

# HP Service Manager

Software Version: 9.41

For the supported Windows® and UNIX® operating systems

## Process Designer Hybrid Guide (for Service Manager 9.41)

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

© 1994-2015 Hewlett-Packard Development Company, L.P.

### 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://softwaresupport.hp.com/>.

This site requires that you register for an HP Passport and to sign in. 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.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

## 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

Chapter 1: Introduction to Service Manager Hybrid mode .....	5
Overview of Service Manager 9.41 Hybrid .....	5
Chapter 2: How to upgrade to Service Manager 9.41 Hybrid .....	12
Automated migration .....	12
Perform preparatory tasks .....	12
Upgrade the Service Manager binary files to the Service Manager 9.41 GA binary files ...	14
Run the Upgrade Utility to upgrade the Service Manager applications to Service Manager 9.41 Hybrid .....	14
Review the migration log file .....	21
Additional manual migration tasks .....	21
Mandatory manual migration tasks .....	21
Review the migrated validity records .....	22
Review the category-based alerts migration .....	23
Configure the Change Management module to use the new Task Planner .....	28
Merge the status lists .....	41
Update the eventin service .....	43
Perform data migration .....	46
Optional manual migration tasks .....	46
Upgrade to the Process Designer related record subform .....	46
Use the Process Designer workflow viewer in your forms .....	47
Configure the Search form .....	49
Migrate the solution matching configuration from security profiles to security settings .....	50
Update reports to use Process Designer-based category tables .....	53
Make the Category field in the Interaction and Incident forms read-only .....	54
Revert to using the "escalate to Change" wizard .....	55
Remove the button in the Incident form that cancels the Interaction escalation process .....	57
Migrate any customization of the knownerror table .....	58
Remove the legacy Request Management dashboard and report definitions .....	58
Enable legacy Request Management and Process Designer-based Request Fulfillment to run in parallel .....	60
Create a production patch and apply it to the production system .....	85

<b>Appendix A: Changes made to Service Manager by the Applications Upgrade Utility .....</b>	<b>86</b>
Workflow migration .....	91
Security role migration .....	96
Object migration .....	97
Category record migration .....	100
Alerts migration .....	106
Incident solution matching migration .....	106
Legacy category table updates .....	108
<b>Appendix B: Updates to the Process Designer framework .....</b>	<b>112</b>
RAD routines updated .....	112
Workflow form updated .....	112
WSDL definition update .....	113
<b>Send Documentation Feedback .....</b>	<b>114</b>

# Chapter 1: Introduction to Service Manager Hybrid mode

HP Service Manager 9.41 Hybrid is a mode of Service Manager in which Process Designer technology is fully implemented, yet that allows you to continue using legacy technology such as Format Control. Service Manager 9.41 Hybrid eases the transition between Service Manager Classic and Service Manager Codeless by enabling you to continue to take advantage of your previous investments in legacy technology. This mode is available only to customers who are upgrading from a Service Manager 9.3x system that has Process Designer Content Pack 9.30.2 (PDCP3) applied.

**Note:** You can also upgrade to Hybrid mode from a Service Manager 9.3x system that has Process Designer Content Pack 9.30.1 (PDCP2) applied by applying PDCP3 before you upgrade to Service Manager 9.41.

To facilitate the migration of your system to Process Designer, we have incorporated a Process Designer Migration tool into the Applications Upgrade Utility. If you are eligible to upgrade to Service Manager 9.41 Hybrid, additional configuration steps will appear when you run the Applications Upgrade Utility, and your system will be migrated to Process Designer.

This document describes:

- How to use the Applications Upgrade Utility embedded in the Applications Upgrade Utility
- How to perform additional manual configuration
- The changes that the Applications Upgrade Utility makes to your Service Manager deployment

## Overview of Service Manager 9.41 Hybrid

As illustrated by the following table, the technology that underlies each module in the Hybrid and Codeless modes of Service Manager is the same.

Module	Underlying technology		
	Classic	Codeless	Hybrid
Service Desk	Classic tailoring	Process Designer	Process Designer
Incident Management	Classic tailoring	Process Designer	Process Designer

Module	Underlying technology		
	Classic	Codeless	Hybrid
Problem Management	Classic tailoring	Process Designer	Process Designer
Change Management	Classic tailoring	Process Designer	Process Designer
Request Management/Request Fulfillment	Classic tailoring	Process Designer	Process Designer
Service Level Management	Process Designer	Process Designer	Process Designer
Knowledge Management	Process Designer	Process Designer	Process Designer

However, the automated migration performed by the Applications Upgrade Utility migrates some legacy technology to Process Designer (for example, the tool automatically generates Process Designer workflows that continue to use processes, format controls, and legacy states). Additionally, further manual tailoring enables you to continue using even more legacy features (for example, you can run the legacy Request Management module and the Process Designer-based Request Fulfillment module in parallel). For detailed information, see the section on the relevant module below.

The following table provides an overview of the functional implementation in a freshly-upgraded Service Manager 9.41 Hybrid system (that is, this table does not include the possibilities presented by additional manual tailoring).

Module		Functional implementation		
		Classic	Codeless	Hybrid
Service Desk	Workflows	Legacy	OOB Process Designer	OOB Process Designer workflow and migrated legacy workflow
	Objects			Process Designer (migrated from legacy object)
	Security			Process Designer (migrated from legacy profiles)
	Category records			OOB Process Designer records and migrated legacy records
Incident Management	Workflows	Legacy	OOB Process Designer	OOB Process Designer workflow and migrated legacy workflow
	Objects			Process Designer (migrated from legacy object)

Module		Functional implementation		
		Classic	Codeless	Hybrid
	Security			Process Designer (migrated from legacy profiles)
	Category records			OoB Process Designer records and migrated legacy records
	Alerts			OoB Process Designer alerts and migrated legacy alerts
	Solution matching			Legacy or Process Designer (you can choose to continue to use legacy solution matching during the upgrade process)
Problem Management	Workflows	Legacy	OoB Process Designer	OoB Process Designer
	Objects			OoB Process Designer
	Security			Process Designer (migrated from legacy profiles)
	Category records			OoB Process Designer records and migrated legacy records
	Known Errors			Process Designer (migrated from legacy Known Errors)
Change Management	Workflows	Legacy	OoB Process Designer	OoB Process Designer
	Objects			OoB Process Designer
	Security			OoB Process Designer
	Category records			OoB Process Designer
Request Management	Workflows	Legacy	N/A	N/A (unless manually configured)
	Objects			
	Security			
	Category records			

Module		Functional implementation		
		Classic	Codeless	Hybrid
Request Fulfillment	Workflows	N/A	OOB Process Designer	OOB Process Designer
	Objects			
	Security			
	Category records			
Service Level Management	Workflows	OOB Process Designer	OOB Process Designer	OOB Process Designer
	Objects			
	Security			
	Category records			
Knowledge Management	Workflows	OOB Process Designer	OOB Process Designer	OOB Process Designer
	Objects			
	Security			
	Category records			

## Service Desk

The Applications Upgrade Utility makes the following changes to the Service Desk module:

- **Workflows:** In addition to creating an out-of-box Process Designer Service Desk workflow, the Applications Upgrade Utility creates a migrated workflow (called "Migrated ServiceDesk") that is based on your legacy process configuration.
- **Security:** The Service Desk module profile is migrated to Process Designer security roles and security rights.
- **Objects:** The legacy Service Desk objects (incident files) are migrated to Process Designer-based objects.



- **Category records:** Records in the legacy category, subcategory, and producttype tables are migrated to the Process Designer-based sdCategory, sdSubcategory, and sdArea tables, respectively.

## Incident Management

The Applications Upgrade Utility makes the following changes to the Incident Management module:

- **Workflows:** In addition to creating an out-of-box Process Designer Incident workflow, the Applications Upgrade Utility creates a migrated workflow (called "Migrated Incident") that is based on your legacy process configuration.
- **Security:** The Incident Management module profile is migrated to Process Designer security roles and security rights.
- **Objects:** The legacy Incident Management objects (probsummary files) are migrated to Process Designer-based objects.
- **Category records:** Records in the legacy category, subcategory, and producttype tables are migrated to the Process Designer-based imCategory, imSubcategory, and imArea tables, respectively.
- **Alerts:** Legacy category-based alerts are migrated to the workflow phases.
- **Solution matching:** During the upgrade process, you have the option to migrate to Process Designer-based solution matching (which is configured in Process Designer security roles). You can also choose to continue using legacy solution matching (which is configured in legacy security profiles).

## Problem Management

Because of structural differences between the legacy and Process Designer-based Problem Management modules, Problem Management must be fully re-implemented on Process Designer. As such, the Applications Upgrade Utility makes the following changes to the Problem Management module:

- **Workflows:** Legacy Problem Management processes are not migrated to Process Designer workflows. Instead, an out-of-box Process Designer Problem workflow is created.

- **Security:** The Problem Management module profile is migrated to Process Designer security roles and security rights.
- **Objects:** Legacy Problem Management objects are not migrated to Process Designer objects. Instead, out-of-box Process Designer Problem objects are created.
- **Category records:** Records in the legacy category, subcategory, and producttype tables are migrated to the Process Designer-based pbmCategory, pbmSubcategory, and pbmArea tables, respectively.
- **Known errors:** The Known Error records in the legacy Problem table are migrated to the Process Designer-based Problem table.

## Change Management

The Change Management module was already reimplemented on Process Designer when Process Designer Content Pack 9.30.2 was applied (this content pack is a prerequisite for upgrading to Service Manager 9.41 Hybrid). Therefore, in Service Manager 9.41 Hybrid Process Designer is fully implemented in the Change Management module.

## Request Management/Request Fulfillment

In Service Manager 9.40, the Request Management Module was reimplemented on Process Designer as the Request Fulfillment module. When you upgrade to Service Manager 9.41 Hybrid, you are upgraded to Request Fulfillment (that is, Process Designer is fully implemented).

However, to enable this transition, Service Manager 9.41 Hybrid enables you to run Request Management and Request Fulfillment side-by-side. For information about how to do this, see ["Enable legacy Request Management and Process Designer-based Request Fulfillment to run in parallel"](#) on page 60.

## Help center for Service Manager 9.41 Hybrid

There is no Hybrid mode-specific version of the Service Manager help center.

For information about the Process Designer framework and how to tailor Process Designer, please refer to the Service Manager Codeless help center.

However, as modules in Service Manager 9.41 Hybrid support functionality from both Classic and Codeless modes (see above for details), you must refer to either the Service Manager Classic or Service

Manager Codeless help center for information about specific functionality, depending on the functionality that you use for each business module.

## Chapter 2: How to upgrade to Service Manager 9.41 Hybrid

To upgrade to Service Manager 9.41 Hybrid, you will need to perform the following tasks:

1. Plan your upgrade (for information about how to do this, refer to the *HP Service Manager Applications Upgrade Guide*)
2. Run the Applications Upgrade Utility to upgrade your system to Service Manager 9.41 and migrate your system to Process Designer (for information about how to do this, refer to "[Automated migration](#)" below)
3. Perform additional manual configuration (for information about how to do this, refer to "[Additional manual migration tasks](#)" on page 21)
4. Test your system (for information about how to do this, refer to the *HP Service Manager Applications Upgrade Guide*)

### Automated migration

An Applications Upgrade Utility is built into the Applications Upgrade Utility to automate many of the steps to migrate your Service Manager system to Process Designer. The additional configuration screens that comprise the tool are only displayed in the Applications Upgrade Utility if you are eligible to upgrade to Service Manager 9.41 Hybrid (that is, if you are upgrading from a Service Manager 9.3x system that has Process Designer Content Pack 9.30.2 (PDCP3) applied).

This section provides a high-level description of how to use the Applications Upgrade Utility to automatically migrate your system to Process Designer. For more detailed information about how to use the Applications Upgrade Utility, refer to the *HP Service Manager Applications Upgrade Guide*.

### Perform preparatory tasks

Before you can upgrade to Service Manager 9.41 Hybrid, you must prepare your system. To do this, perform the following tasks:

1. Increase the value of the *shared\_memory* parameter in the sm.ini file to a value over 96000000.
2. Use the Windows client to disable the **Client side load/unload** option. To do this, click **Window > Preferences**, and then click to clear the option.
3. Disable Development Audit. To do this, navigate to **Tailoring > Audit** in the System Navigator, and then click **Turn Auditing On/Off**. Clear the **Do you want to audit development changes?** option if it is selected, and then save the change.
4. Remove any SMSQL\* dbdict records. To do this, navigate to **Tailoring** in the System Navigator, and then click **Database Dictionary**. Perform a search for SMSQL\* dbdict records, and delete any that exist. Dbdict records are normally temporarily generated during a full-table copy operation.
5. Clean up any records in the following log files:
  - Msglog
  - Syslog
  - Errorlog

**Note:** Back up the table data if you need it later. If the tables contain too many records, the upgrade process will be slow.

6. Clean up the following messaging files:
  - Mail
  - Eventin records
  - Eventout records

**Note:** Back up the table data if you need it later. If the tables contain too many records, the upgrade process will be slow.

7. Clean up any files in the systemperform table.

**Note:** Back up the table data if you need it later. The systemperform dbdict will be converted from lob columns to real database columns. Therefore, if there are too many records in the

systemperform table, transfer.bin will load very slowly.

8. Update the Service Manager configuration files. To do this, follow the instructions in the "Update Service Manager configuration files" step of the "Upgrade tasks on development environment" chapter of the *HP Service Manager Applications Upgrade Guide*.

Restart the Service Manager server before you perform the upgrade in order for these tasks to take effect.

## Upgrade the Service Manager binary files to the Service Manager 9.41 GA binary files

We strongly recommend that you use HP ITSM Deployment Manager to install the Service Manager 9.41 GA binaries. For more information, please refer to the ITSM Deployment Manager documentation.

## Run the Upgrade Utility to upgrade the Service Manager applications to Service Manager 9.41 Hybrid

To upgrade the Service Manager applications, follow these steps.

### Step 1: Purge the existing upgrade files

If you have run an applications upgrade in the past, there may be some artifacts left over from that upgrade process that need to be removed. To do this, follow these steps:

1. In the Service Manager Client command box, type `*aapm.upgrade.purge`, and then press Enter.
2. Select **I'm done, and I want to remove the upgrade files completely**.
3. Click **OK**.

### Step 2: Load the Upgrade Utility files

**preupg.bin**

Use Database Manager to load the `preupg.bin` file, which is located in root folder of the unzipped Service Manager 9.41 Upgrade Utility package. This process may take several minutes. To load the `preupg.bin` file, follow these steps:

1. In the Service Manager Client command box, type `db`, and then press Enter.
2. Right-click the Database Manager form, and select **Import/Load**.
3. Select **preupg.bin**, and then click **Load FG**.

#### **transfer.bin**

Use Database Manager to load `transfer.bin`, which is located in root folder of the unzipped Service Manager 9.41 Upgrade Utility package. This process may take several minutes. To load the `transfer.bin` file, follow these steps:

1. In the Service Manager Client command box, type `smupgrade`, and then press Enter.
2. Click **Load Transfer**, enter the path to the `transfer.bin` file (but do not include "transfer.bin"), and then click **Next**.
3. Log out of Service Manager, and then log back in to Service Manager.

### Step 3: Run the SQL compare utility

#### **Load sqlupgrade.bin**

To load the `sqlupgrade.bin` file, follow these steps:

1. In the Service Manager Client command box, type `db`, and the press Enter.
2. Right-click the Database manager form, click **Import/Load**, and then select **sqlupgrade.bin**.
3. Click **Load FG**.

#### **Run the SQL compare utility**

To run the SQL compare utility, follow these steps:

1. In the Service Manager Client command box, type `smupgrade`, and the press Enter.
2. In the SQL Compare Utility section, click **Run SQL Compare Utility**.
3. Enter the full path to the `upgdbdct.dba` file, and then click **Apply**.

When the utility has finished running, check the two types of SQL compare results (SQL field compare results and SQL unique key compare results).

### **View and export the results**

To view and export the utility results, follow these steps:

1. In the Service Manager Client command box, type `Smupgrade`, and then press Enter.
2. Click **View SQL Compare Results** or **View SQL Unique Key Compare Results**.
3. Click **Search** to display the results in a record list.
4. Configure the display columns as required, and then export the results.

## Step 4: Apply the initial out-of-box upgrade

The following illustrations provide a high-level demonstration of the process to perform an initial out-of-box upgrade.

**Note:** We strongly recommend that you carefully read the *HP Service Manager Applications Upgrade Guide* for detailed information and to achieve a successful upgrade.

To run the HP Service Manager Upgrade Utility, follow these steps:

1. Type `smupgrade` in the Service Manager command line, and then press Enter to launch the Upgrade Utility.
2. In the Upgrade Utility section, click **Service Pack**.
3. In the **Upgrade Processing** section, click **Apply an Upgrade**.
4. On the Welcome screen, verify that the "Applications version upgrading from" field displays your current application version, and then click **Next** to continue.

5. The system displays the following message:

Are you going to use this system to create a custom upgrade for another system?

Select **Yes**, and then click **Next**.

6. The system displays the following message:



What is the fully qualified path to the Service Manager Upgrade patch files?

By default, the text box displays the fully qualified path to the Upgrade folder on the Service Manager server. Verify that the path is correct, and then click **Next**.

7. The system displays the following message:

When HP Service Manager Upgrade doesn't recognize an object it should:

Select **Install HP's Version of the Object Alongside Your Own**, and then click **Next**.

8. The system displays the following message:

Do you want to enable automatic merge and which version will be used as the base version?

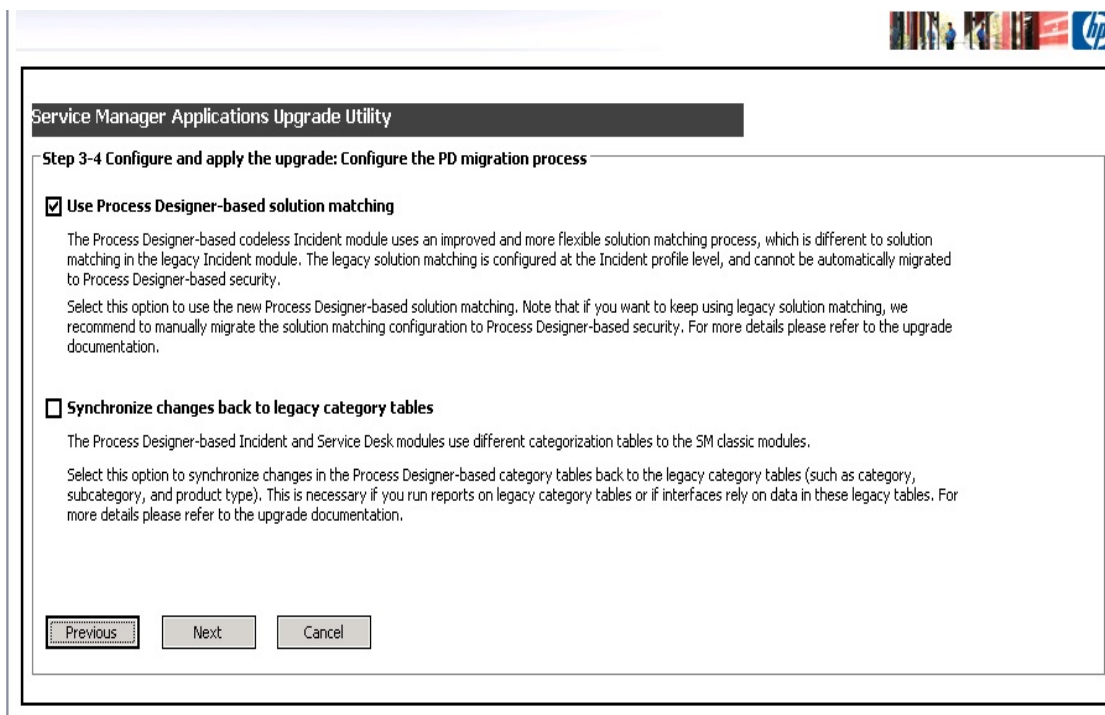
Select the **Enable** check box, select a base version from the list, and then click **Next**.

9. The following message is displayed:

Do you want to force the replacement of the objects?

Select the **Replace RAD** option, and then click **Next**.

10. The following screen is displayed, which allows you to choose whether or not you want to synchronize the legacy categorization tables with the new Process Designer-based tables, and whether or not you want to migrate to Process Designer-based Incident solution matching.



Process Designer-based and legacy versions of Service Manager use different categorization tables. The Applications Upgrade Utility can update the legacy categorization tables (such as the category, subcategory, and product type tables) to synchronize them with the new Process Designer-based categorization tables. This enables you to run reports on legacy categorization tables and to continue to use interfaces that rely on data in these legacy tables.

**Note:** For more information about the changes that are applied to the categorization tables by the Applications Upgrade Utility, see "[Legacy category table updates](#)" on page 108.

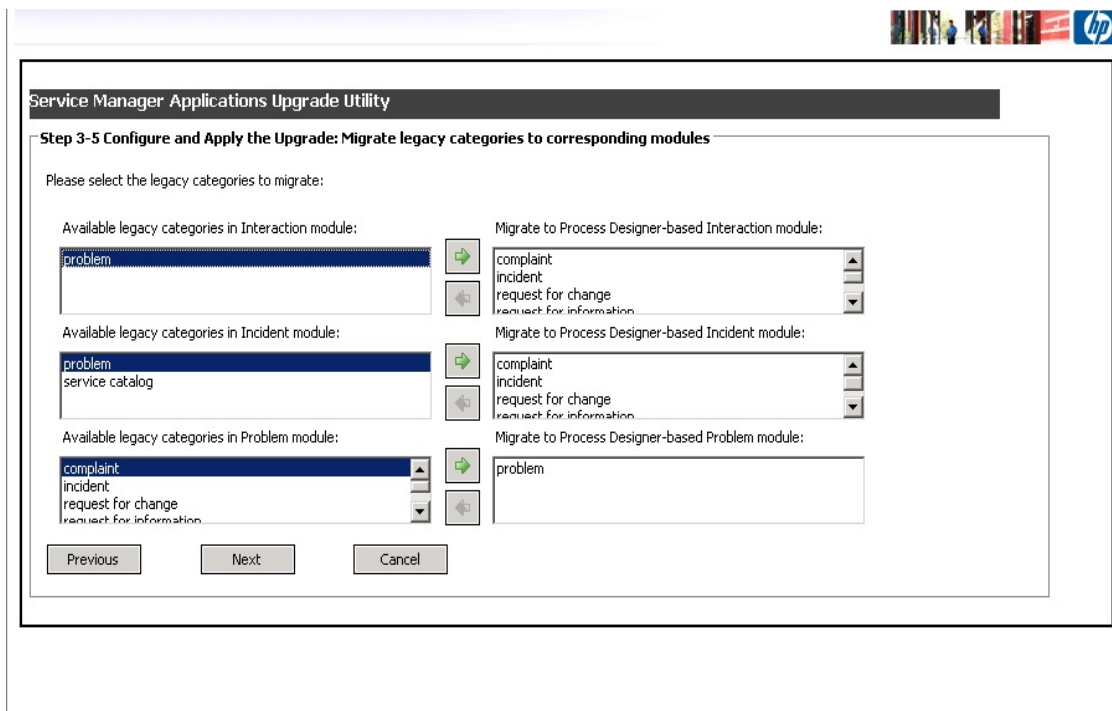
Next, decide whether you want to migrate from legacy solution matching to Process Designer-based solution matching. By default, the option to migrate to Process Designer-based solution matching is selected.

**Note:**

For more information about the changes that are applied to solution matching by the Applications Upgrade Utility, see "[Incident solution matching migration](#)" on page 106.

Legacy solution matching is configured in security profiles. Therefore, if you want to continue to use legacy solution matching, you must manually migrate the solution matching configuration to Process Designer security roles. For more information about how to do this, see ["Migrate the solution matching configuration from security profiles to security settings"](#) on page 50.

11. The following screen is displayed, which enables you to configure the migration of category records.



Select a legacy category from a left-hand pane (legacy categories), and then click the green arrow to move the category to the right-hand pane (categories to be migrated to Process Designer). Repeat this process for all the categories that you want to migrate, and then click **Next**.

For more information about category record migration, see ["Category record migration"](#) on page 100.

12. When you are asked whether you want to proceed, click **Yes**. The Upgrade Utility displays the status as the upgrade is performed.

**Note:** The upgrade process may take several hours to complete.

13. When you receive an Upgrade is complete message, the Upgrade Utility has finished the data processing, and you can follow the instructions in the message to complete the next steps. After you close the message dialog box, you are automatically logged out.

**Note:** We strongly recommend that you back up your database before close the message dialog box.

14. Restart the server and log back in to the client.

## Step 5: Check the upgrade results

For more information about this step, refer to the *HP Service Manager Applications Upgrade Guide*.

## Step 6: Check the upgrade logs

- Check the upgrade.log and detail.log files (in the upgrade folder). Neither log file should contain any significant issue.
- Check the except.log file (in the upgrade folder)
- Check the Process Designer Hybrid migration log file (sm.log) and the Process Designer Hybrid code migration report file (pdmigration.log) in the SM log folder.

## Step 7: Check for and resolve issues with objects

For more information about this step, refer to the *HP Service Manager Applications Upgrade Guide*. To check for and resolve issues with objects, perform the following tasks:

- Resolve dbdict exceptions
- Resolve error objects
- Check auto-merged objects
- Check replaced objects
- Check renamed objects

- Check previously reconciled objects
- Resolve initial conflict objects, such as Object, States, Process, displayscreen, and displayoption
- Resolve any other conflicting objects

## Review the migration log file

After the Applications Upgrade Utility has finished running, the changes made to your system are recorded in the Process Designer migration log file (pdmigration.log) in the SM log folder. It is important that you review this file and make any required changes.

The log file includes the following information:

- A list of Alert definition records that are created based on settings in your legacy categories
- A list of category, subcategory, and producttype records that are migrated to Process Designer-based tables.
- A list of format level link records that are updated to disable the category, subcategory, and product.type link lines.
- A list of the legacy probsummary and incident objects that are migrated to Process Designer-based objects
- A list of updated process records

## Additional manual migration tasks

After you have run the SM9.41 Applications Upgrade Utility to migrate your system to Service Manager 9.41 Hybrid, you must complete a number of manual configuration tasks. Some tasks are optional, depending on how you want to configure your Service Manager 9.41 Hybrid system.

## Mandatory manual migration tasks

**Note:** Some manual migration tasks are only mandatory under certain conditions. See each task for further information.

## Review the migrated validity records

After you migrate to Process Designer, validity records validate against the Process Designer category-related tables instead of the legacy category-related tables. Therefore, a manual review of the validity records is required.

The Process Designer migration report is included in the `pdmigration.log` file, and appears as follows:

The following validity records are updated to perform validation against Process Designer-based category tables (such as `category`, `subcategory`, and `producttype`) instead of legacy category tables. Backups of the original validity records (identified by the suffix `"_disabled"`) have been created.

```
Unique ID:BP;Field Name:category;Sequence:1
Unique ID:BP;Field Name:category;Sequence:3
Unique ID:BP;Field Name:incident.category;Sequence:3
Unique ID:BP;Field Name:product.type;Sequence:1
Unique ID:BP;Field Name:product.type;Sequence:2
Unique ID:BP;Field Name:product.type;Sequence:3
Unique ID:BP;Field Name:product.type;Sequence:5
Unique ID:BP;Field Name:product.type;Sequence:7
Unique ID:BP;Field Name:product.type;Sequence:10
Unique ID:BP;Field Name:product.type;Sequence:13
Unique ID:BP;Field Name:subcategory;Sequence:1
Unique ID:BP;Field Name:subcategory;Sequence:2
Unique ID:BP;Field Name:subcategory;Sequence:3
Unique ID:BP;Field Name:subcategory;Sequence:5
Unique ID:BP;Field Name:subcategory;Sequence:11
Unique ID:BP;Field Name:subcategory;Sequence:12
```

The number of updated validity records depends on your system configuration. If you have not customized any validity records, you can ignore this step.

**Note:** If you have added dbdict fields to the category tables and if you use the validity table to validate against these fields, you must manually change the validation logic for these fields.

## Review the category-based alerts migration

### Stage 1, 2, and 3 alerts and deadline alerts

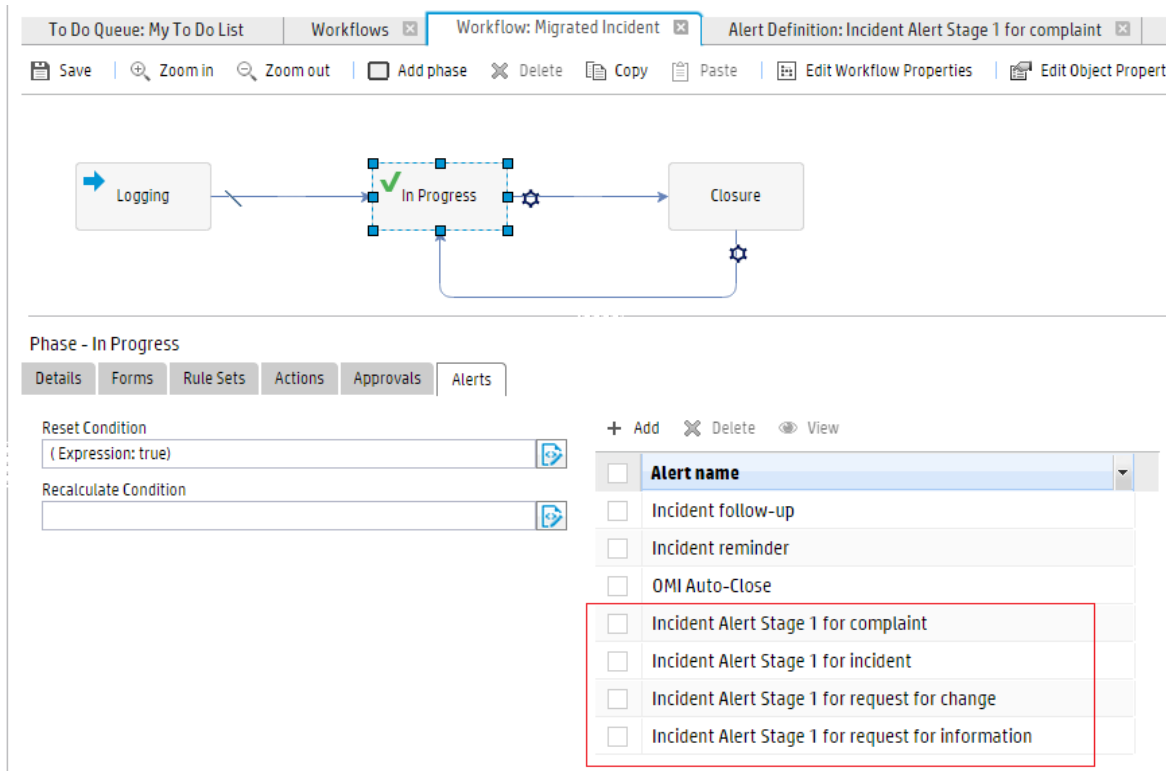
The Applications Upgrade Utility migrates the legacy Stage 1, Stage 2, Stage 3, and Deadline alerts and Deadline Alert Group configurations from legacy categories to the workflow phases of the automatically-generated Incident workflow.

The migrated alerts are named according to their source. For example, "Incident Alert Stage 1 for complaint" indicates that the alert is converted from stage 1 configuration of the complaint category. The following types of alert are created:

- **Incident alert stage 1 for <category name>**. This alert updates the incident alert status to "alert stage 1" if the category has a stage 1 interval or expression specified.
- **Incident alert stage 2 for <category name>**. This alert updates the incident alert status to "alert stage 2."
- **Incident alert stage 3 for <category name>**. This alert updates incident alert status to "alert stage 3."
- **Incident Deadline for <category name>**. This alert updates the incident alert status to "DEADLINE ALERT."

	Interval	Expression
Stage 1:	<input type="text"/>	If (problem.status in \$file)#"Suspend" then (alert.time in \$file+unsuspend.time in \$file) else (alert.time in \$file=NULL
Stage 2:	<input type="text"/>	
Stage 3:	<input type="text"/>	
Deadline:	<input type="text"/>	
Reassignment:	00:00:00	
Reassign Count Threshold:		8
Deadline Alert Group:	<input type="text"/>	

The migrated alerts are located in the alerts tab of each workflow phase.



## Gap and limitations

The migrated alerts implement the main functionality of the legacy alerts; however, there are some gaps and limitations.

- In legacy category-based alert expressions, "\$file" represents the incident. The Applications Upgrade Utility automatically modifies "\$file" to "\$L.file" for compatibility with the migrated alerts. If you use other variables in your legacy system, you must manually modify the expression in the migrated alerts.
- The timing for the generation of stage 2 and 3 alerts is different. In the legacy Incident module, the schedule for stage 2 is generated after the schedule for stage 1 is executed and the incident alert status is updated to stage 1. Likewise, the schedule for stage 3 is generated after the schedule for stage 2 is executed and the incident alert status is updated to stage 2. In Hybrid mode, however, all schedule records for stage 1, stage 2, stage 3, and deadline alerts are generated or re-generated at the same time, when the incident record is updated.

Therefore, if you are using an expression that uses the content of the current record (\$file) as either the condition or the calculation for the alert time in stage 2, stage 3, or deadline alerts, issues may



occur because the content of the current record can differ when the schedule is generated. For example, you configure your category as follows.

	Interval	Expression
Stage 1:	<input type="text"/>	if (problem.status in \$file)#"Suspen" then (alert.time in \$file=unsuspend.ti
Stage 2:	<input type="text"/>	alert.time in \$file=sysmodtime in \$file
Stage 3:	00:00:05	<input type="text"/>
Deadline:	<input type="text"/>	<input type="text"/>
Reassignment:	00:00:00	<input type="text"/>
Reassign Count Threshold:	<input type="text" value="8"/>	
Deadline Alert Group:	<input type="text"/>	

- The migrated alerts are moved to the alerts list of the "In Progress" phase of the migrated Incident workflow. If you add new phases, you need to also add these alerts to new phases so that the alerts can work for incident in the new phases.
- The alert reset and recalculation conditions in objects are changed in Hybrid mode. Therefore your original alerts may not work. For example, the out-of-box alert reset condition in the legacy probsummary object is `category in $L.file="incident"` and not `(null(1 in external.process.reference in $L.file))`. If you use this condition, alerts only work for incidents that are assigned to the incident category. In the migrated probsummary object, the condition is `evaluate(nullsub(parse(str(alertsReset in $L.wfPhase), 2), false)) or not (same(current.phase in $L.file, current.phase in $L.file.save))`, and the alert reset condition of the "In Progress" phase is set to "true." Therefore, all alerts configured in the phase will work for all incidents rather than only for incidents that are assigned to the incident category.

In this case, you can modify the schedule condition and the alert condition in the existing alert definition of the Incident Management module by appending the original reset condition that is configured in the legacy object. For example, update the "OMI Auto-Close" alert definition as displayed in the following image.

### Reassignment threshold

The reassignment interval and expression and the reassignment count threshold in the legacy categories are not migrated automatically.

If you still need reassignment threshold functionality, you can add the relevant fields to the new Process Designer-based category file and form, and then copy the configuration to the new Process Designer categories. The following table lists the fields that you will need to add.

Field name in dbdict	Data type	Label in form/description
count	number	Reassign Count Threshold
reassign	date/time	Reassignment Interval
reass.expression	expression	Reassignment Expression



## Configure the Change Management module to use the new Task Planner

An improved version of the Task Planner was introduced in Service Manager 9.40. To use it, make sure that the following code elements are successfully implemented (to disable the previous version of the Task Planner). To do this, you can either use conflict resolution or manually change the code elements according to following table.

Record type	Record name	Modifications	Comments
Object	changeModel	Uses the "Change Model" workflow, the workflow location is "In Object," and the file-level rule sets are configured to use the new Task Planner.	
	cm3t	<p>The following object-level rule sets are configured to use the new Task Planner:</p> <ul style="list-style-type: none"><li>• <b>After successful add:</b>  <code>.common.taskplan.noneplantasktochangeplan</code></li><li>• <b>On enter:</b>  <code>.common.taskplan.validateoutputontask</code></li><li>• <b>After successful enter:</b>  <code>.common.taskplan.sync.task.titleandstatus.tochangeplan</code>  <code>.common.taskplan.updatedependenttaskstatus</code></li><li>• <b>Initialization:</b></li></ul>	

Record type	Record name	Modifications	Comments
		<pre>.common.taskplan.taskticket.init</pre> <ul style="list-style-type: none"> <li>• On display: <pre>.common.workflow.init.vars</pre> </li> <li>• On update: <pre>.common.taskplan.savecontextconfig</pre> </li> <li>• After successful update: <pre>.common.taskplan.sync.task.titleandstatus.tochangeplan</pre> </li> </ul>	
Process	changeModel.buildtable	<p>The following initial JavaScript is commented:</p> <pre>lib.changeModel.saveFromXML(true);</pre>	<p>This record was used by optionid 441 to save taskxml files to the changeModel table for the legacy Task Editor.</p>
	changeModel.init	<p>The following initial JavaScript is commented:</p> <pre>lib.changeModel.createXML(vars.\$L_file);</pre>	<p>This record was used to initialize a taskxml file to be displayed by the legacy Task Editor.</p>
	changeModel.save	<p>The following initial JavaScript is commented:</p> <pre>lib.changeModel.saveFromXML();</pre>	<p>This record was used to save taskxml files to the changeModel table.</p>

Record type	Record name	Modifications	Comments
	change.open.save	<ul style="list-style-type: none"> <li>• RAD expressions are evaluated before the change.openPlanTasks RAD call.</li> <li>• The following expression is removed:                             <pre>if (\$L.exit="added" and filename(\$L.file)="cm3r") then (\$L.changePlan=jscall("changePlan.createPlan", \$L.file, changeModel in \$L.file);\$L.void=rtecall("copycurrent", \$L.errcode, \$L.phase.save, \$L.phase))</pre> </li> <li>• Disabled calling the change.openPlanTasks RAD.</li> </ul>	
	change.update.save	<ul style="list-style-type: none"> <li>• RAD expressions are evaluated before the apm.mb.ok RAD call.</li> <li>• The following expressions are removed:                             <ul style="list-style-type: none"> <li>◦ if (filename(\$L.file)="cm3r") then (\$L.changePlan=jscall ("changePlan.getPlan", \$L.file))</li> <li>◦ \$L.task.list = "";if (filename(\$L.file) = "cm3r" and not (same(current.phase in \$L.file, \$L.orig.phase))) then (\$L.task.list = jscall ("changePlan.getNeedToCloseTaskList", \$L.orig.phase, \$L.changePlan, nullsub(allow.open.tasks in \$L.phase, false), workflowName in \$L.wfPhase, current.phase in \$L.file))</li> <li>◦ if (\$L.task.list~="") then (\$L.continue = false;\$L.exit="bad.val"; \$L.opentask.msg = scmsg(550, "cm3", {\$L.task.list}))</li> <li>◦ if (\$L.continue=false and not (null(\$L.opentask.msg)) and \$G.bg) then (\$L.void=rtecall("msg", \$L.rc, \$L.opentask.msg))</li> </ul> </li> </ul>	

Record type	Record name	Modifications	Comments
		<ul style="list-style-type: none"> <li>• Disabled calling the apm.mb.ok RAD</li> <li>• RAD expressions are evaluated before the change.openPlanTasks RAD call.</li> <li>• The following expression is removed:   <pre>if (filename(\$L.file)="cm3r") then (\$L.success.flg=rtecall ("refresh", \$L.errcode, \$L.changePlan))</pre> </li> <li>• Disabled calling the change.openPlanTasks RAD</li> </ul>	
	changePlan.buildtable	<ul style="list-style-type: none"> <li>• RAD expressions are evaluated before the change.openPlanTasks RAD call:</li> <li>• The following expression is removed: <ul style="list-style-type: none"> <li>◦ if (filename(\$L.file)="cm3r") then (\$L.changePlan = jscall ("changePlan.getPlan", \$L.file))</li> </ul> </li> <li>• Post RAD expressions: the following expressions are removed: <ul style="list-style-type: none"> <li>◦ \$L.void=jscall("changePlan.createXML", \$L.file)</li> <li>◦ \$L.exit="refresh"</li> </ul> </li> <li>• Disabled calling the change.openPlanTasks RAD</li> <li>• The following initial JavaScript is commented:   <pre>lib.changePlan.saveFromXML(true);</pre> </li> </ul>	This record is used by optionid 441 to save taskxml files into the changeModel table for the legacy Task Editor.
ScriptLibrary	ApplyChangeModel	The following script is commented: addTasksToChangePlan(change, changeModel);	

Additionally, you must check that the following code elements are correctly implemented before you use the new Task Planner. To do this, you can either use conflict resolution or manually change the code elements according to following table.

<b>Record type</b>	<b>Record name</b>	<b>Modifications</b>	<b>Comments</b>
format	chm.cm3r.tasks.horz	The old Task Editor widget is replaced with the new Task Planner widget.	These are the out of box formats. Please customize your own formats accordingly if you are not using out of box formats.



<b>Record type</b>	<b>Record name</b>	<b>Modifications</b>	<b>Comments</b>
	chm.task	A new Task Context tab is added to dynamically display context-related record information that is planned during the task plan.	

<b>Record type</b>	<b>Record name</b>	<b>Modifications</b>	<b>Comments</b>
	chm.standard.registration		

<b>Record type</b>	<b>Record name</b>	<b>Modifications</b>	<b>Comments</b>
	chm.standard.pln.sch		

<b>Record type</b>	<b>Record name</b>	<b>Modifications</b>	<b>Comments</b>
	chm.standard.execution		

<b>Record type</b>	<b>Record name</b>	<b>Modifications</b>	<b>Comments</b>
	chm.standard.pir		

Record type	Record name	Modifications	Comments
	chm.standard.backout		
	chm.normal.registration		
	chm.normal.rsk.imp		
	chm.normal.tcab.approval		
	chm.normal.build.test		
	chm.normal.dcab.approval		
	chm.normal.deployment		
	chm.normal.backout		
	chm.normal.pir		
	chm.emergency.registration		
	chm.emergency.rsk.imp		
	chm.emergency.approval		
	chm.emergency.build.test		
	chm.emergency.implmt		
	chm.emergency.backout		
	chm.emergency.pir		
Triggers	trigger.name="trigger.changeModel.remove.taskplan" and table.name="changeModel"	A new trigger is added to remove task plan-related information once a changeModel record is deleted.	

## Update your customized formats

If you do not plan to use the out-of-box ChangeModel or Task formats, you must configure the Task Planner widget in your tailored ChangeModel and Task formats as described in the following table.

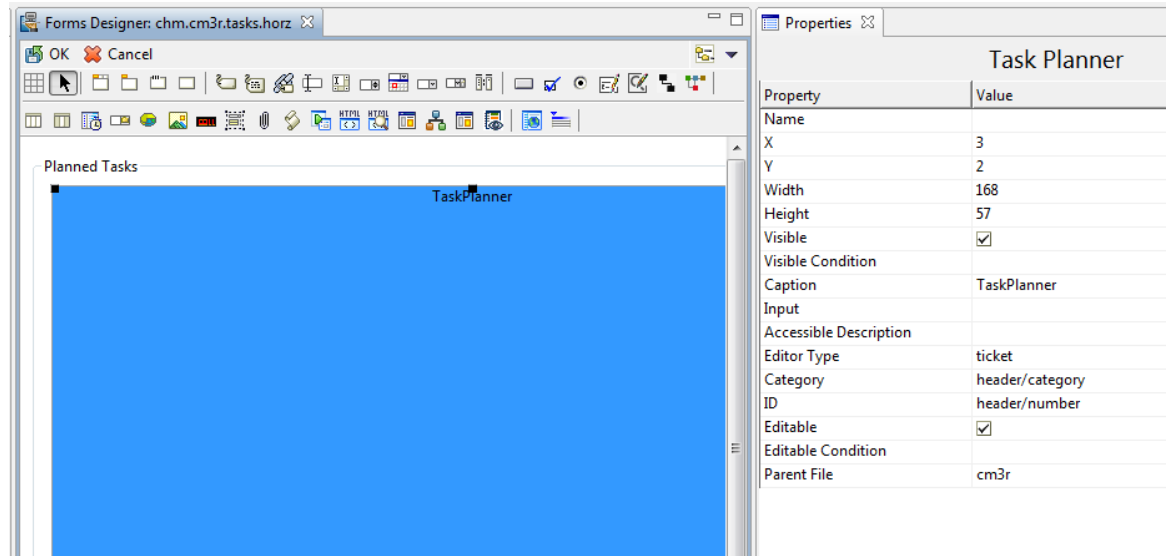
Task Planner property	Value in ChangeModel format	Value in Task format
Editor type	Model	Ticket
Category	Category	Header/category
ID	id	Header/number
Editable	Yes	Yes
Parent file	cm3r	cm3r

### ChangeModel format

The screenshot shows the Forms Designer interface for configuring a Task Planner widget. The main window displays a form with various fields. The Properties panel on the right shows the configuration for the Task Planner widget, with the Editor Type property set to 'model'.

Property	Value
Name	taskplanner1.397638248096
X	3
Y	2
Width	162
Height	57
Visible	<input checked="" type="checkbox"/>
Visible Condition	
Caption	Task Planner
Input	
Accessible Description	
Editor Type	model
Category	category
ID	id
Editable	<input checked="" type="checkbox"/>
Editable Condition	
Parent File	cm3r

## Task format



Additionally, you must manually add the "Additional Properties" tab to the Change and Task details forms, in order to display the Additional Properties information for Changes and Tasks. To do this, manually add references to the following subformats in the relevant formats:

- **Change format**

`common.taskplan.parent.InputOutput.subform`

- **Task format**

`common.taskplan.task.InputOutput.subform`

## Data migration

For more information about how to migrate your data, refer to the ["Perform data migration"](#) section.



## Merge the status lists

The Applications Upgrade Utility implements the Process Designer search form and the Process Designer status list (for more information, see the ["Configure the Search form"](#) section). Therefore, you must merge the non-Process Designer-based status list into the Process Designer-based status list.

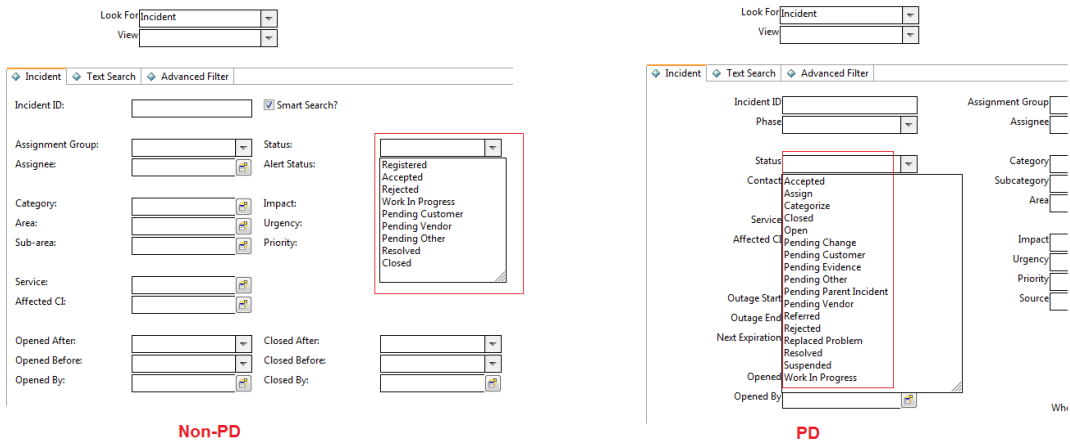
The following table describes the table in each module that you must update, along with information about the Globallist, Globalvariables, and where the data comes from.

We recommend that you modify the data in the ModuleStatus database according to your needs.

<b>Module</b>	<b>Global list name</b>	<b>Process Designer statuses global list variable</b>	<b>Details</b>
Incident	Incident Local Statuses	\$G.imStatuses	Retrieve from file ModuleStatus with Limiting SQL module="probsummary"
Service Desk	Interaction Local Statuses	\$G.sdStatuses	Retrieve from file ModuleStatus with Limiting SQL module="incidents"
Problem	Problem Local Statuses	\$G.pbmStatuses	Retrieve from file ModuleStatus with Limiting SQL module=" rootcause"
Problem	Problem Task Local Statuses	\$G.pbmTaskStatuses	Retrieve from file ModuleStatus with Limiting SQL module=" rootcausetask"

**Note:** Both the Process Designer-based and non-Process Designer-based Change modules use the same status list.

The following illustration displays the difference between the Incident Status list in the non-Process Designer-based and Process Designer-based search forms.



For example, you must update the ModuleStatus list for the Incident module to add or remove statuses according to your requirements.

Module	Status
probsummary	Accepted
probsummary	Assign
probsummary	Categorize
probsummary	Closed
probsummary	Open
probsummary	Pending Change
probsummary	Pending Customer
probsummary	Pending Evidence
probsummary	Pending Other
probsummary	Pending Parent Incident
probsummary	Pending Vendor
probsummary	Referred
probsummary	Rejected
probsummary	Replaced Problem
probsummary	Resolved
probsummary	Suspended
probsummary	Work In Progress



## Update the eventin service



If you use the eventin service to create Incidents, and if the RAD application of your Event Register for creating the Incidents is set to "axces.apm," you must check whether the "add" action in the document engine's state records action blocks externally-created Incidents.

To do this, you need to identify which process maps to the "add" action in the non-Process Designer Open State. The following illustration uses the im.first process as an example from an out-of-box system.

**State Definition**

State:

Display Screen:   

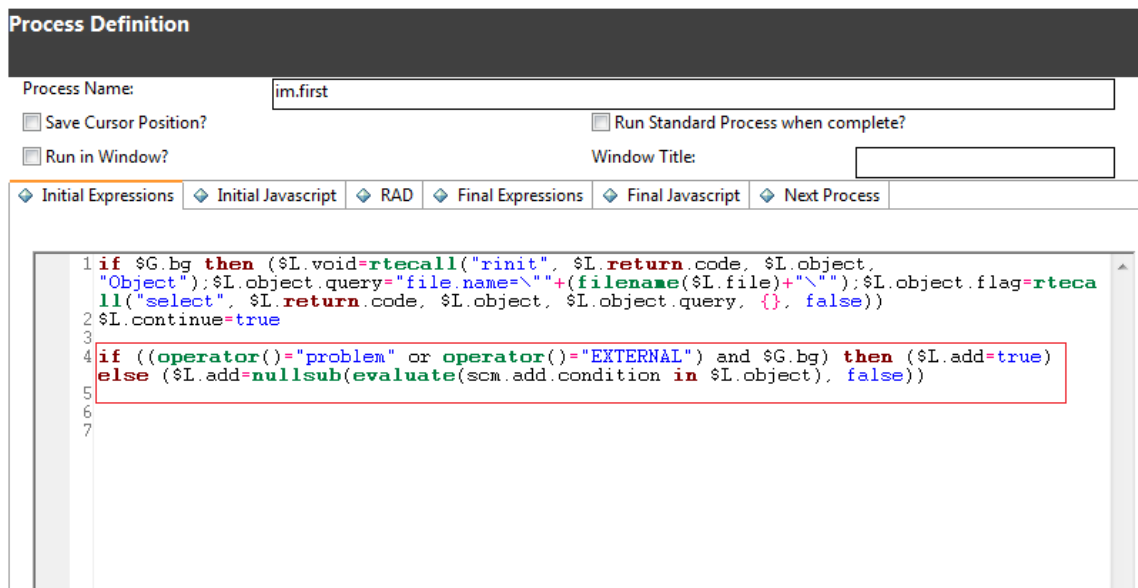
Initialization Process:   

Format:

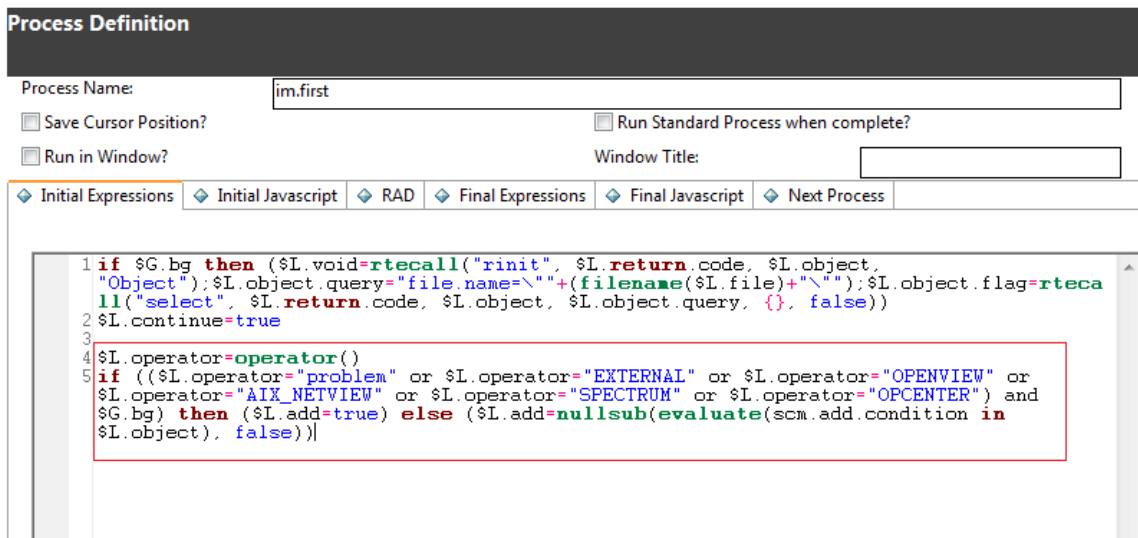
Input Condition (view state only):

Non-base methods			
Display Action	Process Name	Condition	Save First
save	im.first.save	true	
resolve	im.first	true	
new	im.first.new	true	
closeme	im.first	true	
cause	im.lookup.cause	true	
getsla	im.get.sla	true	
find.solution	im.find.solution	true	
getans.search	getans.search.solution	true	
getans.retrieve	getans.retrieve.solution	true	
getans.open	getans.open	true	
getans.create	getans.create.solution	true	
retrieve.iknow	iKnow.getsolution	true	
add	im.first	true	
kmsearch	kmmappedsearch	true	
view.biz.services	view.biz.services	true	
fill	context.pre.fill	false	
runcontext	run.context.wizard	true	
saveandexit	im.first.exit	true	
menu	im.open.cancel	true	

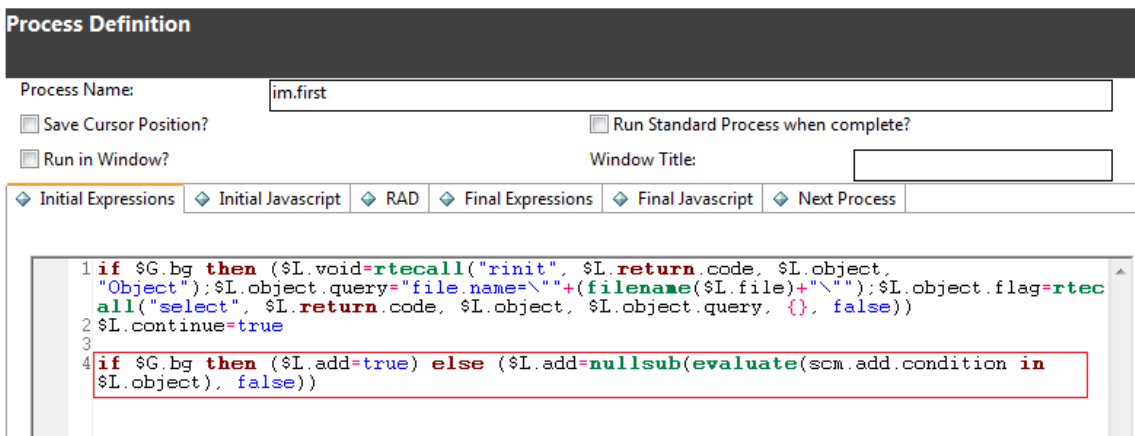
In out-of-box systems earlier than Service Manager 9.41, only the "problem" and "EXTERNAL" operators have permission to add Incident records from the backend.



Make sure that the User IDs of your eventin services are configured as demonstrated in the following image.



Alternatively, make sure the User IDs of your eventin services are not blocked. To do this, you can change your code to allow all backend operations, as illustrated by the following image.



If you use the following applications in your event register, the relevant eventin services are now allowed to trigger the Process Designer rule sets.

RAD application	File	Out-of-box event registers
axces.apm	probsummary	pmu: problem update pmr: problem reopen pmo: problem open pmm: problem mibile checkout/in pmc: problem close ... (total 19 eventregisters)
axces.apm.epmosmu	probsummary	epmosmu: e problem open smu
axces.sm	incidents	esmin: e service management smin: service management
axces.cm3	cm3r cm3t	cm3rin cm3tin ... (total 14 eventregisters)

**Note:**

- The Change module eventin services (such as eventregister cm3rin and cm3tin) can already trigger Process Designer rule sets.
- If you use the axes.database RAD application (the behavior of which is not changed by the Process Designer migration) as the eventregister application, the eventin services will not trigger the Process Designer rule sets.

## Perform data migration

For information about how to perform data migration, refer to the relevant section in the *Process Designer Migration Guide*.

### Problem and Task records

Please refer to the "Problem data migration" section in the *Process Designer Migration Guide*.

### Known Error records to the Problem table

Please refer to the "Known Error data migration" section in the *Process Designer Migration Guide*.

### Task Planner records

Please refer to the "Task Planner data migration" section in the *Process Designer Migration Guide*.

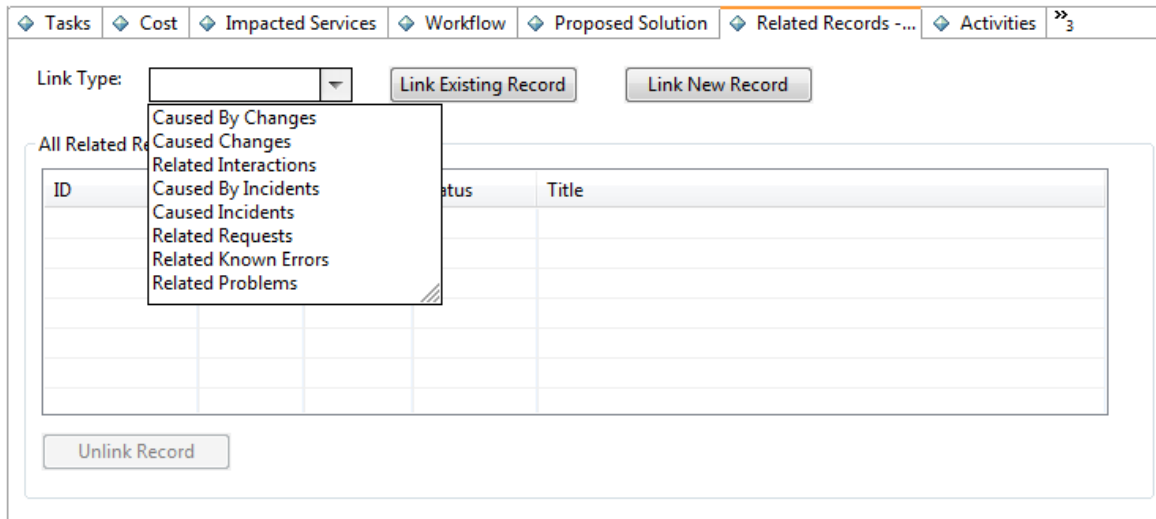
### Service Level Management records

Please refer to the "Service Level Management data migration" section in the *Process Designer Migration Guide*.

## Optional manual migration tasks

### Upgrade to the Process Designer related record subform

The functionality of related records in Process Designer-based systems and in legacy systems is different. For example, the following illustration shows the Process Designer-based Incident Management module's related record subform.



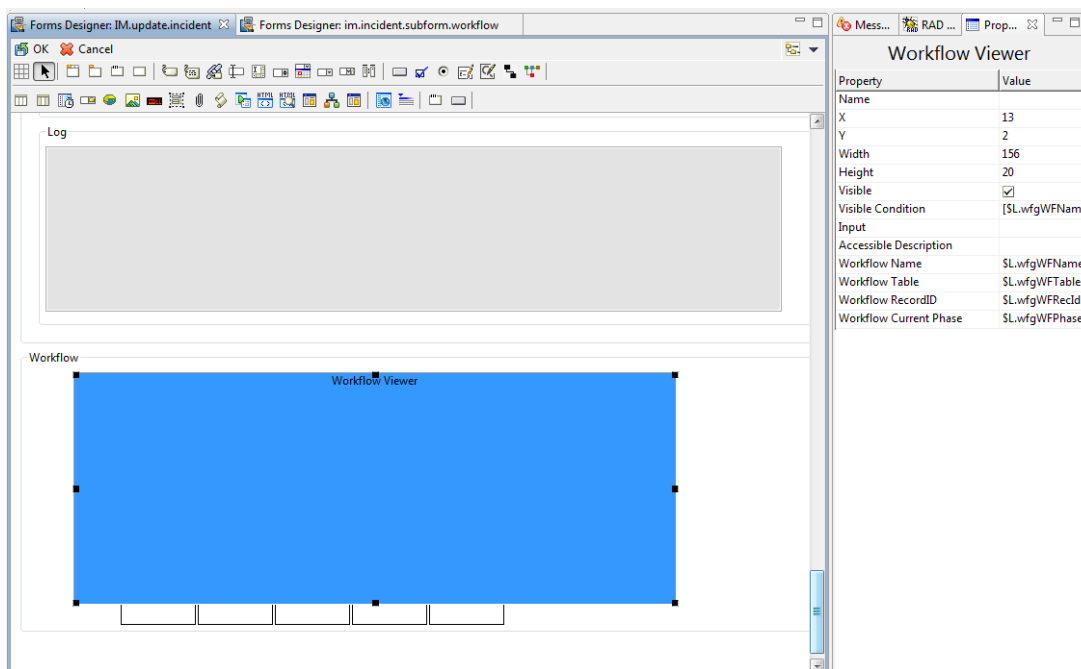
Note that the non-Process Designer related record functionality still works well in Process Designer-based environments. Therefore, it is not mandatory for you to replace this functionality. However, if you decide not to use the Process Designer related record functionality, rule types such as "Run Action" will not work.

If you want to use the Process Designer-based related record functionality, simply update the "Related Records tab" in the corresponding forms to implement the Process Designer related record subform (for example, `im.incident.subform.related` for the Incident module). Additionally, you must migrate the related record data. For more information about how to do this, refer to the "Related Record data migration" section in the "Process Designer Migration Guide" for Service Manager 9.41.

## Use the Process Designer workflow viewer in your forms

When you view a record in a Process Designer-based module, the form uses the new Workflow Viewer widget to display the workflow history of the record. Follow these steps to use this widget in your own forms:

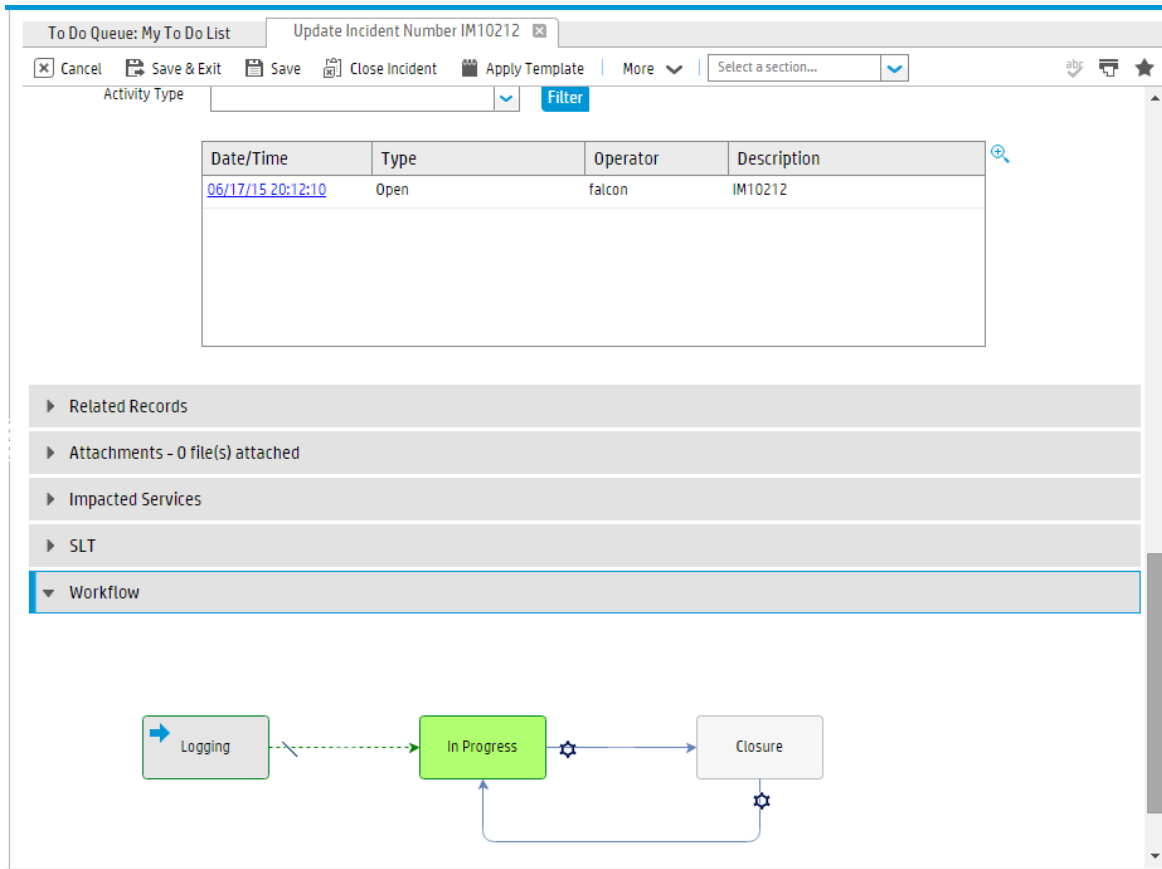
1. Open the out-of-box Process Designer workflow view sub-form with Forms Designer. For example, open the "im.incident.subform.workflow" subform. Then, use design mode to copy the content of this subform.
2. Open the form to which you want to add the new Workflow Viewer widget in Forms Designer. For example, open the "IM.update.incident" Incident Update form. Then, append the content of workflow view subform to the end of the form.



3. Repeat these steps for all your forms.

The following image displays the new Workflow Viewer in an Incident record.





## Configure the Search form

Search forms in Process Designer-based systems and in legacy systems are different. For example, the Process Designer-based Incident module uses the "im.advFind.incident.search" search form, instead of the "advFind.incident.search" search form. If you had previously customized some fields in a search form, you must manually add these fields to the Process Designer-based version of the search form.

We recommend that you use the Process Designer search form. However, if you want to continue to use the legacy search form, you must update the SearchConfig record to revert the changes applied by the Applications Upgrade Utility. The following illustration provides an example of how to configure the Incident search form back to the legacy search form.

### Search Configuration

Table Name:

Search Format:

Initialization Process:

Allow Advanced Find:

Defined Queries		Ranges	
Id	Query	Description	
open	flag=true	Open	
closed	flag=false	Closed	
assigned	assignee.name=operator()	Assigned to me	
highpriority	priority.code="1" or priority.code="2"	High Priority	
tl	total.loss=true	Total Loss of Service	
ucmdb	modEvent=true	Generated by UCMDB Integration	

The following table lists the changed search forms.

Table name	Non-Process Designer search form	Process Designer search form
probsummary	advFind.incident.search	im.advFind.incident.search
cm3r	advFind.search.change	advFind.search.chm
incidents	advFind.SD.search	sd.advFind.search
rootcause	advFind.search.problem	pbm.advFind.search.problem
rootcausetask	advFind.search.problem.task	pbm.advFind.search.problem.task

## Migrate the solution matching configuration from security profiles to security settings

The Incident solution matching functionality in Process Designer-based systems and in legacy systems is different.

We recommend that you use Process Designer solution matching. However, if you want to use non-Process Designer-based solution matching, you must manually migrate the security configuration from legacy security profiles to Process Designer security roles. This is necessary because non-Process Designer solution matching is configured in security profiles (for example, pmenv), which are not used in Process Designer security roles.

**Note:** During the upgrade process, the Applications Upgrade Utility asks you to choose between Process Designer solution matching and non-Process Designer solution matching (for more information, see ["Run the Upgrade Utility to upgrade the Service Manager applications to Service Manager 9.41 Hybrid "](#) on page 14).

The following illustrations use the "Check similar problems" Incident solution matching setting as an example of how to perform the migration.

### Incident Management Security Profile

Profile name :

The screenshot shows the 'Incident Matching Options' tab selected in a configuration window. The 'Check similar problems' checkbox is checked and highlighted with a red box. Below it are three unchecked checkboxes: 'Check similar incidents', 'Check incident duplicates on configuration items', and 'Check incident duplicates on related configuration items'. At the bottom, there are two spinners labeled 'Max levels:' and 'Max hits:'.

The field name of the "Check similar problems" setting is "check.rc." To add a setting with the same name in the "Incident" security area, search for the "Incident" security area. Then, from the **More** menu, click **Administration > Add New Setting**, and then add a new setting as demonstrated in the following illustration.

## New Setting Information

Please enter information for the setting:

Setting Id

Display Label

Description

Setting Type  ▼

Mandatory

< Previous    Next >    Finish    Cancel

Set the security rights value for this new Setting.

Next, you must check all your Incident Management security profile records (in an out-of-box system, there are 49) to identify all the records in which "Check similar problems" is selected. For example, if "Check similar problems" is selected in the sysadmin profile, you need to set the area of the sysadmin security role to "Incident," and select the "Check similar problems" setting as demonstrated in the following illustration.

**Rights**

Role  Area

View  Expert

New  Admin

Update

Delete/Close

Modify Template

Allowed Statuses 


Allowed Categories 


---

**Settings**

If a setting does not have a value specified, the value defaults to the value defined in the associated Area record.

Allow Inefficient Query

Can Create Personal Views

Can Create System Views

Check similar problems

Lock On Display

Finally, add the following line to the bottom of the populateProbsummaryEnvironment function in the "SecuritySetupEnvironment" script library:

```
vars.$G_pm_environment["check.rc"] = _val(_getRights("Incident", " check.rc"), 4);
```

After you manually migrate the "Check similar problems" setting, perform the same tasks for the following settings:

- Check similar incidents
- Check incident duplicates on configuration item
- Check incident duplicates on related configuration item
  - Max levels
  - Max hits

## Update reports to use Process Designer-based category tables

If you plan to use Process Designer-based Incident, Service Desk, and Problem category tables in your reports, you must update your reports to use the new tables.

The following table describes the mapping between category tables in Process Designer-based and legacy systems.

Module	Legacy table	Process Designer-based table
Incident	Category	imCategory
	subcategory	imSubcategory
	producttype	imArea
Service Desk	Category	sdCategory
	subcategory	sdSubcategory
	producttype	sdArea
Problem	Category	pbmCategory
	Subcategory	pbmSubcategory
	producttype	pbmArea

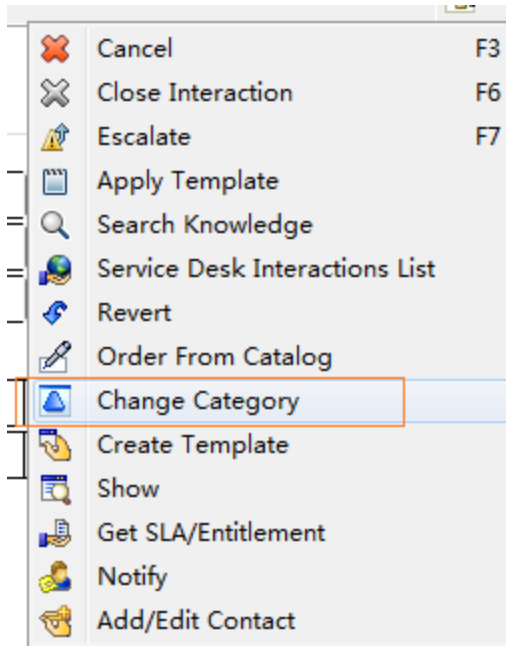
## Make the Category field in the Interaction and Incident forms read-only

In legacy systems, the category field in the Interaction and Incident forms are not read-only; users can change the category of an interaction.

The screenshot shows the 'Interaction Details' form. On the left, there are fields for Interaction ID (SD10176), Handle Time (00:00:03), Contact (CAFFREY, AARON), Service Recipient (CAFFREY, AARON), Location (South America), and Notify By (E-mail). On the right, there are fields for Category (incident), Area, Subarea, Impact (3 - Multiple Users), and Urgency (3 - Average). The 'Category' field is highlighted with a red rectangular box.

However, in Process Designer-based systems, this field is used to determine the specific workflow that the interaction or incident follows. If the category changes, the interaction may need to use a new workflow and forms, but this will not happen if the field is modified directly in the form. Therefore, we recommend that you make the Category field read-only.

To correctly change the category of an interaction or incident, use the new **Change Category** menu option that is automatically created by the Applications Upgrade Utility. This ensures that the correct workflow is used.



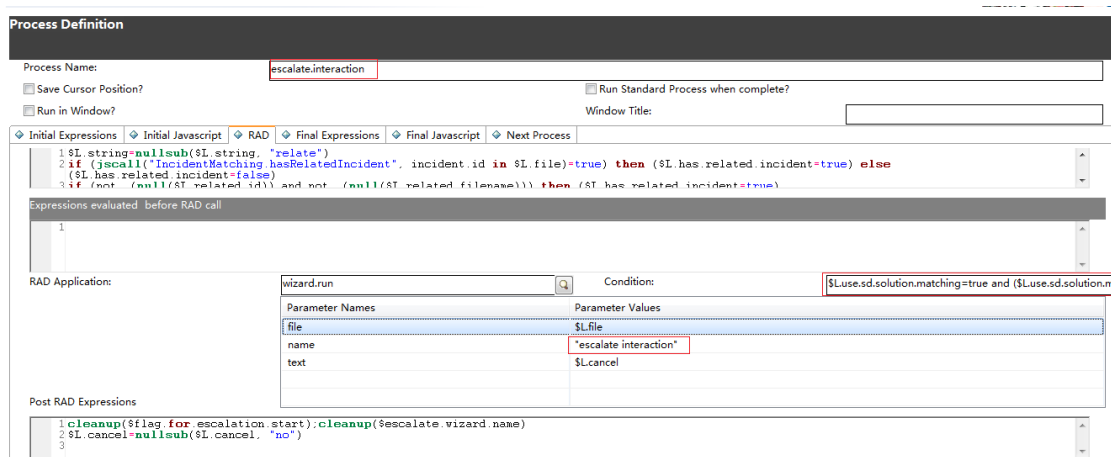
## Revert to using the "escalate to Change" wizard

The escalation to change process in Process Designer-based systems and in legacy systems is different. If you chose to use Process Designer solution matching during the migration process, the escalation wizard no longer appears when you click **Escalate** in an Interaction, and the Change is not created in background. Instead, the Change form is launched for you to input the details.

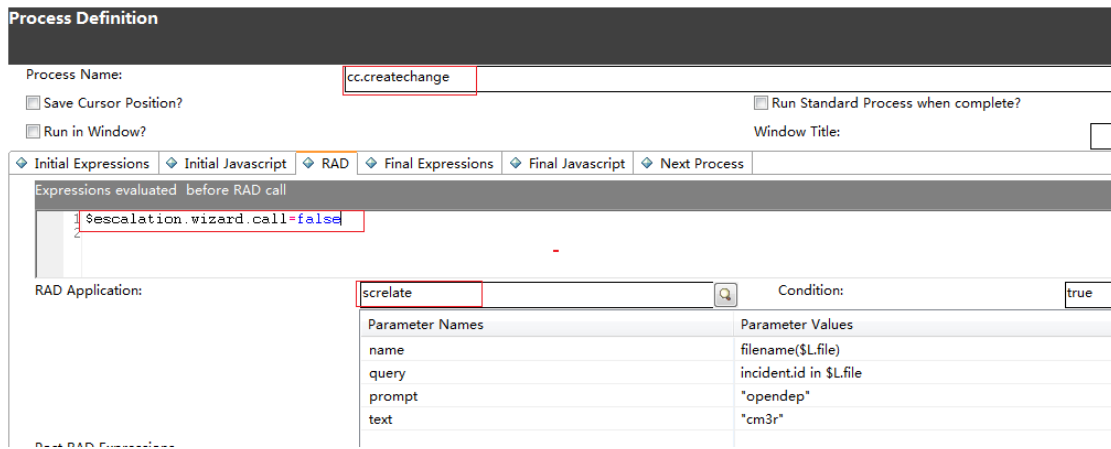
To revert to using the non-Process Designer-based escalation wizard, follow these steps:

1. Open the "escalate.interaction" process, and modify the condition so that it evaluates to "true" for escalation to change. This ensures that the escalation wizard is displayed.

For example, modify the condition from `$L.use.legacy.sd.solution.matching=true` and `($L.string="create" and $exit.code.fc="normal")` to `($L.use.legacy.sd.solution.matching=true or category in $L.file="request for change")` and `($L.string="create" and $exit.code.fc="normal")`.



2. Open the "cc.createchange" process, and modify the expression that is evaluated before the RAD call from `$escalation.wizard.call=false` to `$escalation.wizard.call=true`. This ensures that the change is created in the background.



3. If it does not already exist, add the following expression to the "number" link line of the "screlate.incidents.cm3r" link:

`requestedDate in $L.related=nullsub($requested.end.date, tod())`

**Note:** You can further tailor "escalate to Change" wizard by adding additional expressions or field mapping to copy data to change records.



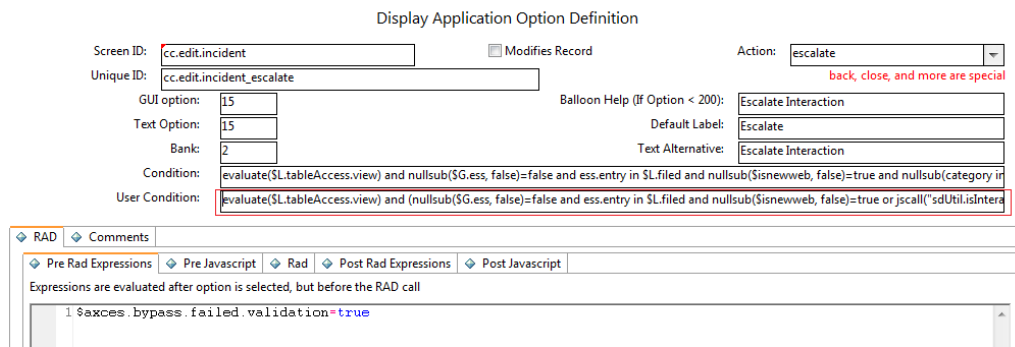
## Remove the button in the Incident form that cancels the Interaction escalation process

The Interaction escalation process in Process Designer-based systems and in legacy systems is different. In Process Designer-based systems, the interaction is added to database before the Process Designer solution matching form is displayed, and the Incident form is displayed when you select to create a new Incident (there is no longer an Escalate button on the Interaction form). When you save the Incident, the Incident is created and associated with the Interaction. If you do not want to escalate the Interaction to an Incident, you can click **Cancel** in the Incident form. In legacy systems, the Incident is created in the background without displaying the Incident form to the end user.

To revert to the old behavior, you must add the **Escalate** button back to the Interaction form and remove the **Cancel** button from the Incident form. To do this, follow these steps:

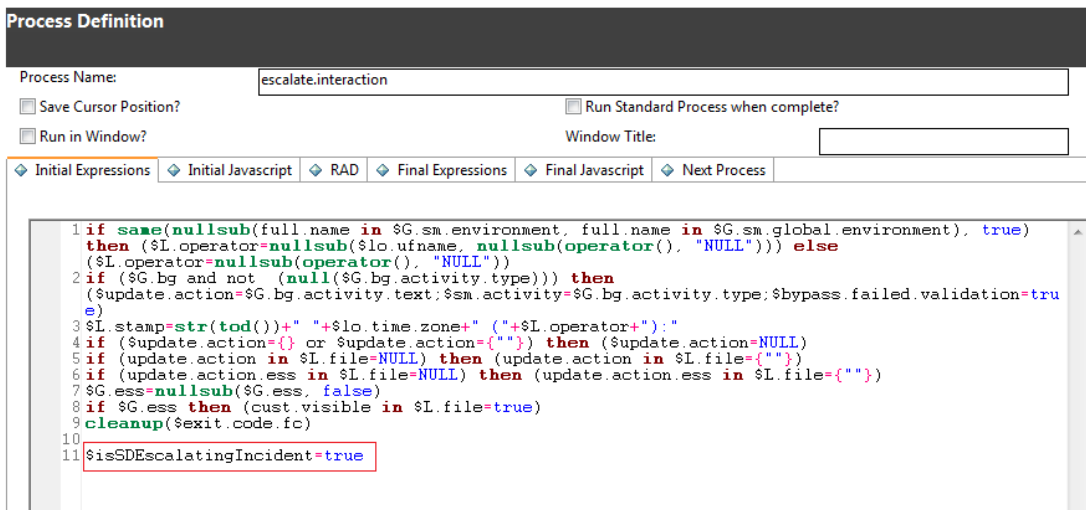
1. To add the **Escalate** button back to the Interaction form, update the user condition in the "cc.edit.incident\_escalate" display option to the following:

```
evaluate($L.tableAccess.view) and (nullsub($G.ess, false)=false and ess.entry in $L.file and nullsub($isnewweb, false)=true or jscall ("sdUtil.isInteractionEscalatedOrFulfilled", $L.file)=false) and nullsub (category in $L.file, "unknown")~="service catalog" and open in $L.file~="Closed"
```



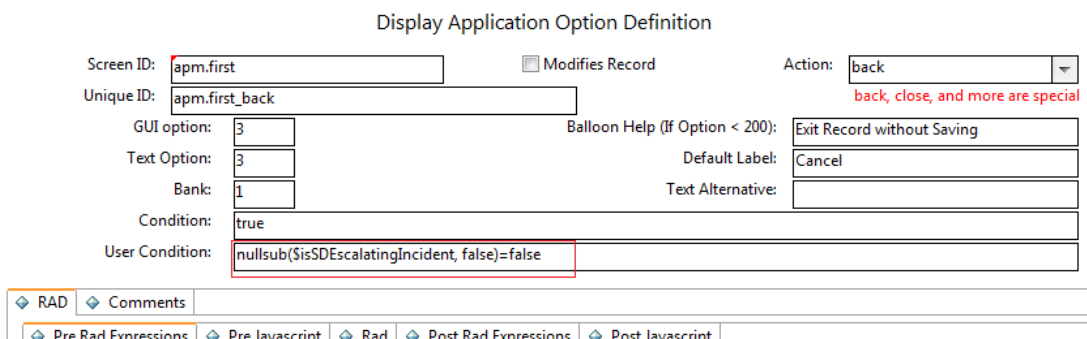
2. Remove the **Cancel** button from the Incident form. To do this, set a flag variable in the "escalate.interaction" process. For example, set the following flag:

```
$isSDEscalatingIncident=true
```



Then, check the following flag in the condition of the **Cancel** button ("apm.first\_back" display option):

`nullsub($isSDEscalatingIncident, false)=false`



## Migrate any customization of the knownerror table

The Process Designer-based Problem module stores Known Error records and Problem records in the same table (rootcause), whereas the legacy Problem module stores Known Error records in the knownerror table. Therefore, if you have made any customizations to the legacy knownerror table that you want to retain, you must manually reapply these customizations to the rootcause table.

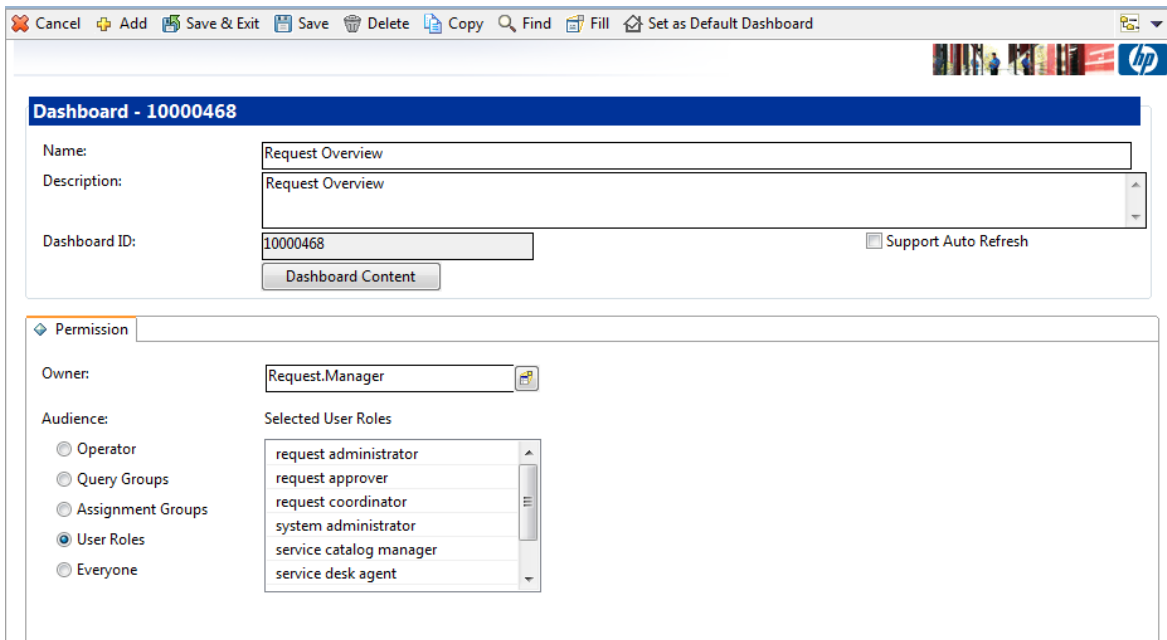
## Remove the legacy Request Management dashboard and report definitions

After you upgrade to Service Manager 9.41 Hybrid, the legacy Request Management dashboard and report definitions remain in the system. If you do not want to run the Legacy Request Management and

Process Designer-based Request Fulfillment modules in parallel, you can remove the legacy Request Management dashboard and report definitions.

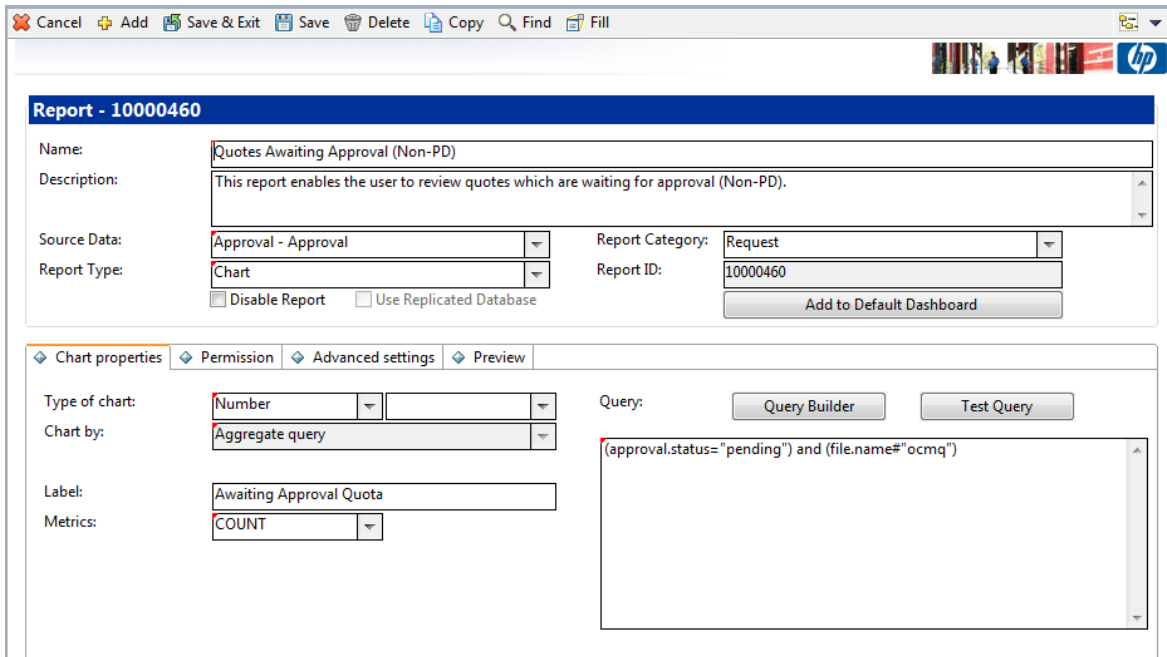
To remove the legacy Request Management dashboard definition record, follow these steps:

1. In the System Navigator, click **Reporting** -> **Search Dashboard**, and then search for ID "10000468."
2. Click **Delete**, and then click **Yes** to remove the Dashboard record.



To remove the legacy Request Management report definition records, follow these steps:

1. In the System Navigator, click **Reporting** -> **Search Report**, and then search for ID "10000460."
2. Click **Delete**, and then click **Yes** to remove the Report record.
3. Repeat steps 1 and 2 for the following IDs:
  - 10000461
  - 10000462
  - 10000463
  - 10000466
  - 10000467



## Enable legacy Request Management and Process Designer-based Request Fulfillment to run in parallel

Migration from legacy Request Management to Process Designer-based Request Fulfillment is a manual process that can be time-consuming if you have heavily tailored the Request Management module. In order to ease this transition Service Manager 9.41 Hybrid enables you to run the legacy Request Management and the Process Designer-based Request Fulfillment modules in parallel. To do this, follow these steps.

### Step 1: Set the value of the global flag to "true"

In the ProcessDesignerEnablement Script Library, locate the **isNonPDRrequestInParallel** function, uncomment the line `$G.nonpd.request.in.parallel` variable, and then set its value to "True."

```

ScriptLibrary
Name: ProcessDesignerEnablement Package: Patch Release HP Proprietary

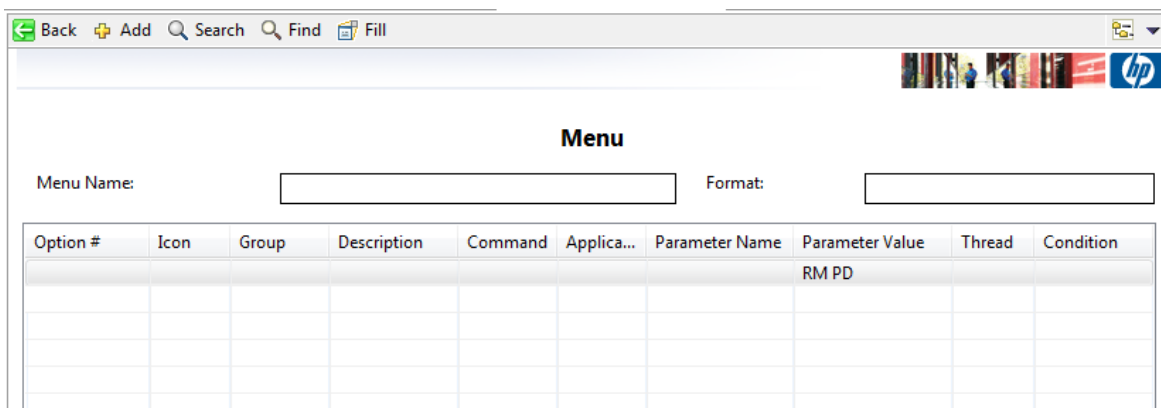
205 /** This method is used to check whether Process Designer has been enabled for Request Ma
206 *
207 * @returns whether request module has been enabled
208 *
209 */
210 function isRequestEnabled(){
211
212     if (vars.$G_pd_request_enabled != null) return vars.$G_pd_request_enabled;
213
214     var obj = $("Object").select('file.name="request\").uniqueResult();
215
216     if(!obj) {
217         vars.$G_pd_request_enabled = false;
218     }else{
219         vars.$G_pd_request_enabled = true;
220     }
221
222     return vars.$G_pd_request_enabled;
223
224 }
225
226 /**
227 * This method is used to identify whether non-PD Request and PD Request modules run
228 * in parallel after the PD Request is enabled.
229 *
230 * @returns whether non-PD Request and PD Request modules run in parallel.
231 */
232 function isNonPDRequestInParallel() {
233     if (vars["$G.nonpd.request.in.parallel"] != null) {
234         return vars["$G.nonpd.request.in.parallel"];
235     }
236
237     // please do NOT un-comment below line if you donot want the non-PD and PD Request
238     // modules run in parallel, by default always return null
239     vars["$G.nonpd.request.in.parallel"] = true;
240     return vars["$G.nonpd.request.in.parallel"];
241 }
242

```

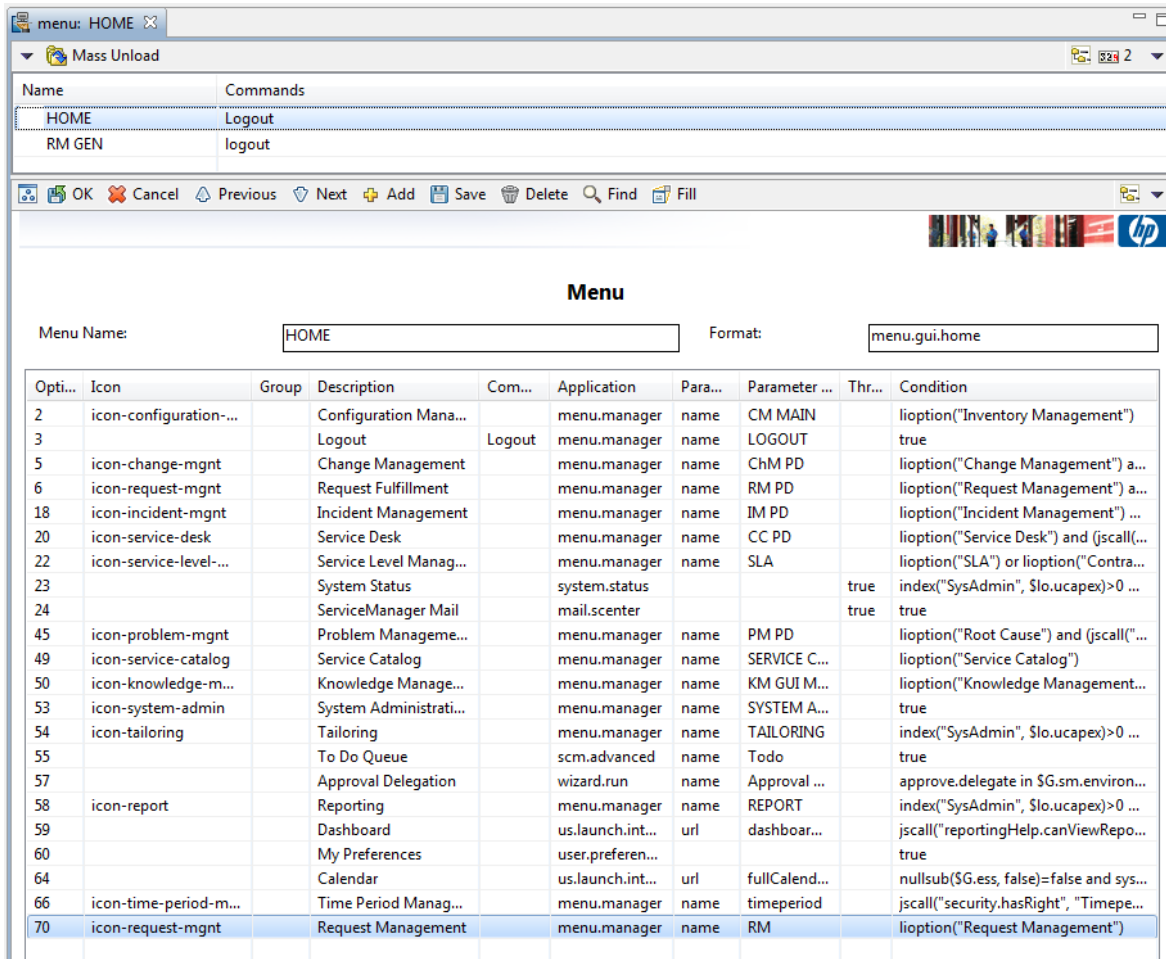
## Step 2: Restore the non-Process Designer Request menu items

### Request module menu items

To restore these menu items, search for menu records that have the parameter value "RM PD."

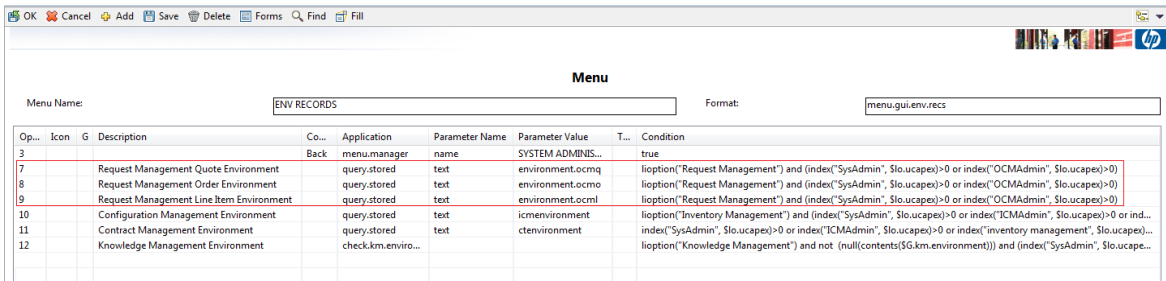


In out-of-box systems, there are three such records. Add the legacy "Request Management" menu item back to the menu. For example, the following image displays the required update to the "Home" menu.



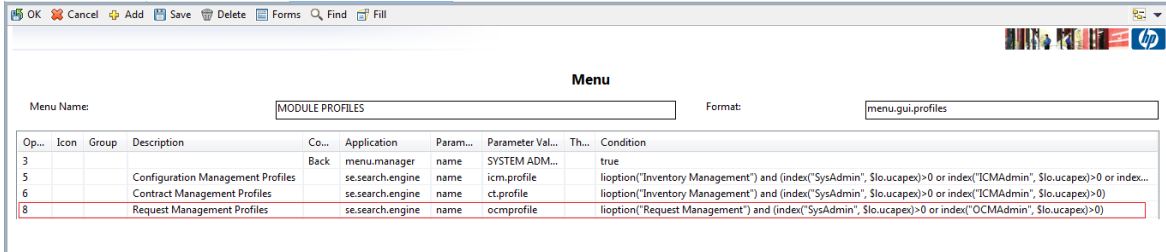
### Request environment menu items

These menu items are located at **System Administration > Ongoing Maintenance > Environment Records**. To restore these menu items, search for the "ENV RECORDS" menu, and then add the legacy request menu items, as demonstrated in the following image.



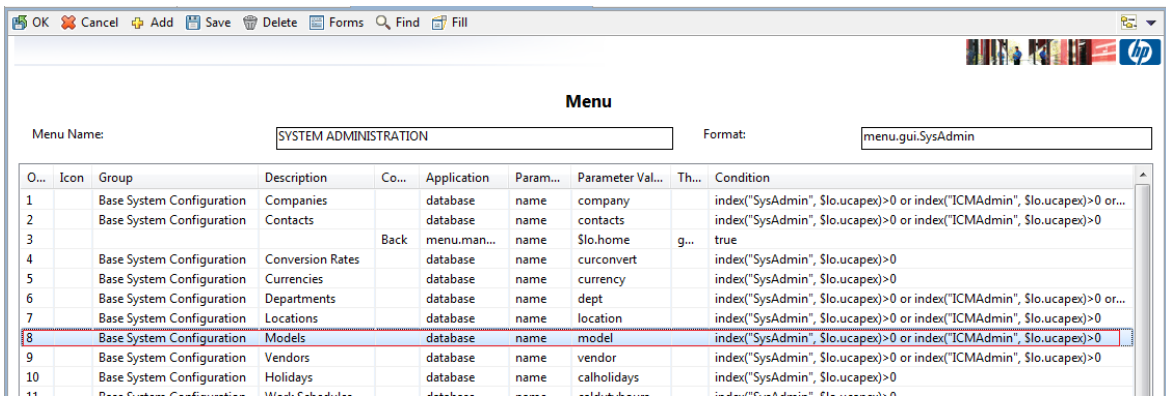
### Request profile menu items

These menu items are located at **System Administration > Ongoing Maintenance > Profiles**. To restore these menu items, search for the "MODULE PROFILES" menu, and then add the legacy request menu items, as demonstrated in the following image.



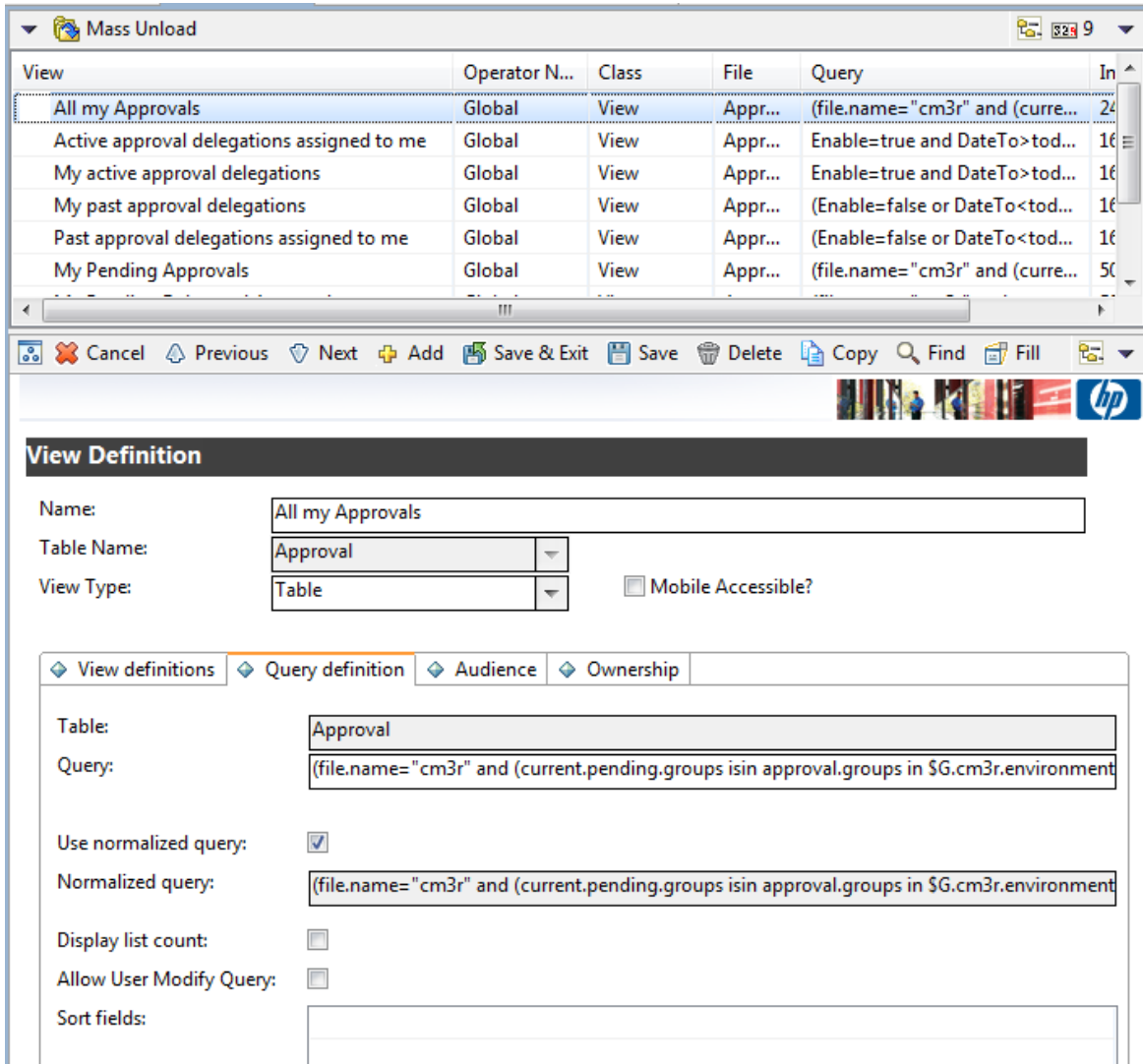
### Models menu item

This menu item is located at **System Administration > Base System Configuration**. To restore this menu item, search for the "SYSTEM ADMINISTRATION" menu, and then add the legacy request menu items, as demonstrated in the following image.



### Step 3: Update the queries in Inbox views

Search all Approval inboxes and check the query condition in each record to determine whether it uses only legacy global variables (\$G.ocmq.environment, \$G.ocmo.environment, and \$G.ocml.environment) or Process Designer-based global variables (\$G.request.environment and \$G.requestTask.environment). If so, modify the queries to use both sets of global variables.



For example, this query in the "All my Approvals" inbox contains a condition that resembles the following:

```
(file.name="request" and (current.pending.groups isin approval.groups in $G.request.environment or current.pending.groups isin {lo.user.name}))
```

In this situation, you need to change the query to resemble the following:

```
(file.name="request" and (current.pending.groups isin approval.groups in $G.request.environment or current.pending.groups isin {lo.user.name})) or (file.name="ocmq" and (current.pending.groups isin approval.groups in $G.ocmq.environment or current.pending.groups isin {lo.user.name}))
```



In the "My Group's To Do List" Inbox view, restore `itemType="ocmq"` or `itemType="ocml"` to the query, and make sure that the "request" and "requestTask" item types are included in the query.

**View Definition**

Name:

Table Name:

View Type:

---

View definitions | Query definition | Audience | Ownership

Table:

Query:

Use normalized query:

Normalized query:

Display list count:

Allow User Modify Query:

Sort fields:

In the "My Pending Delegated Approvals" Inbox view, append the following string to the query:  
 or (file.name="ocmq" and current.pending.groups isin \$G.delegated.ocmq.groups)

**View Definition**

Name:

Table Name:

View Type:   Mobile Accessible?

---

View definitions | Query definition | Audience | Ownership

Table:

Query:

Use normalized query:

Normalized query:

Display list count:

Allow User Modify Query:

Sort fields:

In the "My Pending Approvals" Inbox view, append the following string to the query:  
 or (file.name="ocmq" and (current.pending.groups isin approval.groups in \$G.ocmq.environment or current.pending.groups isin {\$lo.user.name}))

**View Definition**

Name: My Pending Approvals  
Table Name: Approval  
View Type: Table  Mobile Accessible?

View definitions | Query definition | Audience | Ownership

Table: Approval  
Query: ing.groups isin (\$lo.user.name)) or (file.name="ocmq" and (current.pending.groups isin approval.groups in \$G.ocmq.environment or current.pending.groups isin (\$lo.user.name)))  
Use normalized query:   
Normalized query: (file.name="cm3r" and (current.pending.groups isin approval.groups in \$G.cm3r.environment or current.pending.groups isin (\$lo.user.name)))  
Display list count:   
Allow User Modify Query:   
Sort fields:

Step 4: Restore the "OCM Create Order" schedule record

To restore the "OCM Create Order" schedule record, you must unload the record from your system before you upgrade to Service Manager 9.41 Hybrid, and then load it to your system after the upgrade process is complete.

You also can unload this record from a previous version of Service Manager.

To Do Queue: My To Do List | schedule: 1202497

OK Cancel Add Save Delete

**schedule: 1202497**

Name: OCM Create Order Class: ocm  
Schedule ID: 1202497 Expiration: 07/17/15 03:28:06  
Number: Scheduled Class: ocm  
Status: rescheduled Action Time:

Description | Javascript | Strings | Numbers | Booleans / Times | Stacked Queries

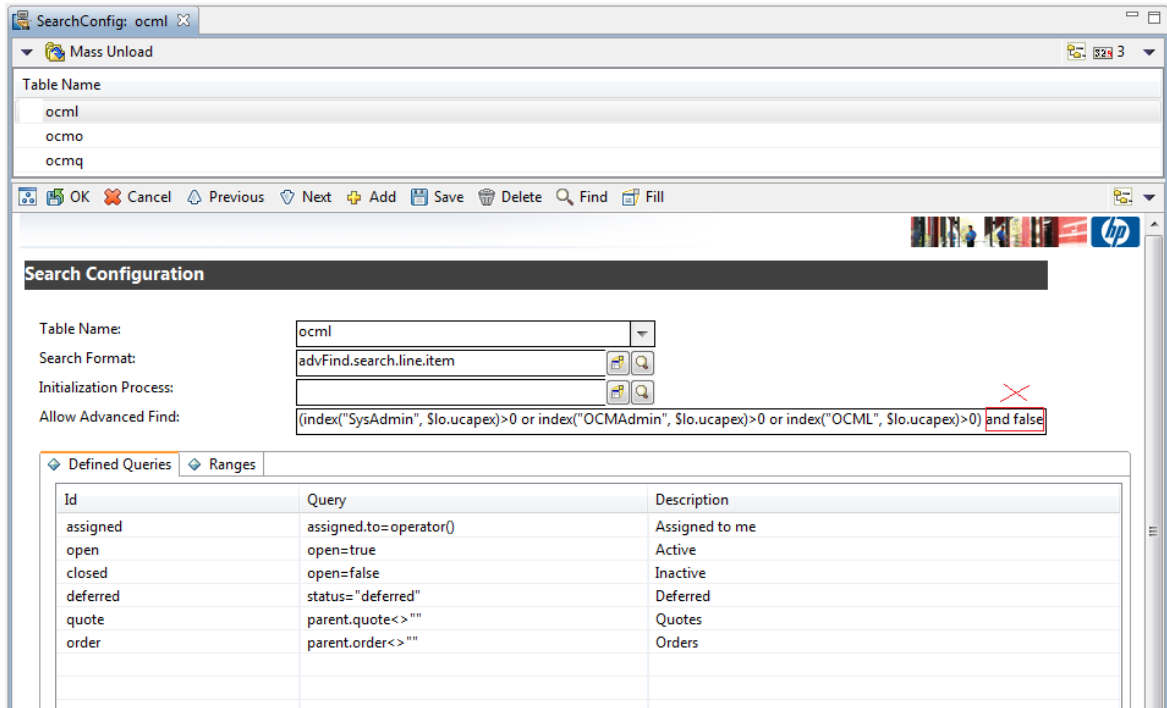
Description

Repeat Interval  
 Monthly  
 Quarterly  
 Semi-Annually  
 Annually  
 Other  
1 00:00:00

Problem Status:  
Application: rmo.create.order  
Query:

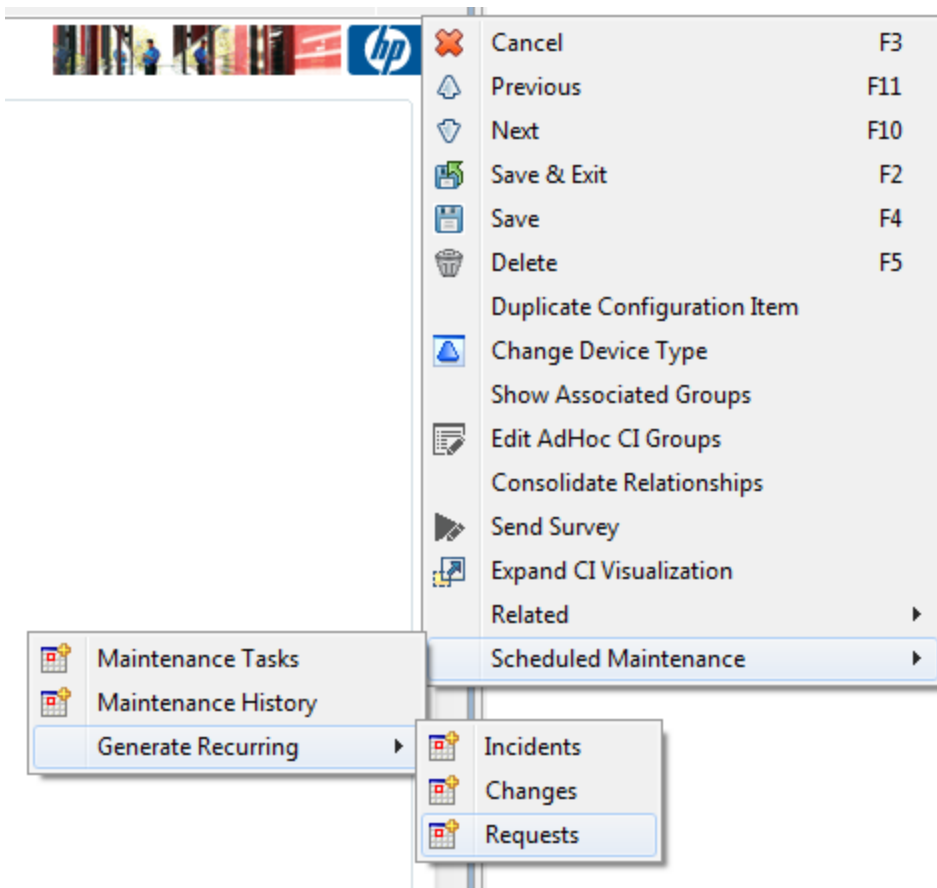
### Step 5: Configure search

Search for SearchConfig records that have a table name that begins with "ocm," and then remove the "false" condition in the Allow Advanced Find query.



### Step 6: Add display options

These display options are located at **More > Scheduled Maintenance > Generate Recurring > Request** in CI record display forms.



The sub-menu in the following illustration refers to Request table records.

Display Application Option Definition

Screen ID:	am.display.joinfile	<input type="checkbox"/> Modifies Record	Action:	do nothing
Unique ID:	am.display.joinfile_do nothing_21			<i>back, close, and more are special</i>
GUI option:	6005	Balloon Help (If Option < 200):		
Text Option:	6005	Default Label:	Scheduled Maintenance>Generate Recurring>Quotes	
Bank:	2	Text Alternative:	G Quotes	
Condition:	gui() and index(type in \$L.filed, \$G.icm.types)=0 and (index("SysAdmin", \$lo.ucapex)>0 or index("ICMAdmin", \$lo.ucapex)>0 or index("data administrator", \$lo.ucapex)>0)			
User Condition:				

RAD Application: patco.rt.tf.template		Separate Thread?: false
Names:	second.file query	Values: \$L.filed ocmq

Create a copy of the display option for this sub-menu, and then modify it to refer to the OCMQ table records, as demonstrated in the following image.

Display Application Option Definition

Screen ID:   Modifies Record Action:  back, close, and more are special

Unique ID:

GUI option:  Balloon Help (If Option < 200):

Text Option:  Default Label:  back, close, and more are special

Bank:  Text Alternative:

Condition:

User Condition:

---

◆ RAD ◆ Comments

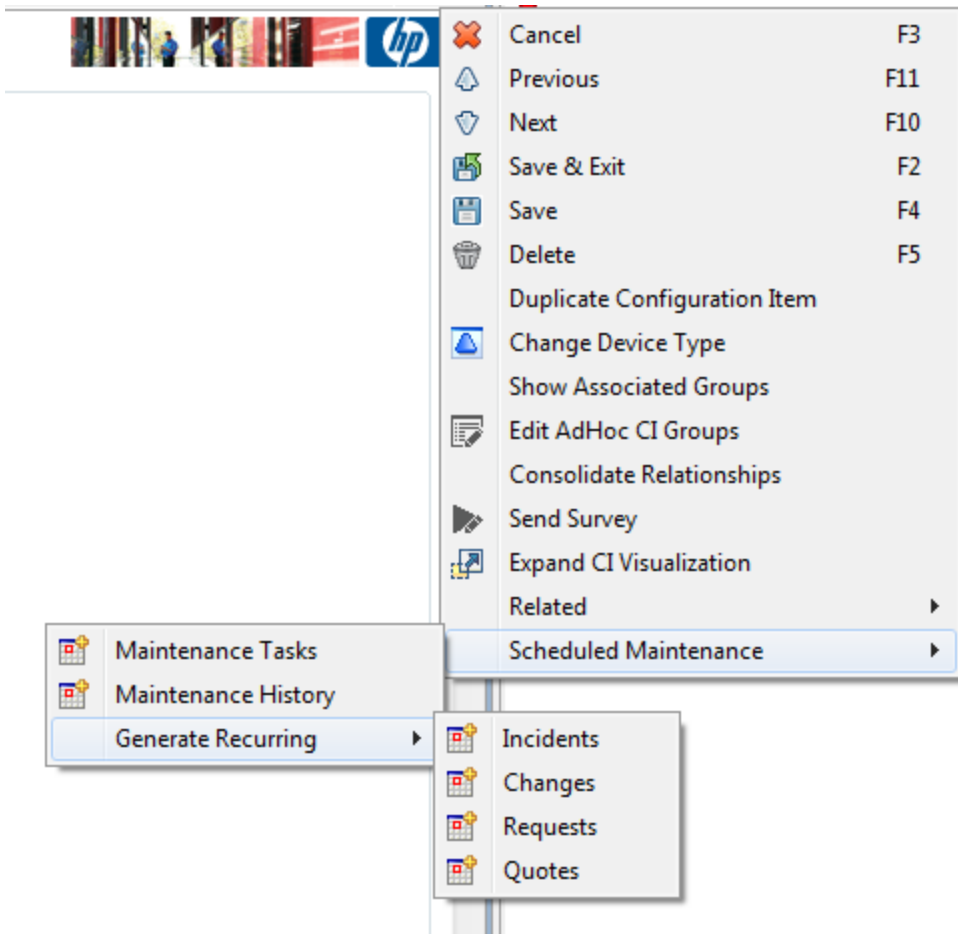
◆ Pre Rad Expressions ◆ Pre Javascript ◆ Rad ◆ Post Rad Expressions ◆ Post Javascript

RAD Application:  Separate Thread?

Names:

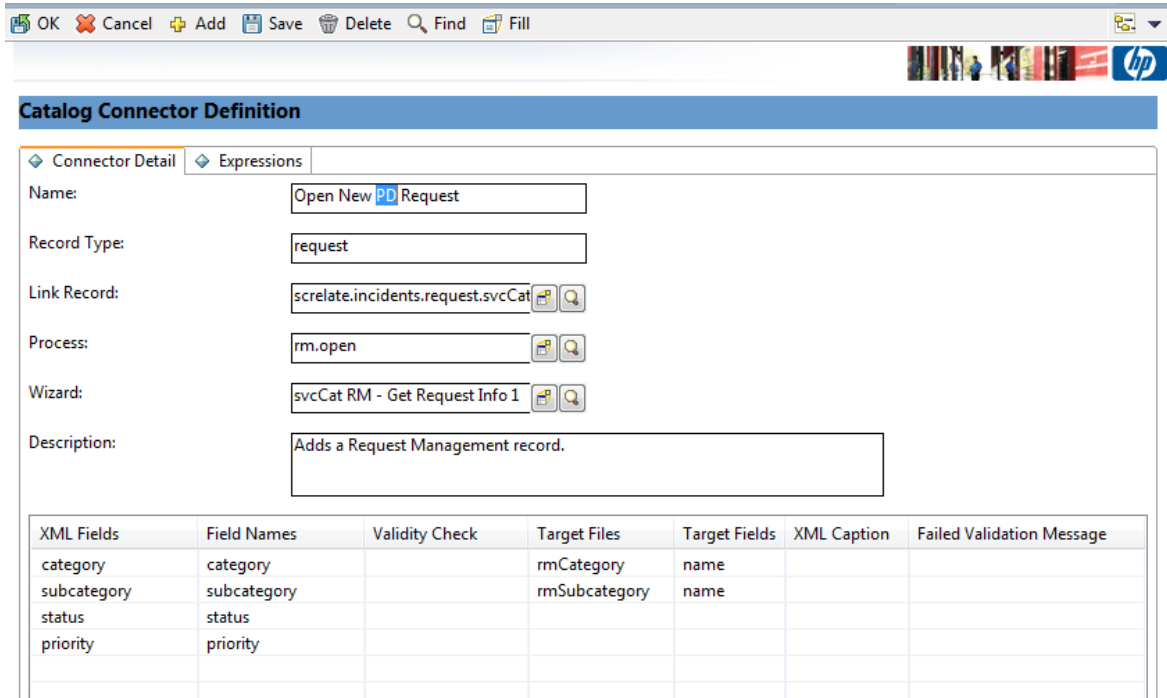
Values:

When you have done this, two sub-menus are displayed in the CI form. One sub-menu refers to the Request table records, and the other sub-form refers to the OCMQ table records.



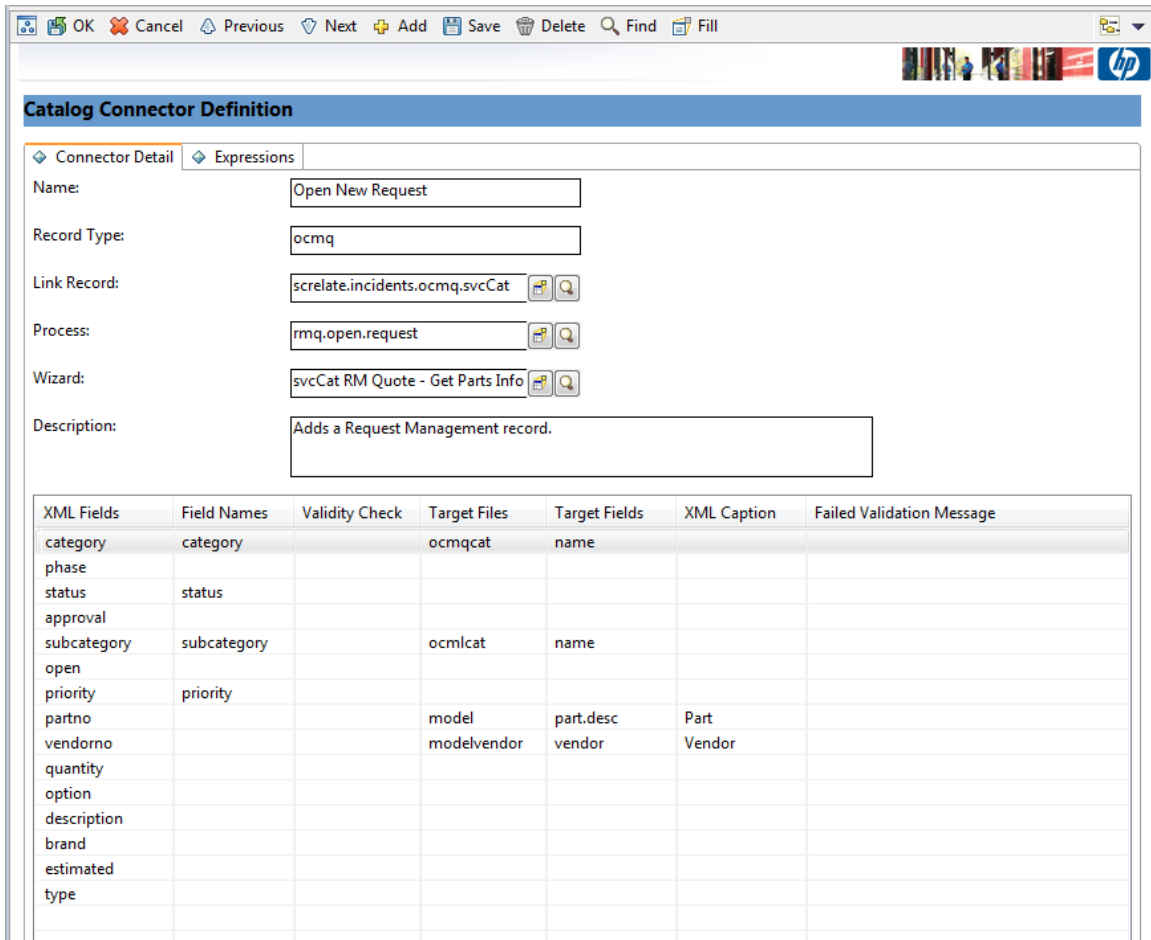
### Step 7: Rename the connector

Rename the Process Designer catalog connector. For example, name the connector "Open New PD Request."



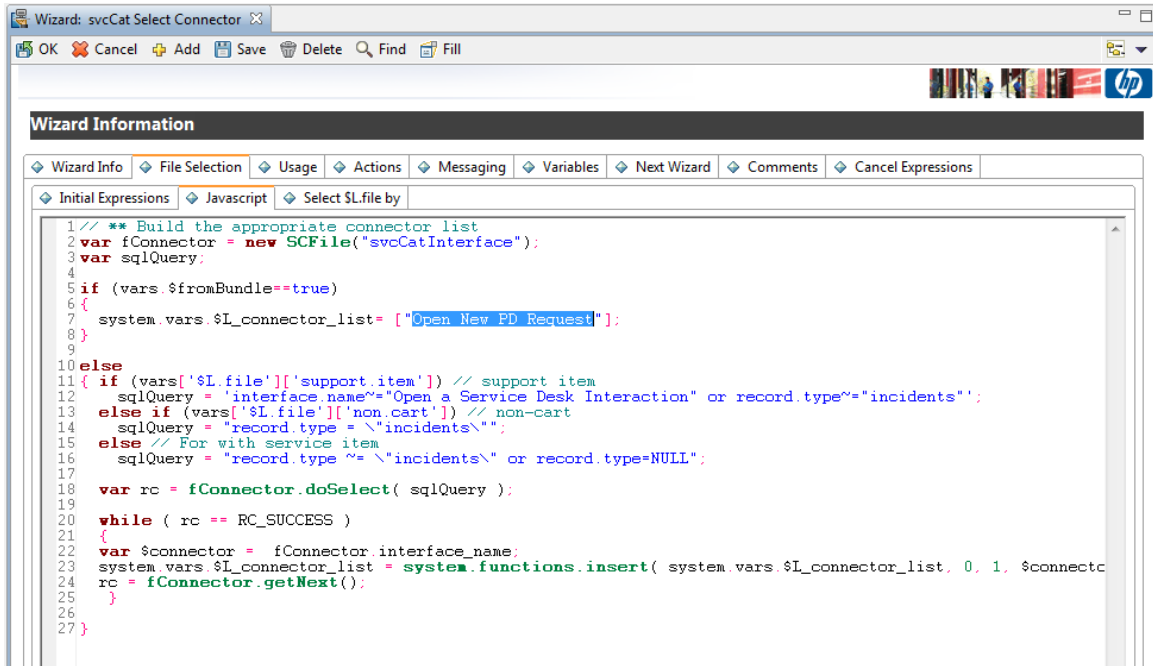
If you customized the "Open New Request" catalog connector, the out-of-box Process Designer catalog connector is named "NEW941Open New PD Request" after the migration process.

If you did not customize the "Open New Request" catalog connector, the out-of-box Process Designer catalog connector is named "Open New Request" after the migration process. In this case, you need to rename the catalog connector to "NEW941Open New PD Request," and then restore the non-Process Designer catalog connector.



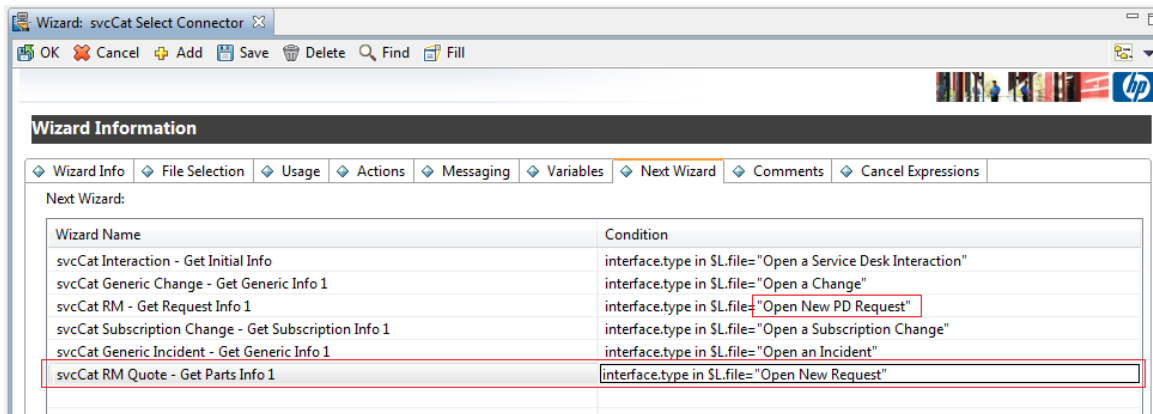
### Step 8: Modify the connector wizard

In the "svcCat Select Connector" wizard, click **File Selection** > **JavaScript**, and then change the connector name for the bundle catalog in the connector list from "Open New Request" to "Open New PD Request" (or to the name that you used in step 6).



Click the **Next Wizard** tab, and then change the connector name for Process Designer Request from "Open New Request" to "Open New PD Request" (or to the name that you used in step 6).

Then, restore the Next wizard for non-Process Designer-based Request (The last line in the following illustration).



**Step 9: Restore the legacy Request profile setting to the Operator form**

Update the JavaScript of the "operator.view" and "operator.search" display screens as follows.

**Before update**

```

if(lib.ProcessDesignerEnablement.isRequestEnabled())
{
    
```



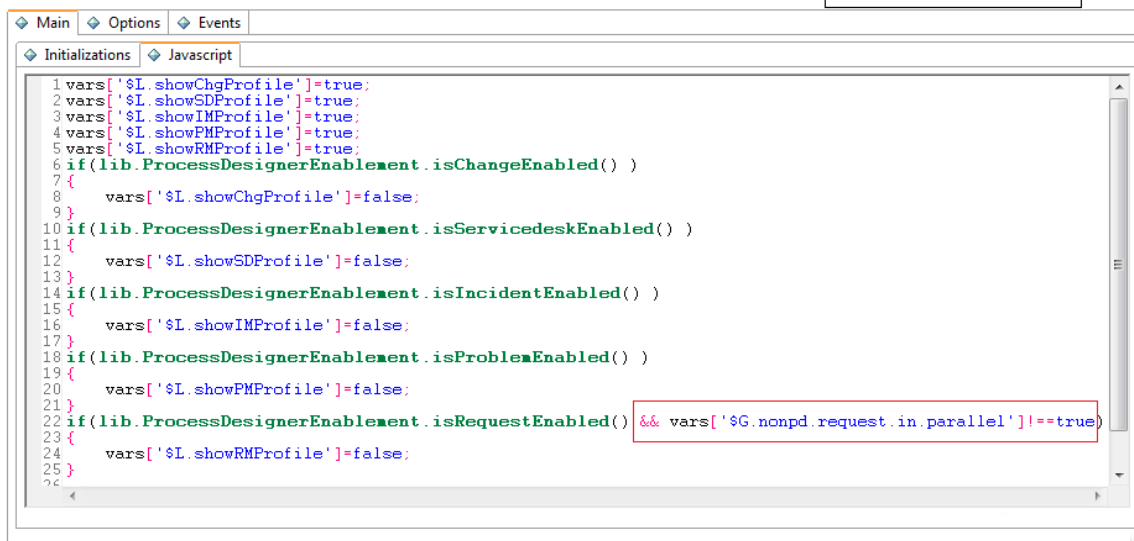
```
vars['$L.showRMPProfile']=false;
}
```

**After update**

```
if(lib.ProcessDesignerEnablement.isRequestEnabled() && vars
['$G.nonpd.request.in.parallel']!=true)
{
vars['$L.showRMPProfile']=false;
}
```

**Display Application Screen Definition**

Screen ID:	operator.view	On option 0:	return to appl.
Title:	SL.title		
Format:	SL.format		
I/O (if RIO):	true	Time (if FDISP):	
	<input type="checkbox"/> Used only for Search?	User Options:	true
		Language:	ENG



**Step 10: Assignment delegation**

In order for legacy Request Management to work in parallel with Process Designer-based Request Fulfilment, you must enable approval delegation to differentiate between these two modules.

To do this, you must modify the ApprovalDelegationGroups and ApprovalUtil script libraries to support both legacy and Process Designer-based Request delegation.

**Note:** In the following illustrations, the updated code is displayed on the left-hand side, and the

original code is displayed on the right-hand side.

### ApprovalDelegationGroups

For the legacy and Process Designer request modules to work in parallel, you must add the module (file name) information to the following functions, which are related to request delegation:

- addDelegateGroups

<pre>else if (module == "ocmq"    module == "request") {     var requestSql = getRequestDelegationGroups(module); }</pre>	<pre>else if(module=="ocmq"    module=="request") {     var requestSql = getRequestDelegationGroups(); }</pre>
---	--

- buildRequestSql

<pre>function buildRequestSql(allRequestGroups, partRequestGroups, module) {     for (var j = 0; j &lt; partRequestGroups.length; ++j) {         if (arrayIndexOf(allRequestGroups, partRequestGroups[j]) &lt; 0)             allRequestGroups.push(partRequestGroups[j]);     }     //build a sql for request     var fileName = module; }</pre>	<pre>function buildRequestSql(allRequestGroups, partRequestGroups) {     for(var j=0;j&lt;partRequestGroups.length;+j)     {         if(arrayIndexOf(allRequestGroups,partRequestGroups[j])&lt;0)             allRequestGroups.push(partRequestGroups[j]);     }     //build a sql for request     var fileName="\ocmq\";     if (lib.ProcessDesignerEnablement.isRequestEnabled()==true)         fileName="\request\"; }</pre>
---	---

- getAllRequestGroups

<pre>function getAllRequestGroups(aDele, module) {     var approver = aDele.approver;     var user = getApprover(approver);     var ocmGroups = new Array();     if (module == "request") {         ocmGroups = lib.MyGroupsSync.getApprovalOfInMyGroups(approver);     } }</pre>	<pre>function getAllRequestGroups(aDele) {     var approver = aDele.approver;     var user = getApprover(approver);     var ocmGroups = new Array();     if (lib.ProcessDesignerEnablement.isRequestEnabled()==true) {         ocmGroups = lib.MyGroupsSync.getApprovalOfInMyGroups(approver);     } }</pre>
---	--

- getRequestDelegationGroups

The value of the delegation module field in approvals delegation is "Request" for legacy request and "PD Request" for Process Designer-based request. You must configure the related JavaScript function accordingly.

<pre>function getRequestDelegationGroups(module) {     //find id there is a delegate all     var allDele = new Array();     var partDele = new Array();     for (var i = 0; i &lt;= delegationList.length; ++i) {         if (delegationList[i].module == "all")             allDele.push(delegationList[i]);     }      var allRequestGroups = new Array();     var partRequestGroups = new Array();      for (var i = 0; i &lt;= allDele.length; ++i) {         var requests = getAllRequestGroups(allDele[i], module);         for (var j = 0; j &lt;= requests.length; ++j) {             if (arrayIndexOf(allRequestGroups, requests[j]) &lt;= 0)                 allRequestGroups.push(requests[j]);         }     }      partRequestGroups = getDelegatedRequestGroups(module);      return buildRequestSql(allRequestGroups, partRequestGroups, module); }</pre>	<pre>function getRequestDelegationGroups() {     //find id there is a delegate all     var allDele = new Array();     var partDele = new Array();     for (var i=0;i&lt;=delegationList.length;+i)     {         if(delegationList[i].module == "all")             allDele.push(delegationList[i]);     }      var allRequestGroups = new Array();     var partRequestGroups = new Array();      for (var i=0;i&lt;=allDele.length;+i)     {         var requests = getAllRequestGroups(allDele[i]);         for (var j=0;j&lt;=requests.length;+j)         {             if (arrayIndexOf(allRequestGroups, requests[j])&lt;=0)                 allRequestGroups.push(requests[j]);         }     }      partRequestGroups = getDelegatedRequestGroups();      return buildRequestSql(allRequestGroups,partRequestGroups); }</pre>
--	---

- **getDelegatedRequestGroups**

This function retrieves the "Request" delegation group for "ocmq" and the "PD Request" delegation group "request."

<pre>function getDelegatedRequestGroups(module) {     var moduleValue = module == "ocmq" ? "Request" : "PD Request";     //find id there is a delegate all     var groupList = new Array();     for (var i = 0; i &lt;= delegationList.length; ++i) {         if (delegationList[i].module == moduleValue)             groupList.push(delegationList[i].group);     }      return groupList; }</pre>	<pre>function getDelegatedRequestGroups() {     //find id there is a delegate all     var groupList = new Array();     for (var i=0;i&lt;=delegationList.length;+i)     {         if(delegationList[i].module == "Request")             groupList.push(delegationList[i].group);     }      return groupList; }</pre>
--	---

- **getDelegationSql**

This function retrieves the delegation SQL for all modules. Therefore, you must configure it to retrieve the delegation SQL for both the ocmq and request tables, and to combine the results.

<pre>function getDelegationSql() {     initDelegation();     if (delegationList.length == 0)         return "";      var changeSql = getChangeDelegationGroups();     var requestSql = getRequestDelegationGroups("ocmq");     var PDRequestSql = getRequestDelegationGroups("request");     var svcSql = getSVCDelegationGroups();     var tpSql = getTPDelegationGroups();      var returnUrl = changeSql + requestSql + svcSql + tpSql + PDRequestSql;      return returnUrl; }</pre>	<pre>function getDelegationSql() {     initDelegation();     if (delegationList.length == 0)         return "";      var changeSql = getChangeDelegationGroups();     var requestSql = getRequestDelegationGroups();     var svcSql = getSVCDelegationGroups();     var tpSql = getTPDelegationGroups();      var returnUrl = changeSql + requestSql + svcSql + tpSql;      return returnUrl; }</pre>
--	---

### ApprovalUtil

Functions in this library use "Request" as the delegation module name for the legacy request module (when Process Designer-based request is not enabled), and "PD Request" as the delegation module name for Process Designer-based request (when Process Designer-based request is enabled).

For the legacy and Process Designer-based request modules to work in parallel, you must configure the following functions to use "Request" as the delegation module name for the legacy request module and "PD Request" as the delegation module name for Process Designer-based request.

- checkDelegates

<pre> } else if (module.indexOf("req") != -1) {     if (lib.ProcessDesignerEnablement.isRequestEnabled() == true) {         var myGroups = new SCFile("myGroups");         var mySql = "name=\"\" + system.functions.operator() + \"\"";         if (myGroups.doSelect(mySql) == RC_SUCCESS) {             profileArray = myGroups.approver_of;         }         for (i in profileArray) {             if (isDuplicate(pendingGroups, profileArray[i]))                 return "";         }         moduleName = "PD Request";     } </pre>	<pre> else if(module.indexOf("req")!=1) {     if(lib.ProcessDesignerEnablement.isRequestEnabled()==true)     {         var myGroups = new SCFile("myGroups");         var mySql = "name=\"\"+system.functions.operator()+\"\"";         if ( myGroups.doSelect( mySql ) == RC_SUCCESS )         {             profileArray = myGroups.approver_of;         }         for (i in profileArray) {             if (isDuplicate(pendingGroups, profileArray[i]))                 return "";         }         moduleName = "Request";     } </pre>
---	---

The function uses the profile ("ocmprofile") to check user rights in legacy systems, and uses the role-based access model ("secRole" and "secRights") in Process Designer-based systems.

- getQuery

<pre> } else if (moduleName == "Request") {     profileName = "ocmprofile";     profileSql = "view=true and approvals=true"; } else if (moduleName == "PD Request") {     profileName = "secRights";     profileSql = "area=\"Request\" and view=true"; } </pre>	<pre> } else if(moduleName=="Request") {     if(lib.ProcessDesignerEnablement.isRequestEnabled())     {         profileName="secRights";         profileSql="area=\"Request\" and view=true";     } else {         profileName="ocmprofile";         profileSql="view=true and approvals=true";     } } </pre>
<pre> var profileArray = new Array(); while (rc == RC_SUCCESS) {     if ((lib.ProcessDesignerEnablement.isChangeEnabled() &amp; amp; &amp; amp; moduleName == "Change")            moduleName == "Timeperiod")            (lib.ProcessDesignerEnablement.isRequestEnabled() &amp; amp; &amp; amp; moduleName == "PD Request")) {         var index = system.functions.index("approvals", profile.settingId);     } } </pre>	<pre> var profileArray = new Array(); while(rc == RC_SUCCESS) {     if ( (lib.ProcessDesignerEnablement.isChangeEnabled() &amp; amp; &amp; amp; moduleName == "Change")           moduleName == "Timeperiod"            (lib.ProcessDesignerEnablement.isRequestEnabled() &amp; amp; &amp; amp; moduleName == "Request"))     {         var index = system.functions.index( "approvals", profile.settingId );     } } </pre>

- addFolderQuery

<pre> } else if (module == "Request") {  var profileArray = currentUser.profile_request;  var cond = "{"; for (var i = 0; i &lt; profileArray.length(); ++i) {   cond += "\" + profileArray[i] + "\",";   if (i &lt; profileArray.length - 1)     cond += ","; } cond = cond.substring(0, cond.length - 1); cond += "}";  sql = "tablename=\"ocm\" and name isin " + cond; tableName = "ocm"; } else if (module == "PD Request") {  var profileArray = currentUser.secRole;  var cond = "{"; for (var i = 0; i &lt; profileArray.length(); ++i) {   cond += "\" + profileArray[i] + "\",";   if (i &lt; profileArray.length - 1)     cond += ","; } cond = cond.substring(0, cond.length - 1); cond += "}";  sql = "tablename=\"request\" and name isin " + cond; tableName = "request"; </pre>	<pre> else if (module=="Request") {   if (lib.ProcessDesignerEnablement.isRequestEnabled()==true) {     var profileArray = currentUser.secRole;   }   else {     var profileArray = currentUser.profile_request;   }    var cond = "{";   for (var i = 0; i &lt; profileArray.length(); ++i) {     cond+="\""+profileArray[i]+"\",";     if (i&lt;profileArray.length-1)       cond+=",";   }   cond = cond.substring(0,cond.length-1);   cond += "}";    if (lib.ProcessDesignerEnablement.isRequestEnabled()==true) {     sql = "tablename=\"request\" and name isin "+cond;     tableName = "request";   }   else {     sql = "tablename=\"ocm\" and name isin "+cond;     tableName = "ocm";   } }  else if (module=="SVC") { </pre>
---	--

• checkProfileForDelegate

<pre> } else {   if (module == "PD request") {     if (lib.ProcessDesignerEnablement.isRequestEnabled() == true) {       if (lib.security.getToken("Request", "approve.delegate") == "true") {         return true;       } else {         return false;       }     }   }   if (module == "Request") {     var profileNames = system.functions.str(system.functions.denum(currentUser.profile_request));     var profileSql = "name isin " + profileNames;     var profiles = new SCFile("comprofile");     var rc = profiles.doSelect(profileSql);      while (rc == RC_SUCCESS) {       if (profiles.approve_delegate == true)         return true;       rc = profiles.getNext();     }   } } </pre>	<pre> } else {   if (module == "request") {     if (lib.ProcessDesignerEnablement.isRequestEnabled()==true) {       if (lib.security.getToken("Request","approve.delegate")=="true"){         return true;       }else {         return false;       }     }   }   else {     var profileNames = system.functions.str(system.functions.denum(currentUser.profile_request));     var profileSql = "name isin "+profileNames;     var profiles = new SCFile("comprofile");     var rc = profiles.doSelect(profileSql);      while(rc == RC_SUCCESS)     {       if (profiles.approve_delegate == true)         return true;       rc = profiles.getNext();     }   } } </pre>
--	---

• canDelegateAll

<pre>//check to see whether a approver can do delegate all approvals //approver can do delegate all only if can delegate flag is set //to true for all his approval module  function canDelegateAll() {   if (system.vars.\$G_sm_environment.approve_delegate != true)     return false;    if (checkProfileForDelegate("change") == false)     return false;   if (checkProfileForDelegate("request") == false)     return false;   if (checkProfileForDelegate("PD request") == false)     return false;   if (checkProfileForDelegate("Timeperiod") == false)     return false;   return true; }</pre>	<pre>//check to see whether a approver can do delegate all approvals //approver can do delegate all only if can delegate flag is set //to true for all his approval module  function canDelegateAll() {   if (system.vars.\$G_sm_environment.approve_delegate != true)     return false;    if (checkProfileForDelegate("change")==false)     return false;   if (checkProfileForDelegate("request")==false)     return false;    if (checkProfileForDelegate("Timeperiod")==false)     return false;   return true; }</pre>
---	--

- **getCanDelegateModulesValue**

<pre>function getCanDelegateModulesValue() {   var modules = new Array();   if (system.vars.\$G_sm_environment.approve_delegate == true)     modules.push("SVC");   if (checkProfileForDelegate("change") == true)     modules.push("Change");   if (checkProfileForDelegate("request") == true)     modules.push("Request");   if (checkProfileForDelegate("PD request") == true)     modules.push("PD Request");   if (checkProfileForDelegate("Timeperiod") == true)     modules.push("Timeperiod");   return modules; }</pre>	<pre>function getCanDelegateModulesValue(){   var modules = new Array();   if (system.vars.\$G_sm_environment.approve_delegate == true)     modules.push("SVC");   if (checkProfileForDelegate("change")==true)     modules.push("Change");   if (checkProfileForDelegate("request")==true)     modules.push("Request");    if (checkProfileForDelegate("Timeperiod")==true)     modules.push("Timeperiod");   return modules; }</pre>
---	--

- **getCanDelegateModules**

<pre>function getCanDelegateModules() {   var modules = new Array();   if (system.vars.\$G_sm_environment.approve_delegate == true)     modules.push(system.functions.scmMsg("SVC", "approval"));   if (checkProfileForDelegate("change") == true)     modules.push(system.functions.scmMsg("Change", "approval"));   if (checkProfileForDelegate("request") == true)     modules.push(system.functions.scmMsg("Request", "approval"));   if (checkProfileForDelegate("PD request") == true)     modules.push(system.functions.scmMsg("PD Request", "approval"));   if (checkProfileForDelegate("Timeperiod") == true)     modules.push(system.functions.scmMsg("Timeperiod", "approval"));   return modules; }</pre>	<pre>function getCanDelegateModules(){   var modules = new Array();   if (system.vars.\$G_sm_environment.approve_delegate == true)     modules.push(system.functions.scmMsg("SVC", "approval"));   if (checkProfileForDelegate("change")==true)     modules.push(system.functions.scmMsg("Change", "approval"));   if (checkProfileForDelegate("request")==true)     modules.push(system.functions.scmMsg("Request", "approval"));    if (checkProfileForDelegate("Timeperiod")==true)     modules.push(system.functions.scmMsg("Timeperiod", "approval"));   return modules; }</pre>
---	---

## Step 11: Use the model table for legacy Request Management

After you run the Applications Upgrade Utility, Process Designer-based Request Fulfilment uses the new productCatalog table instead of the model table. Additionally, the tool updates all references to the model table to refer to the productCatalog table.

In order for the legacy Request Management to work in parallel with Process Designer-based Request Fulfilment, the two modules must use the two tables separately.

To achieve this, first restore the links that reference the model table.

### Links that must be updated

Link name	Field
"ocmlrec"	part.no
"ocml.view.summary"	part.no
"ocml.view.space"	model
"ocml.view.space"	part.no
"ocml.view.sap"	part.no
"ocml.view.detail"	parts.part.no
"ocml.view.detail"	part.no
"ocml.view.default.workorder"	parts.part.no
"ocml.view.default.workorder"	model
"ocml.view.default.workorder"	part.no
"ocml.view.default.telecom"	parts.part.no
"ocml.view.default.telecom"	part.no
"ocml.view.default.office.move"	parts.part.no
"ocml.view.default.office.move"	part.no
"ocml.view.default.internal"	parts.part.no
"ocml.view.default.internal"	part.no
"ocml.view.default"	parts.part.no
"ocml.view.default"	model
"ocml.view.default"	part.no
"ocml.search"	part.no
"ocml.copy.model"	number
"ocml"	part.no
"ocmco"	part.no
"ocmco"	parent.parts
"model"	component.part.no

Link name	Field
"model"	parent.parts
"model"	model
"advFind.search.line.item"	part.no

**Links to update (optional)**

Link name	Field
"Survey.CFG.addFilter"	part.no
"stock"	part.no
"software.counter"	installation.models
"software.counter"	license.models
"problem"	failing.component
"problem"	model
"pc.software.files"	part.no
"modelvendor.lkup"	part.no
"modelvendor"	part.no
"model.icm"	model
"IM.summary"	model
"IM.master.update"	model
"IM.master.open"	model
"IM.master.link"	model
"IM.master.close"	model
"furnishings"	model
"device2"	model
"device.template"	bios.model
"device.template"	part.no
"device.telecom"	part.no



<b>Link name</b>	<b>Field</b>
"device.storage"	part.no
"device.softwarelicense"	part.no
"device.officeelectronics"	print.model
"device.officeelectronics"	part.no
"device.networkcomponents"	part.no
"device.mainframe"	part.no
"device.handhelds"	part.no
"device.furnishings"	part.no
"device.example"	bios.model
"device.example"	part.no
"device.displaydevice"	part.no
"device.display"	part.no
"device.computer"	bios.model
"device.computer"	part.no
"device"	part.no
"DEFAULT ICM"	model
"dec.device.pc"	part.no
"contract.template"	part.no
"contract.template"	model
"contract"	part.no
"contract"	model
"configurationItemNode"	bios.model
"configurationItemNode"	part.no
"configurationItem"	bios.model
"configurationItem"	part.no

Link name	Field
"cm3t"	part.no
"cm3t"	product.no.
"cm3r"	part.no
"cm3r"	product.no.
"advFind.search.contract"	part.no
"advFind.search.contract"	model
"advFind.search.ci"	part.no

Next, configure synchronization between the "model" and "productCatalog" tables. Legacy Request Management and Process Designer-based Request Fulfillment use the two tables separately, but you must ensure that the data in the two tables is consistent.

## Step 12: Modify the KM knowledge base

After you run the Applications Upgrade Utility, the "Request\_Library" KM knowledge base is updated to use the request table. You must add another library that uses the OCMQ table, so that the KM knowledge base can support both legacy and Process Designer-based request.

For example, you might create the following knowledgebase:

- Knowledgebase Name: Quote\_Library
- Display Name: Quote
- Type: sclib
- Search Server Name: <your search server name>

**Knowledgebase Maintenance**

Knowledgebase Name:

Display Name:

Type:  ▼

Search Server Name:  ▼

◆ Status ◆ Type information ◆ Field Definitions ◆ Errors

offline

Refresh Interval:  ▼

0 = No Updates, All others multiply by 5 (eg. 1 interval = 5 minutes)

Then, in the **Field Definitions** tab, configure the following field definitions.

Field name	Alias	Type	Hitlist	Index weight	Match	Sort
number	id	String	true	Level 4	false	
status	problemstatus;kmstatus	String	true	No Index	true	
current.phase	phase	String	true	No Index	true	
brief.description	title	String	true	Level 4	false	
description	description	String	false	Level 3	false	
requested.for	requestedfor	String	true	Default	false	
requestor.name	requestorname	String	true	Default	false	
requested.date	requesteddate	Date	true	No Index	false	
company	company	String	true	No Index	false	
assigned.dept	assignment	String	true	Default	true	
assigned.to	assignee	String	true	Default	false	
priority	priority	String	true	No Index	true	

Field name	Alias	Type	Hitlist	Index weight	Match	Sort
justification	justification	String	false	Default	false	
sysmodtime	sysmodtime	Date	true	No Index	false	

### Step 13: Ensure the related record functionality works correctly

For more information about how to update your forms to use the new related record functionality, please refer to ["Upgrade to the Process Designer related record subform" on page 46.](#)

If you want your system to allow the creation of related records between the legacy Request Management and Process Designer-based Request Fulfillment modules, you must configure the `srelationtype` table appropriately. For example, if you want to allow the creation of related quotes from Incident records, you must add a new record in the `srelationtype` table, as illustrated in the following image.

The screenshot shows the configuration form for 'SM RELATIONSHIP TYPE'. The fields are as follows:

- Type of Relationship: Related Quotes
- Value of Relationship: Related Quotes
- Reversed Type: Related Incidents
- Source File Name: probsummary
- Dependent File Name: ocmq
- Included in Related Record Type List:
- Displayed in Related Records panel:

After you have done this, you can create related quotes from incident records, as illustrated in the following image.

**Incident - IM10211**

Title: IM10211  
 Description: IM10211

Incident ID: IM10211 Requested By: falcon

Status: Categorize Contact Person:   
 Phase: Categorization Location:   
 Primary Affected Service: CI001020 Major Incident:   
 Affected CI: Escalated:   
 CI is operational (no outage)

Outage Start Time: Parent Incident:   
 Outage End Time: Link to Parent Incident:

◆ Categorization an... ◆ Tasks ◆ Cost ◆ Impacted Services ◆ Workflow ◆ Proposed Solution ◆ Related Records -... >> 3

Link Type: Related Quotes

All Related Rec

ID	Status	Title

### Step 14: Restart the server

After you have configured Request Management and Request Fulfillment to run in paralel, you must restart the Service Managerserver.

## Create a production patch and apply it to the production system

For detailed information about how to do this, refer to the *HP Service Manager Applications Upgrade Guide*.

## Appendix A: Changes made to Service Manager by the Applications Upgrade Utility

Appendix A describes the changes that are automatically applied to your system when you use the Applications Upgrade Utility to upgrade to Service Manager 9.41 Hybrid.

### Summary of changes

The following table provides an overview of the functional implementation in a freshly-upgraded Service Manager 9.41 Hybrid system (that is, this table does not include the possibilities presented by additional manual tailoring).

Module		Functional implementation		
		Classic	Codeless	Hybrid
Service Desk	Workflows	Legacy	OOB Process Designer	OOB Process Designer workflow and migrated legacy workflow
	Objects			Process Designer (migrated from legacy object)
	Security			Process Designer (migrated from legacy profiles)
	Category records			OOB Process Designer records and migrated legacy records

Module		Functional implementation		
		Classic	Codeless	Hybrid
Incident Management	Workflows	Legacy	OOB Process Designer	OOB Process Designer workflow and migrated legacy workflow
	Objects			Process Designer (migrated from legacy object)
	Security			Process Designer (migrated from legacy profiles)
	Category records			OOB Process Designer records and migrated legacy records
	Alerts			OOB Process Designer alerts and migrated legacy alerts
	Solution matching			Legacy or Process Designer (you can choose to continue to use legacy solution matching during the upgrade process)
Problem Management	Workflows	Legacy	OOB Process Designer	OOB Process Designer
	Objects			OOB Process Designer
	Security			Process Designer (migrated from legacy profiles)
	Category records			OOB Process Designer records and migrated legacy records
	Known Errors			Process Designer (migrated from legacy Known Errors)

Module		Functional implementation		
		Classic	Codeless	Hybrid
Change Management	Workflows	Legacy	OOB Process Designer	OOB Process Designer
	Objects			OOB Process Designer
	Security			OOB Process Designer
	Category records			OOB Process Designer
Request Management	Workflows	Legacy	N/A	N/A (unless manually configured)
	Objects			
	Security			
	Category records			
Request Fulfillment	Workflows	N/A	OOB Process Designer	OOB Process Designer
	Objects			
	Security			
	Category records			



Module		Functional implementation		
		Classic	Codeless	Hybrid
Service Level Management	Workflows	OOB Process Designer	OOB Process Designer	OOB Process Designer
	Objects			
	Security			
	Category records			
Knowledge Management	Workflows	OOB Process Designer	OOB Process Designer	OOB Process Designer
	Objects			
	Security			
	Category records			

**Service Desk**

The Applications Upgrade Utility makes the following changes to the Service Desk module:

- **Workflows:** In addition to creating an out-of-box Process Designer Service Desk workflow, the Applications Upgrade Utility creates a migrated workflow (called "Migrated ServiceDesk") that is based on your legacy process configuration.
- **Security:** The Service Desk module profile is migrated to Process Designer security roles and security rights.
- **Objects:** The legacy Service Desk objects (incident files) are migrated to Process Designer-based objects.
- **Category records:** Records in the legacy category, subcategory, and producttype tables are migrated to the Process Designer-based sdCategory, sdSubcategory, and sdArea tables, respectively.

## Incident Management

The Applications Upgrade Utility makes the following changes to the Incident Management module:

- **Workflows:** In addition to creating an out-of-box Process Designer Incident workflow, the Applications Upgrade Utility creates a migrated workflow (called "Migrated Incident") that is based on your legacy process configuration.
- **Security:** The Incident Management module profile is migrated to Process Designer security roles and security rights.
- **Objects:** The legacy Incident Management objects (probsummary files) are migrated to Process Designer-based objects.
- **Category records:** Records in the legacy category, subcategory, and producttype tables are migrated to the Process Designer-based imCategory, imSubcategory, and imArea tables, respectively.
- **Alerts:** Legacy category-based alerts are migrated to the workflow phase-based alerts.
- **Solution matching:** During the upgrade process, you have the option to migrate to Process Designer-based solution matching (which is configured in Process Designer security roles). You can also choose to continue using legacy solution matching (which is configured in legacy security profiles).

## Problem Management

The Applications Upgrade Utility makes the following changes to the Problem Management module:

- **Workflows:** Legacy Problem Management processes are not migrated to Process Designer workflows. Instead, an out-of-box Process Designer Problem workflow is created.
- **Security:** The Problem Management module profile is migrated to Process Designer security roles and security rights.
- **Objects:** Legacy Problem Management objects are not migrated to Process Designer objects. Instead, out-of-box Process Designer Problem objects are created.

- **Category records:** Records in the legacy category, subcategory, and producttype tables are migrated to the Process Designer-based pbmCategory, pbmSubcategory, and pbmArea tables, respectively.

### **Change Management**

The Change Management module was already reimplemented on Process Designer when Process Designer Content Pack 9.30.2 was applied (this content pack is a prerequisite for upgrading to Service Manager 9.41 Hybrid). Therefore, in Service Manager 9.41 Hybrid Process Designer is fully implemented in the Change Management module.

### **Request Management/Request Fulfillment**

In Service Manager 9.40, the Request Management Module was reimplemented on Process Designer as the Request Fulfillment module. When you upgrade to Service Manager 9.41 Hybrid, you are upgraded to Request Fulfillment (that is, Process Designer is fully implemented).

However, to enable this transition, Service Manager 9.41 Hybrid enables you to run Request Management and Request Fulfillment side-by-side.

## **Workflow migration**

The Applications Upgrade Utility automatically generates Process Designer-based workflows for the Service Desk and Incident Management modules, based on information in the legacy categories. The workflows are composed of three basic phases, and are named "Migrated Incident" and "Migrated ServiceDesk" respectively.

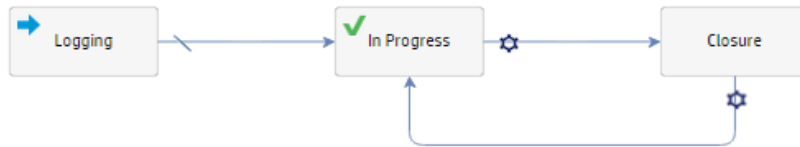


When you view a migrated workflow in the workflow viewer, additional fields are displayed in the Edit Workflow Properties tab. These fields identify the workflow as migrated.

The Applications Upgrade Utility pre-populates these fields using legacy information, but you are able to manually configure them. However, only the Applications Upgrade Utility can configure a workflow as a legacy workflow; you cannot do this manually.

The following image displays the additional fields that are displayed in a "legacy" Process Designer workflow.

Save | Zoom in | Zoom out | Add phase | Delete | Copy | Paste | Edit Workflow Properties | Edit Object Properties



### Workflow Based Configuration

Edit Workflow Properties | Workflow Based Rule Sets | Workflow Based Actions | Workflow Backend Transitions

Workflow Name\*  
Migrated Incident

Table Name\*  
probsummary

HP Proprietary

Migrated Workflow

Legacy Format Control  
probsummary

Legacy Open State  
im.open

Legacy Close State

Legacy Default State  
im.view

Legacy Browse State  
im.browse

### Description

Font [dropdown] | **B** | *I* | U | abc | [color] | [font size] | [bulleted list] | [numbered list] | [link] | [image]

[Large empty text area for description]

**Note:** These additional fields are not displayed when you view non-migrated workflows.

The following table lists the legacy fields from which the values that are used to populate the new fields are obtained.

Field in workflow viewer	Legacy field from which the value is obtained	Default value in the migrated Incident Management workflow	Default value in the migrated Service Desk workflow
Legacy format control	"Master format ctrl" field in the legacy object	probsummary	incidents
Legacy open state	"Open state" field in the legacy object	im.open	sm.open
Legacy close state	"Close state" field in the legacy object	Null	Null
Legacy default state	"Default state field" in the legacy object	im.view	sm.view
Legacy browse state	"Browse state" in the legacy object	im.browse	sm.browse

The value of the "Form Edit Condition" in each phase of the "legacy" Process Designer workflow is retrieved from the legacy display screen.

Phase in legacy workflow	Legacy field from which the value is obtained
Logging phase	Legacy object -> open state -> display screen -> "I/O (if RIO)" field
In Progress phase	Legacy object -> default state -> display screen -> "I/O (if RIO)" field
Closure phase	Legacy object -> default state -> display screen -> "I/O (if RIO)" field

The conditional display forms in each phase of the "legacy" Process Designer workflow are retrieved from the display formats of the legacy category records. For example, the following image displays the conditional display forms of the Incident Management logging phase.

Phase - Logging

Details Forms Rule Sets Actions Approvals Alerts

Default Display form

Additional/Display Forms

+ Add Edit Delete Up Down

<input type="checkbox"/>	Name	Description	Form Condition	Type
<input type="checkbox"/>	IM.open.incident	IM.open.incident	( Category in CurrentRecord = "complaint")	Display Form
<input type="checkbox"/>	IM.open.incident	IM.open.incident	( Category in CurrentRecord = "incident")	Display Form
<input type="checkbox"/>	IM.open.incident	IM.open.incident	( Category in CurrentRecord = "request for change")	Display Form
<input type="checkbox"/>	IM.open.incident	IM.open.incident	( Category in CurrentRecord = "request for informati...	Display Form

## Security role migration

Process Designer-based and legacy versions of Service Manager use different security mechanisms. Therefore, the Applications Upgrade Utility migrates the legacy security profiles to Process Designer-based security roles.

For more information about the items that are migrated or unchanged after you migrate to Service Manager 9.41 Hybrid, refer to Appendix A of the *Process Designer Migration Guide*.

After you migrate to Service Manager 9.41 Hybrid, all profile settings are moved to security rights. However, the legacy profile variables are still populated when users log in. This maintains compatibility and avoids changes to runtime application code that still uses the legacy variables.

The legacy profile variables of the following five modules are populated from security rights (not from the profile) when a user logs in:

- Change Management: \$G.cm3r.environment, \$G.cm3t.environment
- Incident Management: \$G.pm.environment
- Problem Management: \$G.rc.environment



- Service Desk: `$G.sm.environment`
- Knowledge Management: `$G.km.environment`

**Note:** The Request Fulfillment module uses the new `$G.request.environment` and `$G.requestTask.environment` global variables, whereas the legacy Request Management module uses `$G.ocmq.environment` and `$G.ocml.environment`.

If new boolean, number, character, or array fields have been added to the `dbdict` of a Profile table, the Applications Upgrade Utility synchronizes these changes with the new Process Designer security area.

For more information about the mapping between previous security profiles and current PD security roles and the mapping between security profiles and Process Designer security rights/settings, refer to the "Incident security roles and settings" topic in the Service Manager Help.

## Unmapped legacy profile settings

When you migrate the profile settings to Process Designer security rights, not all legacy profile settings are migrated to security rights. For more information about the unmapped legacy profile settings, see the Service Manager 9.41 Help.

## Object migration

The Applications Upgrade Utility migrates the legacy Incident Management objects (probsummary files) and the legacy Service Desk objects (incident files) to Process Designer-based objects. When the migration process is complete, a list of the records that the tool changes is displayed in the migration report.

The following tables describes how the migration of tailored fields is handled.

## Incident Management

Section	Field	Status after migration
Object Info	Number record name	Tailored value is retained
Object Info	Phase table name	Tailored value is retained
Object Info	Paging table name	Tailored value is retained
Object Info	Joindef	Tailored value is retained
Object Info	Status field	Tailored value is retained
Object Info	Assigned to	This is an array type field. Added items are merged into the new object.
Object Info	Workgroup	This is an array type field. Added items are merged into the new object.
Locking	Use locking	Tailored value is retained
Locking	Lock on display	Tailored value is retained
Locking	Lock parent record	Tailored value is retained
Locking	Parent lock information	This is an array type field. Added items are merged into the new object.
Locking	Watch variables	This is an array type field. Added items are merged into the new object.
Variables/Global Lists	Local variables	This is an array type field. Added items are merged into the new object.
Variables/Global Lists	Global lists	This is an array type field. Added items are merged into the new object.

## Service Desk

Section	Field	Status after migration
Object Info	Number record name	Tailored value is retained
Object Info	Phase table name	Tailored value is retained
Object Info	Paging table name	Tailored value is retained
Object Info	Joindef	Tailored value is retained
Object Info	Status field	Tailored value is retained
Object Info	Assigned to fields	This is an array type field. Added items are merged into the new object.
Object Info	Workgroup fields	This is an array type field. Added items are merged into the new object.
Locking	Use locking	Tailored value is retained
Locking	Lock on display	Tailored value is retained
Locking	Lock parent record	Tailored value is retained
Locking	Parent Lock Information	This is an array type field. Added items are merged into the new object.
Locking	Watch Variables	This is an array type field. Added items are merged into the new object.
Variables/Global Lists	Local Variables	This is an array type field. Added items are merged into the new object.
Variables/Global Lists	Global Lists	This is an array type field. Added items are merged into the new object.

## Category record migration

Process Designer-based and legacy versions of Service Manager use different category related tables. Therefore, the Applications Upgrade Utility migrates the category records in the Problem Management, Service Desk, and Incident Management modules from the legacy category tables to the Process Designer-based category tables.

**Note:** You can configure the specific category records that are migrated when you run the Applications Upgrade Utility. For more information about this, see ["Run the Upgrade Utility to upgrade the Service Manager applications to Service Manager 9.41 Hybrid "](#) on page 14.

The Applications Upgrade Utility can detect and migrate the following customizations to the dbdicts of legacy category tables:

- New fields

**Note:**

- The Applications Upgrade Utility cannot migrate new field aliases.
- The Applications Upgrade Utility cannot migrate the datadict and related scmessage for field captions.
- Although Applications Upgrade Utility migrates the data in new fields, remember that you must tailor the related Process Designer-based forms in order to display this data.

- Increases to the length of character-type fields

If there is a conflict between the legacy and Process Designer-based category names, the Applications Upgrade Utility renames the Process Designer-based category name to "*<original name>* codeless."

**Note:** You cannot customize the names of Process Designer-based categories. If you want to change the category name that is displayed to end users, you can use the localization function.

The Applications Upgrade Utility also migrates the following Mandanten settings from the legacy category, subcategory, and producttype tables to the relevant Process Designer-based tables:

- Mandanten field restrictions
- Mandanten restricting queries

A list of records that are modified by the Applications Upgrade Utility is included in the migration report.

## Problem Management

The following records in the legacy category table are migrated to the Process Designer-based pbmCategory table.

Field name in the category table	Field name in the pbmCategory table	Default value
name	name	
N/A	sharedflag	5 (Interaction/Problem)
active	active	
Name	description	
N/A	workflow	Problem
company	company	

The following records in the legacy subcategory table are migrated to the Process Designer-based pbmSubcategory table.

Field name in the subcategory table	Field name in the pbmSubcategory table	Default value
subcategory	subcategory.name	
category	category	
N/A	sharedflag	2 (Problem)
active	active	
subcategory	description	
company	company	

The following records in the legacy producttype table are migrated to the Process Designer-based pbmArea table.

Field name in the producttype table	Field name in the pbmArea table	Default value
product.type	area.name	
subcategory	subcategory	
category	category	
N/A	sharedflag	2 (Problem)
active	active	
product.type	description	
company	company	

## Service Desk

The following records in the legacy category table are migrated to the Process Designer-based sdCategory table.

Field name in the category table	Field name in the sdCategory table	Default value
name	name	
N/A	sharedflag	3 (Interaction/Incident/Problem)
active	active	
N/A	escalate.type (Fullfilment process)	null
Name	description	
N/A	workflow	Migrated ServiceDesk
approvals	approvals	
company	company	

The following records in the legacy subcategory table are migrated to the Process Designer-based sdSubcategory table.

Field name in the subcategory table	Field name in the sdSubcategory table	Default value
subcategory	subcategory.name	
category	category	
N/A	sharedflag	3 (Interaction/Incident/Problem)
active	active	
subcategory	description	
company	company	

The following records in the legacy producttype table are migrated to the Process Designer-based sdArea table.

Field name in the producttype table	Field name in the sdArea table	Default value
product.type	area.name	
subcategory	subcategory	
category	category	
N/A	sharedflag	3 (Interaction/Incident/Problem)
active	active	
product.type	description	
company	company	

## Incident Management

The following records in the legacy category table are migrated to the Process Designer-based imCategory table.

Field name in the category table	Field name in the imCategory table	Default value
name	name	
N/A	sharedflag	3 (Interaction/Incident/Problem)
active	Active	
avail.post	avail.post	
Name	description	
N/A	workflow	Migrated Incident
company	company	



The following records in the legacy subcategory table are migrated to the Process Designer-based imSubcategory table.

Field name in the subcategory table	Field name in the imSubcategory table	Default value
subcategory.name	subcategory	
category	category	
N/A	sharedflag	3 (Interaction/Incident/Problem)
active	Active	
name	description	
company	company	

The following records in the legacy producttype table are migrated to the Process Designer-based imArea table.

Field name in the producttype table	Field name in the imArea table	Default value
product.type	area.name	
subcategory	subcategory	
category	category	
N/A	sharedflag	3 (Interaction/Incident/Problem)
active	Active	
product.type	description	
company	company	

## Alerts migration

Alerts are configured differently in Process Designer-based and legacy versions of Service Manager. In the legacy Incident Management module, alerts are category-based; whereas in Process Designer-based systems alerts are configured in workflow phases. Therefore, the Applications Upgrade Utility migrates the category-based alerts to the workflow phases.

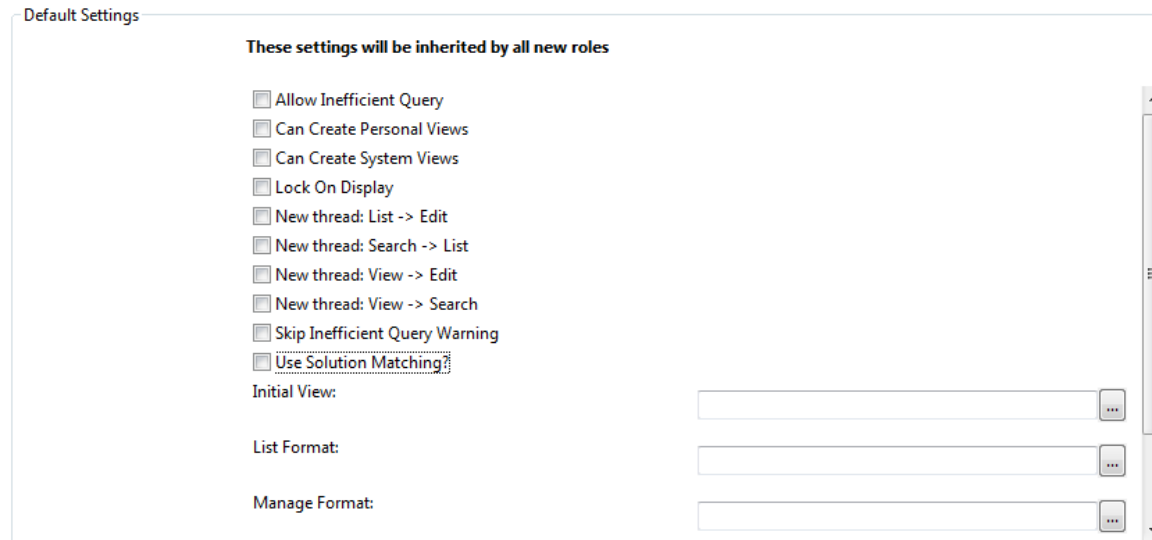
## Incident solution matching migration

Process Designer-based and legacy versions of Service Manager use different Incident solution matching mechanisms. Specifically, legacy Incident solution matching is configured in legacy security profiles, and Process Designer-based solution matching is configured in Process Designer security roles. Therefore, the Applications Upgrade Utility enables you to migrate legacy solution matching to Process Designer-based solution matching.

**Note:** You do not have to migrate to Process Designer-based solution matching in Service Manager 9.41 Hybrid. When you run the Applications Upgrade Utility, you are given the choice to migrate or to continue to use legacy solution matching. For more information, see ["Run the Upgrade Utility to upgrade the Service Manager applications to Service Manager 9.41 Hybrid "](#) on page 14.

If you want to continue to use legacy solution matching, you must manually migrate the solution matching configuration from the legacy security profiles to the Process Designer security roles. For more information, see ["Migrate the solution matching configuration from security profiles to security settings"](#) on page 50.

When you upgrade to Service Manager 9.41, a "Use Solution Matching?" security right is added to the "Incident" security area. This is required in order for the Applications Upgrade Utility to be able to migrate the solution matching configuration from security profiles to security roles. By default, the value of this security right is set to "true."



The Applications Upgrade Utility sets the value of the new "Use Solution Matching?" security right to "true" in the Process Designer security roles if any of the following options is selected in the legacy security profile:

- Check similar problems
- Check similar incidents
- Check incident duplicates on configuration items
- Check incident duplicates on related configuration items

### Incident Management Security Profile

Profile name :

◆ Security
◆ Forms
◆ Incident Matching Options

Check similar problems

Check similar incidents

Check incident duplicates on configuration items

Check incident duplicates on related configuration items

Max levels:

Max hits:

## Legacy category table updates

Process Designer-based and legacy versions of Service Manager use different category tables. The Applications Upgrade Utility can update the legacy category tables (the category, subcategory, and producttype tables) to synchronize them with the new Process Designer-based category tables. This enables you to run reports on legacy category tables and to continue to use interfaces that rely on data in these legacy tables.

**Note:** You can decide whether to update the legacy category tables when you run the Upgrade Utility. For more information about this, see ["Run the Upgrade Utility to upgrade the Service Manager applications to Service Manager 9.41 Hybrid "](#) on page 14.

The following tables describe the field mapping that is applied if you decide to synchronize the legacy category tables.

## Incident Management

The following fields in the Process Designer-based imCategory table are synchronized to the legacy category table.

Field name in imCategory table	Field name in category table	Default value
name	name	
active	active	
avail.post	avail.post	
company	company	

The following fields in the Process Designer-based imsubCategory table are synchronized to the legacy sub category table.

Field name in imSubcategory table	Field name in subcategory table	Default value
subcategory.name	subcategory	
category	category	
active	active	
company	company	

The following fields in the Process Designer-based imArea table are synchronized to the legacy sub producttype table.

Field name in imArea table	Field name in producttype table	Default value
area.name	product.type	
subcategory	subcategory	
category	category	
active	active	
company	company	

## Service Desk

The following fields in the Process Designer-based category table are synchronized to the legacy sdCategory table.

Field name in category table	Field name in sdCategory table	Default value
name	name	
N/A	sharedflag	3 (Interaction/Incident/Problem)
active	active	
N/A	escalate.type (Fullfilment Process)	null
Name	description	
N/A	workflow	ServiceDesk_PDCM
approvals	approvals	
company	company	

The following fields in the Process Designer-based subcategory table are synchronized to the legacy sdSubcategory table.

Field name in subcategory table	Field name in sdSubcategory table	Default value
subcategory	subcategory.name	
category	category	
N/A	sharedflag	3 (Interaction/Incident/Problem)

Field name in subcategory table	Field name in sdSubcategory table	Default value
active	active	
subcategory	description	
company	company	

The following fields in the Process Designer-based producttype table are synchronized to the legacy sdArea table.

Field name in producttype table	Field name in sdArea table	Default value
product.type	area.name	
subcategory	subcategory	
category	category	
N/A	sharedflag	3 (Interaction/Incident/Problem)
active	active	
product.type	description	
company	company	

## Appendix B: Updates to the Process Designer framework

In addition to the changes made by the Applications Upgrade Utility, a number of updates to the Process Designer framework are included in all out-of-box modes of Service Manager 9.41. These changes are required in order to support the Hybrid mode.

### RAD routines updated

The following RAD routines are updated to use a **doAction** call instead of calling processes directly.

- axces.sm
- axces.apm.epmosmu
- axces.apm
- axces.cm3
- cm3r.gs.cancel

These routines are used for input-type event registers and relate to "cm3r," "cm3t," "probsummary," "incidents," "rootcause," and "rootcausetask" files.

### Workflow form updated

The workflow form is updated to display additional fields when you view a legacy workflow (that is, a workflow that the Applications Upgrade Utility generated by using information in legacy categories).

This update is required in order to support the configuration of Process Designer workflows by using legacy functionality, such as format control, legacy open state, legacy close state, and legacy default state.



## WSDL definition update

The Process Designer framework is updated to enable you to retrieve the actions of the legacy states in workflows when you select the action list of a Web Services Description Language (WSDL) definition.

# 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 Hybrid Guide (for Service Manager 9.41) (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](mailto:ovdoc-ITSM@hp.com).

We appreciate your feedback!

