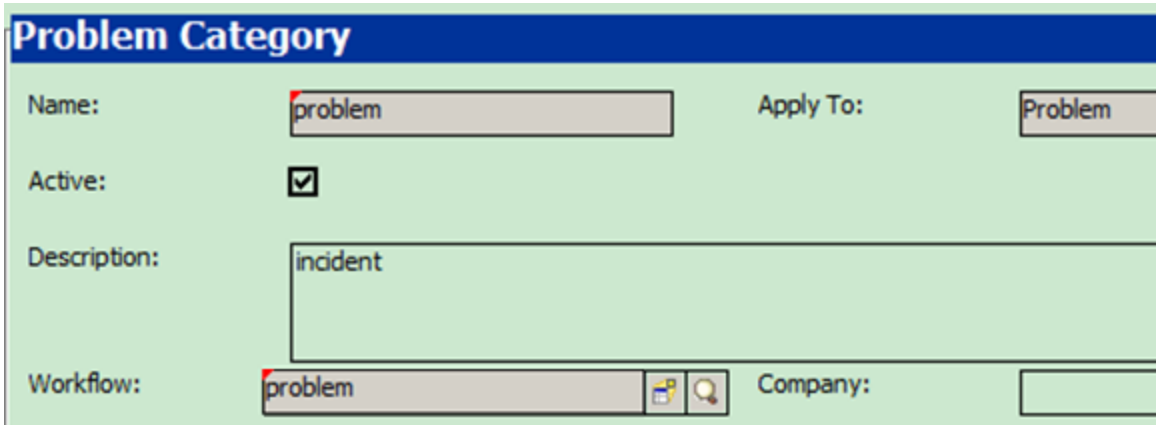
In a Service Manager Classic module, workflow phases are stored in the object-specific phase table, and the workflow logic is defined in the category by using the phases in sequence.



In Service Manager Codeless module, the workflow is configured graphically by using the Workflow Designer in the web UI. You can use this to define more complex logic between phases.

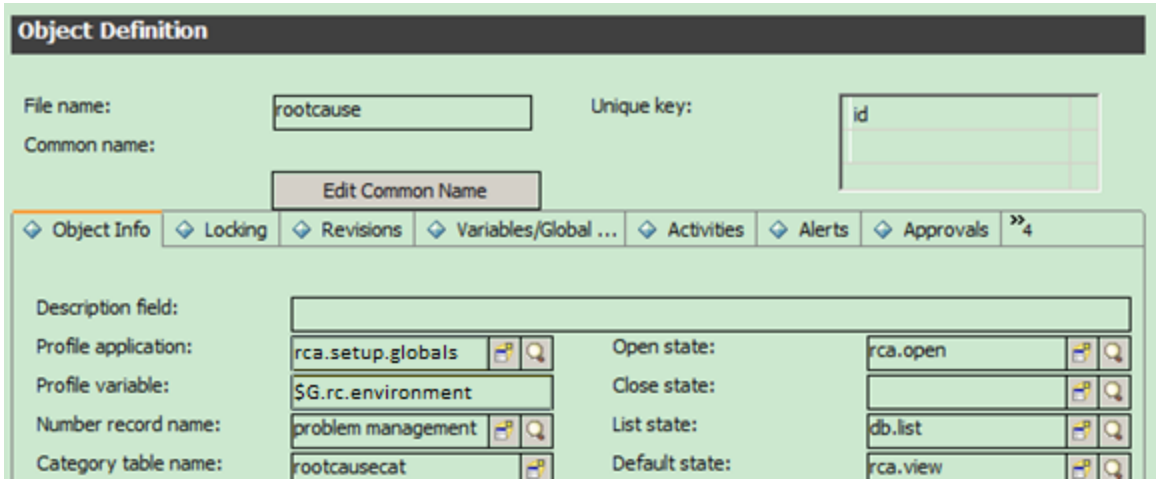


If your workflow location is set by category, the workflow name is specified in the category record:

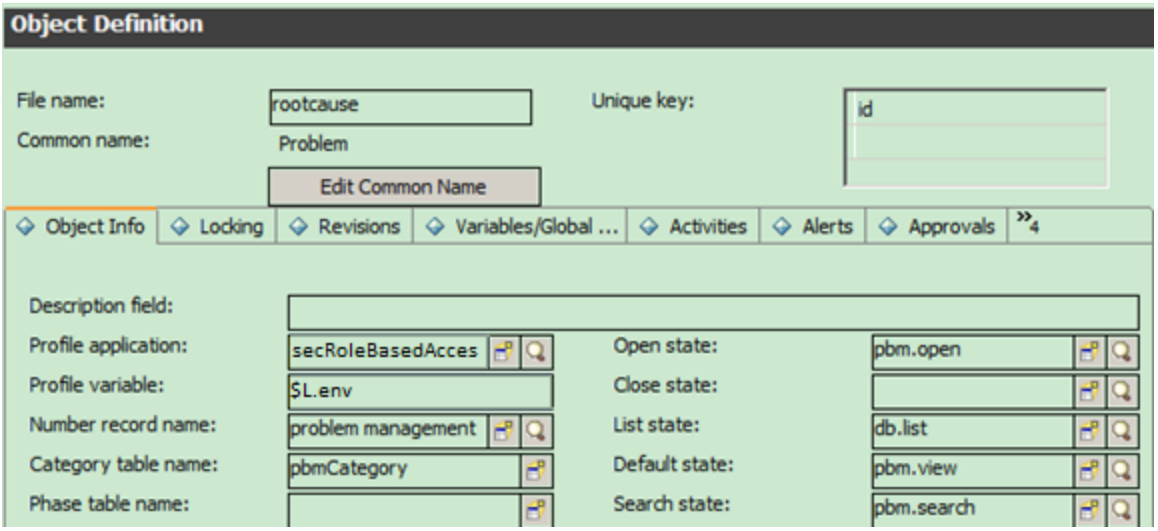


Enable Process Designer role-based security

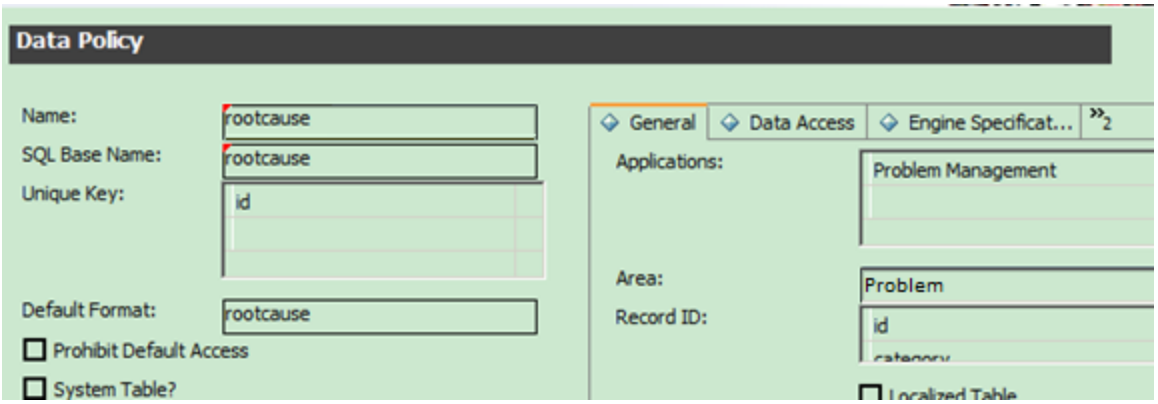
Service Manager Classic modules use the profile-based security model. This is enabled by using the module-specific application (for example, rca.setup.globals for Problem Management) in the object record, and set by the user in the operator table. The settings are stored in module-specific tables (for example, rcenv for Problem Management).



Process Designer provides a common role-base security model. This is enabled by using the secRoleBasedAccess application in the object record, and set by the user in the operator table.



The settings are stored in the secArea, secRights, and secRole tables. The security area is specified in the datadict record.



Migrate the Format Control to rule sets

In Service Manager Classic, business logic is mainly configured in Format Control. Process Designer uses rule sets.

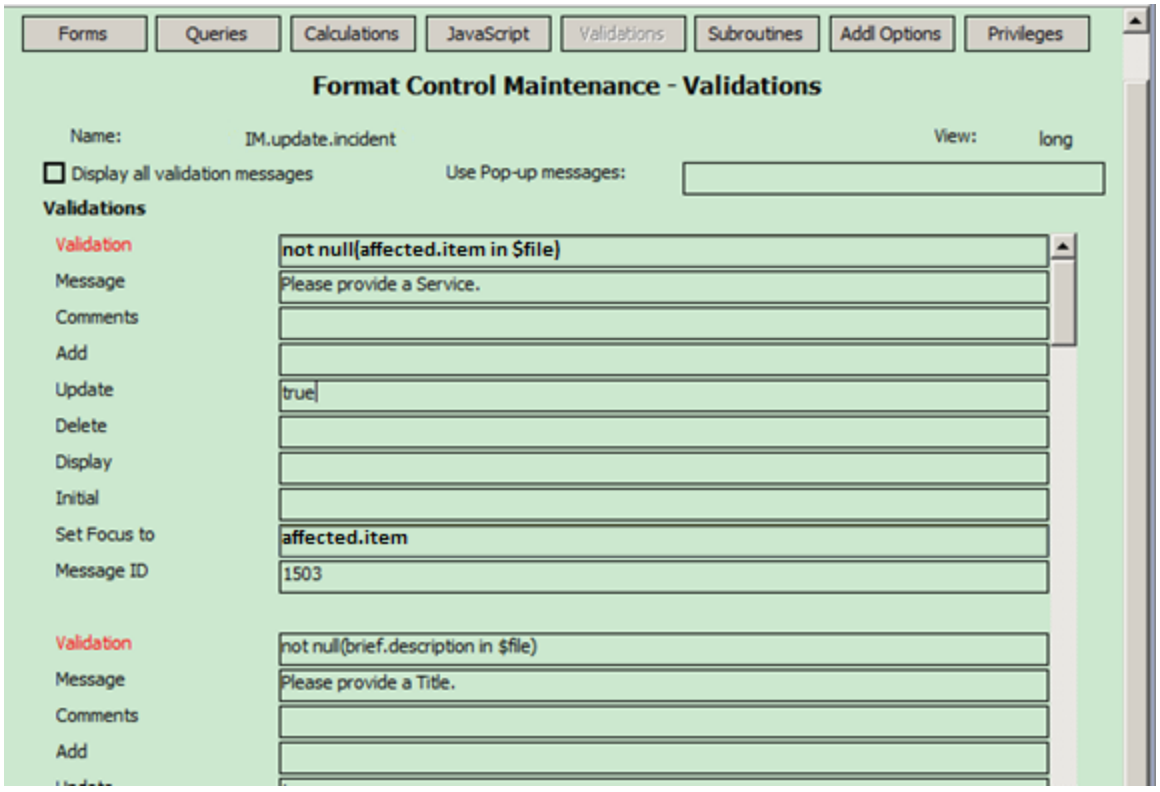
Format Control still works in Service Manager Codeless, but we suggest that you move all logic from Format Control to the workflow rule sets, as follows.

Format Control	Rule set
Master Format Control "Add"	Object or workflow-based rule sets "On add"
Master Format Control "Update"	Object or workflow-based rule sets "On update"

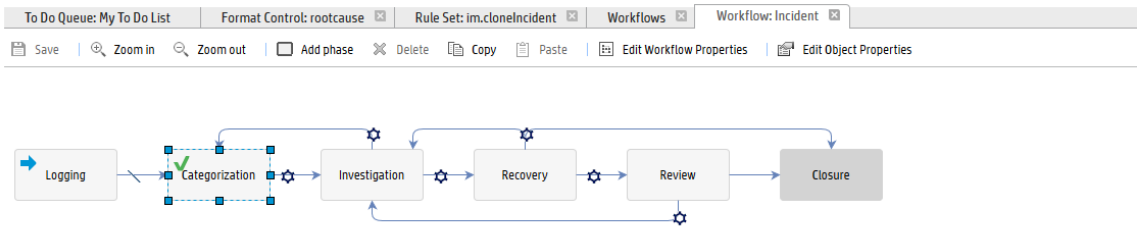
Master Format Control "Display"	Object or workflow-based rule sets "On display"
Master Format Control "Initial"	Object or workflow-based rule sets "Initialization"
Master Format Control "Delete"	Object or workflow-based rule sets "On update"
Form level Format Control "Add"	Workflow phase-based rule sets "On Enter"
Form level Format Control "Update"	Workflow phase-based rule sets "On update"
Form level Format Control "Display"	Workflow phase-based rule sets "On display"
Form level Format Control "Initial"	Workflow phase-based rule sets "Initialization"
Form level Format Control "Delete"	Workflow phase-based rule sets "On update"

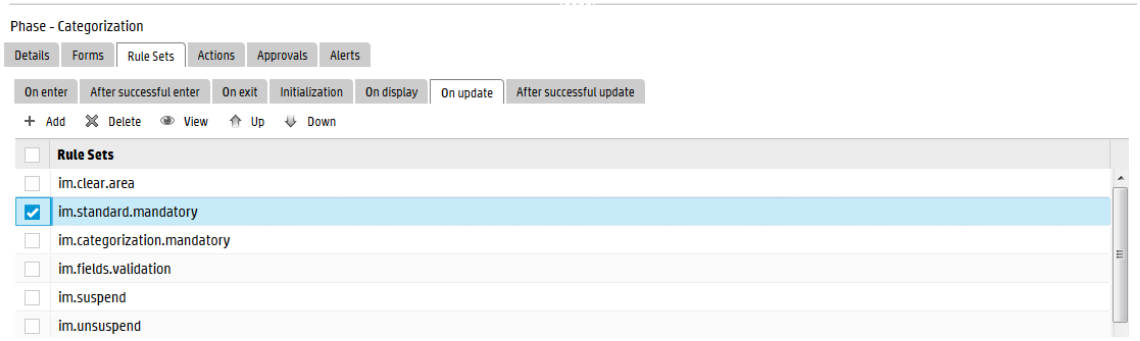
Mandatory validation

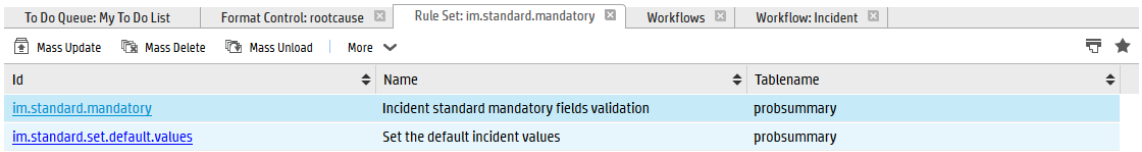
Format Control validations are defined on the master format control or the format level format control.

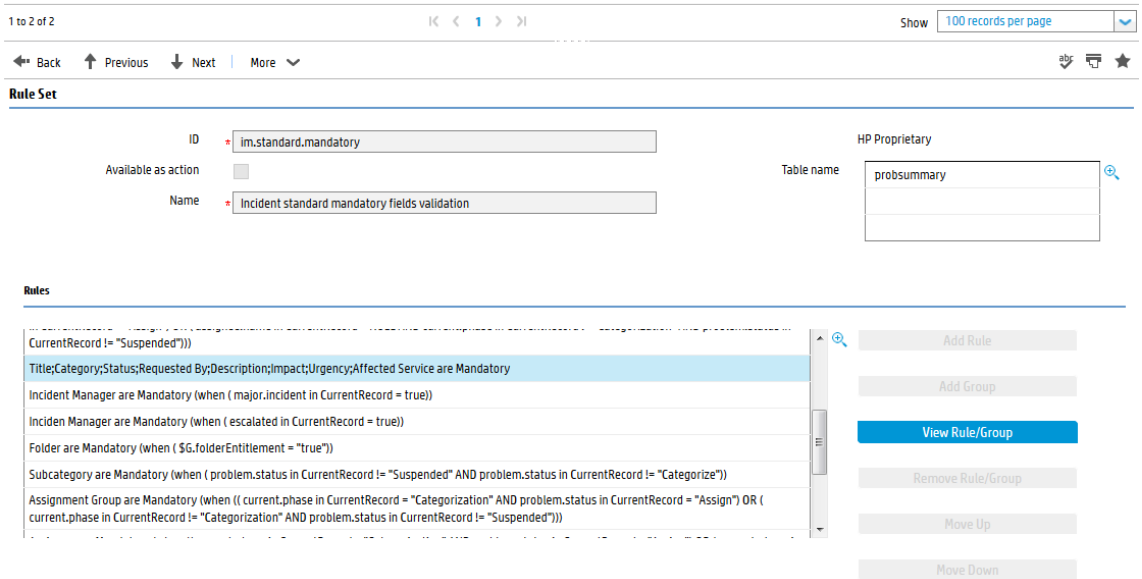


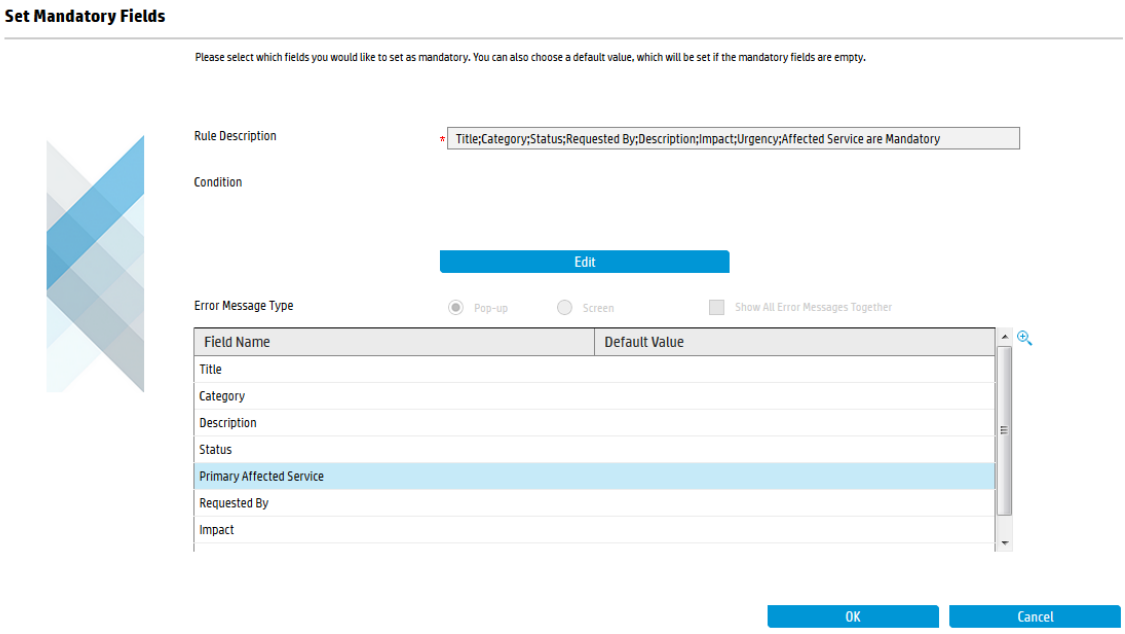
To migrate validation to rule sets, you must define a rule set, set it to corresponding workflow or phase rule sets, and then add the Set Mandatory Fields rule to this rule set.


To Do Queue: My To Do List
Format Control: rootcause
Rule Set: im.cloneIncident
Workflows
Workflow: Incident
Save
Zoom in
Zoom out
Add phase
Delete
Copy
Paste
Edit Workflow Properties
Edit Object Properties
Logging
Categorization
Investigation
Recovery
Review
Closure


Phase - Categorization
Details
Forms
Rule Sets
Actions
Approvals
Alerts
On enter
After successful enter
On exit
Initialization
On display
On update
After successful update
Add
Delete
View
Up
Down
Rule Sets
im.clear.area
im.standard.mandatory
im.categorization.mandatory
im.fields.validation
im.suspend
im.unsuspend

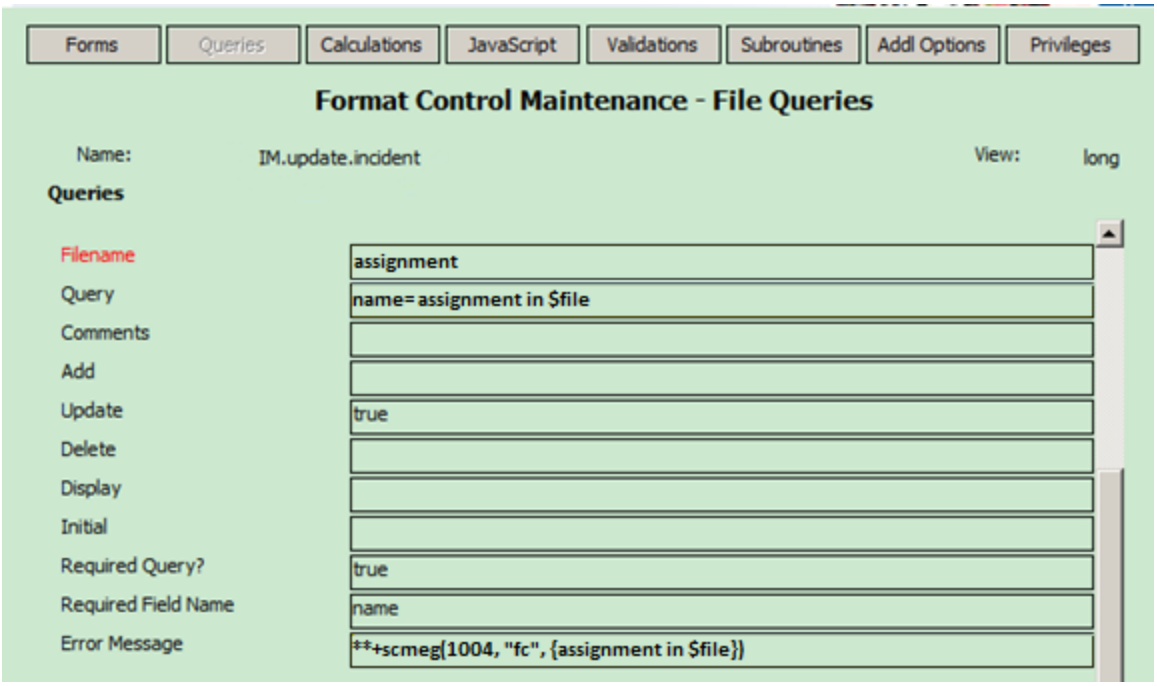

To Do Queue: My To Do List
Format Control: rootcause
Rule Set: im.standard.mandatory
Workflows
Workflow: Incident
Mass Update
Mass Delete
Mass Unload
More
Id
Name
Tablename
im.standard.mandatory
Incident standard mandatory fields validation
probsummary
im.standard.set.default.values
Set the default incident values
probsummary


1 to 2 of 2
Show
100 records per page
Back
Previous
Next
More
Rule Set
ID
im.standard.mandatory
Available as action
Name
Incident standard mandatory fields validation
HP Proprietary
Table name
probsummary
Rules
CurrentRecord != "Suspended")))
Title;Category;Status;Requested By;Description;Impact;Urgency;Affected Service are Mandatory
Incident Manager are Mandatory (when (major.incident in CurrentRecord = true))
Inciden Manager are Mandatory (when (escalated in CurrentRecord = true))
Folder are Mandatory (when ($G.folderEntitlement = "true"))
Subcategory are Mandatory (when (problem.status in CurrentRecord != "Suspended" AND problem.status in CurrentRecord != "Categorize"))
Assignment Group are Mandatory (when ((current.phase in CurrentRecord = "Categorization" AND problem.status in CurrentRecord = "Assign") OR (current.phase in CurrentRecord != "Categorization" AND problem.status in CurrentRecord != "Suspended")))
Add Rule
Add Group
View Rule/Group
Remove Rule/Group
Move Up
Move Down



Validation against a table

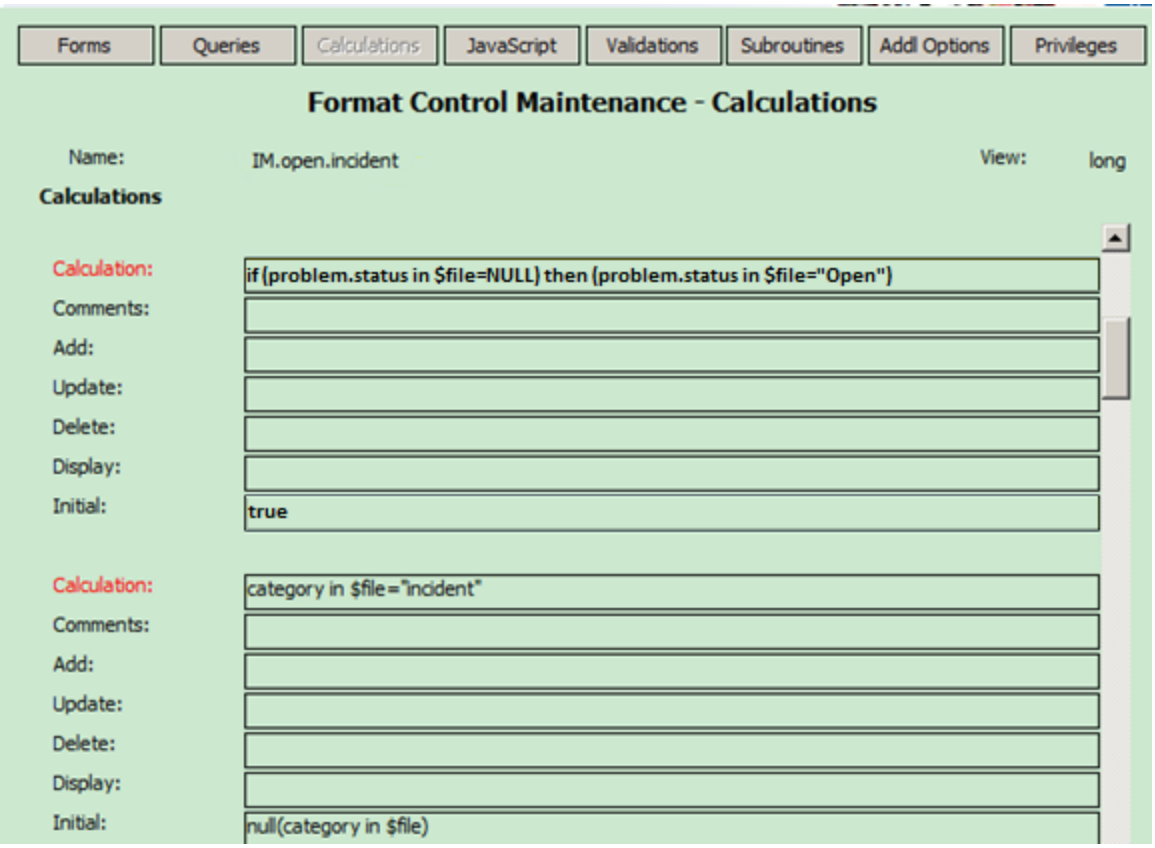
To validate a field against another field in a table, you must define queries in master format control or in format level format control. You must write the code for the queries, manually create the scmessage, and indicate the ID if the message needs to be localized.



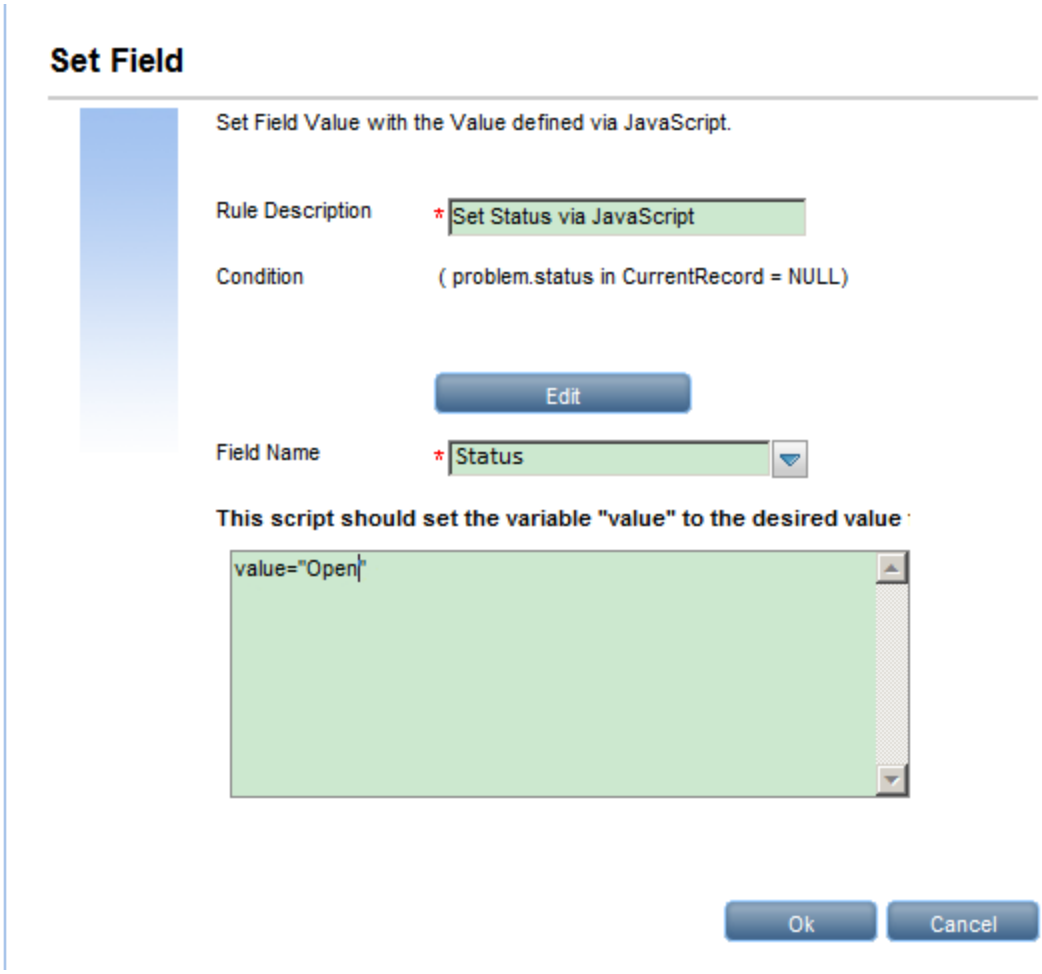
To do this in Service Manager Codeless, you can define a Validate Against Table rule.

Set a field value

To set a field value by using Format Control, you must define the calculations in master format control or in format level format control. Additionally, you must use a RAD expression to write the field set statement (and the condition, if necessary).

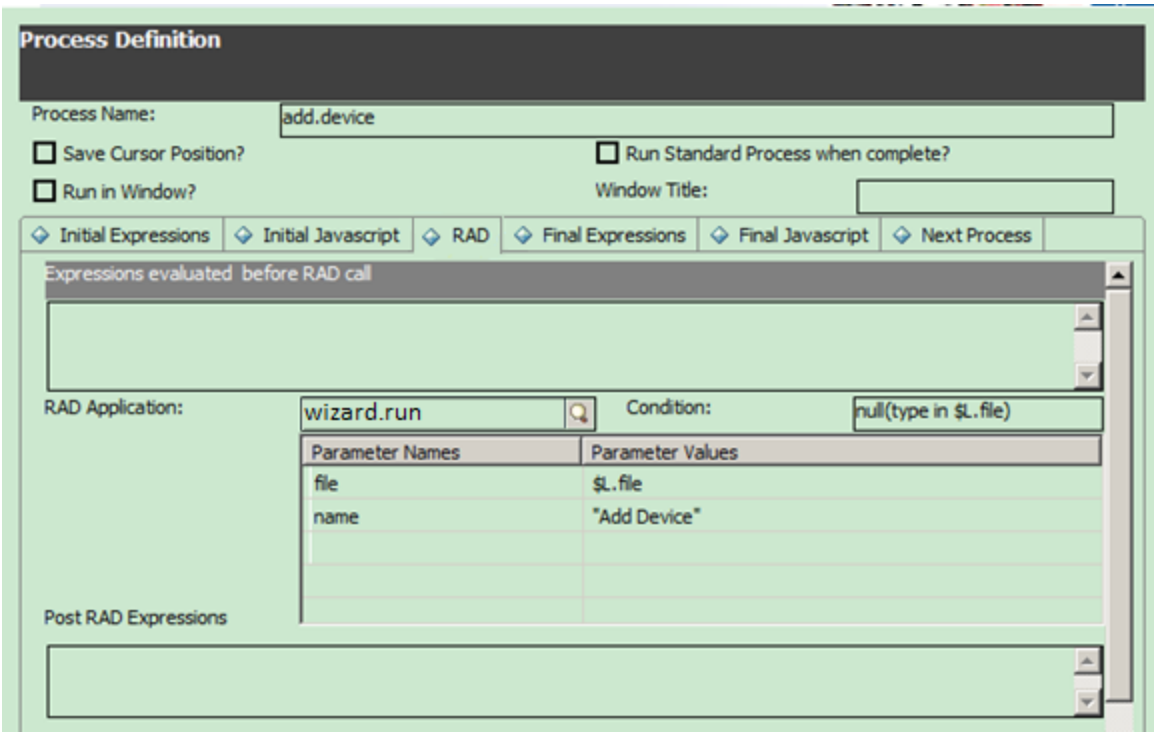


To achieve this by using a Process Designer rule set, you can define a rule set and assign it to the desired workflow or phase, and then add a Set Field rule to this rule set.

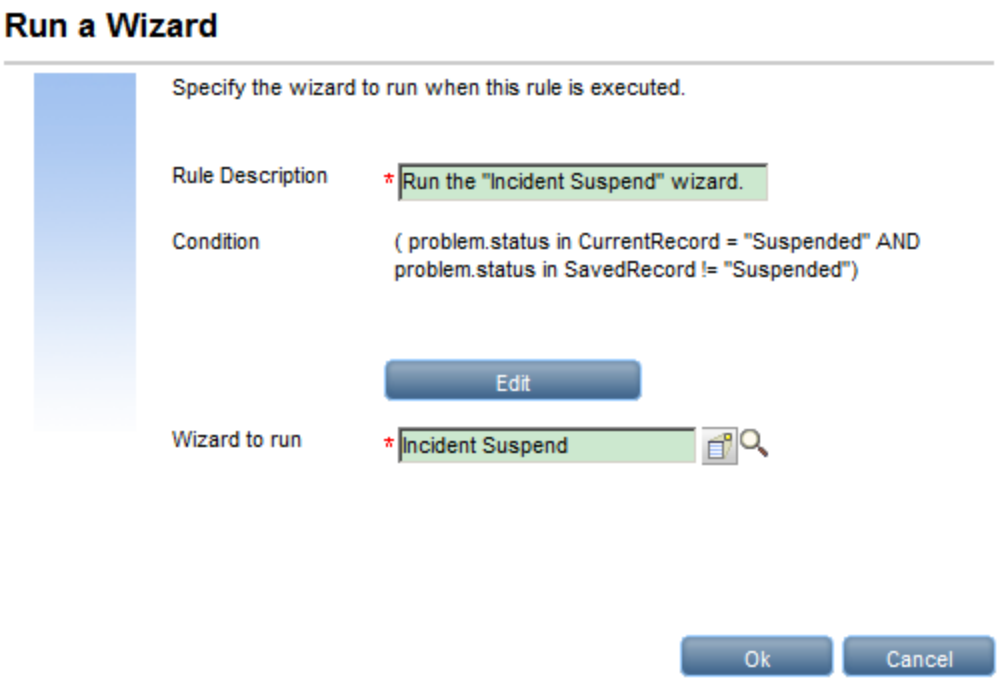


Run a Wizard

In Service Manager Classic, wizards are run by calling the wizard.run RAD application.

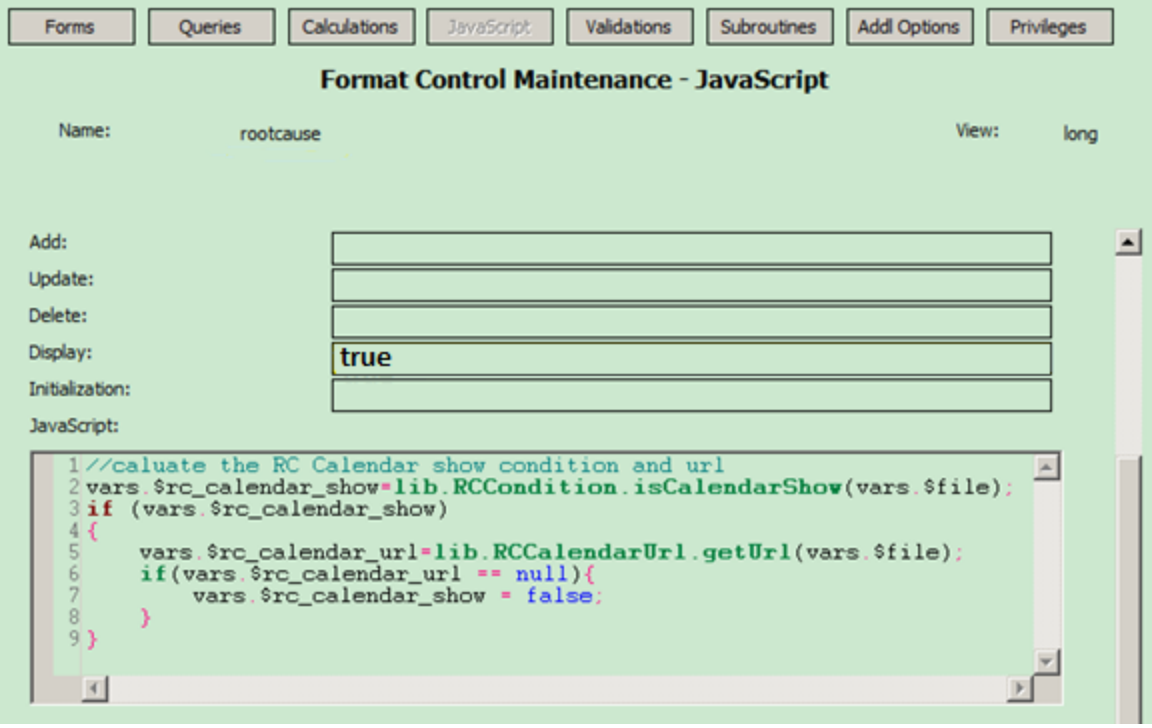


To do this in Service Manager Codeless, you can simply use the Run Wizard rule.

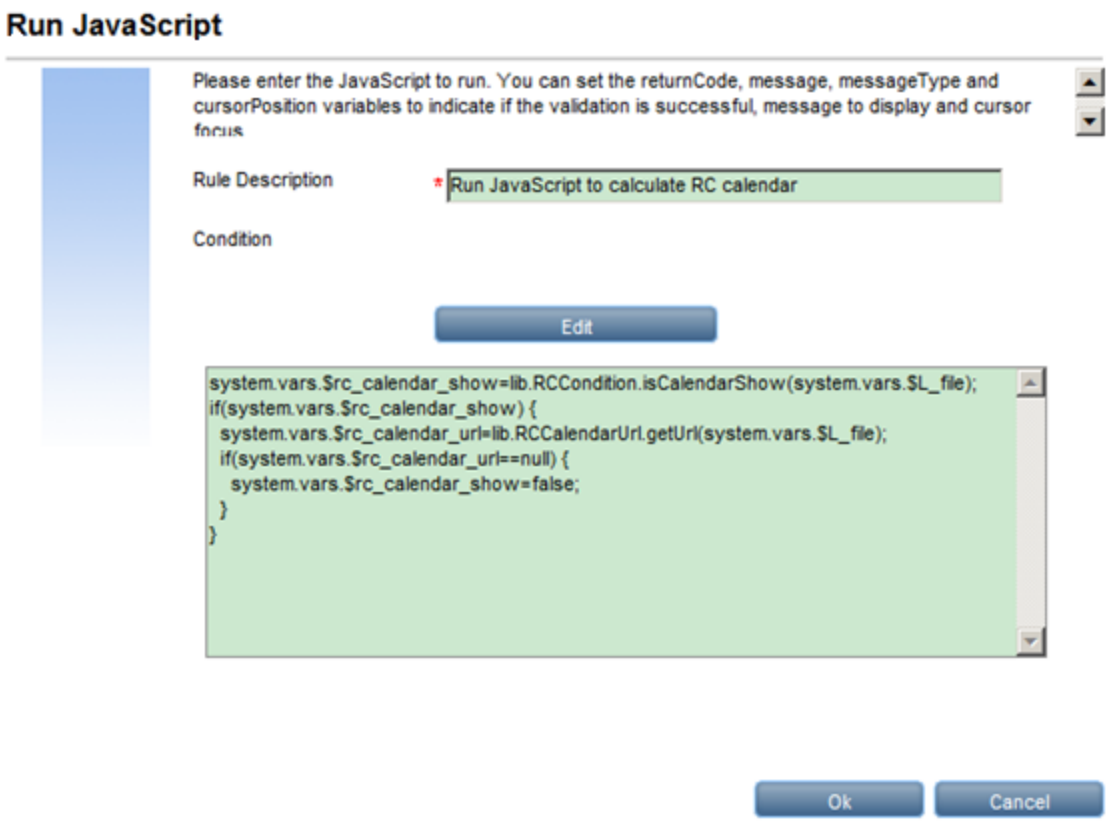


Run a JavaScript

To run a JavaScript by using Format Control, you must define the JavaScript in master format control or format level format control.



To do this in Service Manager Codeless, you can define a Run Java Script rule.



Additional supported rule types

The following rule types are available in out-of-box Service Manager deployments.

Rule type	Description
Launch a URL	Call a URL to launch a web page
Call a process	Call a Service Manager process record
Case Exchange	Trigger certain activities for the Case Exchange integration
Run a wizard	Run a Service Manager wizard
Clear Fields	Clear the specified field and related fields
JavaScript Validation	Use JavaScript to perform actions and validations
Run JavaScript	Use JavaScript to perform actions and validations
Mandatory Fields	Set fields as mandatory

Rule type	Description
Mandatory Variables	Set variables as mandatory
Send Notifications	Send Service Manager notifications
Launch a Script	Launch a Service Manager script
Send HTML Email	Send an HTML Email to users or a group
Start or Stop Clock	Start and stop a Service Manager clock
Set Field	Set a field value using JavaScript
Set Field from Number	Set field based on a number record
Validate Date	Validate a date against a date range
Field Validation Against a List	Validate a field against a list (global or defined)
Validate against Table	Validate a field against a field in another table and fill data into other fields
Validate Text/Number	Validate a field against a range of text or a number in another field of same table
Field Validation Against a Table	Validate a field against a different table
Popup Message Box	Create and configure popup message boxes that appear to end users
Assignment	Automatically distribute records (such as tasks or records) to the groups and individuals who are most able to process them
Run Action	Run actions (defined by rule sets) on records that have a specified relationship to the record that triggers the rule
Run Scheduled Action	Run actions (defined by rule sets) on records after a specified length of time has passed
Group rules	Group multiple rules into a rule group with an overall condition.

Migrate the display options to actions

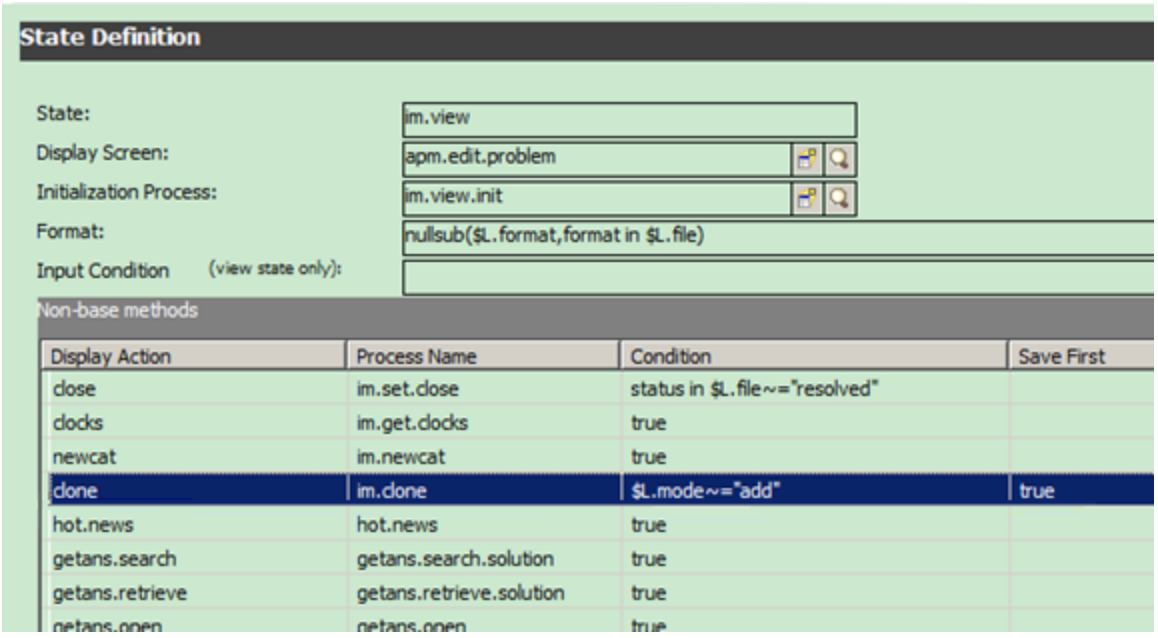
In Service Manager Classic, you must use display options to configure the buttons in a tray, more options list, or form.

In Service Manager Codeless, you can use actions to do this. Workflow actions and Manual workflow transitions appear in trays and more options lists as "virtual" display options, or on forms as buttons.

Display options still work in Service Manager Codeless, but we recommend that you move all display options to workflow actions. To do this, follow these steps:

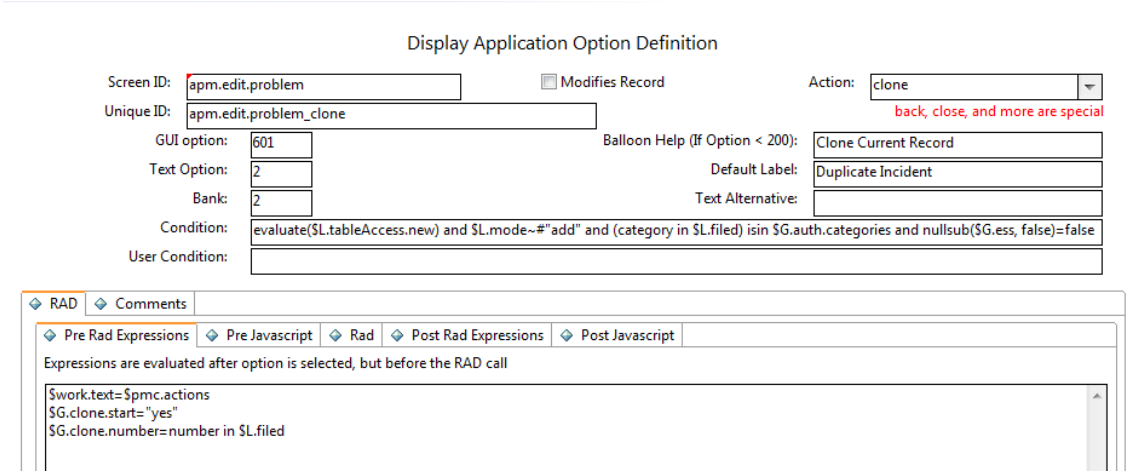
1. Identify your custom actions in the legacy states.
2. Examine the Display Screen field of each state to identify how the action is used in display options.
3. For each of your new custom actions, create a new rule set that calls the process (mapped to the action in the state definition) directly. The "Available as action" option in the rule sets must be selected.
4. Add the new actions to the workflows.

The mapping between the display action and the process is set in the state definition.

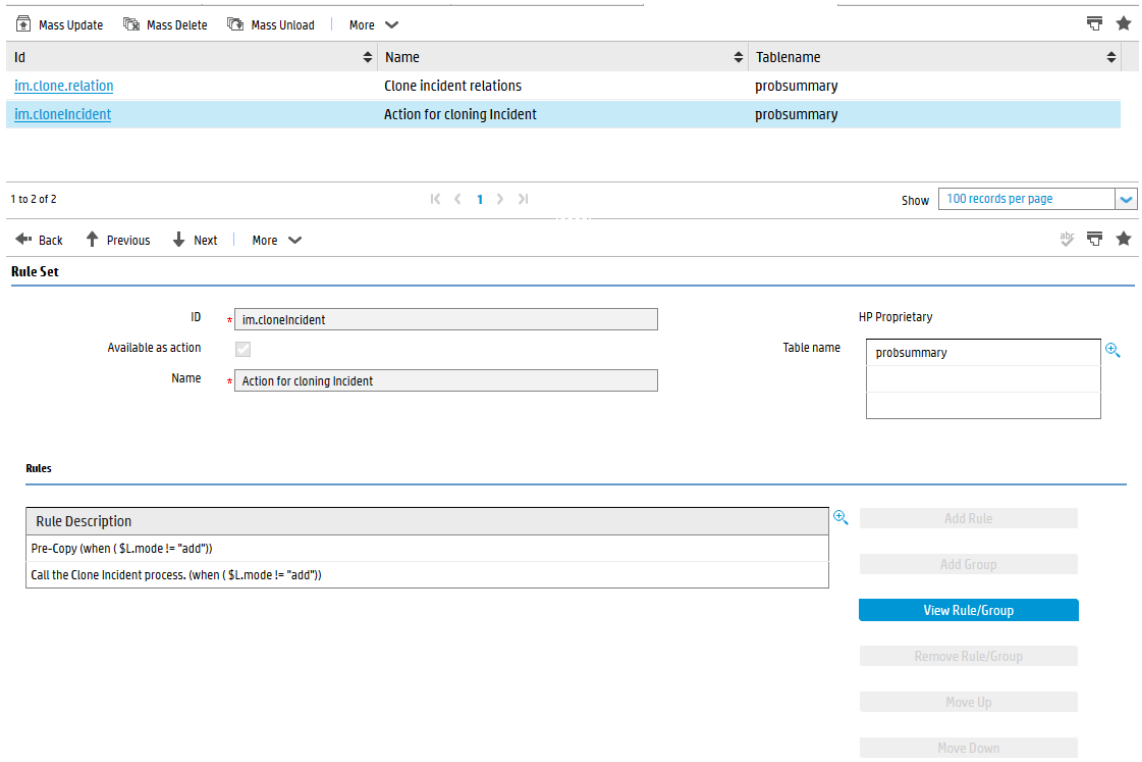


The definition of the display option that uses the action contains the following settings:

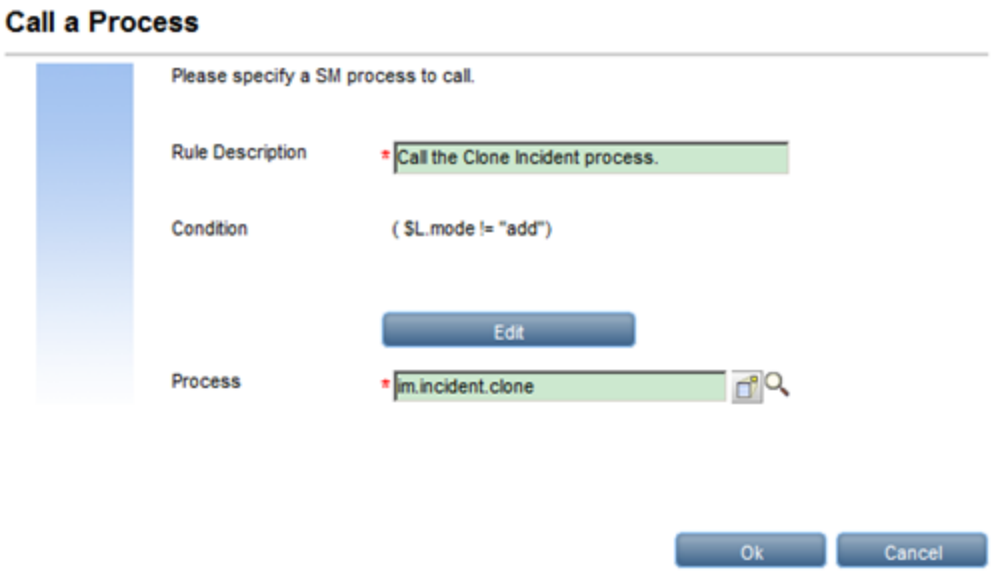
- The value in the **GUI option** field indicates where the display option is located (tray, more option list, or button).
- The **Default Label** field defines the display name.
- The **Condition** field contains a RAD Expression that definesthe security control.



To do this by using a Process Designer action, you must first define a rule set in which the "Available as action" option is selected.

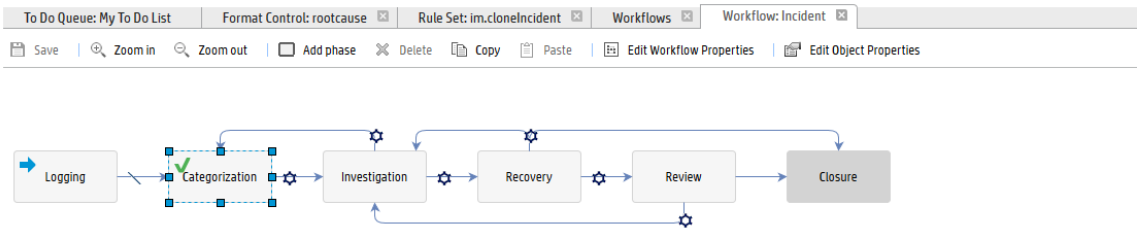


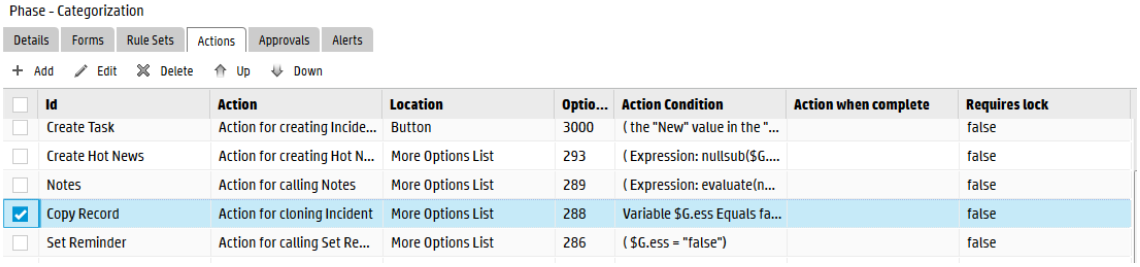
Then, you must add a Call a Process rule to this rule set.



You can use this "action-type" rule set at the workflow or phase levels.

- The Location column indicates where the display option is located (tray, more option list, or button)
- The ID column is set to the display name
- You can use the condition editor to set security controls


To Do Queue: My To Do List
Format Control: rootcause
Rule Set: im.cloneIncident
Workflows
Workflow: Incident
Save
Zoom in
Zoom out
Add phase
Delete
Copy
Paste
Edit Workflow Properties
Edit Object Properties
Logging
Categorization
Investigation
Recovery
Review
Closure


Phase - Categorization
Details
Forms
Rule Sets
Actions
Approvals
Alerts
Add
Edit
Delete
Up
Down
Id
Action
Location
Optio...
Action Condition
Action when complete
Requires lock
Create Task
Action for creating Incide...
Button
3000
(the "New" value in the "...
false
Create Hot News
Action for creating Hot N...
More Options List
293
(Expression: nullsub($G....
false
Notes
Action for calling Notes
More Options List
289
(Expression: evaluate(n...
false
Copy Record
Action for cloning Incident
More Options List
288
Variable $G.ess Equals fa...
false
Set Reminder
Action for calling Set Re...
More Options List
286
($G.ess = "false")
false

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Process Designer Tailoring Best Practices Guide (Codeless Mode) (Service Manager 9.41)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to ovdoc-ITSM@hp.com.

We appreciate your feedback!


Java™
COMPATIBLE