Service Manager

Software Version: 9.50

For the supported Windows® and Linux® operating systems

Upgrade Guide

Document Release Date: October 2016

Software Release Date: October 2016

Hewlett Packard Enterprise

Legal Notices

Documentation Updates

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hpe.com/.

This site requires that you register for an HPE Passport and to sign in. To register for an HPE Passport ID, click **Register for HPE Passport** on the HPE Software Support site or click **Create an Account** on the HPE Passport login page.

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

Support

Visit the HPE Software Support site at: https://softwaresupport.hpe.com/.

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

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

HPE Software Solutions Now accesses the HPSW Solution and Integration Portal website. This site enables you to explore HPE Product Solutions to meet your business needs, includes a full list of Integrations between HPE Products, as well as a listing of ITIL Processes. The URL for this website is https://softwaresupport.hpe.com/km/KM01702731.

Contents

Introduction

This guide provides instructions on how to upgrade from an existing implementation of Service Manager to version 9.50. This guide covers both platform components (the Service Manager Server, Windows Client, Web Tier, Service Request Catalog (SRC), Smart Analytics, Solr Search Engine, and so on) and the Service Manager applications.

For instructions on performing a new installation and setup of Service Manager, see the *Service Manager Installation Guide*, which is available in HTML format from the Service Manager Help Center and in PDF format from the HPE Software Support Online (SSO) website.

Important: You have the option to upgrade only the platform components and run an earlier version of the applications. However, to take full advantage of the new features in this version, you are recommended to upgrade the applications as well.

Note: For compatibility information of the Service Manager platform and applications, refer to the Service Manager Support Matrix on the HPE Support Matrices website.

Upgrade the platform components

Before you can upgrade the applications, you must upgrade the Service Manager Server and Windows Client. Additionally, you may also want to or need to upgrade other components, such as the Web Tier, or Smart Analytics. Service Manager components other than the applications are referred to as "platform components".

> **Note:** For an RDBMS upgrade, see your specific database documentation. For applications upgrade, see "Upgrade the applications" on page 39.

A platform upgrade may include the following components.

Upgrade the Service Manager Server

> **Note:** This procedure only upgrades the Service Manager Server. For information about upgrading your Service Manager applications and data from previous versions, see the *Service Manager Upgrade Guide*.

To upgrade the Service Manager Server, follow these steps:

1. Before you install the Service Manager server, make a backup of your existing server installation folder. For example, `C:\Program Files\HPE\Service Manager` 9.40`\Server`.

   > **Note:** If you have a horizontally scaled system, be sure to back up the server installation folder for each server instance.

2. Install the new version of the Service Manager Server. For detailed steps, see the Service Manager Installation Guide.
3. Restore your old Service Manager server configuration files in the Server's RUN folder, including the sm.ini and sm.cfg files.
4. Perform additional configuration for the server upgrade.
   a. Stop the Service Manager Server service if necessary.
   b. Open the initialization (sm.ini) file with a text editor.
   c. Review the following list of deleted parameters. Compare this list with the parameters in sm.ini and delete any parameters from this list that exist in your sm.ini file:
      - alterlog: Alert log entries are now placed in the sm.log file.
      - apiserver: Client listener processes now connect through dedicated HTTP and HTTPS ports.
      - clustername: You can set up a failover configuration by starting a second load balancer process on a separate system.
      - cstrace
      - debugdtevents
      - debugdtrecords
      - debugdttrace
      - debugdtworld
      - debuglog
      - debugrpc
      - debugtransport
      - immediateshadow: Use your RDBMS utilities for reporting or data backup.
      - keepalive
      - odbccharacterarray
      - scdebug: Contact customer support if you need to debug customized RAD applications from previous versions.

- servletcontainer: You no longer need this parameter to start client listener processes.
- sqldetect: The database dictionary automatically detects changes to RDBMS tables and columns.
- sqlidentify
- sqllogintime
- sqlmodcount: The database dictionary automatically detects changes to RDBMS tables and columns.
- validateodbcfieldnames

d. If necessary, make the required changes for each of the following parameters:

Old name	New name	Required change
autopass_dir	licensefile	The default path to the license file is now in the server’s RUN directory. If you want to retain your previous license file path, you must specify it with the new licensefile parameter.
cacertpem	truststoreFile	Move your CA certificate from a PEM file to a Java keystore and provide the path to this keystore with the truststoreFile parameter.
certpem	keystoreFile	Move your server certificate from a PREM file to a Java keystore and provide the path to this keystore with the keystoreFile parameter.
dhpem	keystoreFile	This parameter is obsolete. You can set DH encryption properties when you create your Java keystore.
pkpem	keystoreFile	Move your server private key from a PEM file to a Java keystore and provide the path to this keystore with the keystoreFile parameter.
pkpempass	keystorePass	Move your server private key from a PEM file to a Java keystore and provide the password to this keystore with the keystorePass parameter.

Old name	New name	Required change
sccluster	group	Use the new parameter to shut down your virtual group.
scclusterbindaddress	groupbindaddress	Use this parameter to specify which IP Service Manager should bind to when your system has multiple IP addresses available from multiple network interfaces.
scclustermcastaddress	groupmcastaddress	Use this parameter to define your horizontally scaled system.
scclustername	groupname	Use the new parameter to define your server group name.
scclusterport	groupport	Use the new parameter to define your server group port.
scemail	emailout	Use the new parameter to process outbound e-mail from Service Manager.
schost	host	Use the new parameter to shut down or standby your system
sctimeramount	heartbeatinterval	Use the new parameter to keep client connections open.
soapaccepttimeout	sessiontimeout	Web services client connections now use the client time out limits.
soapreceivetimeout	sessiontimeout	Web services client connections now use the client time out limits.
ssl_trustedClientspem	trustedclientsJKS	Move your list of trusted clients from a PEM file to a Java keystore and provide the path to this keystore with the trustedclientJKS parameter.
timeoutlimit	sessiontimeout	Use the new parameter to set your client’s session time out.

e. Save and close the sm.ini file.

5. If you have configured SSL or single sign-on, you must copy the following files from the previous installation folder to the new installation folder (your own keystore file names may differ):

- <SM_install>\Server\RUN\cacerts
- <SM_install>\Server\RUN\server.keystore
- <SM_install>\Server\RUN\trustedclients.keystore

6. Restart the Service Manager Server.

Upgrade the Windows Client

To upgrade the Windows Client, follow these steps:

1. Make a backup copy of the following items of your existing Windows Client.
   - Your Windows Client home folder. For example, `C:\Users\<username>\ServiceManager`. Your connections and personalized settings are stored in this folder.

     > **Note:** This is the out-of-the-box home directory, and could differ from yours if you made changes to `<`*`Client`*`>\configuration\config.ini` file. If so, back up the files from the location specified in that file.

   - Your certificate configuration files if any (**Window** > **Preferences** > **HPE Service Manager** > **Security**). For example, your CA certificates file and client keystore file.
   - The following folder: `<Client>\plugins\com.hp.ov.sm.client.eclipse.user_x.xx.xxxx\src\resources\icons\obj16\`.
2. Uninstall the existing Windows client. Your connection and personalized settings are retained.
3. Install the new Windows Client. For detailed steps, see the Service Manager Installation Guide.
4. Copy the backup items of your old Windows Client to the new Windows Client, and restore the configurations.
5. Check the version in **Help** > **About Service Manager Client**.

Upgrade the Web Tier

To upgrade the Service Manager Web Tier, follow these steps:

1. Before deploying the new web tier, make a backup of the following items:
   - web.xml file
   - application-context.xml

- log4j.properties
- splash screen
- style sheets
- The folder that is defined in the *customize-folder* parameter in the web.xml file or the webtier.properties file if there is no such a folder
- any other customizations you made, including your webtier-<*version*>.war (webtier-ear-<*version*>.ear) file.

2. Delete or uninstall the existing web tier .war (or the .ear) file.
3. Clear the cache of your web application server.
4. Deploy the new webtier-9.50.war or webtier-ear-9.50.ear file. For details, see the installation section in the Service Manager Help Center.

   > **Note:** It is best practice to deploy with a unique context root. For example, /webtier-9.50.

5. Use a diff utility to compare the new web tier's web.xml file against your backed-up version to ensure that any new parameters are properly merged into the files used in your final deployment. Do this for **application-context.xml** as well as any other files you may have customized (such as style sheets and splash screens).
6. Make any new customizations that are necessary for your deployment.
7. Restart the web application server.
8. Check the version by clicking the HPE logo (**About HPE Service Manager**) icon.

Upgrade the language packs

If you have one or more old language packs installed in your system, upgrade them through an applications upgrade. For details, see Localized systems.

Upgrade Service Request Catalog (SRC)

To upgrade SRC, follow these steps:

1. Make a backup copy of your existing SRC deployment folder.
2. Deploy the new SRC. For detailed steps, see the SRC Interactive Installation Guide, which is available in the Service Manager Help Center.
3. Migrate your customizations from the old deployment to new one.

Upgrade the Mobility Client

To upgrade the Mobility Client (also referred to as the "Mobile Applications Client"), follow these steps:

1. Make a backup copy of your old Mobility Client deployment folder.
2. Deploy the new Mobility Client. For detailed steps, see the Mobile Applications Client installation instructions, which are available in the Service Manager Help Center.
3. Restore your old customizations in the new deployment.

Upgrade Smart Analytics

Smart Analytics is installed either on a Windows or on a Linux system.

To upgrade Smart Analytics, follow the instructions in these sections:

Upgrade to Service Manager 9.50 Smart Analytics on Windows

If you have installed an earlier version of Smart Analytics, follow the instructions in this section to upgrade your Smart Analytics to SM 9.50 Smart Analytics.

To upgrade your Smart Analytics, follow these steps:

1. Before you upgrade your Smart Analytics, complete these tasks:
   a. Make sure that all the index tasks are finished.
   b. Stop and then delete all the Smart Analytics services.

      > **Note:** To delete a service, run `sc delete "SERVICE_NAME"` in the command window.

   c. Back up all the Smart Analytics files.
2. Obtain the Smart Analytics installer for Windows.
3. Unpack the .zip file and then double-click the setup application (`setupSmartAnalyticsWindowsX64.exe`).
4. Read the introduction, and then click **Next**.
5. Read the License Agreement. To continue the installation, select **I accept the terms of the License Agreement**, and then click **Next**.
6. Select **Upgrade**, and then click **Next**.
7. Review the message on the Preparation for Upgrade dialog box, and then click **Next**.
8. Specify the version of Smart Analytics installed by selecting either of the following options, and then click **Next**:
   - Smart Analytics 9.40 or earlier
   - Smart Analytics 9.41
9. Continue with prompted upgrade steps, which vary based on the Smart Analytics version you selected in Step 8.

   Smart Analytics 9.40 or former

   If you selected Smart Analytics 9.40 or former, continue with these steps:
   a. Choose the current architecture of the Smart Analytics installed:
      - Standalone
      - Distributed
   b. Specify the IP address of the machine on which you have installed Service Manager server.

      You need to specify the IP addresses (or host names) of the Service Manager servers that are permitted to send administrative and query actions to the Smart Analytics servers. Use commas to separate multiple addresses (do not use a space before or after a comma).

      > **Note:** use a valid FQDN or IP address for the server address. Do not use the localhost or

> 127.0.0.1.

c. Select the components that you want to upgrade. The servers that are available for upgrade vary depending on the Smart Analytics architecture you specified:

> **Note:**
>
> - For Smart Analytics components which were installed in the same root folder, upgrade them at the same time in a single upgrade process. Otherwise, components may not be uninstalled completely when you uninstall Smart Analytics.
> - Upgrade the components separately by performing the upgrade process multiple times under the following conditions:
>   - Your current version of Smart Analytics is 9.34
>   - The Smart Analytics components to be upgraded are installed in different parent folders or physical machines

Architecture	Components Available for Upgrade
Standalone	IDOL Server Image Server
Distributed	Smart Analytics Proxy Content Server Image Server Image Proxy Server

d. Specify the parent folder where the selected Smart Analytics components are installed, and then click **Next**.

> **Note:**
>
> - If you are upgrading the Smart Analytics proxy server, the parent folder should include the IDOL folder.
> - If you are upgrading a distributed content server, the folder should be the parent of the content folder.

e. Follow the on-screen steps to configure the components you selected. Click **Next** after each step.

> **Note:** The number of copies (that is, the value of Replicas plus one) must be no more than the number of child servers. If you create more copies than there are child servers, DIH does not start.

f. Review the pre-installation summary, and then click **Install**. If you want to change your configuration, click **Previous**.

Smart Analytics 9.41

If you selected Smart Analytics 9.41, continue with these steps:

a. Specify the parent folder where the current Smart Analytics is installed, and then click **Next**.
b. The installer automatically detects and lists which components have been installed. Review the detected components, and then click **Next**.
c. Specify content folder names as needed. By default, the installer automatically specifies all the content folder names.
d. Specify image server folder names as needed. By default, the installer automatically specifies all the image server folder names.
e. Review the message that Smart Analytics is ready to upgrade, and then click **Next**.
f. Check the pre-installation summary, and then click **Install**. If you want to change your configuration, click **Previous**.

10. Wait for the upgrade to complete, and then click **Done**.

Post-upgrade actions

Perform the following actions after you upgrade your Smart Analytics:

- When the installation is complete, check the Start.bat and Stop.bat files under the <Smart Analytics Installation>/Scripts folder for any commands that start or stop services not on your local machine. Delete or comment out such commands and then add `sc start "[Service Name of your previous content]"` or `sc stop "[Service Name of your previous content]"` as needed.
- If you upgrade from Service Manager 9.41 to Service Manager 9.50, you need to do full re-index of Catalog_Library after you upgrade to Smart Analytics 9.50.
- If you upgraded from Smart Analytics 9.40 or earlier versions to Smart Analytics 9.50, then during the upgrade process, the existing configuration files would be renamed to `<component name>-old.cfg`. After you finish the upgrade, compare it with the new configuration file. If there is any customized change made in the previous .cfg file and it is still valid, manually add it back to the new

.cfg file.

- If you upgraded from Smart Analytics 9.40 or earlier versions to Smart Analytics 9.50, you need to re-index the Hot Topic Analytics data. However, do not start a training for Smart Ticket.
- If you upgraded from a distributed Smart Analytics to Smart Analytics 9.50, add the following lines before the `[actions]` section in the <*Smart Analytics Installation*>/Content#/Content#.cfg file for the system to clean up expired documents:

```
//-------------------------- Schedules ----------------------//
//expire and compact data everyday
[Schedule]
Expire=TRUE
ExpireTime=23:00
ExpireInterval=24:00
```

> **Note:** With this configured, the system automatically deletes all the expired documents at 23:00 every day. By default, the expiration date for documents is set to 365 days.

- If you upgraded from a standalone Smart Analytics with version 9.34 to Smart Analytics 9.50, comment out the `[License]` section in the <*ImageServer1*>/imageserver1.cfg file, and then add the following:

```
[Modules]
```

```
Enable=ocr
```

- If you upgraded a distributed Smart Analytics proxy server with version 9.40 or earlier to Smart Analytics 9.50, the previous configuration files would be renamed to `<component name>-old.cfg`, and you need to manually modify the following configurations in the <*SmartAnalytics*>/IDOL/IDOLServer.cfg file:

Configurations after upgrade	**Configurations in your previous file (<component name>-old.cfg)**	**You need to modify the value to**
VirtualDatabases=1	VirtualDatabases=<x>	VirtualDatabases=<x>

[vdb0]
dbname=Users
type=combinator
mapsto=0:Users
	[VDB0]
	DbName=News
	Type=Combinator
	MapsTo=0:News
		Do not change

	[VDB<x>] DbName=my_database	[VDB<x>] DbName=my_database

Configurations after upgrade	Configurations in your previous file (<component name>-old.cfg)	You need to modify the value to

	Type=Combinator
	MapsTo=<y>:my_database
	Note: The number of x starts from 1.
		Type=Combinator
		MapsTo=<y+1>:my_database
		◦ You must add the VDB configurations by their number sequence.
		◦ There may be other content between two VDB configuration sections.
		◦ Make sure the total number of the VDB configuration sections equals to the number specified in the VirtualDatabases parameter.
		◦ There is no difference if you use the upper or lower case for the section name of [VDB<x>].

[DistributionIDOLServers] Number=2	[DistributionIDOLServers] Number=<x>	[DistributionIDOLServers] Number=<x+1>

[IDOLServer0]
Name=SmartSearch
Host=127.0.0.1
Port=20010
DistributeByFieldsValues=GlobalSearch
		Do not change

Configurations after upgrade	Configurations in your previous file (<component name>-old.cfg)	You need to modify the value to

[IDOLServer1]
Name=Content1
Host=127.0.0.1
Port=
DistributeByFieldsValues=CONTENT1
	[IDOLServer<x>]
	Host=12.3.4.56
	Port=20010
	DistributeByFieldsValues=CONTENT<x>
	Note: The number of x starts from 0.
		[IDOLServer<x+1>]
		Name=Content<x+1>
		Host=12.3.4.56
		Port=20010
		DistributeByFieldsValues=CONTENT<x+1>
		Note:
		◦ You must add the IDOL server configurations by their number sequence.
		◦ Make sure the total number of the IDOL server configuration sections equals to the number specified in the [DistributionIDOLServers] Number parameter.

- If you upgrade from Smart Analytics 9.41 to Smart Analytics 9.50, you need to manually modify the following configuration sections:

  In the <*SmartAnalytics*>/IDOL/IDOLServer.cfg file:

Existing configurations	You need modify the configuration to

[IndexQueue]
IndexQueueInitialSize=10240
IndexQueueMaxHistorySize=500
IndexQueueMaxPendingItems=10240
 	
[SetDateFields]
// Fields containing the document date
Property=DateFields
 	
PropertyFieldCSVs==*/DREDATE,*/DREDATE_*,*/DATE,*/DATE_*,*/*_DATE,*/*_TIME
	[IndexQueue]
	IndexQueueInitialSize=10240
	IndexQueueMaxHistorySize=10240
	IndexQueueMaxPendingItems=0
	
	[SetDateFields]
	// Fields containing the document date
	Property=DateFields
	PropertyFieldCSVs=*/DREDATE,*/DREDATE_*,*/DATE,*/DATE_*,*/*_DATE,*/*_TIME,*/MODIFIED_EPOCH,*/LASTMODIFIED
	
	[Logging]
	LogExpireAction=Day

Existing configurations	You need modify the configuration to

[Logging]
LogExpireAction=compress
LogOldAction=move
LogMaxSizeKBs=20480
	LogMaxOldFiles=7
	LogOldAction=delete
	LogMaxSizeKBs=102400

In the <*SmartAnalytics*>/Level2proxy/IDOLServer.cfg file:

Existing configurations	You need modify the configuration to

[Server]
 	
[IndexQueue]
 	
IndexQueueInitialSize=10240
 	
IndexQueueMaxHistorySize=500
 	
IndexQueueMaxPendingItems=10240
 	
[Logging]
 	
LogExpireAction=compress
 	
LogOldAction=move
 	
LogMaxSizeKBs=20480
	[Server]
	//store index data on disk only when the index cache is full or when the MaxSyncDelay(120 seconds by default) is reached.
	DelayedSync=TRUE
	
	[IndexCache]
	IndexCacheMaxSize=102400
	
	[IndexQueue]
	IndexQueueInitialSize=10240
	IndexQueueMaxHistorySize=10240
	IndexQueueMaxPendingItems=0
	
	[Logging]
	LogExpireAction=Day
	LogMaxOldFiles=7
	LogOldAction=delete
	LogMaxSizeKBs=102400

In the <*SmartAnalytics*>/Content<x>/Content<x>.cfg file:

Existing configurations	You need modify the configuration to

[IndexQueue]
 	
IndexQueueInitialSize=10240
 	
IndexQueueMaxHistorySize=500
 	
IndexQueueMaxPendingItems=10240
 	
[SetDateFields]
// Fields containing the document date
Property=DateFields
 	
PropertyFieldCSVs=*/DREDATE,*/DREDATE_*,*/DATE,*/DATE_*,*/*_DATE,*/*_TIME
 	
[SetMatchFields]
PropertyFieldCSVs=*/*_MATCH,*/EDK_*,*/*_MATCH_INDEX, */*_MATCH_WEIGHT*,*/*_BOOLEAN
 	
[Logging]
LogExpireAction=compress
LogOldAction=move
LogMaxSizeKBs=20480
 	
[LanguageTypes]
 	
GenericTransliteration=TRUE
Transliteration=false
	//The amount of memory to allocate to store the most commonly occurring terms. (using 500MB as term cache here, recommended range is 10240 to 2048000.)
	[TermCache]
	TermCachePersistentKB=512000
	
	//-------------------------- Schedules ----------------------//
	//expire and compact data everyday
	[Schedule]
	Expire=TRUE
	ExpireTime=23:00
	ExpireInterval=24:00
	
	[IndexQueue]
	IndexQueueInitialSize=10240
	IndexQueueMaxHistorySize=10240
	IndexQueueMaxPendingItems=0
	
	[SetDateFields]
	// Fields containing the document date
	Property=DateFields
	PropertyFieldCSVs=*/DREDATE,*/DREDATE_*,*/DATE,*/DATE_*,*/*_DATE,*/*_TIME,*/MODIFIED_EPOCH,*/LASTMODIFIED
	
	[SetMatchFields]
	PropertyFieldCSVs=*/*_MATCH,*/EDK_*,*/*_MATCH_INDEX, */*_MATCH_WEIGHT*,*/*_BOOLEAN,*/IMPORTMAGICEXTENSION
	
	[Logging]
	LogExpireAction=Day
	LogMaxOldFiles=7
	LogOldAction=delete
	LogMaxSizeKBs=102400
	
	[LanguageTypes]
	GenericTransliteration=TRUE
	Transliteration=true

After you modify the configurations, restart all related Smart Analytics components of which such configuration are modified. Then, on the machine where level2proxy is located, reload updated configuration files dynamically and regenerate match index by visiting the following URLs :

- Reload updated configuration files dynamically:

  http://localhost:20011/DRERESET

- Regenerate the match fields:

  http://localhost:20011/DREREGENERATE?Type=Match

  Wait for a few minutes for the regeneration to complete.

Upgrade to Service Manager 9.50 Smart Analytics on Linux

If you have installed an earlier version of Smart Analytics, follow the instructions in this section to upgrade your Smart Analytics to SM 9.50 Smart Analytics.

To upgrade your Smart Analytics on a Linux system, follow these steps:

1. Before you upgrade your Smart Analytics, complete these tasks:
   a. Make sure that all the index tasks are finished.
   b. Stop all the Smart Analytics services.
   c. Back up all the Smart Analytics files.
2. Obtain the Smart Analytics installer for Linux.
3. Unpack the .zip file and then execute the binary file (`setupSmartAnalyticsLinuxX64.bin`) from the command line.
4. Read the introduction, and then press **Enter**.
5. Read the License Agreement, and then press **Enter** repeatedly until you see **DO YOU ACCEPT THE TERMS OF THIS LICENSE AGREEMENTS? (Y/N)**.
6. Type **Y**, and then press **Enter**.
7. Type **2** to select Upgrade, and then press **Enter**.
8. Choose the version of Smart Analytics installed by typing the corresponding number, and then

press **Enter**:

- 1-Smart Analytics 9.40 or earlier
- 2-Smart Analytics 9.41

9. Continue with on-screen instructions, which vary based on the Smart Analytics version you selected in Step 7.

   Smart Analytics 9.40 or former

   If you selected Smart Analytics 9.40 or former, continue with these steps:

   a. Choose the current architecture of the Smart Analytics installed typing the corresponding number, and then press **Enter**:

      - 1-Standalone
      - 2-Distributed

   b. Specify the IP address of the machine on which you have installed Service Manager server, and then press **Enter**.

      You need to specify the IP addresses (or host names) of the Service Manager servers that are permitted to send administrative and query actions to the Smart Analytics servers. Use commas to separate multiple addresses (do not use a space before or after a comma).

      > **Note:** use a valid FQDN or IP address for the server address. Do not use the localhost or 127.0.0.1.

   c. Select the components that you want to upgrade by typing the corresponding numbers. Use commas to separate multiple choices (do not use a space before or after a comma). The servers that are available for upgrade vary depending on the Smart Analytics architecture you specified.

      > **Note:**
      >
      > - For Smart Analytics components which were installed in the same root folder, upgrade them at the same time in a single upgrade process. Otherwise, components may not be uninstalled completely when you uninstall Smart Analytics.
      > - Upgrade the components separately by performing the upgrade process multiple times under the following conditions:

- Your current version of Smart Analytics is 9.34
- The Smart Analytics components to be upgraded are installed in different parent folders or physical machines

Architecture	Components Available for Upgrade
Standalone	IDOL Server Image Server
Distributed	Smart Analytics Proxy Content Server Image Server Image Proxy Server

d. Specify the parent folder where the selected Smart Analytics components are installed, and then press **Enter**.

> **Note:**
>
> - If you are upgrading the Smart Analytics proxy server, the folder should include the IDOL folder.
> - If you are upgrading a distributed content server, the folder should be the parent of the content folder.

e. Follow the configuration steps to configure the components you selected. Press **Enter** after each configuration step.

> **Note:** The number of copies (that is, the value of Replicas plus one) must be no more than the number of child servers. If you create more copies than there are child servers, DIH does not start.

f. Check the pre-upgrade summary, and then press **Enter**.

Smart Analytics 9.41

If you selected Smart Analytics 9.41, continue with these steps:

a. Specify the parent folder where the current Smart Analytics is installed, and then press **Enter**.
b. The installer automatically detects and lists which components have been installed. Review the detected components, and then press **Enter**.

c. Specify content folder names as needed, and then press **Enter**. By default, the installer automatically specifies all the content folder names.

d. Specify image server folder names as needed, and then press **Enter**. By default, the installer automatically specifies all the image server folder names.

e. Review the message that Smart Analytics is ready to upgrade, and then press **Enter**.

f. Check the pre-installation summary, and then press **Enter**.

10. Wait for the installation to complete, and then press **Enter** to exit the installer.

Post-upgrade actions

Perform the following actions after you upgrade your Smart Analytics:

- When the installation is complete, do the following:
  - Check the StartAll.sh and StopAll.sh files under <Smart Analytics Installation>/Scripts folder for any commands that start or stop services not on your local machine. Delete or comment out such commands and then add `source ./Start[Service Name of your previous content]` or `source ./Stop[Service Name of your previous content]` as needed.
  - Delete all the start[Name of Service not on local machine].sh and stop [Name of Service not on local machine].sh files under <Smart Analytics Installation>/Scripts folder.
- If you upgrade from Service Manager 9.41 to Service Manager 9.50, you need to do full re-index of Catalog_Library after you upgrade to Smart Analytics 9.50.
- If you upgrad from Smart Analytics 9.40 or earlier versions to Smart Analytics 9.50, then during the upgrade process, the existing configuration files would be renamed to `<component name>-old.cfg`. After you finish the upgrade, compare it with the new configuration file. If there is any customized change made in the previous .cfg file and it is still valid, manually add it back to the new .cfg file.
- If you upgraded from Smart Analytics 9.40 or earlier versions to Smart Analytics 9.50, you need to re-index the Hot Topic Analytics data. However, do not start a training for Smart Ticket.
- If you upgraded from a distributed Smart Analytics to Smart Analytics 9.50, add the following lines before the [actions] section in the `content[x].cfg file` for the system to clean up expired documents:

```
//-------------------------- Schedules ----------------------//
//expire and compact data everyday
[Schedule]
Expire=TRUE
```

```
ExpireTime=23:00
ExpireInterval=24:00
```

> **Note:** With this configured, the system automatically deletes all the expired documents at 23:00 every day. By default, the expiration date for documents is set to 365 days.

- If you upgraded a standalone Smart Analytics with version 9.34, locate the `[Security]` section in the <*Content1*>/content1.cfg file, and then add the // comment characters to the parameters as displayed in the following:

  ```
  [Security]

  SecurityInfoKeys=123,144,564,231

  DebugDecrypt=true

  //0=NT_V4

  //1=SharePoint
  ```

- If you upgraded a distributed Smart Analytics proxy server with version 9.40 or former, the previous configuration files would be renamed to `<component name>-old.cfg`, and you need to manually modify the following configurations in the <*SmartAnalytics*>/IDOL/IDOLServer.cfg file:

Configurations after upgrade	Configurations in your previous file (<component name>-old.cfg)	You need to modify the value to
VirtualDatabases=1	VirtualDatabases=<x>	VirtualDatabases=<x>

[vdb0]
dbname=Users
type=combinator
mapsto=0:Userss
	[VDB0]
	DbName=News
	Type=Combinator
	MapsTo=0:News
		Do not change

	[VDB<x>]
	DbName=my_database
	Type=Combinator
	MapsTo=<y>:my_database
	Note: The number of x starts from 1.
		[VDB<x>]
		DbName=my_database
		Type=Combinator
		MapsTo=<y+1>:my_database
		Note:
		◦ You must add the VDB configurations by their number sequence.

Configurations after upgrade	Configurations in your previous file (<component name>-old.cfg)	You need to modify the value to
		◦ There may be other content between two VDB configuration sections. ◦ Make sure the total number of the VDB configuration sections equals to the number specified in the VirtualDatabases parameter. ◦ There is no difference if you use the upper or lower case for the section name of [VDB<x>].
[DistributionIDOLServers] Number=2	[DistributionIDOLServers] Number=<x>	[DistributionIDOLServers] Number=<x+1>

[IDOLServer0]
Name=SmartSearch
Host=127.0.0.1
Port=20010
DistributeByFieldsValues=GlobalSearch
		Do not change

[IDOLServer1]
Name=Content1
Host=127.0.0.1
Port=
DistributeByFieldsValues=CONTENT1
	[IDOLServer<x>]
	Host=12.3.4.56
	Port=20010
	DistributeByFieldsValues=CONTENT<x>
	Note: The number of x starts from 0.
		[IDOLServer<x+1>]
		Name=Content<x+1>
		Host=12.3.4.56
		Port=20010
		DistributeByFieldsValues=CONTENT<x+1>
		Note:
		◦ You must add the IDOL server configurations by

Configurations after upgrade	Configurations in your previous file (<component name>-old.cfg)	You need to modify the value to
		their number sequence. ◦ Make sure the total number of the IDOL server configuration sections equals to the number specified in the [DistributionIDOLServers] Number parameter.

- If you upgrade from Smart Analytics 9.41 to Smart Analytics 9.50, you need to manually modify the following configuration sections:

  In the <*SmartAnalytics*>/IDOL/IDOLServer.cfg file:

Existing configurations	You need modify the configuration to

`[IndexQueue]`
`IndexQueueInitialSize=10240`
`IndexQueueMaxHistorySize=500`
`IndexQueueMaxPendingItems=10240`
 	
`[SetDateFields]`
`// Fields containing the document date`
`Property=DateFields`
 	
`PropertyFieldCSVs==*/DREDATE,*/DREDATE_*,*/DATE,*/DATE_*,*/*_DATE,*/*_TIME`
 	
`[Logging]`
`LogExpireAction=compress`
`LogOldAction=move`
`LogMaxSizeKBs=20480`
	`[IndexQueue]`
	`IndexQueueInitialSize=10240`
	`IndexQueueMaxHistorySize=10240`
	`IndexQueueMaxPendingItems=0`
	
	`[SetDateFields]`
	`// Fields containing the document date`
	`Property=DateFields`
	`PropertyFieldCSVs=*/DREDATE,*/DREDATE_*,*/DATE,*/DATE_*,*/*_DATE,*/*_TIME,*/MODIFIED_EPOCH,*/LASTMODIFIED`
	
	`[Logging]`
	`LogExpireAction=Day`
	`LogMaxOldFiles=7`
	`LogOldAction=delete`
	`LogMaxSizeKBs=102400`

  In the <*SmartAnalytics*>/Level2proxy/IDOLServer.cfg file:

Existing configurations	You need modify the configuration to

[Server]
 	
[IndexQueue]
 	
IndexQueueInitialSize=10240
 	
IndexQueueMaxHistorySize=500
 	
IndexQueueMaxPendingItems=10240
 	
[Logging]
 	
LogExpireAction=compress
 	
LogOldAction=move
 	
LogMaxSizeKBs=20480
	[Server]
	//store index data on disk only when the index cache is full or when the MaxSyncDelay(120 seconds by default) is reached.
	DelayedSync=TRUE
	
	[IndexCache]
	IndexCacheMaxSize=102400
	
	[IndexQueue]
	IndexQueueInitialSize=10240
	IndexQueueMaxHistorySize=10240
	IndexQueueMaxPendingItems=0
	
	[Logging]
	LogExpireAction=Day
	LogMaxOldFiles=7
	LogOldAction=delete
	LogMaxSizeKBs=102400

In the <*SmartAnalytics*>/Content<x>/Content<x>.cfg file:

Existing configurations	You need modify the configuration to

[IndexQueue]
 	
IndexQueueInitialSize=10240
 	
IndexQueueMaxHistorySize=500
	//The amount of memory to allocate to store the most commonly occurring terms. (using 500MB as term cache here, recommended range is 10240 to 2048000.)
	[TermCache]
	TermCachePersistentKB=512000

Existing configurations	You need modify the configuration to

IndexQueueMaxPendingItems=
10240
 	
[SetDateFields]
// Fields containing the
document date
Property=DateFields
 	
PropertyFieldCSVs=*/DREDAT
E,*/DREDATE_
,/DATE,*/DATE_*,*/*_
DATE,*/*_TIME
 	
[SetMatchFields]
PropertyFieldCSVs=*/*_
MATCH,*/EDK_*,*/*_MATCH_
INDEX, */*_MATCH_
WEIGHT*,*/*_BOOLEAN
 	
[Logging]
LogExpireAction=compress
LogOldAction=move
LogMaxSizeKBs=20480
 	
[LanguageTypes]
 	
GenericTransliteration=TRU
E
Transliteration=false
	//-------------------------- Schedules ----------
	------------//
	//expire and compact data everyday
	[Schedule]
	Expire=TRUE
	ExpireTime=23:00
	ExpireInterval=24:00
	
	[IndexQueue]
	IndexQueueInitialSize=10240
	IndexQueueMaxHistorySize=10240
	IndexQueueMaxPendingItems=0
	
	[SetDateFields]
	// Fields containing the document date
	Property=DateFields
	PropertyFieldCSVs=*/DREDATE,*/DREDATE_
	,/DATE,*/DATE_*,*/*_DATE,*/*_TIME,*/MODIFIED_
	EPOCH,*/LASTMODIFIED
	
	[SetMatchFields]
	PropertyFieldCSVs=*/*_MATCH,*/EDK_*,*/*_MATCH_
	INDEX, */*_MATCH_WEIGHT*,*/*_
	BOOLEAN,*/IMPORTMAGICEXTENSION
	
	[Logging]
	LogExpireAction=Day
	LogMaxOldFiles=7
	LogOldAction=delete
	LogMaxSizeKBs=102400
	
	[LanguageTypes]
	GenericTransliteration=TRUE
	Transliteration=true

After you modify the configurations, restart all related Smart Analytics components of which such configuration are modified. Then, on the machine where level2proxy is located, reload updated configuration files dynamically and regenerate match index by visiting the following URLs :

- Reload updated configuration files dynamically:

  http://localhost:20011/DRERESET

- Regenerate the match fields:

http://localhost:20011/DREREGENERATE?Type=Match

Wait for a few minutes for the regeneration to complete.

Upgrade the Solr Search Engine

To upgrade the Solr Search Engine, follow these steps:

1. Make a backup of the following items in your existing installation:
   - The search engine installation folder. For example, `C:\Program Files\HPE\Service Manager 9.40\Search Engine Backup`
   - The files to be modified by the unload files in the KM installation package.
   - The schemastub.xml file under the `<SM server>/RUN/km/styles/` directory.
2. Make sure your Service Manager server and clients have upgraded to version 9.50.
3. Stop your Solr Search Engine.
4. Install the new Solr Search Engine. For details, see the Solr Search Engine installation instructions in the Service Manager Help Center.
5. If you are running a 9.3x version of the Service Manager applications, load QCCR1E91035_SM940_SM930.unl into your Service Manager system.

   Caution: Skip this step if you are running the Service Manager 9.40 or later applications.
6. Restart your Solr search engine.
7. Reconfigure your search servers and knowledgebases, and then perform a full re-index of all of your knowledgebases. For details, see the Solr Search Engine installation instructions in the Service Manager Help Center.

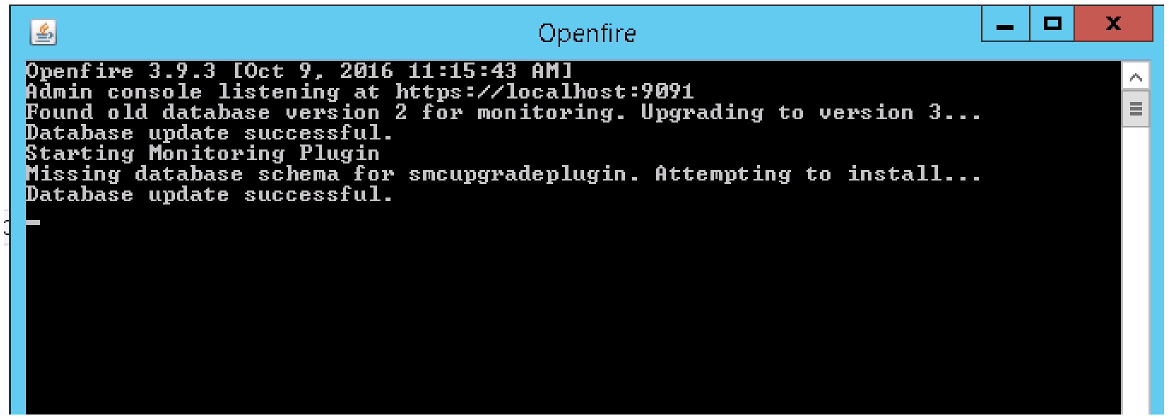
Upgrade the chat server

Important: Collaboration versions earlier than 9.41 (which are based on the legacy HPE Enterprise Collaboration Server) are no longer supported. If you are upgrading from the legacy EC based Collaboration, no backup is needed and only data migration is needed.

Note: You must upgrade the chat server to version 9.50 if you have upgraded the Service Manager server to 9.50.

Follow these steps to upgrade the chat server from version 9.41 to 9.50:

1. Stop your existing Openfire.
2. Open a DOS command prompt. Navigate to the C:\Program Files (x86)\HPE\Service Manager 9.41\ChatServer\bin directory, and then run the **openfire-service /uninstall** command to remove the original Openfire service.
3. Rename the existing Openfire folder.
4. Make a backup of the following items:
   - <Openfire_Home>\conf\crowd.properties
   - <Openfire_Home>\conf\openfire.xml
   - <Openfire_Home>\conf\security.xml
   - <Openfire_Home>\plugins\lwssoplugin\lwssofmconf.xml
   - <Openfire_Home>\lib\ojdbc-xxxx.jar (the JDBC driver if you copied it to this folder before)
   - <Openfire_Home>\resources\security\client.truststore
   - <Openfire_Home>\resources\security\keystore
   - <Openfire_Home>\resources\security\truststore
   - Any other customizations that you made
5. Save the chat server installer from the Service Manager 9.50 installation package to your computer.
6. Navigate to the C:\Program Files (x86)\HPE\Service Manager 9.50 folder and create a new directory called ChatServer.
7. Extract the chat-server-9.50.zip file to the ChatServer folder.
8. Copy the files that are backed up in step 3 to the corresponding directories in the ChatServer folder.
9. Open a DOS command prompt. Navigate to the C:\Program Files (x86)\HPE\Service Manager 9.50\ChatServer\bin folder, and then run **openfire.bat**. The chat server is updated automatically.



```
Openfire 3.9.3 [Oct 9, 2016 11:15:43 AM]
Admin console listening at https://localhost:9091
Found old database version 2 for monitoring. Upgrading to version 3...
Database update successful.
Starting Monitoring Plugin
Missing database schema for smcupgradeplugin. Attempting to install...
Database update successful.
```

Tip: You can also add the new chat server service to Windows services by running the **install-service.bat** command.

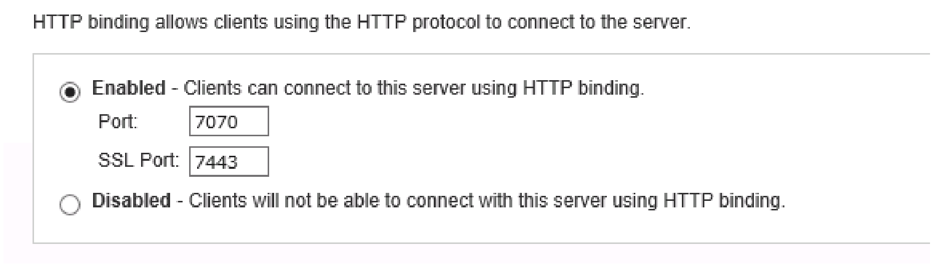
(Optional) Open the HTTP port for the chat server

Follow these steps:

1. Browse to the <Openfire_Home>\conf directory, and then open openfire.xml with a text editor.
2. Update the HTTP port as illustrated below:

```
<adminConsole>
  <!-- Disable either port by setting the value to -1 -->
  <port>9090</port>
  <securePort>9091</securePort>
</adminConsole>
```

3. Save your changes and close this file.
4. Access https://localhost:9091 and log on to the Administraton Console page.
5. Go to **Server** > **Server Settings** > **HTTP Binding**, and then fill the HTTP port as illustrated below:



HTTP binding allows clients using the HTTP protocol to connect to the server.

● **Enabled** - Clients can connect to this server using HTTP binding.
Port: 7070
SSL Port: 7443

○ **Disabled** - Clients will not be able to connect with this server using HTTP binding.

6. Save your changes ans then restart the chat server.

Install platform unloads

New features and enhancements introduced in the latest version of a platform component (that is, a component other than the applications, such as the Server, Windows Client, Web Tier, SRC, Smart Analytics, and so on) sometimes require updates to both the component and the applications.

Some customers may select to upgrade only certain platform components and still run an old version of the applications. For these customers to take advantage of the full benefits of the latest version of a platform component, HPE provides unload files as needed in an **unloads** directory in the installation package of the component. These unload files contain the application-side changes made for the latest version of the component, and you can manually install them in your system.

> **Note:** If you plan to upgrade or have already upgraded the applications to the latest version, ignore these unload files, which are already included in the latest applications.

The following table lists the unload files for different platform components.

Name	Applicable applications version	Component	Description
QCCR1E112012_ SM950_ SM934.unl	9.34+PD4, 9.35+PD4	Server	Solves the issue that PD Framework components (including Condition Editor , Query Editor , Workflow Editor, and Task Planner) do not work correctly if an earlier version of the applications is running on the 9.50 RTE and web tier. **Note:** This unload is required if you are running 9.34 or 9.35 applications with Process Designer Content Pack 4 applied.
QCCR1E112815_ SM950_ SM934.unl	9.34	Server	Fixes an issue in which Time Period Management menus are not displayed correctly due to incorrect menu condition settings.
QCCR1E118520_ SM950_ SM934.unl	9.34, 9.35, and 9.40	Server	Enables the enhanced query hash algorithm for the web client.
QCCR1E131250_ SM950_ SM940.unl	940 and 941	Server	Fixes the issue that Smart Analytics cannot recognize the text in a photo.
QCCR1E135272_	9.41	Server	Enables the Collaboration functionality to work

Name	Applicable applications version	Component	Description
SM950_ SM941.unl			correctly when you run Service Manager applications version 9.41 with version 9.50 or later of the Service Manager Server and clients.
QCCR1E66893_ SM950_ SM934.unl	9.34, 9.35, 9.40, and 9.41	Server	Enables the Document Engine to enforce application-level security for web service Retrieve requests.
QCCR1E99207_ SM950_ SM934.unl	9.34 and 9.35	Server	Enables Service Manager to lazy load global lists. This reduces the login time in environments that contain a large number of global lists.
QCCR1E129149_ SM950_ SM941.unl	9.41	Web Tier	Enables To-Do Alerts to be automatically acknowledged without the user having to click the **Acknowledge** button.
QCCR1E124742_ SM950_ SM934.unl	9.34, 9.35, and 9.40	Web Tier	Merges the module configuration requests that are sent from the web client to the server during the login process. This helps to reduce the login time.
QCCR1E116757_ SM950_ SM940.unl	9.40 and 9.41	Web Tier	Enables look up function support for the security manual list type setting.
QCCR1E112070_ SM950_ SM934.unl	9.34	Web Tier	For Service Catalog User Selection multi-selection support.
QCCR1E104944_ SM950_ SM941.unl	9.41	Web Tier	Enables a fix that hides the User Selections of Service Catalog items if their Visible Condition is set to false.
QCCR1E125775_ SM950_ SM941.unl	9.41	Web Tier	Enables IT users to manually trigger context-aware search. Additionally, enables context-aware search to be enabled from ESS.
MOBILITY_ESS_ SM950_ SM934.unl	9.34	Mobility Client	Enables the Mobility Client self-service interface
QCCR1E123225_ SM950_ SM934.unl	9.34, 9.35, 9.40, and 9.41	Mobility Client	Enables the "Set Reminder" feature for the Mobility Client.
QCCR1E125623_ SM950_ SM934.unl	9.34, 9.35, 9.40, and 9.41	Mobility Client	Fixes the issue that buttons are not grayed out as expected when users use wizards in the Mobility Client.

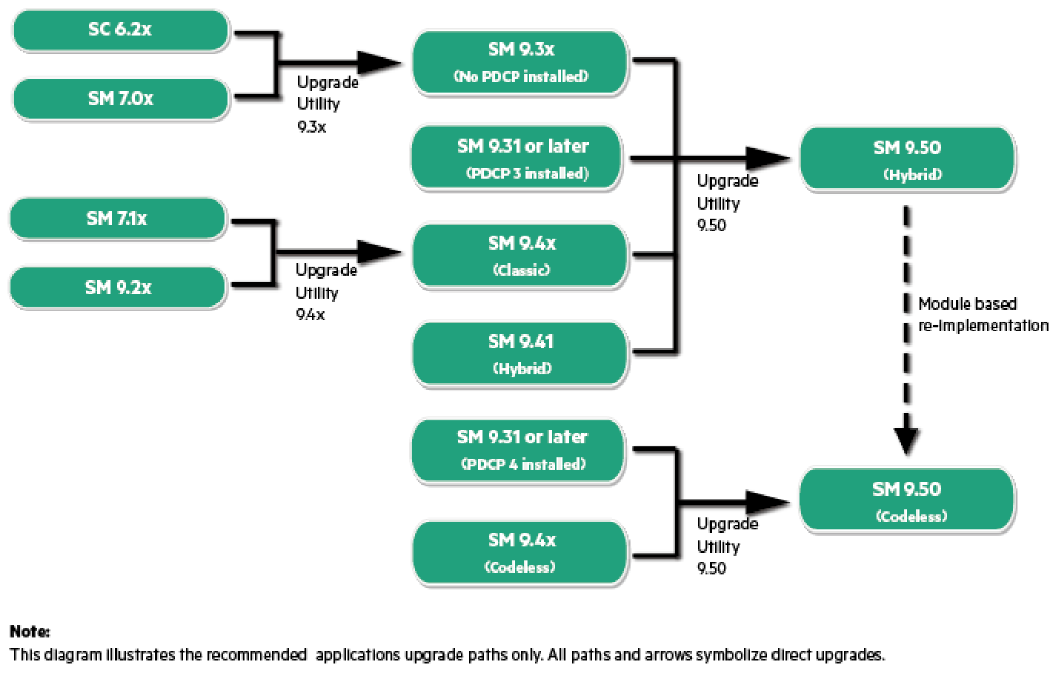
Name	Applicable applications version	Component	Description
QCCR1E128475_SM950_SM940.unl	9.40 and 9.41	Solr Search Engine	Provides the ability to enable HTTPS connections for the Solr Search Engine. Loading this unload will add a new option, **SSL Enabled**, in the Knowledge Management Environment record (**Knowledge Management** > **Administration** > **Environment**).

Review the description of each unload file to determine which unload files are needed for your system.

For the steps to load an unload into Service Manager, see the the Service Manager Help Center (**Administer** > **Database Administration** > **Data persistence** > **Importing records** > **Load an unload file**).

Upgrade the applications

The following diagram illustrates the recommended HPE Service Manager 9.50 applications upgrade paths. The paths and arrows symbolize direct upgrades.


SC 6.2x
SM 7.0x
Upgrade Utility 9.3x
SM 7.1x
SM 9.2x
Upgrade Utility 9.4x
SM 9.3x (No PDCP installed)
SM 9.31 or later (PDCP 3 installed)
SM 9.4x (Classic)
SM 9.41 (Hybrid)
Upgrade Utility 9.50
SM 9.50 (Hybrid)
Module based re-implementation
SM 9.31 or later (PDCP 4 installed)
SM 9.4x (Codeless)
Upgrade Utility 9.50
SM 9.50 (Codeless)
Note:
This diagram illustrates the recommended applications upgrade paths only. All paths and arrows symbolize direct upgrades.

Note: If you are using SM7.1x or SM9.2x, you must firstly upgrade the applications version to 9.41 before you can further upgrade to SM9.50 applications. Refer to Applications Upgrade Guide (from SM7.1x) and Applications Upgrade Guide (from 9.2x) for more information.

Follow these steps to run the Upgrade Utility with the Upgrade package to upgrade your applications from version 9.3x/9.4x:

1. Make sure you have upgraded your Service Manager server and clients to version 9.50.

   For details, see "Upgrade the platform components" on page 9.

2. Run the SM 9.50 Upgrade Utility with the Upgrade package. For details, see "Applications upgrade overview" on page 46.

Supported upgrade paths

HPE Service Manager 9.50 is available in two modes: Service Manager Codeless and Service Manager Hybrid.

HPE Service Manager Codeless

As of version 9.40, Service Manager Codeless incorporates the Process Designer functionality that was previously available through content packs, together with further enhancements that are not available to users of Service Manager Classic. Primarily, this involves implementing Process Designer-based workflows for a number of modules, including Change Management, Service Desk, Incident Management, Problem Management, Request Fulfillment, Knowledge Management, and Service Level Management.

> **Note:** Process Designer is a graphical interface that enables you to develop the workflows that control the flow of a single record throughout its lifecycle within HPE Service Manager, without being an expert in RAD programming.

HPE Service Manager 9.50 Codeless is available to customers who are installing Service Manager for the first time or who are upgrading from the following scenarios:

- A Service Manager 9.3x system that has Process Designer Content Pack 9.30.3 (PDCP4) applied
- A Service Manager 9.4x system (Codeless mode)

HPE Service Manager 9.50 Hybrid

HPE Service Manager 9.50 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 9.50 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 to customers who are upgrading from the following scenarios:

- A Service Manager 9.3x system without PDCP installed
- A Service Manager 9.3x system that has Process Designer Content Pack 9.30.2 (PDCP3) applied.
- A Service Manager 9.4x system (Classic mode or Hybrid mode)

> **Important:**
> Service Manager 9.50 does not support the following upgrades:

- Upgrade from SM9.3x with PDCP4 installed to SM9.50 Hybrid.
- Upgrade from SM9.4x Codeless to SM9.50 Hybrid.

Meanwhile, Service Manager 9.50 does not support the following direct upgrades:

- Direct upgrade from SC6.2/SM7.0x to SM9.50 Hybrid/Codeless.
- Direct upgrade from SM7.1x/SM9.2x to SM9.50 Hybrid/Codeless.
- Direct upgrade from SM9.3x without PDCP installed/SM9.3x with PDCP3 installed to SM9.50 Codeless.
- Direct upgrade from SM9.4x Classic to SM9.50 Codeless.
- Direct upgrade from SM9.4x Hybrid to SM9.50 Codeless.

After you upgraded to Service Manager9.50 applications, refer to *HPE Service Manager Support Matrix* for a list of supported databases. If your existing database is not supported, upgrade your database by following the instructions in your specific database documentation.

Upgrade to HPE Service Manager 9.50 Codeless

Recommended upgrade order from SM9.4x Codeless

1. Install the Service Manager 9.50 server.
2. Install the Service Manager 9.50 Windows and web clients.
3. Upgrade to the Service Manager 9.50 applications using Service Manager 9.50 Upgrade Utility.
4. Install or upgrade other components.

Recommended upgrade order from SM9.3x with Process Designer Content Pack 9.30.3 (PDCP4) installed

1. Install the Service Manager 9.50 server.
2. Install the Service Manager 9.50 Windows and web clients.
3. Upgrade to the Service Manager 9.50 applications using Service Manager 9.50 Upgrade Utility.
4. Install or upgrade other components.

Migration procedure from Service Manager 9.50 Hybrid to 9.50 Codeless

Service Manager Hybrid has all the necessary prerequisites and can be migrated to Service Manager Codeless.

Upgrade to HPE Service Manager 9.50 Hybrid

Recommended upgrade order from SM9.4x Classic/SM9.3x without Process Designer Content Pack (PDCP) installed

1. Install the Service Manager 9.50 server.
2. Install the Service Manager 9.50 Windows and web clients.
3. Upgrade to the Service Manager 9.50 applications using Service Manager 9.50 Upgrade Utility.
4. Install or upgrade other components.

Recommended upgrade order from SM9.3x with Process Designer Content Pack 9.30.2 (PDCP3) installed/SM 9.41 Hybrid

1. Install the Service Manager 9.50 server.
2. Install the Service Manager 9.50 Windows and web clients.
3. Upgrade to the Service Manager 9.50 applications using Service Manager 9.50 Upgrade Utility.
4. Install or upgrade other components.

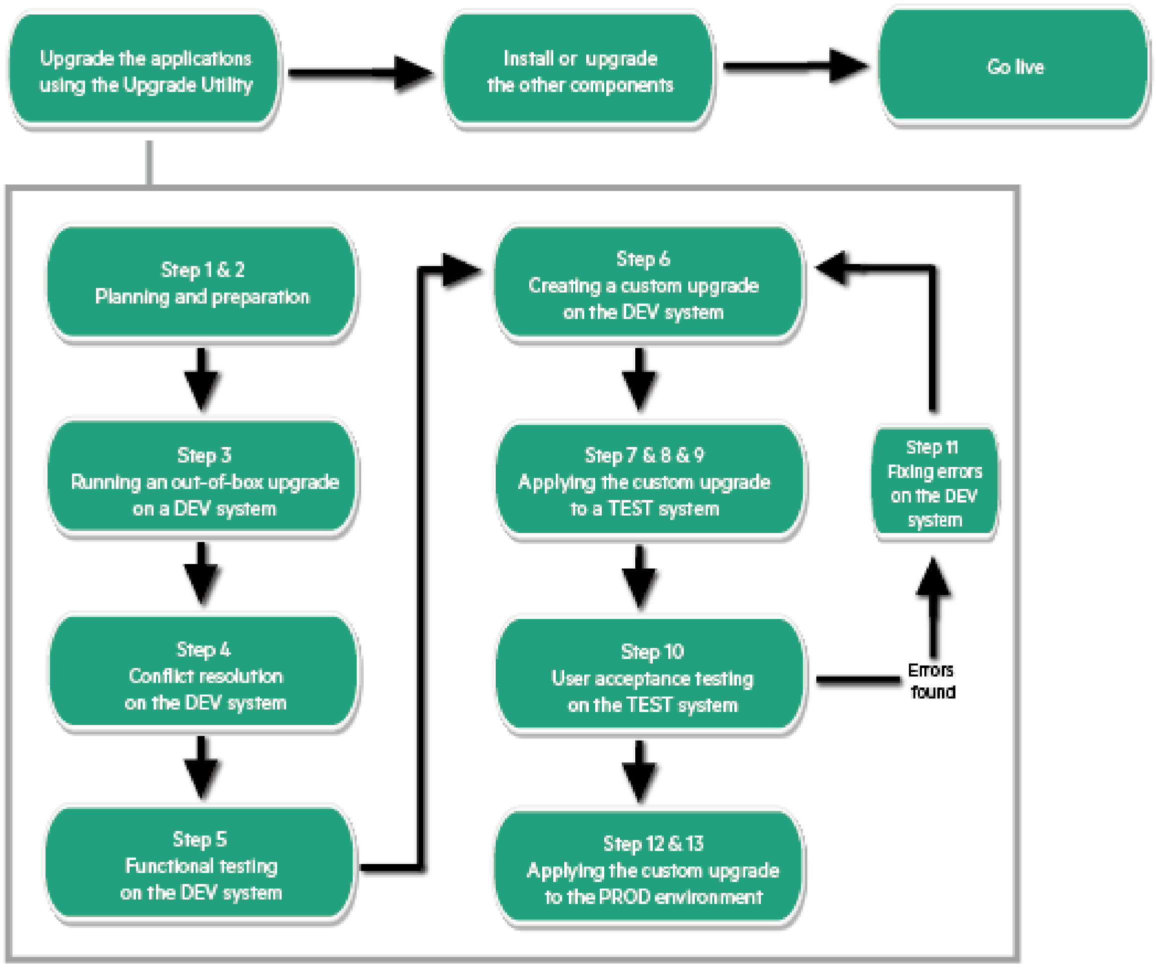
Note:
- If your applications version is earlier than 9.31, you must firstly upgrade the applications to version 9.31 before you can further upgrade to Service Manager 9.50 Hybrid.
- If your Service Manager system has Process Designer Content Pack 9.30.1 (PDCP2) installed, you must firstly upgrade to PDCP3 before you can further upgrade to Service Manager 9.50 Hybrid.

Recommended upgrade order from SM7.1x/9.2x

1. Install the Service Manager 9.50 server.
2. Install the Service Manager 9.50 Windows and web clients.
3. Upgrade to the Service Manager 9.41 applications using Service Manager 9.41 Upgrade Utility.
4. Upgrade to the Service Manager 9.50 applications using Service Manager 9.50 Upgrade Utility.
5. Install or upgrade other components.

Process of a major upgrade

The following diagram illustrates the process of a major upgrade.


Upgrade the applications using the Upgrade Utility
Install or upgrade the other components
Go live
Step 1 & 2 Planning and preparation
Step 3 Running an out-of-box upgrade on a DEV system
Step 4 Conflict resolution on the DEV system
Step 5 Functional testing on the DEV system
Step 6 Creating a custom upgrade on the DEV system
Step 7 & 8 & 9 Applying the custom upgrade to a TEST system
Step 10 User acceptance testing on the TEST system
Step 12 & 13 Applying the custom upgrade to the PROD environment
Step 11 Fixing errors on the DEV system
Errors found

The high-level steps of the applications major upgrade process are explained as follows. For more detailed information of each step, see "Applications upgrade overview" on page 46.

Step 1. Create a development system by duplicating your production system.

You will run an out-of-box upgrade on this system, and then fix errors and resolve conflicts. Make sure the development system meets the requirements in the support matrix of the target upgrade version, and copy your production data to the development system.

Step 2. Prepare the development system for an upgrade.

Prepare the data and binaries (server and client) of the development system for an upgrade.

1. Purge and archive unnecessary data to optimize upgrade performance.

   Purge and archive unnecessary data in the following files (which may contain large amounts of data): mail, msglog, syslog, spool, eventin, eventout, schedule, sysperform, and devaudit (on a production system, development auditing should be turned off).

   For information about purging and archiving records, see the Service Manager Help Center.

2. Meet database requirements.

   a. Keep the table space name same as the production system for an Oracle database.

   b. Convert all column names to uppercase for a case-sensitive database.

   c. Remove some indexes and constraints.

3. Upgrade your Service Manager server and client to the latest version.

4. Update the server configuration files (sm.cfg and sm.ini).

   Before running the Upgrade Utility, you need to make changes to the Service Manager configuration files before running the Upgrade Utility. Record all changes that you have made so that you can revert them to the original status after the upgrade.

Step 3. Run an out-of-box upgrade on the development system.

1. Check the size of the applications upgrade data, and compare it with the out-of-box upgrade package to make sure the downloaded data has no problems.

2. Load applications upgrade file preupg.bin from the out-of-box upgrade package.

3. Log off and then log back in.

4. Apply the out-of-box upgrade package to the development system.

5. Log out, restart the server, and then log back in.

Step 4. Conflict resolution on the development system.

1. Fix all exceptions in the except.log file, and verify your fixes.

2. Resolve all conflicts in the upgraderesults table, and verify your conflict resolution results.

3. Perform additional manual tasks.

Step 5. Functional testing on the development system.

Return the development system to normal operation, and test the upgraded development system.

Step 6. Create a custom upgrade package in the development system.

Now you have resolved all the exceptions and conflicts. It is necessary to package all the reconciled objects together into a "custom upgrade" so that they can be moved into the production environment automatically.

Step 7. Create a test system by duplicating your production system.

You need to apply the newly-created custom upgrade to your test system for user acceptance testing. Make sure the test system meets the requirements in the support matrix of the target upgrade version, and then copy your production data to the test system.

Step 8. Prepare the test system for an upgrade.

Follow the same steps for the development system, as described in step 2.

Step 9. Apply the custom upgrade package to the test system.

1. Upgrade the Service Manager server and client to the latest version for your test system.
2. Load applications upgrade file preupg.bin from the custom upgrade package.
3. Log off and then log back in.
4. Apply the custom upgrade package to the test system.
5. Fix all exceptions in the except.log file, and verify your fixes.
6. Resolve all "Error" results in the upgraderesults table, and verify your resolution results.

Step 10. User acceptance testing on the test system.

Return the test system to normal operation, and test the custom upgrade.

Test all features that your users will access. Pay particular attention to areas that were modified on your system.

Step 11: (Optional) Fixing errors on the development system.

If there are errors found in the test system, fix them in the development system and then go to *Step 6*.

Step 12. Prepare the production system for an upgrade.

Follow the same steps for the development system, as described in step 2.

Step 13. Apply the customer upgrade package to your production system.

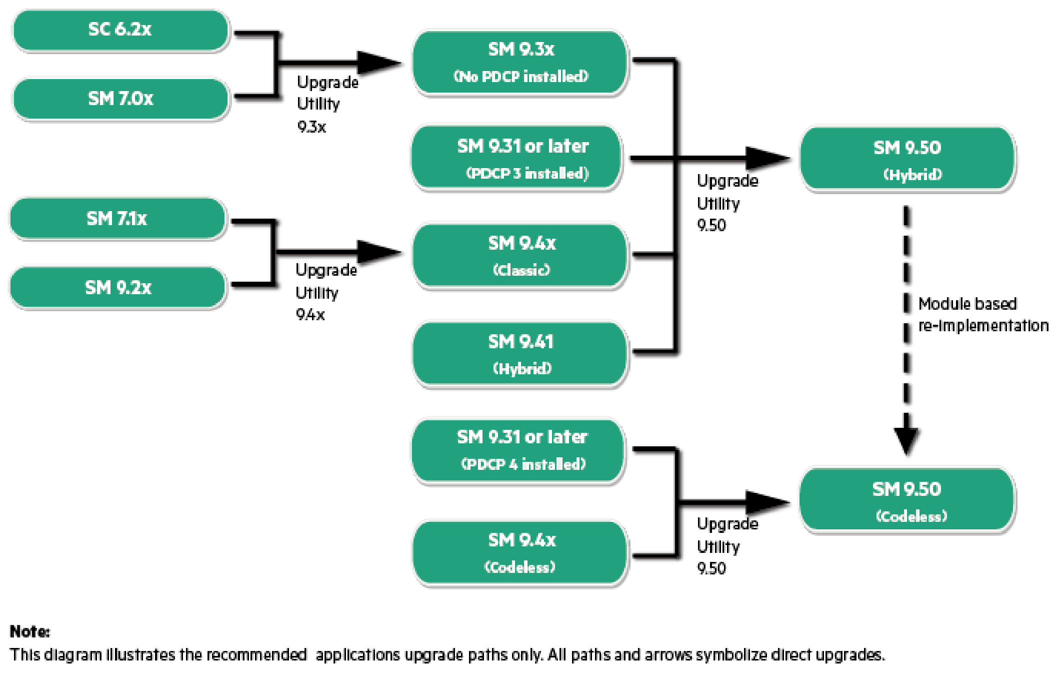
Make sure you have thoroughly tested the custom upgrade, before you apply the newly-created custom upgrade to your production system. This process is identical to the one you followed when applying your upgrade to your test system.

> **Note:** The production system should not be available to users while you are applying the custom upgrade.

Applications upgrade overview

The following sections describe how to upgrade your HPE Service Manager applications to 9.50 by using the HPE Service Manager 9.50 Upgrade Utility.

The following diagram summarizes the recommended upgrade paths. The paths and arrows symbolize direct upgrades.


SC 6.2x
SM 7.0x
Upgrade Utility 9.3x
SM 7.1x
SM 9.2x
Upgrade Utility 9.4x
SM 9.3x (No PDCP installed)
SM 9.31 or later (PDCP 3 installed)
SM 9.4x (Classic)
SM 9.41 (Hybrid)
Upgrade Utility 9.50
SM 9.50 (Hybrid)
Module based re-implementation
SM 9.31 or later (PDCP 4 installed)
SM 9.4x (Codeless)
Upgrade Utility 9.50
SM 9.50 (Codeless)
Note:
This diagram illustrates the recommended applications upgrade paths only. All paths and arrows symbolize direct upgrades.

Before you begin an upgrade

Before you begin an upgrade, ensure that you:

- Read through the *Upgrade Guide* to familiarize yourself with the upgrade process and all of the upgrade requirements.
- Are an experienced HPE Administrator who is familiar with HPE Service Manager.

If you do not have the administrative experience necessary to manage the upgrade, you may need assistance from your local applications developers and database administrators. You can also contact HPE Service Manager Customer Support for help with troubleshooting upgrade errors. For additional information and support, contact your HPE sales representative.

Applications upgrade

You can upgrade your existing HPE Service Manager applications to version 9.50 applications using the Upgrade Utility and resolving the differences between the two versions.

What are applications?

Applications are the Service Manager modules and their related configuration files. For example, Incident Management and Change Management are Service Manager applications.

New features that require an applications upgrade

Some new features provided by the release of Service Manager 9.50 require an applications upgrade. The following new features provided by the release of Service Manager 9.50 require an applications upgrade:

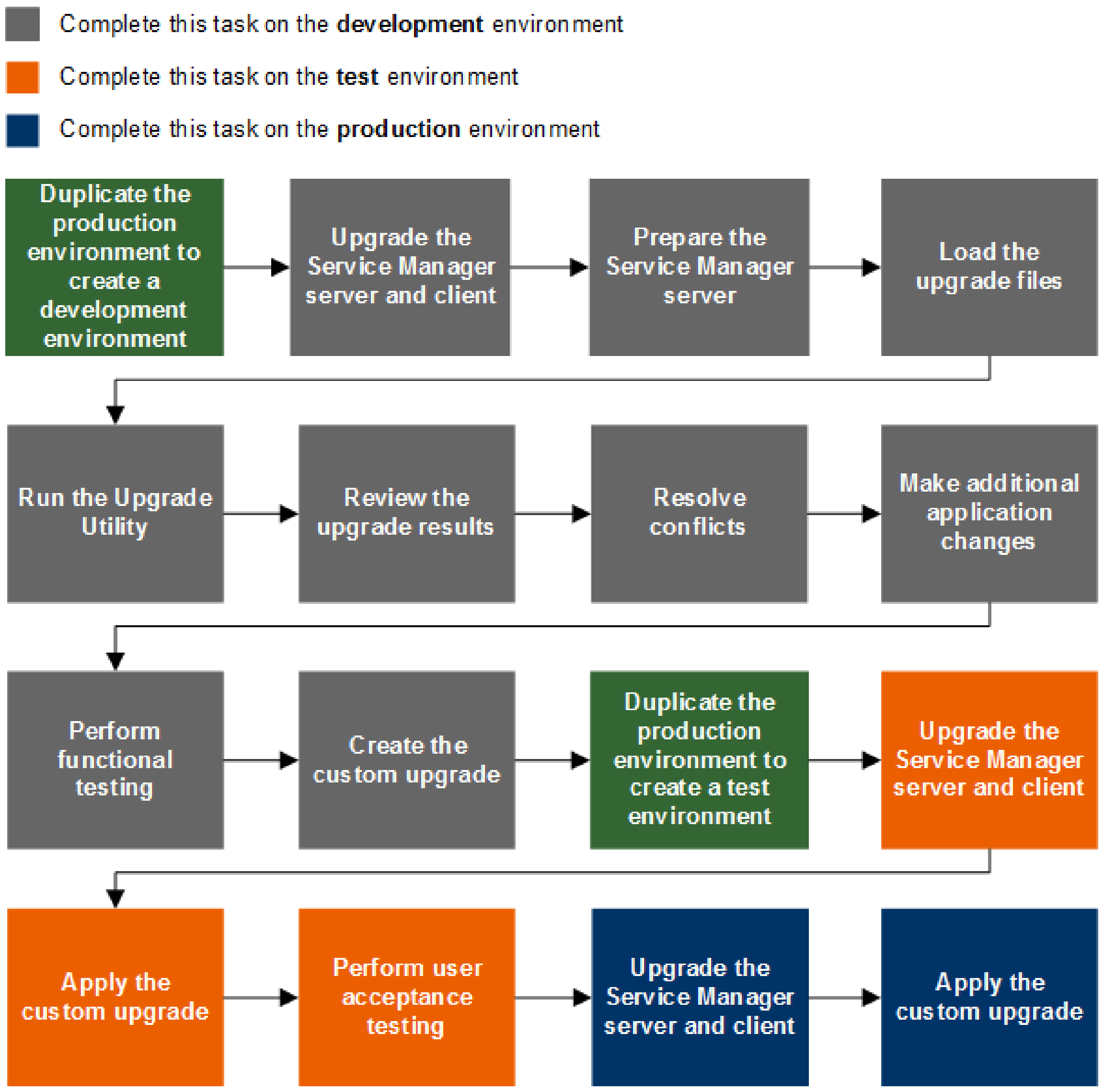
- Enhanced Service Desk, Incident Management, Problem Management, Change Management, Request Fulfillment Management, Service Catalog Management (provided since Service Manager 9.50), and Service Level Management based on Process Designer (provided since Service Manager 9.40)
- Process Designer framework
- Smart Analytics (provided since Service Manager 9.40)
- Mobile Applications
- Service Request Catalog (SRC)
- Case exchange
- Smart email
- Service Portal (provided since Service Manager 9.50)

- Simplified interaction (provided since Service Manager 9.41)
- Logical Name (provided since Service Manager 9.41)
- Smart search (provided since Service Manager 9.41)
- Service Manager Collaboration (provided since Service Manager 9.41)
- Accessibility for the embedded Service Manager Calendar
- Service Manager Reports (provided since Service Manager 9.40)
- Service Manager Survey (provided since Service Manager 9.40)
- Entity Relationship Diagram (ERD and Data integrity check) (provided since Service Manager 9.40)
- The Primary Key and `Not Null` constraints (provided since Service Manager 9.32)

Applications upgrade lifecycle

You can use three environments to complete the upgrade process: your current production environment, a new development environment, and a new test environment. You will duplicate your production environment to create the development and test environments. Run the out-of-box upgrade and create the custom upgrade on the development environment, and then run and test the custom upgrade on the test environment before you apply it to your production environment. The figure below provides an overview of the steps in the upgrade process.

The following flow chart illustrates the lifecycle of a typical upgrade of HPE Service Manager applications.


Complete this task on the **development** environment
Complete this task on the **test** environment
Complete this task on the **production** environment
Duplicate the production environment to create a development environment
Upgrade the Service Manager server and client
Prepare the Service Manager server
Load the upgrade files
Run the Upgrade Utility
Review the upgrade results
Resolve conflicts
Make additional application changes
Perform functional testing
Create the custom upgrade
Duplicate the production environment to create a test environment
Upgrade the Service Manager server and client
Apply the custom upgrade
Perform user acceptance testing
Upgrade the Service Manager server and client
Apply the custom upgrade

Upgrade phases and sub-phases

The following table describes the phases and sub-phases in the entire applications upgrade lifecycle. These sub-phases are logged in the upgrade log files during the upgrade. When an error occurs, the log files can help you find out during which phase and sub-phase the error occurs.

Phase	Sub-phases
Planning and preparation	• Load Transfer
Running an out-of-box upgrade	• Pre Upgrade Action Check

Phase	Sub-phases
	• Pre Upgrade Action Update • Pre Upgrade Action Purge • Pre Upgrade Action • Load Upgrade File • Upgrade Dbdicts • Load Upgrading Data • Upgrade Data • Post Upgrade Action • Post Upgrade Action Prior to SM940 • Post Upgrade Action Prior to SM950 • Post Upgrade Action Auto Merge • Post Upgrade Action Purge • Post Upgrade Action Update • Post Upgrade Action Notification • Post Upgrade Action Restore
Creating a custom upgrade	• Pre Create Action Check • Build Signatures • Build Distribution • Export Data • Transfer Data
Applying the custom upgrade	• Pre Upgrade Action Check • Pre Upgrade Action Update • Pre Upgrade Action Purge • Pre Upgrade Action • Load Upgrade File • Upgrade Dbdicts • Load Upgrading Data • Upgrade Data • Post Upgrade Action • Post Upgrade Action Prior to

Phase	Sub-phases
	SM940 • Post Upgrade Action Prior to SM950 • Post Upgrade Action Purge • Post Upgrade Action Update • Post Upgrade Action Notification • Post Upgrade Action Restore

How does customization affect the upgrade process?

The following explains how customization affects the upgrade process.

Conflicts

Object changes: The Upgrade Utility compares the objects in your database with their out-of-box versions and the corresponding objects provided from the upgrade package. The Upgrade Utility compares objects by their signatures. Each data record in Service Manager has a unique signature, which changes once that data record is updated. When processing object changes, the Upgrade Utility behaves as described in the following tables.

Changed object

Object in DB has been tailored?	OOB version of object matches upgrade package?	Out-of-box upgrade	Custom upgrade
Yes	No	The Upgrade Utility tries to merge the object from the upgrade package with your tailored object. If the merge is successful, the Upgrade Utility marks the new merged object as "Auto Merged." If the merge fails, the Upgrade Utility marks the object as	The Upgrade Utility marks the object from the upgrade package as “Forced,” and then copies the object in its revision.

Changed object, continued

Object in DB has been tailored?	OOB version of object matches upgrade package?	Out-of-box upgrade	Custom upgrade
		"Renamed." In both cases, the Upgrade Utility prefixes the object from the upgrade package with "NEW950," and then copies the object to your database and prefixes that object as "PRE<*version_number*>". For example, an object from applications version 9.31.0022 PDCP4 would be prefixed with "PRE9.31.0022_PDCP4".	
Yes	Yes	The Upgrade Utility keeps your local version, even if your version has been tailored and marks your local version as “Kept Customer”.	The Upgrade Utility marks your local version object as “Already Current.”
No	No	The Upgrade Utility overwrites the object in your database with the object from the upgrade package, and marks the object as “Upgraded.”	The Upgrade Utility marks the object from the upgrade package as “Forced,” and then copies the object in its revision.
No	Yes	The Upgrade Utility marks the object as "Already Current" regardless of whether this is the first upgrade or a custom upgrade.	

Added object

Object is added in ...	Behavior	Note
DB	The Upgrade Utility always keeps your local version and does nothing else.	The object in your database does not have a corresponding object from the upgrade package.
Upgrade package	The Upgrade Utility adds the object to your database and marks the object as "Added" regardless of whether this the first upgrade or a custom upgrade.	The object from the upgrade package does not have a corresponding object from your database.

Dbdict changes: The Upgrade Utility automatically adds new dbdict and merges new fields to existing dbdicts. The Upgrade Utility does not delete any existing field. For field and key changes, check "Field mapping changes" and "Key changes."

Field mapping changes: Normally the Upgrade Utility applies field mapping changes automatically, but there may be some exceptions. For example, when a length change is required, the Upgrade Utility automatically expands the length mapping. However, if the field mapping is a LOB-type change, the Upgrade Utility will not the change the type mapping. For detailed exceptions, check SQL Field Compare results before the upgrade and the except.log file after the upgrade.

Key changes: Normally the Upgrade Utility applies key changes automatically, but there may be some exceptions. For example, the Upgrade Utility automatically adds a new key and updates the existing key of the pre-upgrade out-of-box version. However, if a Unique key has been tailored, the Upgrade Utility will not apply the key change. For detailed exceptions, check the SQL Unique Key Compare Results before the upgrade and the except.log file after the upgrade.

Customization during upgrade

If any tailoring changes are made to your production system, for example, by applying an applications "hot fix," after you have initiated the upgrade process, it is highly recommended that you apply those same changes to the development system that is being used for conflict resolution before you create the final custom upgrade package. Or, these tailoring changes may be lost after the custom upgrade has been applied to production, and any conflict resolution that needs to be done in a production environment may slow down the production upgrade.

Upgrade Utility contents

The following table lists the files that are included in the HPE Service Manager Upgrade Utility.

List of Upgrade Utility files

File	Contents
AppUpgVersion.txt	Contains Upgrade Utility version and build number information to help you identify which applications upgrade version you have available. For example: A version of "SM930-9.50.*00xx* v9.50 *00xx* Upgrade Build *00xx*" indicates the following: • The Upgrade Utility upgrades Service Manager 9.30 and later releases to Service Manager 9.50. • The Upgrade Utility version number is *9.50.00xx*. • The Upgrade Utility build number for this version is *00xx*.

List of Upgrade Utility files, continued

File	Contents
preupg.bin	Files that allow you to access the various features of the Upgrade Utility.
transfer.bin	Files that allow for the execution of the upgrade.
sqlupgrade.unl	Files that allow you to run SQL compare, a feature of the Upgrade Utility. **Note:** preupg.bin includes the files in sqlupgrade.unl. To run SQL Compare, you do not need to load sqlupgrade.unl again after you load preupg.bin.
upgrade.inf	Signature information for the upgrade objects.
upgrade.str	Database dictionaries to be upgraded.
upgrade.ver	Version stamp for this upgrade.
*.dta (in the data folder)	The data files for each table that needs to be upgraded. For example, upgradeactivityactions.dta and upgradeactivitytype.dta.
upgrade.mak	Signature definitions for the upgrade objects.
upgdbdct.dta	Temporary dbdicts needed for the SQL Compare process.
DeltaMigrationTool.unl	Files that allow you to run Delta Migration Tool.
.zip (in the 3waymerge\oob folder)	Each zip file includes the XML representation of the objects that have been signatured in the pre-upgrade out-of-box version for the Three-Way Merge tool. **Note:** Only four base versions are included, namely, 9.3.zip, 9.3PD.zip, 9.4.zip and 9.4PD.zip.

Planning an upgrade

Good planning allows your upgrade to run as smoothly and quickly as possible, and helps you to avoid retracing your steps. When preparing for your upgrade, you will need to consider how long each step will take and when users need to be logged off the system so that you can schedule each phase of your upgrade.

Step 1: Identify the upgrade resources

Make sure that you have access to the following resources

- Service Manager tools: The utilities you will use most during the upgrade process include Database Manager and Forms Designer.
- Documentation resources: See the Service Manager 9.50 Help Center for client/server installation instructions and Service Manager knowledge.
- HPE Software Support: The HPE Software Support web site has operating system and compatibility information, product documentation, and release notes.

Step 2: Meet the software requirements

Before you start your upgrade, make sure that you meet the following HPE Service Manager system requirements:

- Your RDBMS version, operating system, and client/server environment must meet all criteria listed in the Support Matrix for the target version. See the *HPE Service Manager Support Matrices* to review the Support Matrix.
- To directly upgrade your existing applications to Service Manager 9.50, your existing Service Manager applications release level must be one of the following:
  - HPE Service Manager 9.3x without Process Designer installed
  - HPE Service Manager 9.31 or later with Process Designer Content Pack 9.30.2 or 9.30.3 installed
  - HPE Service Manager 9.4x Classic, Codeless, or Hybrid
- The Service Manager server process (sm) must have read-write access to the database.

Backups

It is highly recommended, at a minimum, that you back up the database at the following strategic points in the upgrade lifecycle:

- After applying an upgrade
- After resolving conflicts

NFS-mounted partitions

Do not install either Service Manager or the Service Manager Upgrade Utility on an NFS-mounted remote partition. This can cause serious performance degradation. The performance of an NFS-mounted partition drops significantly if it reads data in many small pieces instead of one large chunk. Service Manager generates a lot of database read/write activity. An NFS-mounted partition is significantly slower than a local drive when running the Upgrade Utility process.

Step 3: Perform a system health check

A well-maintained production system is the easiest to upgrade. Before starting the upgrade process, perform all regular maintenance on your production system. If necessary, contact HPE Customer Support for recommended actions. Suspend all customization activity on the production system.

Step 4: Create development and test environments

Plan to have at least two copies of your existing production environment:

- A development system that mirrors your current production environment. Use the development system to run the out-of-box upgrade and build a custom upgrade.
- A test system that mirrors your current production environment. Apply the custom upgrade on the test system and verify it there.

Step 5: Develop an upgrade strategy

In standard HPE Service Manager terminology:

- *Customization* refers to changes to RAD applications.
- *Tailoring* refers to changes made by using Service Manager tailoring tools, such as Forms Designer and Format Control.
- *Configuration* refers to local settings (for example, in your environment records and the system information record).

The upgrade process affects different parts of the Service Manager system. Besides upgrading the standard Service Manager applications, an upgrade affects the RDBMS where Service Manager is running and any customized files or RAD applications. For more information, see "How does customization affect the upgrade process?" on page 51.

Tailored systems

A list of tailored files can help you resolve differences quickly between your existing files and new files. You can also use the SQL Compare utility to determine how dbdict files differ.

To list all upgrade files, click **Tailoring** > **Differential Upgrade** > **Patch Records**. Select **SM95** in the drop-down list and then press Enter.

RDBMS-mapped systems

Because Service Manager tables (data files) must be mapped to an RDBMS, you must choose one of the following options before beginning the upgrade:

- Allow the Upgrade Utility to modify your RDBMS tables for you.
- Use SQL Compare utility to generate dbdict difference report and manually update the RDBMS databases before beginning the upgrade process.

The upgrade can affect certain mappings and tables. Contact your database administrator for assistance and to discuss the impact on the RDBMS.

Localized systems

You can upgrade a localized system with the Upgrade Utility. Before you begin to upgrade a localized system, ensure that you have the correct language pack for the language to which you will be upgrading. For more information and instructions on how to install the language pack, refer to *Service Manager Help Center* > *Install language packs*. The Upgrade Utility detects the presence of a localized system and runs the upgrade just as it would for an English system. You will have to make any customization and tailoring changes, based on the requirements described in this document for each of your system configurations.

Customized RAD applications and ScriptLibrary records

A list of customized RAD applications and ScriptLibrary records and the extent of the customization is useful. If it is not available, the programmer who made the changes may be able to supply information. Or, you may need to run a comparison between the existing applications or script and the new version.

Step 6: Meet database requirements

Before upgrading your system, verify that your system is pointing to the correct database.

- "Convert all tables and fields from lowercase to uppercase" below

Note: For Oracle users, you must have a granted role that includes “connect” and “resource” along with a granted system privilege of “select any dictionary” as a minimum in order to avoid errors generated by Oracle.

Convert all tables and fields from lowercase to uppercase

HPE Service Manager does not generate lowercase table names or field names. Therefore, if your database is case sensitive, you must convert all tables and fields from lowercase to uppercase before you can upgrade the server and client or applications.

Upgrade tasks on the development environment

Step 1: Duplicate the production environment

To achieve the best results, develop and test the custom upgrade on a system that resembles your production environment as closely as possible.

To duplicate the production environment, follow these steps:

1. Identify a server to use for the development and test environments.
   - Linux: You can copy the files to a new location on your production machine.
   - Windows: You must create the development system on a different machine from your production system.
2. Ensure that adequate memory and disk space is available and accessible. Frequent backups are necessary.
3. Ensure that your development and test systems meet all upgrade requirements. For more information, see "Step 2: Meet the software requirements" on page 55.
   - Upgrade your RDBMS to a version compatible with HPE Service Manager 9.50. See the *HPE Service Manager 9.50 Support Matrix*.
   - Convert your RDBMS code page to Unicode. See your RDBMS vendor documentation.
4. Set up the environment of your development and test machines to resemble your production server as closely as possible. The operating system version and service pack level should match.
5. Copy your existing production system data onto your development system.

   HPE recommends you use the native RDBMS backup utilities to back up your data. Refer to your RDBMS documentation for backup instructions.

6. Install a Service Manager 9.50 run time environment on the duplicated system. Do not load the Service Manager 9.50 demonstration data files.
7. Install a Service Manager 9.50 client on the duplicated system.

Step 2: Upgrade the server and the client

Make sure that you have upgraded your server and the client to the latest version before you attempt to run an applications upgrade. This allows the upgrade utility to call the new functions in the latest server and client and to take full advantage of all of the applications features following an upgrade.

> **Note:**
> - If you have deployed Knowledge Management, you must also upgrade the Knowledge Management Search Engine to the new version. For details, see *Service Manager Help Center > Upgrading from the K2 Search Engine* and "Upgrade the Solr Search Engine" on page 33.
> - You must upgrade the IDOL server if your are working with IDOL. For details, see "Upgrade Smart Analytics" on page 15.

To upgrade your applications by using this Upgrade Utility, you must first perform a server and client upgrade to 9.50 platform (server and client).

- Obtain the latest client from the installation package, and then follow the instructions in "Upgrade the Windows Client" on page 13.
- Obtain the latest server from the installation package, and then follow the instructions in "Upgrade the Service Manager Server" on page 9.

Step 3: Perform preparatory tasks

You must prepare your system before upgrade. 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 256000000.
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.

   > **Caution:** Failure to disable this option will cause the upgrade process to fail.

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. 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.
5. 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.
6. 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.
7. Update the Service Manager configuration files.

   The following tables list the changes that you need to make to the HPE Service Manager configuration files before running the Upgrade Utility. Record all changes that you have made so that you can revert them to the original status after the upgrade.

   Stop the Service Manager server, apply the required changes to the configuration files, and then restart the Service Manager server.

   sm.cfg

Parameter	Changes	Description
sm system.start	If this parameter exists, comment it out by changing it to:	Commenting this parameter out disables the background processes.

sm.cfg, continued

Parameter	Changes	Description
	`#sm system.start`	
sm -sync	Add this parameter to the end of the file if it does not exist yet.	This parameter starts the sync process, which identifies and releases locks owned by inactive processes and shared memory that is not in use.
sm -httpPort	If there is more than one instance of the **sm -httpPort** parameter, keep only one instance.	Each **sm -httpPort** parameter starts a Service Manager server process that can handle a certain number of client sessions (see the Service Manager Help Center documentation for more information). Keeping one process alive will be enough for the upgrade process.
Other parameters	Comment out all other parameters except the ones mentioned in this table.	Commenting out those parameters disables all the other Service Manager processes that are not required during an upgrade.

sm.ini

Parameter	Changes	Description
ir_disable:1	Add this parameter to the end of the file if it does not exist.	This parameter disables all IR keys on your existing Service Manager system. This will make the upgrade process run faster.
sessiontimeout:1200	Add this parameter to the end of the file if it does not exist. If this parameter already exists, update it to an appropriate value.	This parameter defines the number of minutes that the server waits for a client heartbeat signal before the server assumes that the client session has timed out and closes the connection. A value of 1200 sets the timeout to 20 hours (1200 minutes), a period that should be enough for an upgrade phase to complete in a typical scenario.
shared_ memory:256000000	Replace the default **shared_ memory:32000000** with **shared_ memory:256000000**.	This sets the shared memory size to 256 MB. However, if you have a large database, you may need to allocate more shared memory to accommodate the upgrade processing.
heartbeatinterval:120	Add this parameter to the end of the file	This parameter controls the client heartbeat frequency. If the server does not receive a

sm.ini, continued

Parameter	Changes	Description
	if it does not exist.	heartbeat from the client within the time-out limit as defined by the sessiontimeout parameter, the server terminates the client. All unsaved data is lost and the client must establish a new connection.

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

Step 4: Load the applications upgrade file

You must load the preupg.bin file into HPE Service Manager before you can use the Upgrade Utility.

> **Note:** If you are performing a custom upgrade on a test or production system, use the preupg.bin file included in your custom upgrade instead.

To load the applications upgrade file, follow these steps:

1. On the Service Manager server, create a folder (referred to as the Upgrade folder later in this document).

   > **Note:** Make sure that the Service Manager server process (sm) has write and execute privileges for this folder.
   >
   > If you are connecting to the Service Manager server from a client that is installed on a remote client computer, make sure that the folder is created on the Service Manager server instead of the client computer.

2. Extract the Service Manager applications Upgrade Utility files to the Upgrade folder. Make sure the extraction does not miss any Upgrade Utility files. Otherwise, the upgrade process will fail.
3. Log in to the Service Manager Windows client as a system administrator.

   > **Note:** Select **English** as the language when logging into the system for an upgrade.

4. Backup the `SYSTEM ADMINISTRATION` menu record.
5. Load the preupg.bin file using Service Manager Database Manager.
6. Log out of Service Manager and then log on again.

Step 5: Run the Upgrade Utility

Now that you have a functional environment, you are ready to run the Upgrade Utility. Follow the steps in this step to run the out-of-box upgrade against the data in your development environment and to run your custom upgrade against your test and production environments. You must perform these steps in a HPE Service Manager Windows client, instead of a web client.

> **Caution:** If the upgrade fails while the Upgrade Utility is running, fix possible issues and rerun the Upgrade Utility, and you should be able to resume the upgrade from the failure point; if the upgrade process cannot be resumed, you must restore the database to the last backup point and fix possible issues before you can rerun the Upgrade Utility.

The running of the Upgrade Utility involves the following three primary phases:

1. In the first phase, the Upgrade Utility guides you through several questions and collects information needed for the upgrade.
2. The second phase is the dbdict update phase, where the utility updates dbdicts.
3. The third phase is the data update phase, where the utility updates applications data.

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

1. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **Apply Upgrade** to launch the Upgrade Utility.
2. In the text box, type the fully qualified path to the folder that hosts transfer.bin, and then click **Next**.

   Example:

   Windows: `c:\temp\upgrade\`

   Linux: `/tmp/upgrade/`
3. The system displays a series of information for your verification, including Applications version upgrading from, Applications version upgrading to, Applications base version, Full path to the Upgrade Utility files and Language(s) to be upgraded. Verify these information and then click **Next** to continue.

   > **Note:** If this screen does not display the correct information, do not continue with the upgrade. Instead contact HPE Software Customer Support.

4. (Optional) The system prompts you to migrate legacy categories to the corresponding Process Designer-based modules. To move your selection between the legacy categories lists and the Process Designer-based modules lists, select a category and click the arrow button. Click **Next**.

   > **Note:** This screen appears only when you are upgrading from the following versions:
   >
   > - Service Manager 9.3x without Process Designer Content Pack installed
   > - Service Manager 9.3x with Process Designer Content Pack 9.30.2 (PDCP3) installed
   > - Service Manager 9.4x Classic
   >
   > Before you proceed, refer to "Appendix A: Additional manual migration tasks on Service Manager 9.50 Hybrid" on page 113 for more information.

5. The upgrade is now ready to start. Click **Next** to start the upgrade.
6. When you are asked whether you want to proceed, click **Yes**.
7. The Upgrade Utility displays the status when the upgrade is being processed. The loading process may take a long time.
8. 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 click **Finish**, you are automatically logged out.
9. Restart the server and log back in to the client.
10. Open the scversion table in the Database Manager, and verify that the Applications Version field is 9.50.00xx. If this field displays a value other than 9.50.00xx, check the log files to identify the issue that occurred.

Upgrade Utility logs and error messages

The Upgrade Utility creates a set of log files during the upgrade process. These files reside in the same directory as the upgrade files.

List of upgrade log files

Log file	Contents
detail.log	This file contains specific information about the upgrade, including the following: • Almost all information in upgrade.log. • Name of the file being purged. For example, “dbdict: upgradestatus is purged.” • Progress of a file loading. For example, “Adding record # 100 from table upgradeobjects”.

List of upgrade log files, continued

Log file	Contents
	• Changes made to fields during file processing. For example, “Increasing field length for incidentlib.company in kmquery dbdict from VARCHAR(40) to VARCHAR(70)” • Signature of a file and the action on it. For example, “Processing Format Record : cc.get.dependen, signature=(current=3843738292, oob=NONE, upgrade=3843738292), upgraderesult=current”
except.log	This file contains information about any exceptions reported by the upgrade, including the following: • Messages about data type mismatches that failed to be resolved, or database dictionaries failed to be upgraded. For example, "dbdict:FolderRights, field:delete, field type is logical -- expected to be:character" See "Data type mismatches" on page 75. • Messages about the unique key changes that failed to be resolved. For example, "dbdict:Todo, Unique Key is {"record.id", "itemType"} -- expected to be:{"record.id"}" If there are exceptions logged in this file, HPE recommends you to resolve them in the "Resolving exceptions and conflicts" phase.
upgrade.log	This file contains information about where the upgrade is at any point. This file contains only the main steps of the upgrade, including the following: • Starting and ending of each sub-phase. For example, “**** Start Phase [Pre Upgrade Action Update] ****” • Main activities during each sub phase. For example, “Purging upgrade files...” • Number of files to be processed. For example, “There are 608 dbdicts to be processed.” • Names of the files being processed. For example, "Processing dbdict, AdvFilter”
pdmigration.log	The log file, which is stored in the Service Manager log directory, includes the following information: • A list of Alert definition records that are created based on settings in your Classic 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 Classic probsummary and incident objects that are migrated to Process Designer-based objects • A list of updated process records

Step 6: View the upgrade results

While you are applying an out-of-box upgrade on the development system, the Upgrade Utility stores information regarding the upgrade result of each object. You can access this information in the Upgrade Utility through **View Upgrade Results and Merge Conflicts**.

To view the upgrade results, follow these steps:

1. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **View/Merge Results**.
2. If you are upgrading from SM9.3x with PDCP3 installed or non-PD environment or SM9.4x Classic, you can see a wizard page. Select **View Upgrade Results and Merge Conflicts**, and then click **Next**.

   Note: This screen appears only after Process Designer Hybrid migration.

3. In the **Result** drop-down list, select the type of results you want to search for.

   Example: Renamed

   When you select a result type from the drop-down list, the description of that result type appears under the drop-down list.

4. Click **Search**.
5. A list is returned that displays all the result records of the specified type.

   Note: Some types of results are only informational and do not require any follow-up action.

Description of upgrade results

The search criteria, search results, and a description of the applicable action for each result are described in the table below.

Field	Definition
Object Type	Enter the type of object you want to search for, or leave this field blank to return all object types. The object types you could search for include Applications Cluster, Object, Process, ScriptLibrary, displayoption, format, formatctrl, help, joindefs, link, scmessage, screlconfig, triggers, validity, and wizard.
Object Name	Enter the name of the object you want to search for, or leave this field blank to return objects with any name. The object name is typically the unique identifier in the database table specified for the object type. The unique identifier for some object types (for example, format) may contain concatenate values of multiple fields according to the key setting in the signaturemake signature definition table.
Result: Added	Select this option to search for new objects that the Upgrade Utility added to the system. These objects did not exist in your system before this upgrade. No further action is necessary for these objects.
Result: Already Current	Select this option to search for objects that were already the latest version. No further action is necessary for these objects.
Result: Auto Merged	Select this option to search for objects that the Upgrade Utility automatically merged by using your local version, the out-of-box version and the upgrade version of the objects. **Note:** The result occurs only after applying the first out-of-box upgrade; it no longer occurs after applying the custom upgrade. **Required Action**: If the object was not merged the way you expect, use the **Revert** or **Mass Revert** option from the options menu to revoke the auto-merge and merge the objects manually.
Result: Error	Select this option to search for objects that encountered an error while being updated by the Upgrade Utility. For more information about the error, review the sm.log and except.log files.

Field	Definition
	Required Action: Fix the cause of the error, or apply the upgrade again in a copy of your production system if it is needed.
Result: Forced	Select this option to search for objects that were not only tailored on your Service Manager system but also changed in the upgrade version. After the upgrade, your objects were automatically replaced with the objects in your custom upgrade package. The Upgrade Utility copied your object of the original version to its revision. **Note:** This result occurs only after applying the custom upgrade. No further action is necessary for these objects.
Result: Kept Customer	Select this option to search for objects that were tailored on your Service Manager system but not changed on the upgrade version. These objects were not changed. **Note:** The result occurs only after applying the first out-of-box upgrade; it no longer occurs after applying the custom upgrade. **Tip:** If an object is Applications Cluster and you want to use the upgrade version, you can select **Choose Upgrade**. The object <*object name*> is then renamed to PRE<*old version number*><*object name*> and the object NEW950<*object name*> is renamed to <*object name*>. No further action is necessary for these objects.
Result: Kept Customer Non-OOB	Select this option to search for objects that did not exist in the original version but were added on your Service Manager system, and also added on the upgrade version. **Note:** The result occurs only after applying the first out-of-box upgrade; it no longer occurs after applying the custom upgrade. **Tip:** If you want to use the upgrade version for the object, you can select **Choose Upgrade**. If the object is Application Cluster, the object <object name> is then renamed to PRE<old version number><object name> and the object NEW950<object name> is renamed to <object name>. If the object is not Application Cluster, the object NEW950<object name> is copied to replace <object name>. The result is then set to **Replaced**. **Required Action**: Choose one of the following solutions for each object with this result.

Field	Definition
	• Keep the old version — No further action is necessary. • Keep the new version — Select the object in the merge view and click **Copy all from left to right** on the toolbar. • Merge new and current versions — Determine which of the new features need be incorporated into your tailored object, and then modify the tailored object. When finished, delete the new object NEW941<object name> and the copied object PRE<old version number><object name>.
Result: Merged	Select this option to search for dbdict objects that were tailored on your Service Manager system, which Upgrade Utility has merged with the version in this upgrade. **Required Action**: Test these objects, and when satisfied change their result to **Reconciled**.
Result: Previously Reconciled	Select this option to search for objects that were tailored on your Service Manager system, that were marked as Reconciled during a previous upgrade or patch release, or where your object was not changed and the Upgrade Utility added a new object NEW950<object name>. **Note:** The result occurs only after applying the out-of-box upgrade; it no longer occurs after applying the custom upgrade. **Required Action**: Choose one of the following for each object with this result. • Keep the old version — No further action is necessary. • Keep the new version — Select the object in the merge view and click **Copy all from left to right** on the tool bar. • Merge new and old versions — Determine which of the new features should be incorporated into your tailored object, and then make the changes in your tailored object. When finished, delete the new object NEW950<object name> and the copied object PRE<old version number><object name>. **Tip:** If you want to use the upgrade version for the object, you can select **Choose Upgrade**. If the object is Application Cluster, the object <object name> is then renamed to PRE<old version number><object name> and the object NEW950<object name> is renamed to <object name>. If the object is not Application Cluster, the object NEW950<object name> is copied to replace <object name>. The result is then set to **Replaced**.
Result: Reconciled	Select this option to search for objects that you have already marked as Reconciled.

Field	Definition
	Note: The result occurs only after applying the out-of-box upgrade; it no longer occurs after applying the custom upgrade. No further action is necessary for these objects.
Result: Renamed	Select this option to search for objects that were not only tailored on your Service Manager system but also changed on the upgrade version. After upgrade, your tailored object was not changed, the Upgrade Utility added a new object NEW950<object name> and copied your tailored object as a backed up object PRE<old version number><object name>. **Note:** The result occurs only after applying the out-of-box upgrade; it no longer occurs after applying the custom upgrade. **Required Action**: Choose one of the following for each object with this result. • Keep the old version — No further action is necessary. • Keep the new version — Select the object in the merge view and click **Copy all from left to right** on the tool bar. • Merge new and old versions — Determine which of the new features should be incorporated into your tailored object, and then make the changes in your tailored object. When finished, delete the new object NEW950<object name> and the copied object PRE<old version number><object name>. **Tip:** If you want to use the upgrade version for the object, you can select **Choose Upgrade**. If the object is Application Cluster, the object <object name> is then renamed to PRE<old version number><object name> and the object NEW950<object name> is renamed to <object name>. If the object is not Application Cluster, the object NEW950<object name> is copied to replace <object name>. The result is then set to **Replaced**.
Result: Upgraded	Select this option to search for objects that were automatically replaced with the upgrade version objects. These are objects that were not tailored on your Service Manager system, but changed on the upgrade version. **Note:** The result occurs only after applying the out-of-box upgrade; it no longer occurs after applying the custom upgrade. No further action is necessary for these objects.
Result: Replaced	Select this option to search for objects that were not only tailored on your Service Manager system but also changed on the upgrade version and automatically replaced with the upgrade version objects. These objects have 3 types: RAD Application, ScriptLibrary with HPE Propertary and the objects that are necessary for Process Designer functionality. After upgrade, your

Field	Definition
	tailored object was renamed to PRE<old version number><object name> and the Upgrade Utility added a new object <object name>. **Note:** The result occurs only after applying the out-of-box upgrade; it does not occur after applying the custom upgrade. No further action is necessary for these objects. **Tip:** If you do not want to replace the objects, you may select **Revert**. If the object is Application Cluster, the object <object name> is then renamed to NEW950<object name> and the object PRE<old version number><object name> is renamed to <object name>. If the object is not Application Cluster, the object PRE<old version number><object name> is copied to replace <object name>. The result is then set back to **Renamed**, **Previously Reconciled**, **Kept Customer** or **Kept Customer Non-OOB**.

See the following table for some examples about how the upgrade result types are marked for an existing record.

OOB record (9.30)	OOB record (9.33)	Customer record (9.33)	OOB record (9.5x)	Upgrade Utility check sequence	Upgrade Utility check condition	Conflict flag	Upgrade result type	Upgrade result value
111	112	112	112	1	Customer record (9.33) = OOB record (9.5x)	No	Already Current	112
111	112	112	113	2	Customer record (9.33) != OOB record (9.5x) and Customer record (9.33) = OOB record (9.33)	No	Upgraded	113
111	112	113	112	3	Customer record (9.33) != OOB record (9.5x) and Customer record (9.33) != OOB record (9.33) and OOB record (9.33) = OOB record (9.5x)	No	Kept Customer	113

OOB record (9.30)	OOB record (9.33)	Customer record (9.33)	OOB record (9.5x)	Upgrade Utility check sequence	Upgrade Utility check condition	Conflict flag	Upgrade result type	Upgrade result value
111	112	113	114	4	Customer record(9.33)!=OOB record(9.5x) and Customer record(9.33)!=OOB record(9.33) and OOB record(9.33)!=OOB record(9.5x)	Yes	Renamed	113
		113	112	5	OOB record (9.33) does not exist and Customer record (9.33) != OOB record (9.5x)	Yes	Kept Customer Non-OOB	113

Step 7: Manage the upgrade result data

To manage the upgrade result data more easily, do either of the following:

- Open the Upgrade Results list, click **More** or the More Actions icon, and click **Export to Excel** to manage the data in a Microsoft Excel document, or
- Open the Upgrade Results list, click **File** > **Print** > **List View** to print the list of records. (See the HPE Service Manager Help Center documentation for more information.)

After you complete an applications upgrade on your development system, you are ready to resolve the exceptions and conflicts that originate from tailoring on an upgraded object. Before resolving conflicts, HPE Service Manager features may not function as expected.

Step 8: Resolve exceptions

Exceptions are logged if the Upgrade Utility cannot add or update an object. After running an upgrade, you can identify exceptions by viewing error messages in the except.log or sm.log file. These exceptions are reported in the Upgrade Results list as "Error."

Data type mismatches

If the data type of a field in your dbdict does not match the data type of the like-named field defined in the dbdict provided by the upgrade package, the Upgrade Utility cannot merge these dbdicts. For example, if an existing dbdict has a scalar field and the Upgrade Utility attempts to add a structure field with the same name, this discrepancy prevents the dbdict from being updated.

To fix this issue, you can change the data type and the SQL type of the field, and use Complex Update to migrate existing data on that field to the target data type. Follow these steps to modify field data type by using the Dbdict Utility:

1. Navigate to **System Navigator** > **Tailoring** > **Database Dictionary**.
2. Type the dbdict name in the File Name field, and then click **Search**.
3. Double-click the field you want to change the data type, and then make updates as expected.
4. Save your changes.

The following is an example that shows the process of fixing a typical data type mismatch.

Note: The error messages in the except.log file identify each data type by an index number:

Index number	Data type
1	number
2	character
3	date/time
4	logical
8	array
9	structure
11	expression

The following table displays a list of data type mismatches that may appear in the except.log file for an Oracle database. Only some examples are listed for each category of data type mismatch. If you are using an MSSQL database, the actual error message may vary slightly. For example, the following list highlights the different errors in the different databases:

- Oracle Error : dbdict:ApprovalDef, field:appr.condition, SQL type is CHAR(1) -- expected to be:RAW(255)
- MSSQL Error: dbdict:ApprovalDef, field:appr.condition, SQL type is CHAR(1) -- expected to be:VARBINARY(255)

Error message in Oracle:	Solution:
dbdict:incidents, field:svc.options, SQL type is VARCHAR2(90) -- expected to be:BLOB dbdict:SMISTaskQueue, field:failedReason, SQL type is VARCHAR2(60) -- expected to be:BLOB dbdict:Workflow, field:description, SQL type is VARCHAR2(255) -- expected to be:BLOB	To fix these issues, change the SQL type to RAW (255) or BLOB by using the Dbdict utility. Additionally, you will need to set the “SQL RC” to true to allow the field to store RAD expressions. Note that the stored value of the field in the database is encoded by Service Manager.
dbdict:cm3eventack, field:number, field type is number -- expected to be:character	To fix this issue, change the field type to character by using the Dbdict utility.
dbdict:Rule, field:tablename, SQL type is VARCHAR2(60) -- expected to be:CLOB	Ignore the error message. No further action is necessary.
dbdict:svcItemCount, field:access.filter, SQL type is RAW(255) -- expected to be:CLOB	To fix these issues, change the SQL type to CLOB by using the Dbdict utility. Additionally, you need to

Error message in Oracle:	**Solution:**
	switch the value of "SQL RC" to update the type.
dbdict:eventin, field:evnumber, field type is number -- expected to be:character dbdict:eventout, field:evnumber, field type is number -- expected to be:character	These fields reside in the descriptor structure field of BLOB SQL type. To fix this issue, change the field type to character by using the Dbdict utility.
dbdict:svcCatLanguage, field:catalogid, field type is character -- expected to be:number	This field resides in the catalog structure field of BLOB SQL type. To fix this issue, change the field type to number by using the Dbdict utility.
dbdict:licenseinfo, field:id, field type is character -- expected to be:number	The licenseinfo table is used to track license information by Service Manager server. This issue should be ignored.
dbdict:svcCatalog, field:id.attach, field type is character -- expected to be:number	This id.attach field is an alias of id field in svcCatalog table. To fix the issue, change the field type to by using the Dbdict utility.
dbdict:FolderRights, field:close, field type is logical -- expected to be:character	The close field is an alias of delete field in FolderRights table. To fix the issue, change the field type to character by using the Dbdict utility.
dbdict:FolderRights, field:delete, field type is logical -- expected to be:character	This issue can be fixed by following the steps in the "Fixing the FolderRights delete field" on page 79 section.
dbdict:cm3r, field:assets, SQL type is CLOB -- expected to be: VARCHAR2(200) dbdict:cm3r, field:assets, field type:character, SQL type is CLOB -- expected to be:VARCHAR2(200), SQL table is m1 -- expected to be:a3	This kind of error message is generated because the Upgrade Utility does not re-map the field of array type from the main table to the alias table and the subsequent full-table copy may slow down the upgrade. Follow these steps to fix the issue: 1. Double-click the field of array type which contains this field in the related table, and then update the value in "SQL Table" to a<*number*> which is not used in the alias table. For example, a3. 2. Double-click the field of character type in the related table. Update the value in "SQL TYPE" to the expected value, and then update the value in "SQL Table" to the same a<*number*> which is not used in the alias table. For example, a3. 3. Click **OK** to save your changes.

Error message in Oracle:	Solution:
	Caution: If the related table contains more than 100,000 records, fixing the data type mismatches in the field may take more than one hour.
dbdict:inbox, Primary Key is inbox.id -- the primay key can not be added. Since SM doesn't support operation on table with primary key when applications version is older than 9.32, then add inbox.id as unique key.	To fix this issue, change the unique key to primary key by using the Dbdict utility.
Update field type for record.number in kmreindex dbdict from number to character -- You need to update the relevant data in dbdict kmreindex manually.	The field type is updated from number to character. You need to check whether the data values are updated and then update the data values to the correct ones.
Failed to add Primary key: {"mailno"} to table mail. You must add it manually.	Check whether the primary key is added to dbdict mail. If not, please check whether the values of this field are filled for all records in this table. You need to manually fill this field and add the primary key by using the Dbdict utility.

Note: Ignore the message if the child fields of an array of structure field have been mapped to the database columns. For example:

```
dbdict:computer, field:video, SQL type is EMPTY -- expected to be:BLOB
dbdict:computer, field:vid.card.id, SQL type is VARCHAR2(30) -- expected to
be:EMPTY
dbdict:computer, field:vid.card.bios, SQL type is VARCHAR2(30) -- expected to
be:EMPTY
dbdict:computer, field:vid.manufacturer, SQL type is VARCHAR2(30) -- expected to
be:EMPTY
dbdict:computer, field:vid.model, SQL type is VARCHAR2(30) -- expected to
be:EMPTY
dbdict:computer, field:vid.chip.type, SQL type is VARCHAR2(30) -- expected to
be:EMPTY
dbdict:computer, field:vid.memory, SQL type is NUMBER -- expected to be:EMPTY
dbdict:computer, field:vid.resolution, SQL type is VARCHAR2(30) -- expected to
be:EMPTY
dbdict:computer, field:vid.resolution.x, SQL type is VARCHAR2(30) -- expected to
be:EMPTY
dbdict:computer, field:vid.resolution.y, SQL type is VARCHAR2(30) -- expected to
be:EMPTY
```

Fixing the FolderRights delete field

Example: The dbdict for the FolderRights table has a delete field with the "logical" data type. The Upgrade Utility tries to update the delete field with the "character" data type, which has possible values of "always," "never," "workgroup," and "assigned."

> **Note:**
> The field type of delete field in FolderRights dbdict may be updated to character. There is another information in except.log:
>
> ```
> Update field type for delete in FolderRights dbdict from logical to character --
> You need to update the relevant data in dbdict FolderRights manually.
> ```
>
> The value of the field also need to be updated following the steps manually.

1. In the dbdict for the FolderRights table, add a field named delete.tmp with a data type of character, and update the dbdict.
2. Log out of the system and log back in.
3. Make sure that the Complex Update feature is enabled for the FolderRights table.

   > **Note:** To enable the Complex Update feature, open the FolderRights Format Control record, go to **Privileges** and open the **Advanced** tab, and then update the values in the Simple Mass Update field and in the Complex Mass Update field to `true`.

4. In the Database Manager, search for all records in the FolderRights table.
5. From the More Actions menu, click **Mass Update**.
6. When you are asked whether you want to update all records, click **Yes**.
7. Click **Complex Update** on the toolbar.
8. In the statements area under **Instructions for action on EACH RECORD**, add statements using standard RAD expressions to migrate data from the delete field to the delete.tmp field.

   Example:

   ```
   if delete in $file=true then delete.tmp in $file="always" else delete.tmp in
   $file="never"
   ```

9. In the dbdict for the FolderRights table, edit the **delete** field and add the **Type** (character) and **SQL Type** (same as the SQL Type automatically assigned for delete.tmp).
10. Log out of the system and log back in.

11. In the Database Manager, search for all records in the FolderRights table.
12. From the More Actions menu, click **Mass Update**.
13. Click **Complex Update** on the toolbar.
14. In the statements area under **Instructions for action on EACH RECORD**, add statements using standard RAD expressions to migrate data from the delete.tmp field to the delete field and empty the delete.tmp field.

    Example:

    ```
    delete in $file=delete.tmp in $file; delete.tmp in $file=NULL
    ```

15. In the dbdict for the FolderRights table, follow these steps to remove the delete.tmp field:
    a. In the Database Manager, search for all records in the dbdict table.
    b. Select the dbdict format.
    c. Type `FolderRights` in the **Name** field, and then click **Search**.
    d. Locate the row for the delete.tmp field. Remove the values in the Name, Type, Index, and Level fields respectively.
    e. Click **Save** and **OK**.
16. Log out of the system and log back in.
17. Test the change by updating records in the FolderRights table and populating the **delete** field with "always," "never," "workgroup," or "assigned."

Add new object record failure

All the upgraded objects are saved in the upgradeobjects table, and you need to add some objects from this table to the current upgradeobjects table. If the target table contains more than one unique type key, some errors may occur when adding new objects during the upgrade process. Refer to the following error message as an example.

```
2014-03-20 14:04:29 file:inbox, updated the field inbox.id value from 10000313 to
14042954310000313
```

The value of the inbox.id field is 10000313 in this example, which is used in the current upgradeobjects table. If the Upgrade Utility adds an inbox record that contains the same value for this field, the current value of the inbox.id field is automatically prefixed with a timestamp to avoid duplication errors. The timestamp is composed of hour (two bits), minute (two bits), second (two bits) and millisecond (three bits). In this example, the timestamp is 140429543.

After upgrading, you can update the temporary value of the inbox.id field as necessary.

Handle key change failure

If the Upgrade Utility fails to apply certain key changes, error information is logged into the except.log file. Review the log file and make appropriate operations:

Error message 1: Failed to add <*key_type*> key: <*field_name*> to table <*table_name*>. You must add it manually.

Follow these steps to handle this error:

1. Type `dbdict` in the Service Manager command line, and press **Enter**.
2. In the **File Name** field, type the table name indicated by the error message, and click **Search**.
3. Click the **Keys** tab.
4. Position the mouse point in the key name part of an empty key structure, and click **New Field/Key**.
5. Select the appropriate key type from the combo list, and add the appropriate field to the key, as indicated by the error message.
6. Click **Add**, and click **Yes** to confirm.
7. Click the **Keys** tab, and click **OK** to save the change.
8. Continue to follow *steps 1* through *step 7* for each key that failed to be added.

Error message 2: Failed to update <*key_type*> key: <*old_field_name*> to <*new_field_name*> in table <*table_name*>. You must update it manually.

Follow these steps to handle this error:

1. Type `dbdict` in the Service Manager command line, and press **Enter**.
2. In the **File Name** field, type the table name indicated by the error message, and click **Search**.
3. Click the **Keys** tab.
4. From the key list, select the key name that is indicated by the error message, and click **Edit Field/Key**.
5. Update the fields in the key according to the field names indicated by the error message.
6. Click **OK**, and click **Yes** to confirm.

7. Click the **Keys** tab, and click **OK** to save the change.
8. Continue to follow *steps 1* through *step 7* for each key that failed to be updated.

Error message 3: Failed to remove <*key_type*> key: <*field_name*> from table <*table_name*>. You must remove it manually.

Follow these steps to handle this error:

1. Type `dbdict` in the Service Manager command line, and press **Enter**.
2. In the **File Name** field, type the table name indicated by the error message, and click **Search**.
3. Click the **Keys** tab.
4. From the key list, select the key name that is indicated by the error message, and click **Edit Field/Key**.
5. Click **Delete**, and click **Yes** to confirm.
6. Click the **Keys** tab, and click **OK** to save the change.
7. Continue to follow *steps 1* through *step 6* for each key that failed to be removed.

In addition, you may encounter the following Unique Key errors:

Error: `dbdict:Approval, Unique Key is {"unique.key", "file.name", "name"} -- expected to be:{"unique.key", "file.name", "name", "component"}`

Error: `dbdict:ApprovalLog, Unique Key is {"counter", "file.name", "unique.key"} -- expected to be:{"counter", "file.name", "unique.key", "component"}`

Unexpected errors

If the except.log file or the Upgrade Results list reports any errors other than data type mismatches, review the sm.log file for more information, and if needed, contact Customer Support for assistance.

Step 9: Resolve conflicts

In this step, you are going to resolve the conflicts between your tailored objects and the new objects provided in the upgrade package.

Standard conflict resolution process

The standard process of resolving conflicts are listed as follows:

1. To view the conflicts, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **View/Merge Results**.
2. (Optional) Select **View Upgrade Results and Merge Conflicts**, and then click **Next**.

   Note: This screen appears only after Process Designer Hybrid migration.

3. Search for records with a status of "Renamed."
4. For each renamed object, you can choose one of the following options.
   - **Option 1:** Use your customized object instead of the new object.

     In this case, delete the new object that is prefixed with NEW950 and the copied object that is prefixed with PRE<*version_number*>.

   - **Option 2:** Use the new object instead of your customized object.

     In this case, delete your customized object and the copied object that is prefixed with PRE<*version_number*> and then rename the new object by removing the NEW950 prefix, or you can select **Choose Upgrade** to replace the object with the new object that is prefixed with NEW950.

     The Mass Choose Upgrade feature can help you replace multiple customized objects with new objects that are prefixed with NEW950. For more information, see the "Using the Mass Choose Upgrade feature" on page 95.

   - **Option 3:** Merge the changes shipped with the new object into your customized object.

     In this case, find out what changes the new object includes, manually apply those changes to your customized object, and then delete the new object that is prefixed with NEW950 and the copied object that is prefixed with PRE<*version_number*>.

   Note: The Merge tool, Auto-Merge and Revert options, and third-party three-way compare and merge tools can assist you in comparing the objects and merging the code during the out-of-box upgrade. See "Using the Merge tool" on page 87 and "Using the Auto Merge and Revert options" on page 95 for more information.

5. After resolving each conflict, you must mark the object as "Reconciled." Marking an object as reconciled will remove it from the current conflict list and provides the version with which the

object was last reconciled.

> **Note:** The Mark as Reconciled feature can assist you to mark the object as Reconciled. For more information, see "Using the Mark as Reconciled feature" on page 96.

For more specific guidelines and examples for conflict resolution, see white paper *Conflict Resolution for Upgrade to Service Manager (SM) 9.3x*.

Display components

The Upgrade Results list also includes conflicts in display components. You can follow the standard conflict resolution process to resolve these conflicts. However, you must pay special attention to following types of display components:

- displayscreen records
- displayoption records
- displayevent records

Display screen records

Display screens are individual records identified by a unique screen ID. The displayscreen records define the attributes of a screen and provide access to the individual records for options and events. A display screen is not the same as a form.

> **Caution:** There are triggers attached to the displayscreen file. Changes to the records in this file affect their associated display options and events.

If any displayscreen objects appear in your upgrade results list, perform conflict resolution on those records by following the standard conflict resolution process.

Display options and display events

The Upgrade Utility upgrades the display components in the same manner as all other components. The displayoption table and the displayevent table have a unique identifier, stored in the ID field. The upgrade process assigns an ID to every display option following the pattern: **<screen id>_<action>_<number>**, where the screen ID and action are from the display option (or event), and the number is an optional field added when multiple options have the same screen ID and action.

If options that have been added to your system have the same action as others in the same display screen, the upgrade process assigns a <number> in the order of the options' GUI option number.

If an added option was not the last option in terms of GUI option number, the upgrade process does not add the additional numbers in the ID field in the same order as they would have for an out-of-box system. The upgrade process renames the added option and any option after it (in GUI option order), it does not upgrade them automatically.

To ensure that this type of renaming does not happen in future upgrades, when performing conflict resolution on these options, use the ID of the renamed option, **NEW950<screen id>_<action>_<number>** and manually change the identifier of the added options. Rename all other options to match the ID of the renamed ones.

When renaming an option, use an identifier to specify that this is a customized option, added for your installation. For example, an ID might look like: “**apm.edit.problem_do nothing_ACME1**”.

This table gives an example of part of the display screen conflict resolution for **apm.edit.problem**.

Example conflict resolution for the apm.edit.problem display option

Screen ID	GUI Action	Upgrade Action	
300	do nothing	update	Name: apm.edit.problem_do_nothing_1 Result: This item was updated correctly. User Action: No action necessary.
400	do nothing	update	Name: apm.edit.problem_do_nothing_2 Result: This item was updated correctly. User Action: No action necessary.
450 This is an option you added.	do nothing	rename	Name: apm.edit.problem_do_nothing_3 Result: This item was renamed. It is your customized option. User Action: Rename this object to give it a unique new name, such as: apm.edit.problem_do_nothing_ACME1 Name: NEW950apm.edit.problem_do_nothing_3 Result: This is the new SM 9.50 option. User Action: Perform conflict resolution. To perform conflict resolution, open apm.edit.problem and look at the options. Compare this option with apm.edit.problem_do_nothing_3 and NEW950apm.edit.problem_do_nothing_3
500	do nothing	The upgrade	Result: This option does not appear in the reports.

Example conflict resolution for the apm.edit.problem display option, continued

Screen ID	GUI Action	Upgrade Action	
		ignores this option.	User Action: Perform conflict resolution. To perform conflict resolution, open apm.edit.problem and look at the options. Compare this option with apm.edit.problem_do_nothing_3 and NEW950apm.edit.problem_do_nothing_3

Using the Merge tool

For each object marked as "Renamed" in the Upgrade Results list, the Upgrade Utility generates XML objects for the three versions of the object: base, customer, and upgrade.

Version	Location	Description
base	*Upgrade* \3waymerge\work\base	An XML representation of every object that has been signatured in the pre-upgrade out-of-box version.
customer	*Upgrade* \3waymerge\work\customer	An XML representation of all objects that were tailored in the customer version and resulted in a conflict during the upgrade.
upgrade	*Upgrade* \3waymerge\work\upgrade	An XML representation of the object provided by the upgrade package of all objects that resulted in a conflict.

Each of the three folders described above contains a sub-folder for each signatured table. You can find the XML representations of the objects in the table within these sub-folders.

The built-in, Two-Way/Three-Way Merge tool allows you to examine the upgrade and customer versions of a record in a side-by-side view as well as the base, upgrade, and customer versions of a record in a three-way view. This will help you to determine which changes to include in the final record.

This tool assists the conflict resolution process in these two ways:

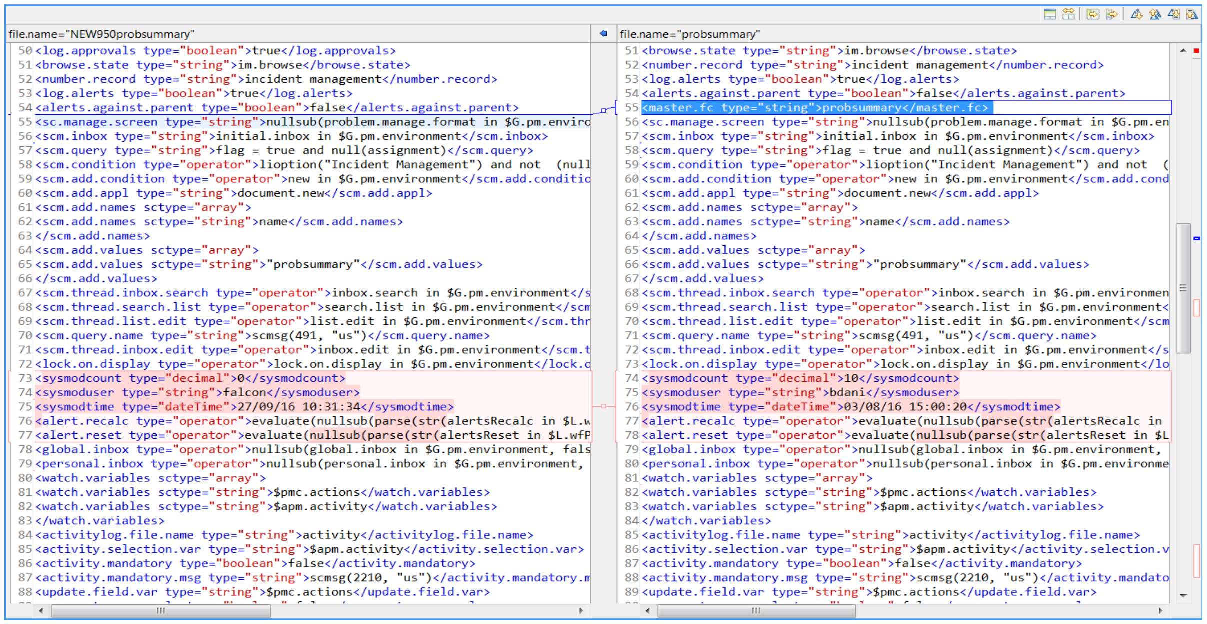
- It allows you to identify where changes are located before you can visually compare the objects and to make changes manually, such as in format records.
- It allows you to identify and merge changes directly between objects, such as ScriptLibrary records.

To use the Merge tool, follow these steps:

1. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **View/Merge Results**.
2. In the **Result** drop-down list, select **Renamed**.

   > **Note:** The Two-way/Three-way Merge tool is available only for “Renamed” records.

3. Click **Search**.
4. A list is returned that displays all the result records of the “Renamed” type. Select a record that you want to examine, and then click **Merge** on the toolbar.
5. The current default merge mode is Three-Way Merge mode. The merge option is not available for format records. Instead, you can click **Compare** option from the **More Actions** menu to start the merge tool in read-only mode. You can use this mode to identify the differences in the side-by-side view or three-way view and then merge records manually in Forms Designer.



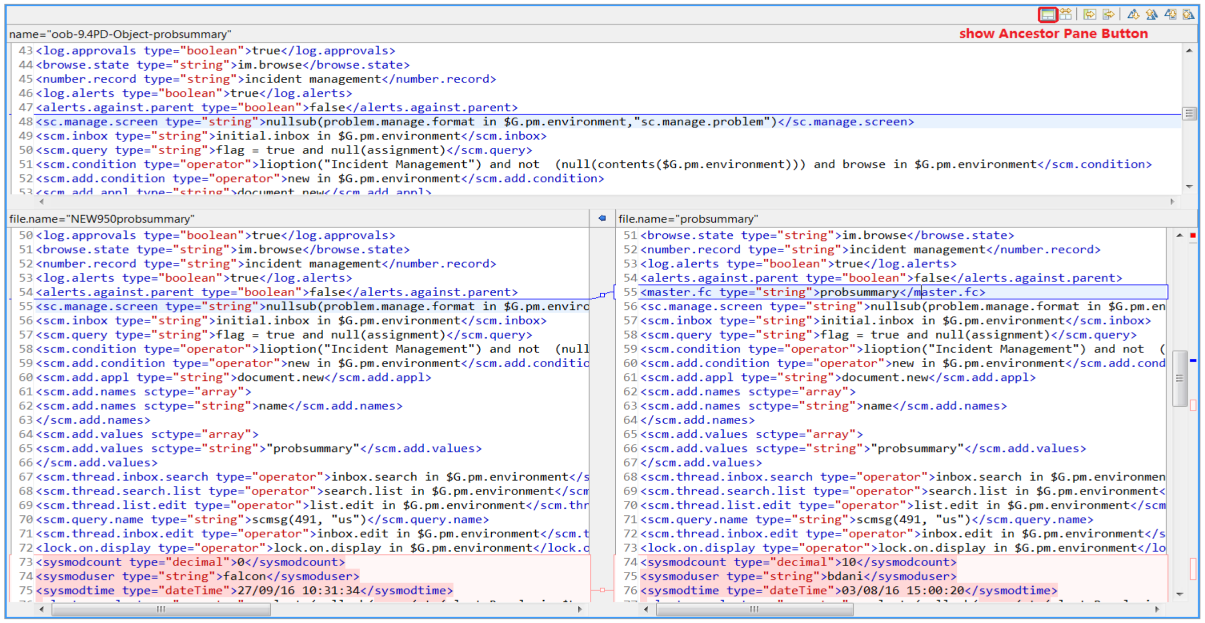
file.name="NEW950probsummary"

```
<log.approvals type="boolean">true</log.approvals>
<browse.state type="string">im.browse</browse.state>
<number.record type="string">incident management</number.record>
<log.alerts type="boolean">true</log.alerts>
<alerts.against.parent type="boolean">false</alerts.against.parent>
<sc.manage.screen type="string">nullsub(problem.manage.format in $G.pm.enviro
<scm.inbox type="string">initial.inbox in $G.pm.environment</scm.inbox>
<scm.query type="string">flag = true and null(assignment)</scm.query>
<scm.condition type="operator">lioption("Incident Management") and not  (null
<scm.add.condition type="operator">new in $G.pm.environment</scm.add.conditic
<scm.add.appl type="string">document.new</scm.add.appl>
<scm.add.names sctype="array">
<scm.add.names sctype="string">name</scm.add.names>
</scm.add.names>
<scm.add.values sctype="array">
<scm.add.values sctype="string">"probsummary"</scm.add.values>
</scm.add.values>
<scm.thread.inbox.search type="operator">inbox.search in $G.pm.environment</s
<scm.thread.search.list type="operator">search.list in $G.pm.environment</scm
<scm.thread.list.edit type="operator">list.edit in $G.pm.environment</scm.thr
<scm.query.name type="string">scmsg(491, "us")</scm.query.name>
<scm.thread.inbox.edit type="operator">inbox.edit in $G.pm.environment</scm.t
<lock.on.display type="operator">lock.on.display in $G.pm.environment</lock.c
<sysmodcount type="decimal">0</sysmodcount>
<sysmoduser type="string">falcon</sysmoduser>
<sysmodtime type="dateTime">27/09/16 10:31:34</sysmodtime>
<alert.recalc type="operator">evaluate(nullsub(parse(str(alertsRecalc in $L.w
<alert.reset type="operator">evaluate(nullsub(parse(str(alertsReset in $L.wfP
<global.inbox type="operator">nullsub(global.inbox in $G.pm.environment, fals
<personal.inbox type="operator">nullsub(personal.inbox in $G.pm.environment,
<watch.variables sctype="array">
<watch.variables sctype="string">$pmc.actions</watch.variables>
<watch.variables sctype="string">$apm.activity</watch.variables>
</watch.variables>
<activitylog.file.name type="string">activity</activitylog.file.name>
<activity.selection.var type="string">$apm.activity</activity.selection.var>
<activity.mandatory type="boolean">false</activity.mandatory>
<activity.mandatory.msg type="string">scmsg(2210, "us")</activity.mandatory.m
<update.field.var type="string">$pmc.actions</update.field.var>
```

file.name="probsummary"

```
<browse.state type="string">im.browse</browse.state>
<number.record type="string">incident management</number.record>
<log.alerts type="boolean">true</log.alerts>
<alerts.against.parent type="boolean">false</alerts.against.parent>
<master.fc type="string">probsummary</master.fc>
<sc.manage.screen type="string">nullsub(problem.manage.format in $G.pm.en
<scm.inbox type="string">initial.inbox in $G.pm.environment</scm.inbox>
<scm.query type="string">flag = true and null(assignment)</scm.query>
<scm.condition type="operator">lioption("Incident Management") and not  (
<scm.add.condition type="operator">new in $G.pm.environment</scm.add.cond
<scm.add.appl type="string">document.new</scm.add.appl>
<scm.add.names sctype="array">
<scm.add.names sctype="string">name</scm.add.names>
</scm.add.names>
<scm.add.values sctype="array">
<scm.add.values sctype="string">"probsummary"</scm.add.values>
</scm.add.values>
<scm.thread.inbox.search type="operator">inbox.search in $G.pm.environmen
<scm.thread.search.list type="operator">search.list in $G.pm.environment<
<scm.thread.list.edit type="operator">list.edit in $G.pm.environment</scm
<scm.query.name type="string">scmsg(491, "us")</scm.query.name>
<scm.thread.inbox.edit type="operator">inbox.edit in $G.pm.environment</s
<lock.on.display type="operator">lock.on.display in $G.pm.environment</lo
<sysmodcount type="decimal">10</sysmodcount>
<sysmoduser type="string">bdani</sysmoduser>
<sysmodtime type="dateTime">03/08/16 15:00:20</sysmodtime>
<alert.recalc type="operator">evaluate(nullsub(parse(str(alertsRecalc in
<alert.reset type="operator">evaluate(nullsub(parse(str(alertsReset in $L
<global.inbox type="operator">nullsub(global.inbox in $G.pm.environment,
<personal.inbox type="operator">nullsub(personal.inbox in $G.pm.environme
<watch.variables sctype="array">
<watch.variables sctype="string">$pmc.actions</watch.variables>
<watch.variables sctype="string">$apm.activity</watch.variables>
</watch.variables>
<activitylog.file.name type="string">activity</activitylog.file.name>
<activity.selection.var type="string">$apm.activity</activity.selection.v
<activity.mandatory type="boolean">false</activity.mandatory>
<activity.mandatory.msg type="string">scmsg(2210, "us")</activity.mandato
<update.field.var type="string">$pmc.actions</update.field.var>
```

a. In Three-Way Merge mode, click the **Show Ancestor Pane** button to show the base record reference. Click the button again to hide the base record reference.



name="oob-9.4PD-Object-probsummary"

```
<log.approvals type="boolean">true</log.approvals>
<browse.state type="string">im.browse</browse.state>
<number.record type="string">incident management</number.record>
<log.alerts type="boolean">true</log.alerts>
<alerts.against.parent type="boolean">false</alerts.against.parent>
<sc.manage.screen type="string">nullsub(problem.manage.format in $G.pm.environment,"sc.manage.problem")</sc.manage.screen>
<scm.inbox type="string">initial.inbox in $G.pm.environment</scm.inbox>
<scm.query type="string">flag = true and null(assignment)</scm.query>
<scm.condition type="operator">lioption("Incident Management") and not  (null(contents($G.pm.environment))) and browse in $G.pm.environment</scm.condition>
<scm.add.condition type="operator">new in $G.pm.environment</scm.add.condition>
<scm.add.appl type="string">document.new</scm.add.appl>
```

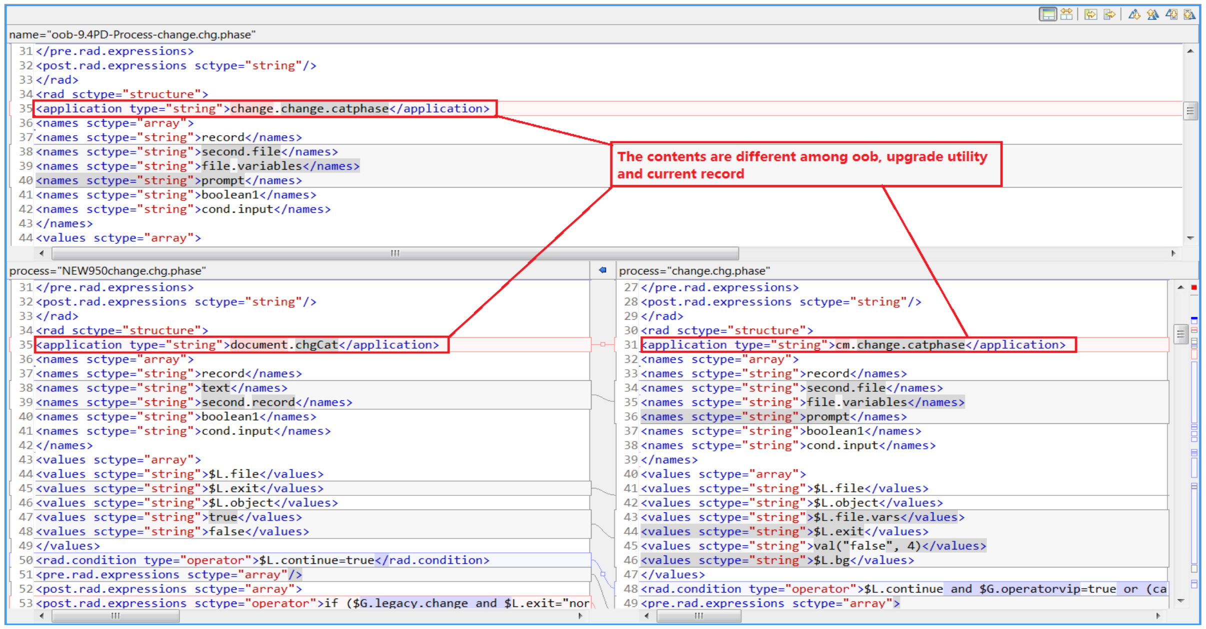
file.name="NEW950probsummary"

```
<log.approvals type="boolean">true</log.approvals>
<browse.state type="string">im.browse</browse.state>
<number.record type="string">incident management</number.record>
<log.alerts type="boolean">true</log.alerts>
<alerts.against.parent type="boolean">false</alerts.against.parent>
<sc.manage.screen type="string">nullsub(problem.manage.format in $G.pm.enviro
<scm.inbox type="string">initial.inbox in $G.pm.environment</scm.inbox>
<scm.query type="string">flag = true and null(assignment)</scm.query>
<scm.condition type="operator">lioption("Incident Management") and not  (null
<scm.add.condition type="operator">new in $G.pm.environment</scm.add.conditic
<scm.add.appl type="string">document.new</scm.add.appl>
<scm.add.names sctype="array">
<scm.add.names sctype="string">name</scm.add.names>
</scm.add.names>
<scm.add.values sctype="array">
<scm.add.values sctype="string">"probsummary"</scm.add.values>
</scm.add.values>
<scm.thread.inbox.search type="operator">inbox.search in $G.pm.environment</s
<scm.thread.search.list type="operator">search.list in $G.pm.environment</scm
<scm.thread.list.edit type="operator">list.edit in $G.pm.environment</scm.thr
<scm.query.name type="string">scmsg(491, "us")</scm.query.name>
<scm.thread.inbox.edit type="operator">inbox.edit in $G.pm.environment</scm.t
<lock.on.display type="operator">lock.on.display in $G.pm.environment</lock.c
<sysmodcount type="decimal">0</sysmodcount>
<sysmoduser type="string">falcon</sysmoduser>
<sysmodtime type="dateTime">27/09/16 10:31:34</sysmodtime>
```

file.name="probsummary"

```
<browse.state type="string">im.browse</browse.state>
<number.record type="string">incident management</number.record>
<log.alerts type="boolean">true</log.alerts>
<alerts.against.parent type="boolean">false</alerts.against.parent>
<master.fc type="string">probsummary</master.fc>
<sc.manage.screen type="string">nullsub(problem.manage.format in $G.pm.en
<scm.inbox type="string">initial.inbox in $G.pm.environment</scm.inbox>
<scm.query type="string">flag = true and null(assignment)</scm.query>
<scm.condition type="operator">lioption("Incident Management") and not  (
<scm.add.condition type="operator">new in $G.pm.environment</scm.add.cond
<scm.add.appl type="string">document.new</scm.add.appl>
<scm.add.names sctype="array">
<scm.add.names sctype="string">name</scm.add.names>
</scm.add.names>
<scm.add.values sctype="array">
<scm.add.values sctype="string">"probsummary"</scm.add.values>
</scm.add.values>
<scm.thread.inbox.search type="operator">inbox.search in $G.pm.environmen
<scm.thread.search.list type="operator">search.list in $G.pm.environment<
<scm.thread.list.edit type="operator">list.edit in $G.pm.environment</scm
<scm.query.name type="string">scmsg(491, "us")</scm.query.name>
<scm.thread.inbox.edit type="operator">inbox.edit in $G.pm.environment</s
<lock.on.display type="operator">lock.on.display in $G.pm.environment</lo
<sysmodcount type="decimal">10</sysmodcount>
<sysmoduser type="string">bdani</sysmoduser>
<sysmodtime type="dateTime">03/08/16 15:00:20</sysmodtime>
```

b. In Three-Way Merge mode, if the records of base, upgrade, and customer versions are all different, the background color is red. If you put the cursor into the records with the red background and then click the **Copy Current Change from Left to Right** button, the selected records from the left pane will be appended to selected records in the right pane.


The contents are different among oob, upgrade utility and current record

name="oob-9.4PD-Process-change.chg.phase"

```
</pre.rad.expressions>
<post.rad.expressions sctype="string"/>
</rad>
<rad sctype="structure">
<application type="string">change.change.catphase</application>
<names sctype="array">
<names sctype="string">record</names>
<names sctype="string">second.file</names>
<names sctype="string">file.variables</names>
<names sctype="string">prompt</names>
<names sctype="string">boolean1</names>
<names sctype="string">cond.input</names>
</names>
<values sctype="array">
```

process="NEW950change.chg.phase"

```
</pre.rad.expressions>
<post.rad.expressions sctype="string"/>
</rad>
<rad sctype="structure">
<application type="string">document.chgCat</application>
<names sctype="array">
<names sctype="string">record</names>
<names sctype="string">text</names>
<names sctype="string">second.record</names>
<names sctype="string">boolean1</names>
<names sctype="string">cond.input</names>
</names>
<values sctype="array">
<values sctype="string">$L.file</values>
<values sctype="string">$L.exit</values>
<values sctype="string">$L.object</values>
<values sctype="string">true</values>
<values sctype="string">false</values>
</values>
<rad.condition type="operator">$L.continue=true</rad.condition>
<pre.rad.expressions sctype="array"/>
<post.rad.expressions sctype="array">
<post.rad.expressions sctype="operator">if ($G.legacy.change and $L.exit="nor
```

process="change.chg.phase"

```
</pre.rad.expressions>
<post.rad.expressions sctype="string"/>
</rad>
<rad sctype="structure">
<application type="string">cm.change.catphase</application>
<names sctype="array">
<names sctype="string">record</names>
<names sctype="string">second.file</names>
<names sctype="string">file.variables</names>
<names sctype="string">prompt</names>
<names sctype="string">boolean1</names>
<names sctype="string">cond.input</names>
</names>
<values sctype="array">
<values sctype="string">$L.file</values>
<values sctype="string">$L.object</values>
<values sctype="string">$L.file.vars</values>
<values sctype="string">$L.exit</values>
<values sctype="string">val("false", 4)</values>
<values sctype="string">$L.bg</values>
</values>
<rad.condition type="operator">$L.continue and $G.operatorvip=true or (ca
<pre.rad.expressions sctype="array">
```

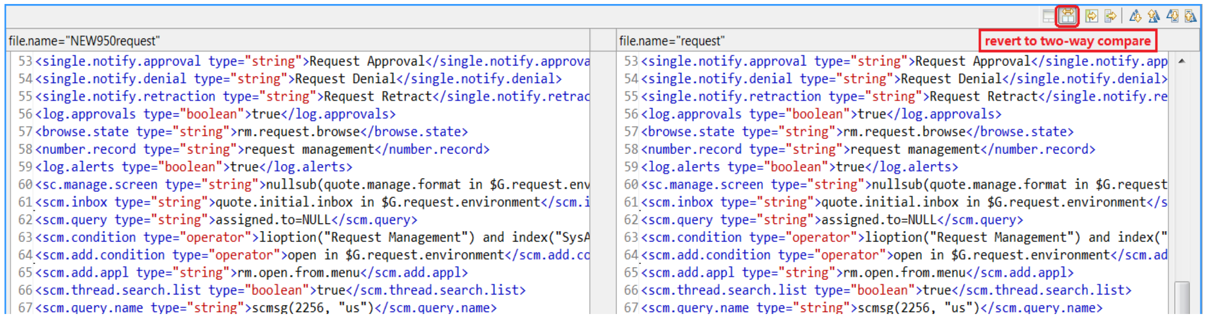
c. In Three-Way Merge mode, if the records of the customer version are identical to the base version, but different from the upgrade version, the background color is blue. If you put the cursor into the records with the blue background, and then click the **Copy Current Change from Left to Right** button the selected records from the right pane will be replaced by the selected records from the left pane.

d. In Three-Way Merge mode, if the records on the left contain red background records, and you then click the **Copy All from Left to Right** button, the records in the right pane with blue background will be replaced with selected records from the left pane.

> **Note:** The right records with red background will not be changed. To do this, you will need to switch to Two-Way Merge mode to use Copy All from Left to Right feature.

6. To enter Two-Way Merge mode, click the **Two-Way Compare** (Ignore Ancestor) button. In this mode, the Show Ancestor Pane button is disabled.

   When you do this, the button label changes to Three-Way Compare. Click this again to revert to Three-Way Compare mode.



file.name="NEW950request"

```
<single.notify.approval type="string">Request Approval</single.notify.approva
<single.notify.denial type="string">Request Denial</single.notify.denial>
<single.notify.retraction type="string">Request Retract</single.notify.retrac
<log.approvals type="boolean">true</log.approvals>
<browse.state type="string">rm.request.browse</browse.state>
<number.record type="string">request management</number.record>
<log.alerts type="boolean">true</log.alerts>
<sc.manage.screen type="string">nullsub(quote.manage.format in $G.request.env
<scm.inbox type="string">quote.initial.inbox in $G.request.environment</scm.i
<scm.query type="string">assigned.to=NULL</scm.query>
<scm.condition type="operator">lioption("Request Management") and index("SysA
<scm.add.condition type="operator">open in $G.request.environment</scm.add.co
<scm.add.appl type="string">rm.open.from.menu</scm.add.appl>
<scm.thread.search.list type="boolean">true</scm.thread.search.list>
<scm.query.name type="string">scmsg(2256, "us")</scm.query.name>
```

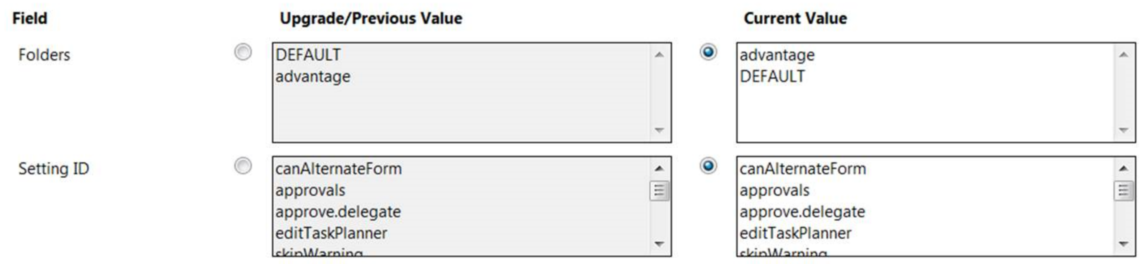
file.name="request"

```
<single.notify.approval type="string">Request Approval</single.notify.app
<single.notify.denial type="string">Request Denial</single.notify.denial>
<single.notify.retraction type="string">Request Retract</single.notify.re
<log.approvals type="boolean">true</log.approvals>
<browse.state type="string">rm.request.browse</browse.state>
<number.record type="string">request management</number.record>
<log.alerts type="boolean">true</log.alerts>
<sc.manage.screen type="string">nullsub(quote.manage.format in $G.request
<scm.inbox type="string">quote.initial.inbox in $G.request.environment</s
<scm.query type="string">assigned.to=NULL</scm.query>
<scm.condition type="operator">lioption("Request Management") and index("
<scm.add.condition type="operator">open in $G.request.environment</scm.ad
<scm.add.appl type="string">rm.open.from.menu</scm.add.appl>
<scm.thread.search.list type="boolean">true</scm.thread.search.list>
<scm.query.name type="string">scmsg(2256, "us")</scm.query.name>
```

a. In Two-Way-Merge mode, if one record differs upgrade and customer versions, you can click **Copy Current Change from Left to Right** button to replace a selected record from the left pane to the right pane.

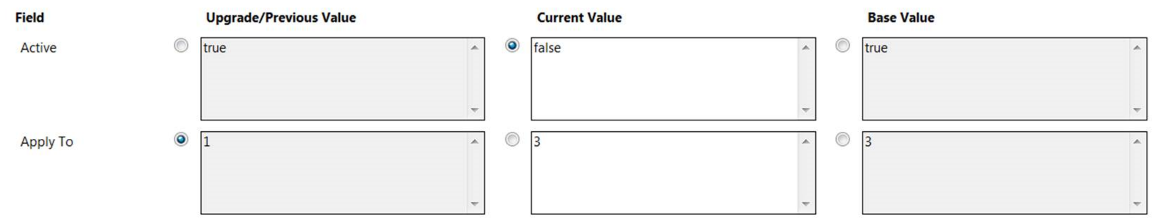
b. In Two-Way-Merge mode, click **Copy All from Left to Right** button to replace all records in the right pane with the records from the left pane.

Note: Service Manager 9.50 provides a Field-Level Merge mode for some simple structure records. This field compare mode displays either Two-Way-Merge or Three-Way-Merge. You can select the value you want to keep. Refer to the following screenshots:



Field	Upgrade/Previous Value	Current Value
Folders	○ DEFAULT advantage	● advantage DEFAULT
Setting ID	○ canAlternateForm approvals approve.delegate editTaskPlanner skipWarning	● canAlternateForm approvals approve.delegate editTaskPlanner skipWarning

The field compare mode displays Two-Way-Merge by default if there is no related out-of-box xml file in the <Upgrade Utility root folder>/3waymerge/oob/<base version>/<dbdict name> folder. Otherwise, the field compare mode displays Three-Way-Merge.



Field	Upgrade/Previous Value	Current Value	Base Value
Active	true	[✓ false]	true
Apply To	[✓ 1]	3	3

To switch between the field merge mode and the XML merge mode, click **XML Merge**.

Using a third party tool to visually compare objects

You may also visually compare the three versions of each object using a three-way compare and merge tool outside HPE Service Manager, and then merge them manually in Service Manager. For example, you can use a tool, such as KDiff3 for Windows, to compare and merge objects.

To download and learn about KDiff3, visit the KDiff3 Web site.

A brief example of using KDiff3 to compare the three versions of an object is provided in the following steps:

1. Install KDiff3 on the Service Manager server host.
2. Open KDiff3. For more information, refer to the KDiff3 documentation.
3. For the A(Base) parameter, specify the path to the Upgrade\3waymerge\work\base folder.
4. For the B parameter, specify the path to the Upgrade\3waymerge\work\customer folder.
5. For the C parameter, specify the path to the Upgrade\3waymerge\work\upgrade folder.
6. Click **OK**.
7. Navigate the folder structure and double-click the file that is named after the object you want to merge.

8. Compare the three versions of the object in the 3-way compare view.

9. Manually apply the changes you have identified to the object in Service Manager.

Using the Auto Merge and Revert options

During upgrade processing, the Upgrade Utility attempts to merge your objects with the corresponding objects provided with the upgrade. You can search for records with a result type of "Auto Merged" in the Upgrade Results list to review these objects.

> **Note:** An auto-merged object needs to be thoroughly tested for syntax and functional integrity.

If you encounter issues with objects that have been auto-merged or reconciled, you can click the **Revert** button on the toolbar to restore "Auto Merged" or "Renamed" objects, or "Replaced" Applications Cluster objects to their previous states. Use **Revert** to restore one object and **Mass Revert** to restore multiple objects. After restoring an object, you can also re-attempt to auto-merge that object using the **Auto Merge** or **Mass Auto Merge** option from the More Actions menu.

> **Note:** If you do not choose to use the auto-merge option, you must manually unzip the OOB data to the same folder in which you extracted the Merge Tool. If you do choose the auto-merge option, the OOB data is extracted automatically by the Merge Tool.

Using the Mass Choose Upgrade feature

During the Upgrade process, you can use the Mass Choose Upgrade feature to overwrite your systems old objects with the newer versions from the upgrade utility. You can use this feature to quickly update the objects of the following statuses, which are generated during the upgrade:

- Auto Merged
- Renamed
- Previously Reconciled
- Reconciled
- Kept Customer Non-OOB

You can use this feature to quickly update Applications Cluster objects of the following statuses:

- Kept Customer
- Kept Customer Non-OOB

- Renamed
- Previously Reconciled

To use the Mass Choose Upgrade feature, follow these steps:

1. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **View/Merge Results**.
2. (Optional) Select **View Upgrade Results and Merge Conflicts**, and then click **Next**.

   Note: This screen appears only after Process Designer migration.
3. In the **Result** drop-down list, filter the set of objects (Auto Merged; Renamed; Previously Reconciled; Reconciled) on which you wish to use the Mass Choose Upgrade feature and then click **Search**.
4. If more than two objects exist in the resulting search, click the **Mass Choose Upgrade** button from **More Actions** menu in the returned list and then click **Yes**.

After you click **Yes**, the objects that you selected will be updated with the contents of the newer versions from the upgrade utility.

Using the Mark as Reconciled feature

During the Upgrade process, you must mark conflicting objects as “Reconciled” after resolving each conflict. To help with this process, you can use the Mass Mark as Reconciled feature to mark multiple objects as “Reconciled.” You can use this feature on objects with the following statuses:

- Auto Merged
- Renamed
- Previously Reconciled

To use the Mass Mark as Reconciled feature, follow these steps:

1. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **View/Merge Results**.
2. If you are upgrading from SM9.3x with PDCP3 installed or non-PD environment or SM9.4x Classic, you can see a wizard page. Select **View Upgrade Results and Merge Conflicts**, and then click **Next**.

> **Note:** This screen appears only after Process Designer migration.

3. In the **Result** drop-down list, filter the set of objects (Auto Merged; Renamed; Previously Reconciled) on which you wish to use the Mass Mark as Reconciled feature and then click **Search**.
4. If more than two objects exist in the resulting search, click the **Mass Mark as Reconciled** button from the **More Actions** menu and then click **Yes**. After you click **Yes**, all objects that you selected will be marked as "Reconciled" and removed from current conflict list.

   > **Note:** If only one object exits in the resulting search, or if you want to resolve conflicts for the selected objects individually, use the **Mark as Reconciled** button on the toolbar instead.

Resolve Process Designer related data

To resolve the Process Designer related data, follow these steps:

1. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **View/Merge Results**.
2. If you are upgrading from SM9.3x with PDCP3 installed or non-PD environment or SM9.4x Classic, you can see a wizard page. Select **View Upgrade Results and Merge Conflicts**, and then click **Next**.

   > **Note:** This screen appears only after Process Designer Hybrid migration.

3. Click the **Expert Search** button from the More Actions menu, type `version.reconciled#"PD Enable"` in the **Query** field, and then click **Search**.
4. In the search result, resolve the data by following the actions described in the table below.

Reconciled Version	Result	Action
PD Enable: Updated	Renamed	Merge the newer object from the Upgrade Utility to the object.
PD Enable: Replaced	Renamed	Overwrite your systems old object with the newer version from the Upgrade Utility.
PD Enable: Need to be removed	Any	Remove the objects manually.

Step 10: Perform additional manual tasks

This section lists changes that cannot be automated by the Upgrade Utility and changes that are required only for certain customers. Make these changes before testing and backing up your system.

Adopt the logical name solution

In previous versions of Service Manager, the logical.name field in the device table was used as both the unique identifier and display name for a CI, causing loss of CI data integrity across Service Manager modules and complexity of CI reconciliation between integration products (for example, Service Manager does not allow duplicate CI names while UCMDB does).

Service Manager 9.41 and later versions solves the problems described above with the logical name solution. Once you have upgraded to Service Manager 9.50 from an applications version prior to 9.41, you need to perform additional manual tasks to adopt the solution in your production environment. For a summary of this solution and details of these manual post-upgrade tasks, see the *Service Manager Logical Name Solution* white paper.

Adopt Smart Search

Service Manager 9.41 introduces Smart Search which enables you to search among a variety of content, including Service Manager records, SharePoint documents, static web pages, and KM documents. You can integrate multiple knowledge libraries by configuring different search connectors, so that users can search among the information that they can access.

Once you have upgraded to Service Manager 9.50 from an applications version prior to 9.41, follow these steps to make sure the login.DEFAULT format control record has the expected code for the Smart Search feature:

1. Log on to your Service Manager server as a System Administrator.
2. Click **Tailoring** > **Format Control**. Service Manager opens a blank Format Control Maintenance form.
3. Type `login.DEFAULT` in the Name field, and then click **Search**.
4. Click **JavaScript**.

5. Check whether the following codes exist:

   ```
   var idolserver = new SCFile('idolserverinfo');
   var rc = idolserver.doSelect('true');
   if (rc === RC_SUCCESS) {
                  vars['$lo.idol.enabled'] = idolserver['enable'];
                  vars['$lo.idol.img.enabled'] = idolserver['image.enable'];
                  if (vars['$lo.idol.enabled']) {
                          vars['$G.kmsearchengine'] = 'IDOL';
                   }
   }
   ```

   If these codes do not exist, you must type `true` in the Add field and then add these codes manually.

6. Click **Save** and **OK**.

Adopt Chat

Perform the manual steps described in the "Deploy the chat server" task of the Collaboration deployment procedure. For details, search for "Deploy Service Manager Collaboration with HTTP" or "Deploy Service Manager Collaboration with HTTPS" in the Service Manager Help Center or the Installation Guide in PDF format.

Update the support.groups fields in the device table

Once you have upgraded to Service Manager 9.50, follow these steps to update the support.groups fields in the device table:

1. Log on to your Service Manager server as a System Administrator.
2. Click **Tailoring** > **Database Dictionary**.
3. Type `device` in the File Name field, and then click **Search**.
4. Click the **SQL Tables** tab, and then add a row as follows:

Alias	Name	Type
a1	DEVICE2A1	<*Specify this value as appropriate*>

5. Click **OK** to save your changes.
6. Click **support.groups** of the array type.

7. Update the value in the SQL Table field to `a1`, and then click **OK** to save your changes.
8. Click **support.groups** of the character type.
9. Update the value in the SQL Table field to `a1`, update the value in the SQL Type field to `VARCHAR (60)`, and then click **OK** to save your changes.
10. Click **OK** to save your changes and close the device table.

Caution: If the device table contains more than 100,000 records, updating the support.groups fields may take more than one hour.

Additional manual migration tasks on Service Manager 9.50 Hybrid

After you have run the Service Manager 9.50 Applications Upgrade Utility to migrate your system to Service Manager Hybrid, you must complete a number of manual configuration tasks. Some tasks are optional, depending on how you want to configure your Service Manager Hybrid system. For detailed instructions, refer to "Appendix A: Additional manual migration tasks on Service Manager 9.50 Hybrid" on page 113.

Note: Perform these manual migration tasks on the development environment only.

Step11: Migrate Process Designer data

You can use the data migration tool to migrate data from legacy process records to new process records within the same Service Manager system. It is not used to migrate data across different Service Manager systems. You can configure the migration tool from the GUI, and it requires administrator privileges.

The migration tool allows an operator to configure the field/value mapping between two files. It also covers customized fields. The out-of-box Process Designer provides several sample migration scripts that you can choose to use according to your migration scenario.

Caution: During data migration, there are no dependencies between the modules, however the scripts for each specific module must be executed in the listed order.

Module	Script
PD Framework	1. Legacy change model taskplanner data to new taskplanner 2. Legacy change instance taskplanner data to new taskplanner 3. Legacy change/task data to new taskplanner
Problem	1. Legacy problem to new problem 2. Legacy problem task to new problem task 3. Legacy known error to new known error 4. Legacy Process Designer known error to new Process Designer known error 5. Legacy known error related record migration 6. Legacy known error attachment to new known error attachment

Module	Script
Service Level Agreement	1. Legacy SLA to new Agreement
Service Catalog	1. Legacy Catalog Items to new Catalog Items

If further migrations are needed, an administrator can use these samples to define their own. Considering that some tables typically have a large data volume, the Data Migration tool supports an SQL batch update mode to ensure good performance.

Note: We recommend that you back up the production data before migration.

Run the scripts for specific modules as follows:

Module		9.3x non PD	9.3x PD3	9.3x PD4	9.40 Classic / 9.41 Classic	9.40 Codeless / 9.41 Hybrid / 9.41 Codeless
PD Framework	Legacy change model taskplanner data to new taskplanner		√	√		
	Legacy change instance taskplanner data to new taskplanner		√	√		
	Legacy change/task data to new taskplanner	√			√	

Module		9.3x non PD	9.3x PD3	9.3x PD4	9.40 Classic / 9.41 Classic	9.40 Codeless / 9.41 Hybrid / 9.41 Codeless
Problem	Legacy problem to new problem	√	√		√	
	Legacy problem task to new problem task	√	√		√	
	Legacy known error to new known error	√	√		√	
	Legacy PD known error to new PD known error			√		
	Legacy known error related record migration	√	√		√	
	Legacy known error attachment to new known error attachment	√	√		√	
Service Level Agreement	Legacy SLA to new Agreement	√	√	√		
Service Catalog	Legacy Catalog Items to new Catalog Items	√	√	√	√	√

How to run a migration script

> **Note:** If you are upgrading an existing Service Manager system to Process Designer, you need to customize the legacy out-of-box migration script samples for your specific environment.

To run a migration script, follow these steps:

1. Type `db` in the command line to open Database Manager.
2. Type **migrationSetting** in the **Table** field, and then click **Search**.
3. In the Migration Script setting detail form, click **Migrate Data** to run the migration script.
4. When using the Batch Update mode, the Data Migration pop-up window shows the RDBMS SQL Statements that will be run on the database. Check that the SQL statements are correct, and change these SQL statements if needed. Click **Next**, and then click **Finish** to start the migration in Batch Update mode.

   > **Note:** If there is no data to migrate, or if the data is already migrated, you receive the following notification:
   >
   > "No records match the filter condition. At least one matching record is required."

Step12: Return the system to normal operation

After the upgrade, the system may exhibit abnormal behavior until you return it to its normal operating environment.

> **Note:** All upgrade-related files should be removed from the system after successful completion of the upgrade. For information on removing these files, see "Step 10: Perform additional manual tasks" on page 98.

To return to a normal operating environment, follow these steps:

1. Log out.
2. Stop the server.
3. Remove the comments from the system.start entry from the sm.cfg file.
4. Restore all the parameters that you updated in sm.ini and sm.cfg to their original state.

5. Add all parameters that are documented as necessary for your upgraded system to run properly.
6. Start the Service Manager server.
7. Log on.
8. Wait for the background processes to start up.

> **Note:** HPE Service Manager does not recompile indexes in your RDBMS. If your RDBMS is not configured to recompile indexes automatically after index changes, you must recompile your indexes manually.

Step 13: Test the development environment

After you resolve all conflicts, test the upgraded system and verify that the conflicts resolutions function properly. If there are problems that you cannot resolve, contact HPE Customer Support.

Step 14: Back up the system

Make a checkpoint backup of the data files to enable you to restore from this point, if necessary. Refer to the documentation for your RDBMS for backup instructions.

Step 15: Build a custom upgrade

If you have followed all the steps to this point, you have already run the upgrade against the duplicate system you created of your production system, and you have performed reconciliation for that system. This section describes how to build a custom upgrade, which is then applied to your test system and production system.

To build the custom upgrade, follow these steps:

1. Log on to the reconciled development system.
2. Before you begin to build the custom upgrade, you need to back up (or duplicate) the production system. For information on duplicating your system, see "Step 1: Duplicate the production environment" on page 60.

3. Create a folder for the custom upgrade on the HPE Service Manager server. You can assign any name you like to this folder. However, for documentation purposes, this folder will be referred to as the *CustomUpgrade* folder. Ensure that the *CustomUpgrade* folder is empty.

   > **Note:** If you are connecting to the Service Manager server from a client that is installed on a remote client computer, make sure that the folder is created on the Service Manager server instead of the client computer.

4. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **Create Custom Upgrade**.
5. The Upgrade Utility automatically checks for duplicate records prefixed with "PRE", "NEW" or "OLD".
6. The system displays the patch file name and all the supported languages that are installed on your Service Manager system. Review these information, and then click **Next**.
7. Type a name for the custom upgrade package in the **The name of this release** field. For example, `SM950`.
8. Type the fully-qualified path to the folder where the Upgrade Utility should create files in the **The path to export the upgrade files** field. This should be the path to the empty *CustomUpgrade* folder that you created in *step 3*. For example, `C:\HPE\ServiceManager\CustomUpgrade`.
9. Click **Next**.
10. The system displays **HPE Service Manager Upgrade Builder is now ready to build the upgrade**. Click **Next** to continue.
11. When you receive a "Finished creating the transfer files for the upgrade" message, the Upgrade Utility has finished the data packaging.

Upgrade Utility logs and error messages

When building the custom upgrade, the Upgrade Utility creates a set of log files, which reside in the same directory as the upgrade files. The following table describes the contents of the log files during this step.

List of upgrade logs

Log file	Contents
detail.log	This file contains specific information about the upgrade, including the following: • All information in transfer.log and upgrade.log

List of upgrade logs, continued

Log file	Contents
	• Name of the file being purged. For example, “2016-09-20 13:58:32 dbdict: upgradestatus is purged.” • Progress of signature creation. For example, “2016-09-20 14:47:36 Created 100 signatures for Object on version SM950” • Building distribution information. For example, “2016-09-20 14:55:07 Building Distribution object for Applications Cluster Action.run”
transfer.log	This file contains information about the object being transferred by the upgrade. For example, “2016-09-20 15:12:27 Initiating an export of scmessage on query " ((class="error" and message.id isin {"10"})) and syslanguage~="xxx""”
upgrade.log	This file contains information about where the upgrade is at any point, including the following: • Starting and ending of each sub-phase. For example, “2016-09-20 14:47:32 **** Start Phase [Pre Create Action Check] ****” • Main activities during each sub-phase. For example, “2016-09-20 14:47:35 Signaturing records.” • Name of the file being exported. For example, “2016-09-20 15:11:37 Exporting: table = Object query = 'true'”

Upgrade tasks on the test environment

Step 1: Apply the custom upgrade to the test environment

You need to apply the newly-created custom upgrade to your test system for user acceptance testing.

> **Note:** If you experience problems, such as a power failure or a network connection error while upgrading the system, you need to restore the database before attempting to run the upgrade again.

To upgrade the custom upgrade, follow these steps:

1. On the test system, which is a copy of your production system, complete all the preparation tasks in "Upgrade tasks on the development environment" on page 59.
2. Load the preupg.bin file using Service Manager Database Manager, and then restart the client.

   > **Note:** Instead of using the file that you extracted from the product installation package, you must use the preupg.bin file in your CustomUpgrade folder.

   > **Note:** The AppUpgVersion.txt file does not exist in the CustomUpgrade folder. Do not copy the file to this folder.

   > **Caution:** Before loading these files, you also need to disable the **Client side load/unload** option from **Window** > **Preferences** > **HPE Service Manager**.

3. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **Apply Upgrade** to launch the Upgrade Utility.
4. In the text box, type the fully qualified path to the folder that hosts transfer.bin, and then click **Next**.

   > **Note:** Instead of using the file that you extracted from the product installation package, you must use the transfer.bin file in your *CustomUpgrade* folder.

   Example:

   Windows: `c:\temp\upgrade\`

   Linux: `/tmp/upgrade/`

5. The loading process may take a long time. Wait until the file is loaded and the system displays the `Transfer files loaded` message.
6. The system displays a series of information for your verification, including Applications version upgrading from, Applications version upgrading to, Applications base version, Full path to the Upgrade Utility files and Language(s) to be upgraded. Verify these information and then click **Next** to continue.

   > **Note:** If this screen does not display the correct information, do not continue with the upgrade. Instead contact HPE Software Customer Support.

7. When you are asked whether you want to proceed, click **Yes**.
8. The Upgrade Utility displays the status when the upgrade is being processed.
9. 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 click **Finish**, you are automatically logged out.
10. Restart the server and log back in to the client.
11. Open the **scversion** table in the Database Manager, and verify that the **Application Version** field is 9.50.00xx. If this field displays a value other than 9.50.00xx, check the log files to identify the issue that occurred.

Note how long it takes to apply the custom upgrade, so you will know how long the production system will be unavailable during the production upgrade. You can check the log file for an estimate.

Upgrade Utility logs and error messages

When applying the custom upgrade to the test system, the Upgrade Utility creates a set of log files, which reside in the same directory as the custom upgrade files.

The contents of these log files are similar to those in the log files when running an out-of-box upgrade. See "Upgrade Utility logs and error messages" on page 66.

Tables and records that are not upgraded by the Upgrade Utility

The Upgrade Utility does not automatically upgrade all tables and records. The patches record lists the tables and records that are packaged into the custom upgrade. Customizations made to any other tables or records will not be part of the custom upgrade. To make sure that the objects that you have reconciled are moved to the production system, verify the following scenarios:

- If an object is in the patches record, and the Result field of its related “upgraderesults" record displays “Already Current”, "Kept Customer” or “Kept Customer Non-OOB”, change the Result field to “Reconciled”.

  For example, follow these steps after you have modified the cm.open.display_newphase displayoption record:

   a. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **View/Merge Results**.

   b. Type `displayoption` in the Object Type field, type `cm.open.display_newphase` in the Object Name field, and then click **Search**.

      > **Note:** Some object names consist of multiple key fields, you can find the definitions in the signaturemake record.

   c. If the Result field in the search results is “Already Current”, “Kept Customer”, “Kept Customer Non-OOB”, manually change the Result field to “Reconciled”

- If an object is in the patches record, and the Result field of its related “upgraderesults" record is not “Already Current”, "Kept Customer” or “Kept Customer Non-OOB”, no additional task is needed.
- If an object is not in the patches record, do one of the following:
    - Create an unload file containing those objects by adding them to an unload script or using the standard HPE Service Manager Unload/Export Facility,
    - Make the same changes manually by directly modifying the objects on the production system. For records that you might have deleted, you can either build a purge script for those records or delete the records manually on the production system.

Resolve Process Designer related data

To resolve the Process Designer related data, follow these steps:

1. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **View/Merge Results**.
2. Click the **Expert Search** button from the More Actions menu, type `version.reconciled#"PD Enable"` in the **Query** field, and then click **Search**.
3. In the search result, resolve the data by following the actions described in the table below.

Reconciled Version	Result	Action
PD Enable: Need to be removed	Any	Remove the objects manually.

Step 2: Perform additional manual tasks

This section lists changes that cannot be automated by the Upgrade Utility and changes that are required only for certain customers. Make these changes before testing and backing up your system. For detailed instructions, refer to "Step 10: Perform additional manual tasks" on page 98.

Step 3: Migrate Process Designer data

See "Step11: Migrate Process Designer data" on page 101 for more information.

Step 4: Test the test environment

After you apply the custom upgrade on the test environment, perform user acceptance testing and load testing for all features, especially customized applications. Test the upgraded system with the new Service Manager client to verify any changes you have made in reconciliation. If the upgrade process has any problems, you need to contact HPE Customer Support. After you complete testing of the upgraded system, you can use it to upgrade your production system.

To test the test environment, follow these steps:

1. Return the system to a normal operating environment.
2. Install and configure the Service Manager client for the target version (see instructions in the *HPE Service Manager Installation Guide*).
3. Use the new Service Manager client to log on.
4. Review the features described in the *Service Manager 9.50 Help Center*.
5. Use the new Service Manager client to thoroughly test the upgraded system. Test all features that your users will access. Pay particular attention to areas that were modified on your system.

Upgrade tasks on the production environment

Step 1: Apply the custom upgrade to the production environment

After you test the custom upgrade on the test system, you need to apply the newly-created custom upgrade to your production system. This process is identical to the one you followed when applying your upgrade to your test system.

> **Note:** Do not apply an upgrade to your production system if it has not been thoroughly tested. If issues were found in the custom upgrade, HPE recommends you to fix the issues in the development system and re-create the customer upgrade.

To apply the custom upgrade to the production system, follow these steps:

1. Make sure the production system is not be available to users while you are applying the custom upgrade.
   a. Have all users log out of the server.
   b. Prevent users from logging into your Service Manager server by running the **sm -quiesce:1** quiesce command from the operating system’s command prompt.
2. Ensure the upgrade files you created are accessible to the production system (the files are located on the same server).
3. If you transfer the files to your production system by FTP, set FTP to binary mode.
4. Complete all the preparation tasks in "Upgrade tasks on the development environment" on page 59.
5. Apply the customer upgrade to the production system.

   > **Note:** If you experience problems, such as a power failure or a network connection error while upgrading the system, you need to restore the database before attempting to run the upgrade again.

6. Log out from your Service Manager server, and then log in again.
7. Allow users to log in to the server by running the **sm -quiesce:0** quiesce command from the operating system’s command prompt.

Step 2: Perform additional manual tasks

This section lists changes that cannot be automated by the Upgrade Utility and changes that are required only for certain customers. Make these changes before testing and backing up your system. For detailed instructions, refer to "Step 10: Perform additional manual tasks" on page 98.

Step 3: Migrate Process Designer data

See "Step11: Migrate Process Designer data" on page 101 for more information.

Step 4: Test the production environment

Test the upgraded environment and verify that it functions properly. If there are problems that you cannot resolve, contact HPE Software Support.

Appendix A: Additional manual migration tasks on Service Manager 9.50 Hybrid

After you have run the SM9.50 Applications Upgrade Utility to migrate your system to Service Manager Hybrid, you must complete a number of manual configuration tasks. Some tasks are optional, depending on how you want to configure your Service Manager 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 Service Manager Hybrid, validity records are validated against the Process Designer category tables instead of the Classic category tables. Therefore, a manual review of the validity records is required.

A Process Designer migration report that resembles the following is included in the pdmigration.log file:

```
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 Classic Stage 1, Stage 2, Stage 3, and Deadline alerts and Deadline Alert Group configurations from Classic 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."

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 Classic alerts; however, there are some gaps and limitations.

- In Classic 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 Classic 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 Classic 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.

- 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 Classic 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 Classic object.

Reassignment threshold

The reassignment interval and expression and the reassignment count threshold in the Classic 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

Review Change Management processes

Note: The following step applies only to customers who upgrade to Service Manager Hybrid from a system that does not have PDCP3 installed.

Renamed processes

The following two Classic processes in DocEngine are renamed by the Applications Upgrade Utility. The old Classic names are now used for Codeless processes.

Process name in Classic mode	Process name in Hybrid mode
cm.open	cm.open.classic
cm.open.save	cm.open.save.classic

Therefore, if you have made any customizations to the processes, a conflict with the new name will be reported in the upgrade results (for example, if you customized “cm.open”, this process will be renamed to “cm.open.classic” and will conflict with the out-of-box “cm.open.classic”.) Additionally, if you have made any customizations that call the Classic processes, you must update these to call the new process names.

Review "Close Phase"

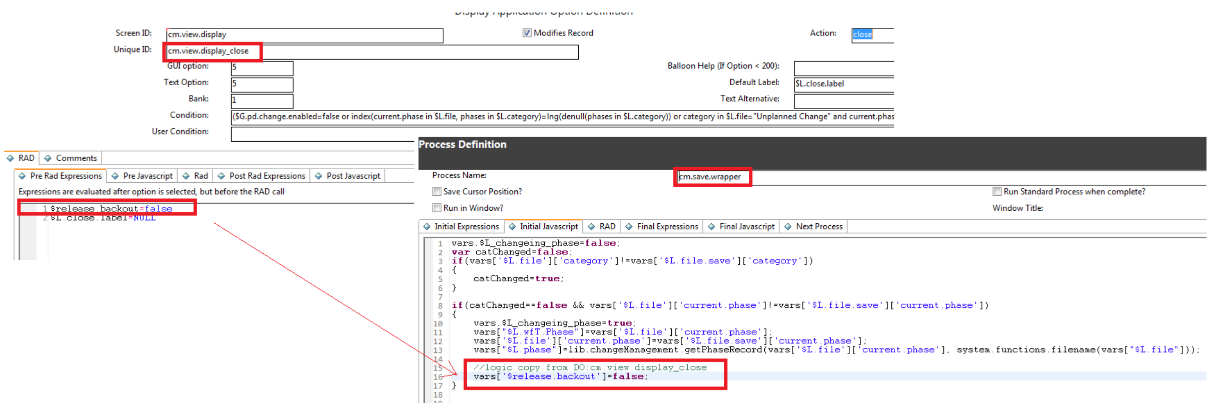
In the migrated Change Management workflow, the "Close Phase" functionality is moved to transitions between phases. In Classic versions of Service Manager, the "close" display option closes intermediate phases and closes the change or change task in the last phase. In the migrated Change Management workflow, the "close" display option closes only changes or change tasks in the last phase in the workflow.

The following condition is added to the original condition of the "cm.view.display_close" display option:

`($G.pd.change.enabled=false or index(current.phase in $L.file, phases in $L.category)=lng(denull(phases in $L.category)) or category in $L.file="Unplanned Change" and current.phase in $L.file="Discovery Assessment") and`

If you have tailored this display option, or if you have configured other display options to perform the "close" action, you must make equivalent modifications to ensure that the close action only works for last phase.

In Hybrid versions of Service Manager, display options are not called during the "close" phase. Therefore, the "cm.view.display_close" display option code is migrated to the "cm.save.wrapper" process, as illustrated by the following image. If you have tailored this display option, you must make equivalent changes to the process.



Screen ID: cm.view.display
Unique ID: cm.view.display_close
Modifies Record ✓
Action: close
GUI option: 5
Text Option: 5
Bank: 1
Balloon Help (If Option < 200):
Default Label: $L.close.label
Text Alternative:
Condition: ($G.pd.change.enabled=false or index(current.phase in $L.file, phases in $L.category)=lng(denull(phases in $L.category)) or category in $L.file="Unplanned Change" and current.phas
User Condition:

Expressions are evaluated after option is selected, but before the RAD call

```
$release.backout=false
$L.close.label=NULL
```

Process Name: cm.save.wrapper
Save Cursor Position? ☐
Run in Window? ☐
Run Standard Process when complete? ☐
Window Title:

```
vars.$L_changeing_phase=false;
var catChanged=false;
if(vars['$L.file']['category']!=vars['$L.file.save']['category'])
{
    catChanged=true;
}

if(catChanged==false && vars['$L.file']['current.phase']!=vars['$L.file.save']['current.phase'])
{
    vars.$L_changeing_phase=true;
    vars["$L.wfT.Phase"]=vars['$L.file']['current.phase'];
    vars['$L.file']['current.phase']=vars['$L.file.save']['current.phase'];
    vars["$L.phase"]=lib.changeManagement.getPhaseRecord(vars['$L.file']['current.phase'], system.functions.filename(vars["$L.file"]));

    //logic copy from DO:cm.view.display_close
    vars['$release.backout']=false;
}
```

In out-of-box Hybrid versions of Service Manager, the transition between phases is named "Close Phase". However, you can modify the label to suit your needs by using the localization functionality in the Workflow Editor.

Modify $L.env to $G .cm3r.environment/$G.cm3t.environment

In the Classic Change module, $L.env is a "cm3profile" record, whereas in Hybrid Change it is a "tableAccess" record. Therefore, references to $L.env are updated to refer to $G.cm3r.environment/$G.cm3t.environment, in case a "cm3profile" record is required (the $G.cm3r.environment/$G.cm3t.environment global environment variable contains the same content as the Classic version of the $L.env record).

In out-of-box Hybrid versions of Service Manager, all the related display screens, display options, and processes in the Change Management module are updated. For example, in "cm.view.display_reopen", `evaluate(reopen in $L.env)` is updated to `(filename($L.file)="cm3r" and evaluate (reopen in $G.cm3r.environment) or filename($L.file)="cm3t" and evaluate(reopen in $G.cm3t.environment))`.

However, you must make equivalent updates for any tailoring you have made that uses $L.env as a "cm3rprofile" record.

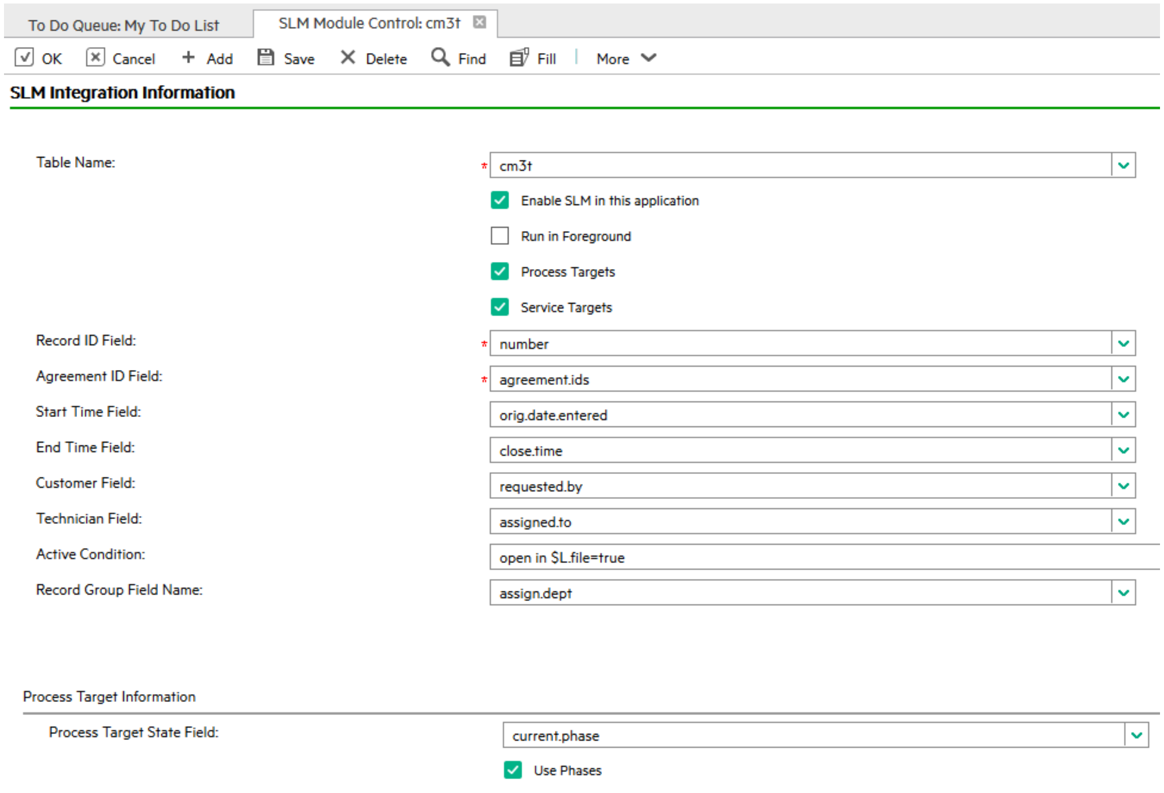
Revert to status-based SLM for Change Tasks

> **Note:** The following step applies only to customers who upgrade to Service Manager Hybrid from a system that does not have PDCP3 installed.

Service Level Management (SLM) for Change Tasks in Codeless versions of Service Manager is based on phases, whereas in Classic versions it is based on statuses. Therefore, you must revert to status-based SLM for Change Tasks in order for your existing SLAs to work with the migrated workflows.

To do this, follow these steps:

1. In the System Navigator, click **Service Level Management** > **Administration** > **Configure Application**.
2. In the **Table Name** drop-down list, select **cm3t**.
3. In the Process Target Information section, click to clear the **Use Phases** option, and then configure the **Process Target State Progression** and **Standard Alerts** fields as required.



SLM Integration Information

Table Name:	cm3t
	✓ Enable SLM in this application
	☐ Run in Foreground
	✓ Process Targets
	✓ Service Targets
Record ID Field:	number
Agreement ID Field:	agreement.ids
Start Time Field:	orig.date.entered
End Time Field:	close.time
Customer Field:	requested.by
Technician Field:	assigned.to
Active Condition:	open in $L.file=true
Record Group Field Name:	assign.dept

Process Target Information

Process Target State Field:	current.phase
	✓ Use Phases

Review the "form edit" condition in workflow phases

The “form edit” condition in Process Designer-based workflows determines whether a form is editable when a user views a record. In Service Manager Classic, this control condition may be located different places. When the Applications Upgrade Utility creates the migrated workflows for Hybrid mode, the condition settings are migrated from the default location of the condition in Service Manager Classic:

- Service Desk: The condition is copied from the state input condition
- Change: The condition is copied from the legacy change phase

If you have configured the condition in any other location, it will not be automatically migrated. In this situation, you must manually combine the Classic condition settings into the “form edit” condition.

> **Note:** Whilst we recommend that you use the “form edit” condition in workflow phases in Service Manager Hybrid and Codeless, the “IO control” condition for the display screen still works. If you set both conditions, the form is editable only when both condition are met.

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 Classic 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, the Globalvariables, and where the data comes from.

We recommend that you modify the data in the ModuleStatus database according to your needs. For example, you must update the ModuleStatus list for the Incident module to add or remove statuses according to your requirements.

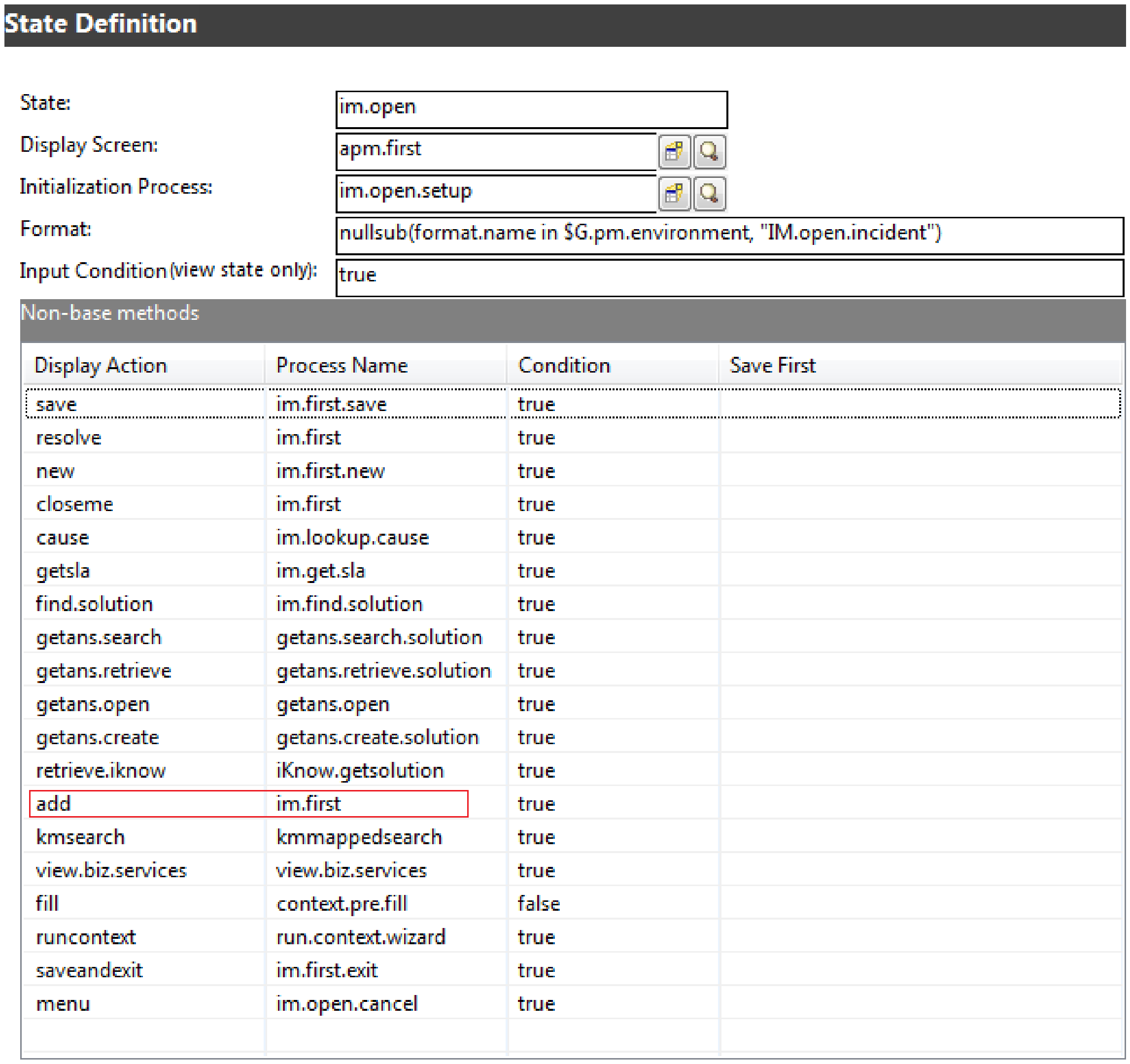
Module	Global list name	Process Designer statuses global list variable	Details
Incident	Incident Local Statuses	$G.imStatuses	Retrieve from file ModuleStatus with Limiting SQL module="probsummary"
Service Desk	Interaction Local Statuses	$G.sdStatuses	Retrieve from file ModuleStatus with Limiting SQL 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 Classic Change modules use the same status list.

Update the eventin service

If you use the eventin service to create Incidents, and if the RAD applications 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, first identify which process maps to the "add" action in the Classic Open State. The following illustration uses the im.first process as an example from an out-of-box system.



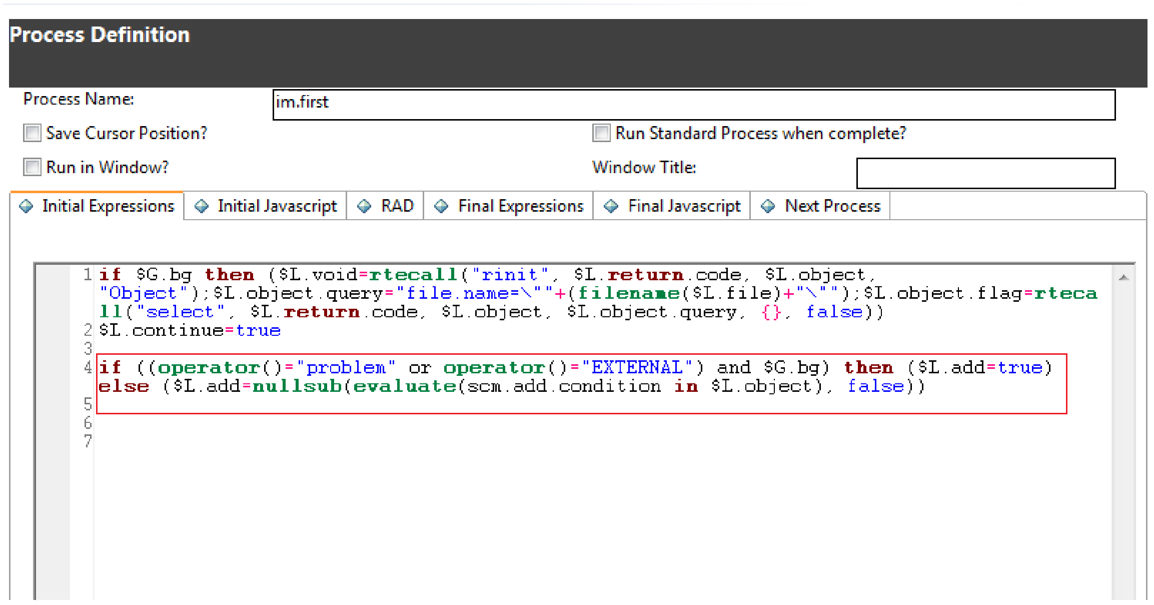
State Definition

State:	im.open
Display Screen:	apm.first
Initialization Process:	im.open.setup
Format:	nullsub(format.name in $G.pm.environment, "IM.open.incident")
Input Condition (view state only):	true

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	

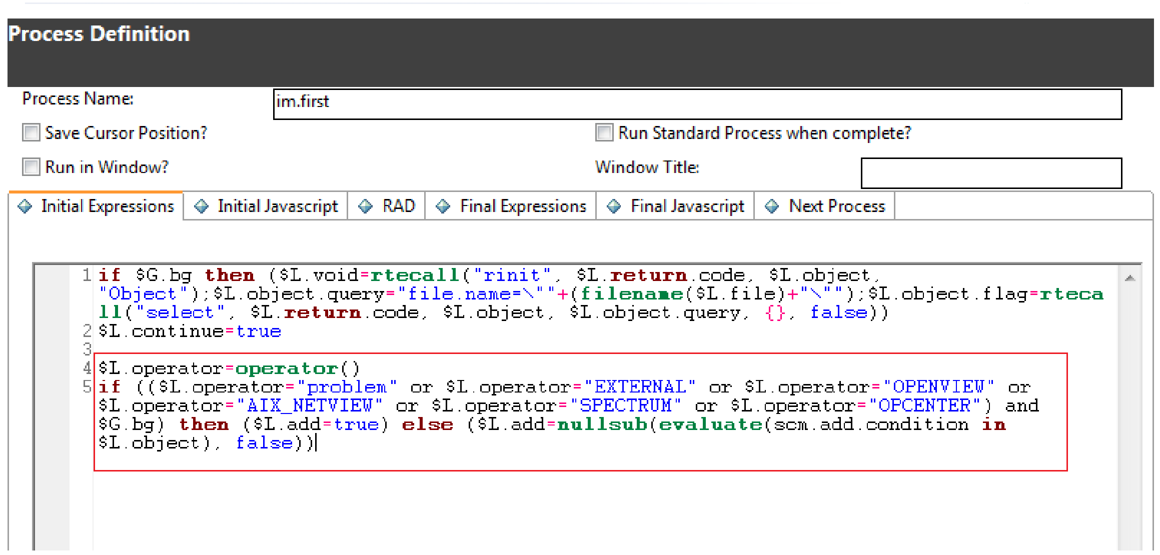
In out-of-box systems earlier than Service Manager 9.41, only the "problem" and "EXTERNAL" operators have permission to add Incident records from the backend.



im.first

```
if $G.bg then ($L.void=rtecall("rinit", $L.return.code, $L.object,
"Object");$L.object.query="file.name=\""+(filename($L.file)+"\"");$L.object.flag=rteca
ll("select", $L.return.code, $L.object, $L.object.query, {}, false))
$L.continue=true

if ((operator()="problem" or operator()="EXTERNAL") and $G.bg) then ($L.add=true)
else ($L.add=nullsub(evaluate(scm.add.condition in $L.object), false))
```

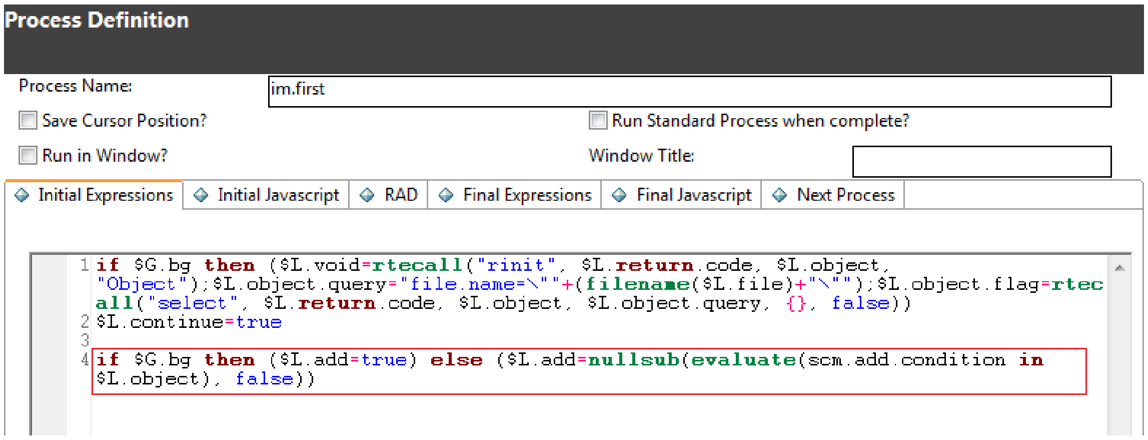
Make sure that the User IDs of your eventin services are configured as demonstrated in the following image.



im.first

```
if $G.bg then ($L.void=rtecall("rinit", $L.return.code, $L.object,
"Object");$L.object.query="file.name=\""+(filename($L.file)+"\"");$L.object.flag=rteca
ll("select", $L.return.code, $L.object, $L.object.query, {}, false))
$L.continue=true

$L.operator=operator()
if (($L.operator="problem" or $L.operator="EXTERNAL" or $L.operator="OPENVIEW" or
$L.operator="AIX_NETVIEW" or $L.operator="SPECTRUM" or $L.operator="OPCENTER") and
$G.bg) then ($L.add=true) else ($L.add=nullsub(evaluate(scm.add.condition in
$L.object), false))
```

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.



Process Definition

Process Name: im.first

☐ Save Cursor Position? ☐ Run Standard Process when complete?

☐ Run in Window? Window Title:

```
if $G.bg then ($L.void=rtecall("rinit", $L.return.code, $L.object,
"Object");$L.object.query="file.name=\""+(filename($L.file)+"\"");$L.object.flag=rtec
all("select", $L.return.code, $L.object, $L.object.query, {}, false))
$L.continue=true

if $G.bg then ($L.add=true) else ($L.add=nullsub(evaluate(scm.add.condition in
$L.object), false))
```

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 applications	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 axces.database RAD applications (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.

Replace the environmental variables

In Service Manager Classic, the `$L.env` variable stores the profile record. However, in Service Manager Codeless, the `$L.env` variable stores the tableAccess record. Therefore, all code in the Change Management module that uses the `$L.env` variable (mostly, this relates to Display Options) must be updated to use the `$G.cm3r.environment` or `$G.cm3t.environment` variables.

For example, the display condition of the **cm.view.display_approve** display option must be updated as follows:

Original:

```
evaluate(approvals in $L.env)
```

After update:

```
evaluate(approvals in $G.cm3r.environment) or evaluate(approvals in
$G.cm3t.environment)
```

Optional manual migration tasks

Configure the Change Management module to use the new Task Planner

Note: The following step applies only to customers who upgrade to Service Manager Hybrid from a 9.3x system that has PDCP3 or PDCP4 installed.

If you are upgrading to Service Manager 9.50 Hybrid from a Classic version of Service Manager, you must add the Task Planner form to your Change format, and then run the "Legacy change/task data to new taskplanner" data migration script, if you want 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 (this disables the previous version of the Task Planner). To do this, you can 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	The following object-level rule sets are configured to use the new Task Planner:
		• After successful add:
		`.common.taskplan.noneplantasktochangeplan`
		• On enter:
		`.common.taskplan.validateoutputontask`	

Record type	Record name	Modifications	Comments

		• After successful enter:
		`.common.taskplan.sync.task.titleandstatus.`
		`tochangeplan`
		`.common.taskplan.updatedependenttaskstatus`
		• Initialization:
		`.common.taskplan.taskticket.init`
		• On display:
		`.common.workflow.init.vars`
		• On update:
		`.common.taskplan.savecontextconfig`
		• After successful update:
		`.common.taskplan.sync.task.titleandstatus.tochangeplan`	

Process	changeModel.buildtable	The following initial JavaScript is commented: `lib.changeModel.saveFromXML(true);`	This record was used by optionid 441 to save taskxml files to the changeModel table for the Classic Task Editor.
Process	changeModel.init	The following initial JavaScript is commented: `lib.changeModel.createXML(vars.$L_file);`	This record was used to initialize a taskxml file to be displayed by the Classic Task Editor.
Process	changeModel.save	The following initial JavaScript is commented:	This record was used

Record type	Record name	Modifications	Comments
		`lib.changeModel.saveFromXML();`	to save taskxml files to the changeModel table.
Process	change.open.save	• RAD expressions are evaluated before the change.openPlanTasks RAD call. • The following expression is removed: `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, $L.phase))` • Disabled calling the change.openPlanTasks RAD.	
Process	change.update.save	• RAD expressions are evaluated before the apm.mb.ok RAD call. • The following expressions are removed: ◦ `if (filename($L.file)="cm3r") then ($L.changePlan=jscall ("changePlan.getPlan", $L.file))` ◦ `$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 $L.file))` ◦ `if ($L.task.list~="") then ($L.continue = false;$L.exit="bad.val"; $L.opentask.msg = scmsg(550, "cm3", {$L.task.list}))` ◦ `if ($L.continue=false and not (null($L.opentask.msg))`	

Record type	Record name	Modifications	Comments

		`and $G.bg) then ($L.void=rtecall("msg", $L.rc, $L.opentask.msg))`
		• Disabled calling the apm.mb.ok RAD
		• RAD expressions are evaluated before the change.openPlanTasks RAD call.
		• The following expression is removed:
		`if (filename($L.file)="cm3r") then ($L.success.flg=rtecall ("refresh", $L.errcode, $L.changePlan))`
		• Disabled calling the change.openPlanTasks RAD	

Process	changePlan.buildtable	• RAD expressions are evaluated before the change.openPlanTasks RAD call: • The following expression is removed: ◦ `if (filename($L.file)="cm3r") then ($L.changePlan = jscall("changePlan.getPlan", $L.file))` • Post RAD expressions: the following expressions are removed: ◦ `$L.void=jscall("changePlan.createXML", $L.file)` ◦ `$L.exit="refresh"` • Disabled calling the change.openPlanTasks RAD • The following initial JavaScript is commented: `lib.changePlan.saveFromXML(true);`	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 use conflict resolution or manually change the code elements according to following table.

Record type	Record name	Modifications	Comments
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.
	chm.task chm.standard.registration chm.standard.pln.sch chm.standard.execution chm.standard.pir 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	A new Task Context tab is added to dynamically display context-related record information that is planned during the task plan.	

Record type	Record name	Modifications	Comments
	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

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 "Appendix B: Data migration" on page 154.

Upgrade to the Process Designer related record subform

The functionality of related records in Process Designer-based systems and in Classic systems is different. However, Classic related record functionality still works in Process Designer-based environments, and it is therefore not mandatory for you to replace this functionality.

> **Caution:** If you decide not to use the Process Designer-based related record functionality, rule types such as "Run Action" will not work.

To use the Process Designer-based related record functionality, simply update the "Related Records tab" in the corresponding forms to implement the Process Designer-based related record subform (for example, `im.incident.subform.related` for the Incident module). Then, migrate the related record data.

Use the Process Designer workflow viewer in your Change Management 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. The Applications Upgrade Utility updates your forms during the upgrade process to enable this feature.

However, if you previously customized Service Manager so that the cm3r.sub.workflow.g subform was not used to hold the Classic workflow widget in the Change Management module, you must manually implement the new Workflow Viewer widget in your Change Management forms.

To do this, follow these steps:

1. In Forms Designer, open the "chm.cm3r.sub.workflow.g" out-of-box Process Designer workflow subform. Then, use design mode to copy the content of this subform.
2. In Forms Designer, open the form to which you want to add the new Workflow Viewer widget. Then, overwrite the content for the existing Classic workflow widget subform with the content that you copied from the "chm.cm3r.sub.workflow.g" subform.

Configure the Search form

Search forms in Process Designer-based systems and in Classic 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 Classic search form, you must update the SearchConfig record to revert the changes applied by the Applications Upgrade Utility. To do this, set the value of the **Search Format** field to the appropriate Classic search form for that table.

The following table lists the mapping between Process Designer-based search forms and their Classic equivalents.

Table name	Classic search form	Process Designer search form
probsummary	advFind.incident.search	im.advFind.incident.search
cm3r	advFind.search.change	advFind.search.chm
cm3t	advFind.search.task	advFind.search.task
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 Classic systems is different.

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

The following example uses the "Check similar problems" Incident solution matching setting to demonstrate how to perform the migration.

1. 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. From the **More** menu, click **Administration** > **Add New Setting**, and then add a new setting.
2. Set the security rights value for this new Setting.
3. 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.
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 Classic systems.

Module	Classic table	Process Designer-based table
Incident	Category	imCategory
	subcategory	imSubcategory
	producttype	imArea

Module	Classic table	Process Designer-based table
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 Classic systems, the category field in the Interaction and Incident forms are not read-only; users can change the category of an interaction. 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 Classic systems is different. After you upgrade to Service Manager 9.50 Hybrid, the escalation wizard no longer appears when you click **Escalate** in an Interaction, and the Change is not created in the background. Instead, the Change form is launched for you to input the details.

To revert to using the Classic 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 Classic 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 Classic systems, the Incident is created in the background without displaying the Incident form to the end user.

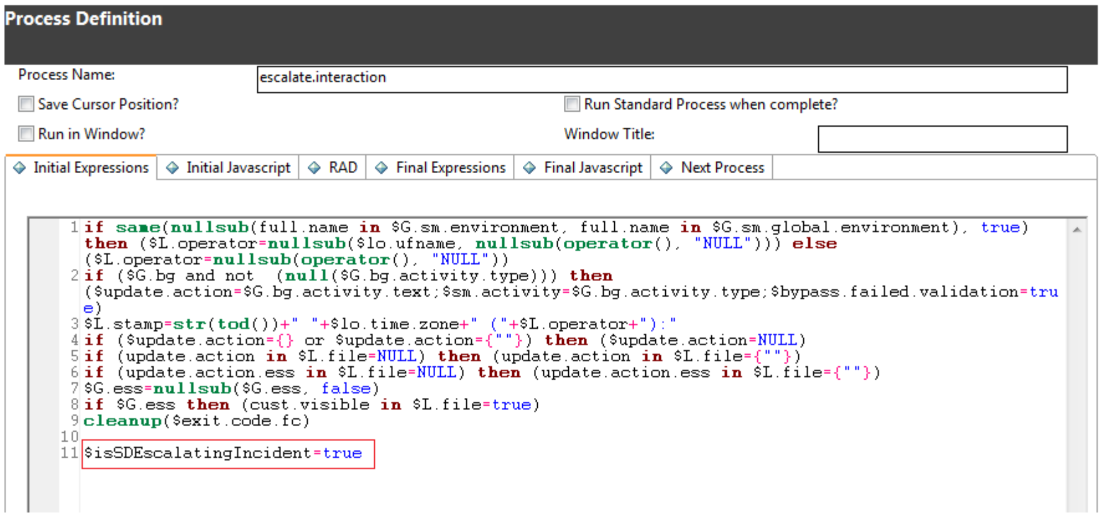
To revert to the old behavior, you must add the **Escalate** button back to the Interaction form and remove the **Cancel** button from the Incident form. To do this, follow these steps:

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 $L.filed and nullsub($isnewweb, false)=true or jscall
   ("sdUtil.isInteractionEscalatedOrFulfilled", $L.file)=false) and nullsub
   (category in $L.filed, "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
   ```



Process Definition

Process Name: escalate.interaction

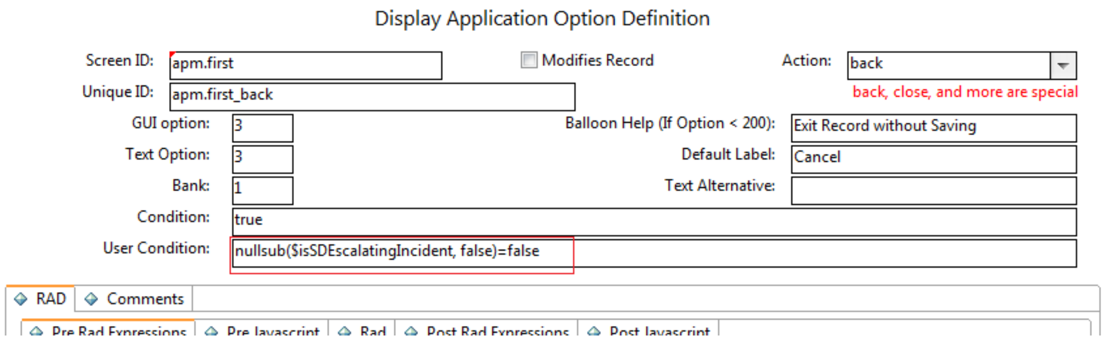
Save Cursor Position? ☐ Run Standard Process when complete? ☐

Run in Window? ☐ Window Title:

```
if same(nullsub(full.name in $G.sm.environment, full.name in $G.sm.global.environment), true)
then ($L.operator=nullsub($lo.ufname, nullsub(operator(), "NULL"))) else
($L.operator=nullsub(operator(), "NULL"))
if ($G.bg and not  (null($G.bg.activity.type))) then
($update.action=$G.bg.activity.text;$sm.activity=$G.bg.activity.type;$bypass.failed.validation=tru
e)
$L.stamp=str(tod())+" "+$lo.time.zone+" ("+$L.operator+"):"
if ($update.action={} or $update.action={""}) then ($update.action=NULL)
if (update.action in $L.file=NULL) then (update.action in $L.file={""})
if (update.action.ess in $L.file=NULL) then (update.action.ess in $L.file={""})
$G.ess=nullsub($G.ess, false)
if $G.ess then (cust.visible in $L.file=true)
cleanup($exit.code.fc)

$isSDEscalatingIncident=true
```

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

```
nullsub($isSDEscalatingIncident, false)=false
```



Display Application Option Definition

Screen ID:	apm.first	Action:	back
Unique ID:	apm.first_back		back, close, and more are special
GUI option:	3	Balloon Help (If Option < 200):	Exit Record without Saving
Text Option:	3	Default Label:	Cancel
Bank:	1	Text Alternative:	
Condition:	true		
User Condition:	nullsub($isSDEscalatingIncident, false)=false		

Modifies Record ☐

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 Classic Problem module stores Known Error records in the knownerror table. Therefore, if you have made any customizations to the Classic knownerror table that you want to retain, you must manually reapply these customizations to the rootcause table.

Remove the Classic Request Management dashboard and report definitions

After you upgrade to Service Manager Hybrid, the Classic Request Management dashboard and report definitions remain in the system. If you do not want to run the Classic Request Management and Process Designer-based Request Fulfillment modules in parallel, you can remove the Classic Request Management dashboard and report definitions.

To remove the Classic 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 Classic 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 Classic Request Management and Process Designer-based Request Fulfillment to run in parallel

Migration from Classic Request Management to Process Designer-based Request Fullfillment 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 Hybrid enables you to run the Classic 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 **isNonPDRequestInParallel** function, un-comment the line `$G.nonpd.request.in.parallel variable`, and then set its value to "True."

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 Classic "Request Management" menu item back to the 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 Classic request menu items.

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 Classic request menu items.

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 Classic Request menu items.

Step 3: Update the queries in Inbox views

1. Search all Approval inboxes andcheck the query condition in each record to determine whether it uses only Classic 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 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})
```

2. 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.
3. 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
   ```

   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})
   ```

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 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.

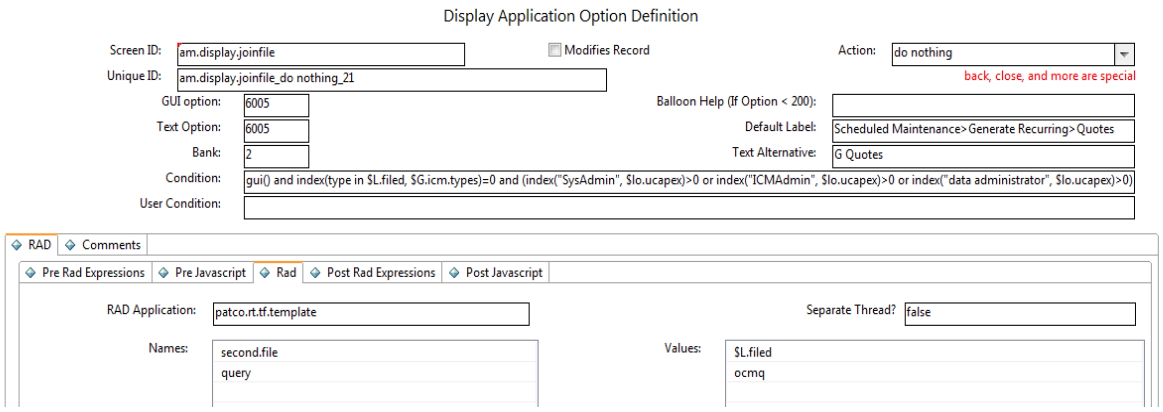
Step 5: Configure search

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.



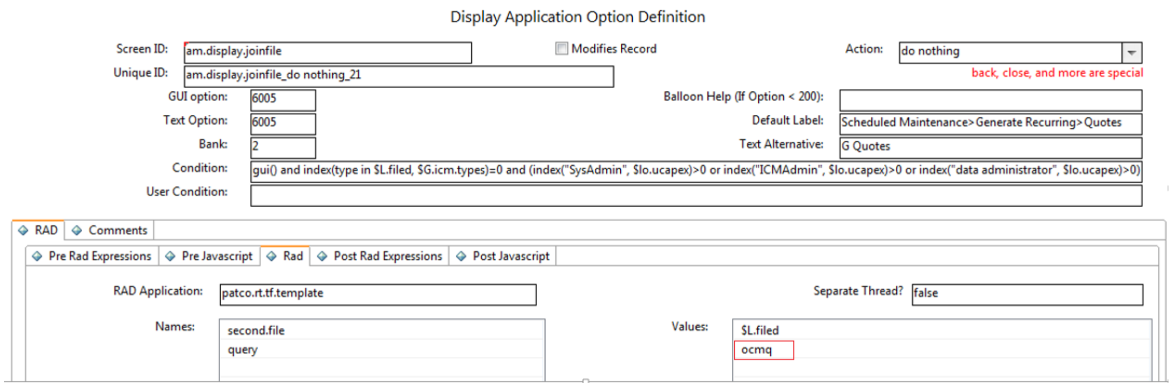
Display Application Option Definition

Screen ID: am.display.joinfile
☐ Modifies Record
Action: do nothing
back, close, and more are special
Unique ID: am.display.joinfile_do nothing_21
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:	Values:
second.file	$L.filed
query	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: am.display.joinfile
☐ Modifies Record
Action: do nothing
back, close, and more are special
Unique ID: am.display.joinfile_do nothing_21
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:	Values:
second.file	$L.filed
query	ocmq

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

1. 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).
2. 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).
3. Restore the Next wizard for Request Management.

Step 9: Restore the Classic 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.showRMProfile']=false;

}
```

After update

```
if(lib.ProcessDesignerEnablement.isRequestEnabled() && vars
['$G.nonpd.request.in.parallel']!==true)

{

vars['$L.showRMProfile']=false;

}
```

Step 10: Assignment delegation

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

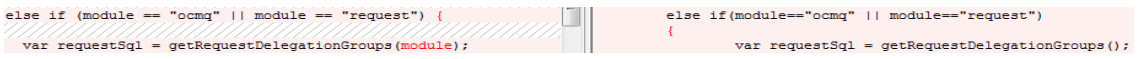
To do this, you must modify the ApprovalDelegationGroups and ApprovalUtil script libraries to support both Classic and Process Designer-based Request delegation.

> **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 Classic Codeless request modules to work in parallel, you must add the module (file name) information to the following functions, which are related to request delegation:

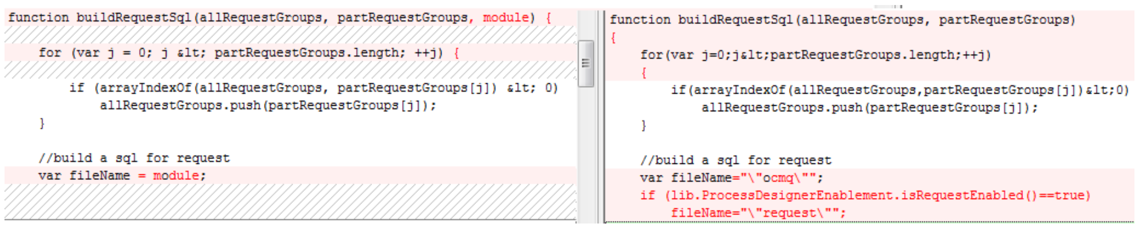
- addDelegateGroups



```
else if (module == "ocmq" || module == "request") {

  var requestSql = getRequestDelegationGroups(module);
```

```
        else if(module=="ocmq" || module=="request")
        {
                var requestSql = getRequestDelegationGroups();
```

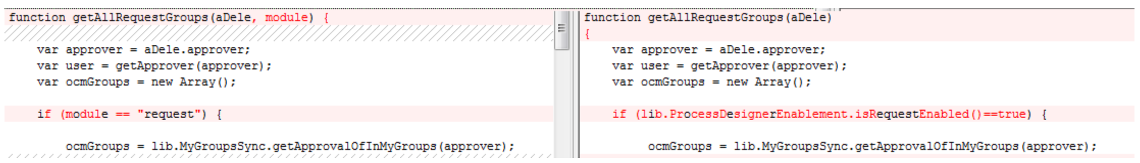
- buildRequestSql



```
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;
```

```
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\"";
```

- getAllRequestGroups

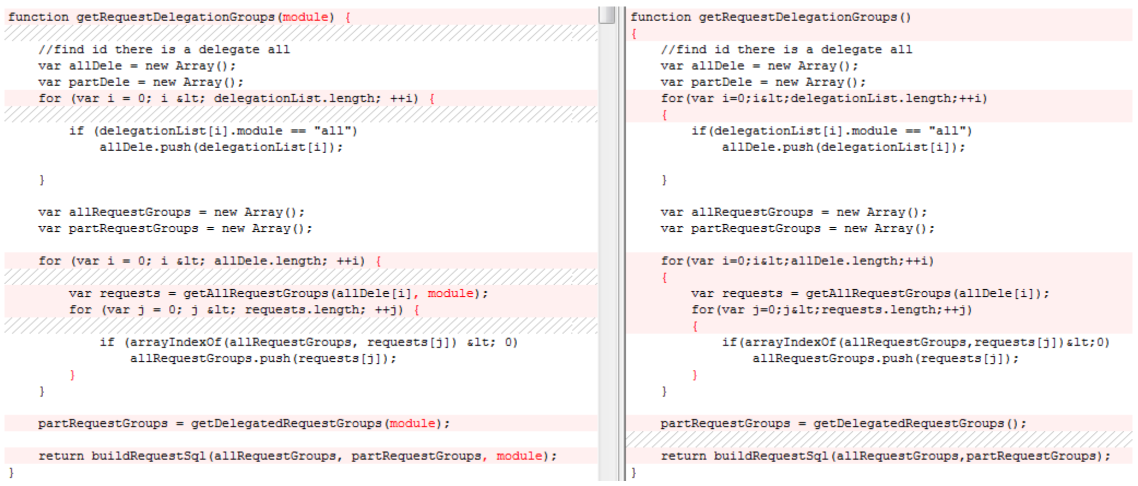


```
function getAllRequestGroups(aDele, module) {

    var approver = aDele.approver;
    var user = getApprover(approver);
    var ocmGroups = new Array();

    if (module == "request") {

        ocmGroups = lib.MyGroupsSync.getApprovalOfInMyGroups(approver);
```

```
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);
```

- getRequestDelegationGroups

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

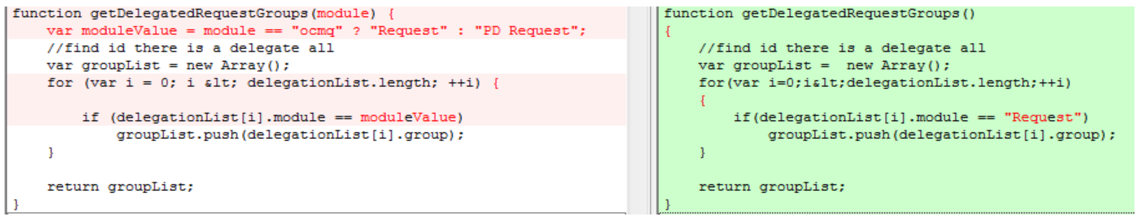


```
function getRequestDelegationGroups(module) {

    //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);
}
```

```
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);
}
```

- getDelegatedRequestGroups

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

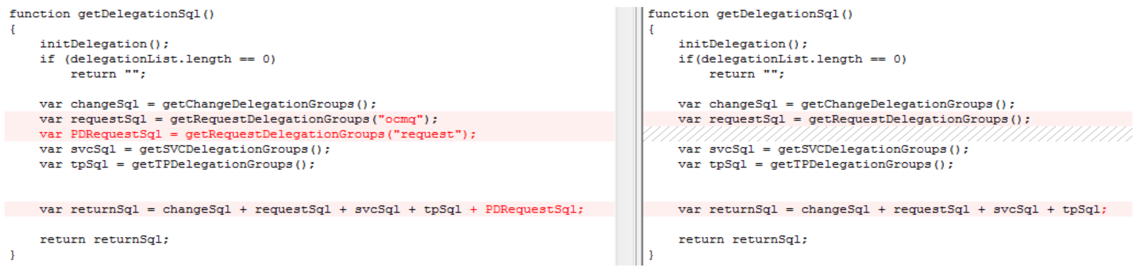


```
function getDelegatedRequestGroups(module) {
    var moduleValue = module == "ocmq" ? "Request" : "PD Request";
    //find id there is a delegate all
    var groupList = new Array();
    for (var i = 0; i &lt; delegationList.length; ++i) {

        if (delegationList[i].module == moduleValue)
            groupList.push(delegationList[i].group);
    }

    return groupList;
}
```

```
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;
}
```

- 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.



```
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 returnSql = changeSql + requestSql + svcSql + tpSql + PDRequestSql;

    return returnSql;
}
```

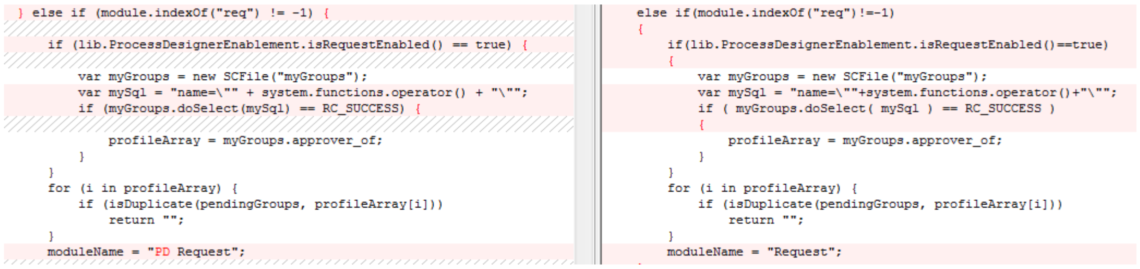
```
function getDelegationSql()
{
    initDelegation();
    if(delegationList.length == 0)
        return "";

    var changeSql = getChangeDelegationGroups();
    var requestSql = getRequestDelegationGroups();

    var svcSql = getSVCDelegationGroups();
    var tpSql = getTPDelegationGroups();


    var returnSql = changeSql + requestSql + svcSql + tpSql;

    return returnSql;
}
```

ApprovalUtil

Functions in this library use "Request" as the delegation module name for the Classic 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 Classic 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 Classic request module and "PD Request" as the delegation module name for Process Designer-based request.

- checkDelegates

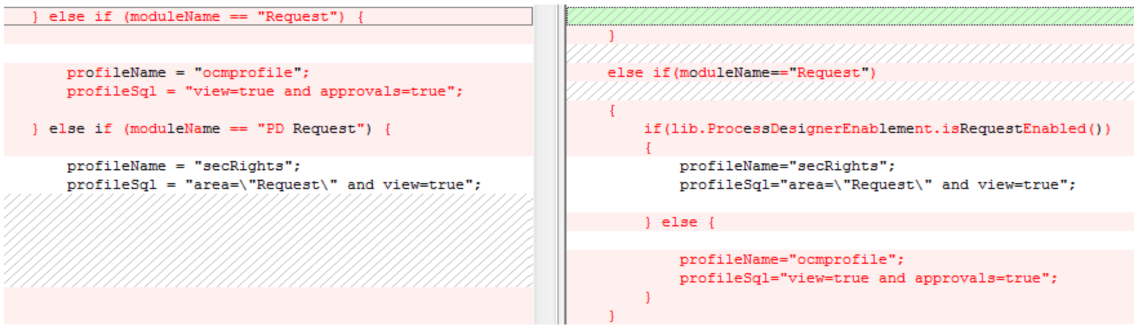


```
} else if (module.indexOf("req") != -1) {

    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";
```

```
    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";
```

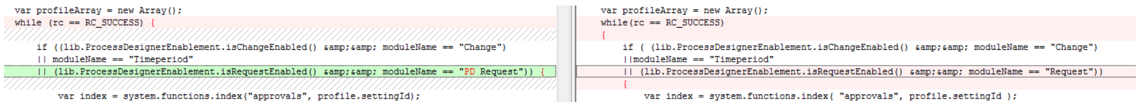
The function uses the profile ("ocmprofile") to check user rights in Classic systems, and uses the role-based access model ("secRole" and "secRights") in Process Designer-based systems.

- getQuery



```
 } 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";
```

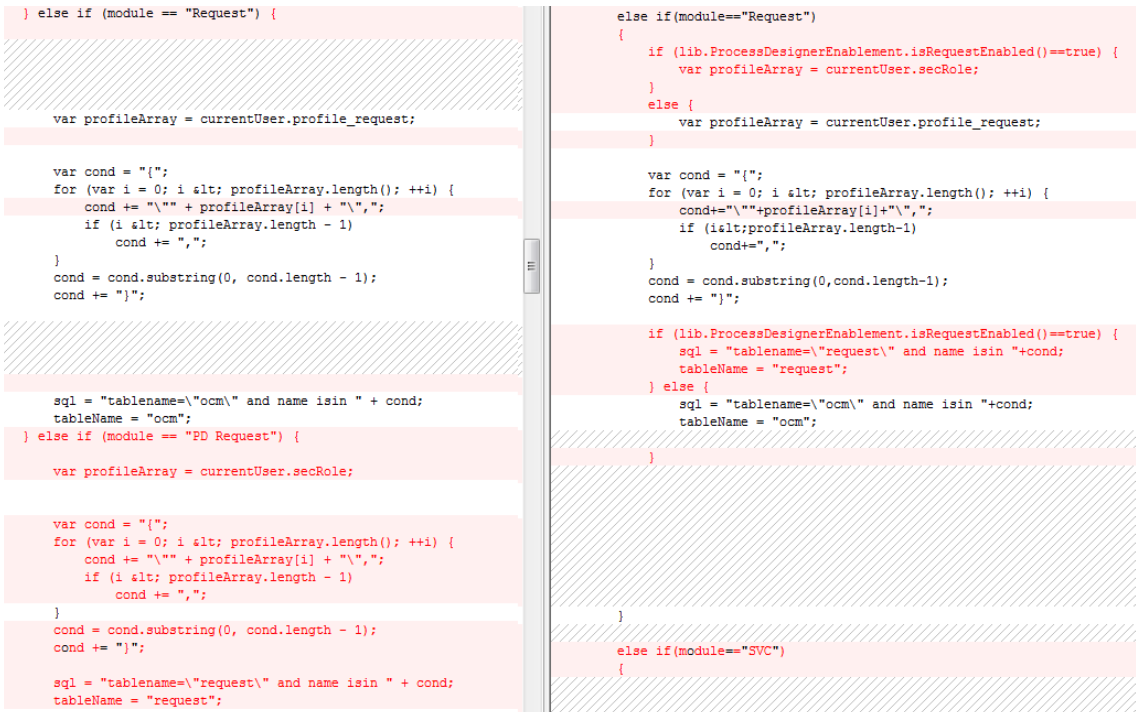
```
    }

    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";
        }
    }
```



```
var profileArray = new Array();
while (rc == RC_SUCCESS) {

    if ((lib.ProcessDesignerEnablement.isChangeEnabled() &amp;&amp; moduleName == "Change")
    || moduleName == "Timeperiod"
    || (lib.ProcessDesignerEnablement.isRequestEnabled() &amp;&amp; moduleName == "PD Request")) {

        var index = system.functions.index("approvals", profile.settingId);
```

```
var profileArray = new Array();
while(rc == RC_SUCCESS)
{
    if ( (lib.ProcessDesignerEnablement.isChangeEnabled() &amp;&amp; moduleName == "Change")
    ||moduleName == "Timeperiod"
    || (lib.ProcessDesignerEnablement.isRequestEnabled() &amp;&amp; moduleName == "Request"))
    {
        var index = system.functions.index( "approvals", profile.settingId );
```

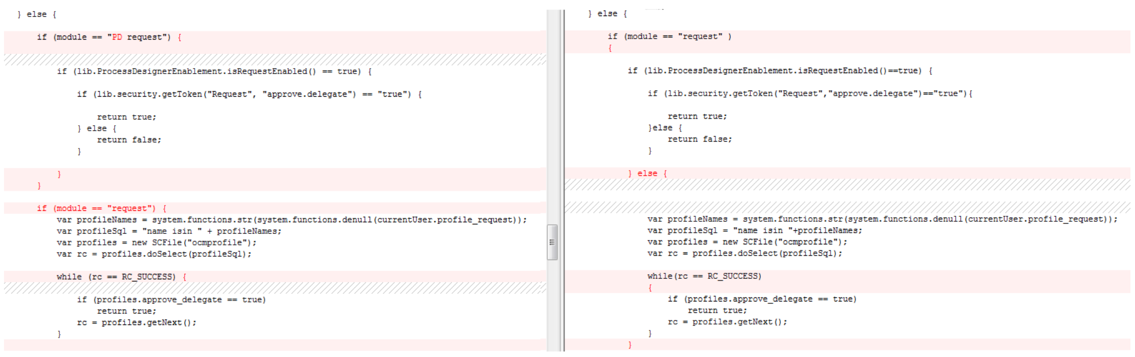
- addFolderQuery



```
} 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";
```

```
    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")
    {
```

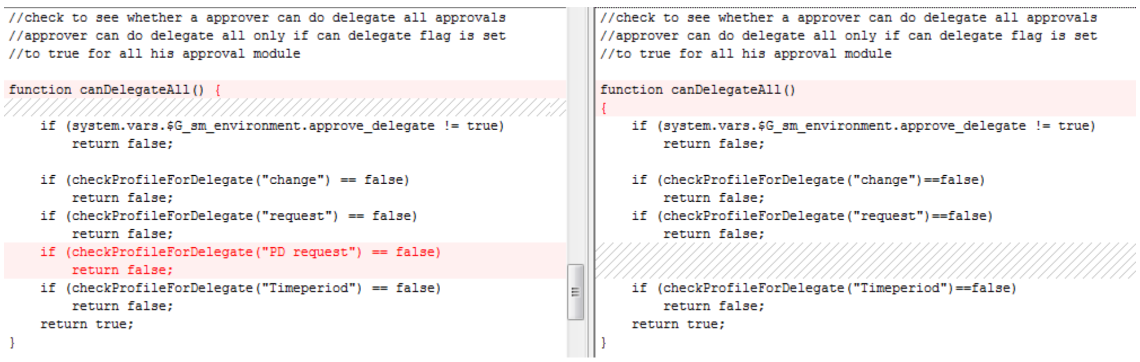
- checkProfileForDelegate



```
} 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.denull(currentUser.profile_request));
        var profileSql = "name isin " + profileNames;
        var profiles = new SCFile("ocmprofile");
        var rc = profiles.doSelect(profileSql);

        while (rc == RC_SUCCESS) {

            if (profiles.approve_delegate == true)
                return true;
            rc = profiles.getNext();
        }
```

```
} 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.denull(currentUser.profile_request));
            var profileSql = "name isin "+profileNames;
            var profiles = new SCFile("ocmprofile");
            var rc = profiles.doSelect(profileSql);

            while(rc == RC_SUCCESS)
            {
                if (profiles.approve_delegate == true)
                    return true;
                rc = profiles.getNext();
            }
        }
```

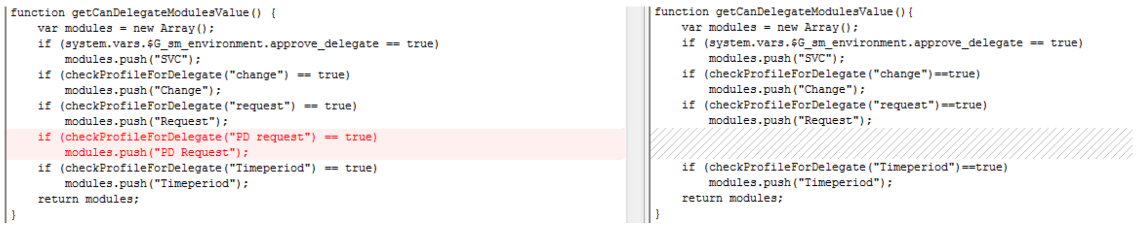
- canDelegateAll



```
//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;
}
```

```
//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;
}
```

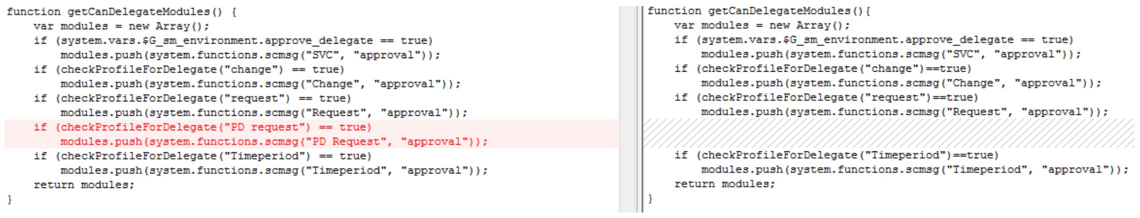
- getCanDelegateModulesValue



```
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;
}
```

```
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;
}
```

- getCanDelegateModules



```
function getCanDelegateModules() {
    var modules = new Array();
    if (system.vars.$G_sm_environment.approve_delegate == true)
        modules.push(system.functions.scmsg("SVC", "approval"));
    if (checkProfileForDelegate("change") == true)
        modules.push(system.functions.scmsg("Change", "approval"));
    if (checkProfileForDelegate("request") == true)
        modules.push(system.functions.scmsg("Request", "approval"));
    if (checkProfileForDelegate("PD request") == true)
        modules.push(system.functions.scmsg("PD Request", "approval"));
    if (checkProfileForDelegate("Timeperiod") == true)
        modules.push(system.functions.scmsg("Timeperiod", "approval"));
    return modules;
}
```

```
function getCanDelegateModules(){
    var modules = new Array();
    if (system.vars.$G_sm_environment.approve_delegate == true)
        modules.push(system.functions.scmsg("SVC", "approval"));
    if (checkProfileForDelegate("change")==true)
        modules.push(system.functions.scmsg("Change", "approval"));
    if (checkProfileForDelegate("request")==true)
        modules.push(system.functions.scmsg("Request", "approval"));

    if (checkProfileForDelegate("Timeperiod")==true)
        modules.push(system.functions.scmsg("Timeperiod", "approval"));
    return modules;
}
```

Step 11: Use the model table for Classic 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 Classic 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

Link name	Field
"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
"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

Link name	Field
"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
"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

Link name	Field
"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
"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. Classic Request Management and Process Designer-based Request Fullfillment 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 Classic 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*>

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	
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, refer to "Upgrade to the Process Designer related record subform".

If you want your system to allow the creation of related records between the Classic Request Management and Process Designer-based Request Fullfillment modules, you must configure the screlationtype 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 screlationtype table.

For more information about how to perform data migration for screlation records, refer to "Related Records data migration".

After you have done this, you can create related quotes from incident records.

Step 14: Restart the server

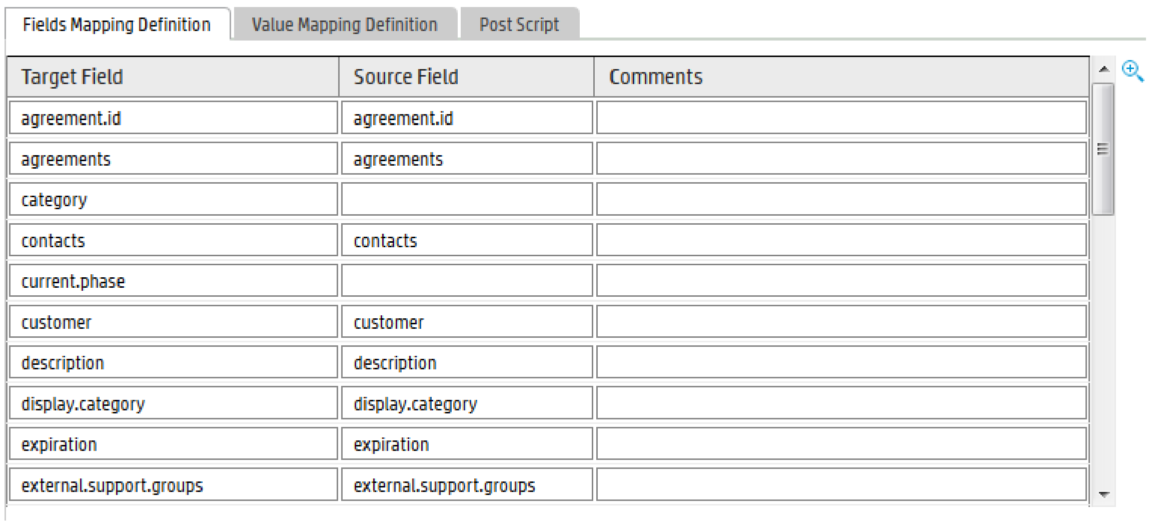
After you have configured Request Management and Request Fulfillment to run in paralel, you must restart the Service Managerserver.

Appendix B: Data migration

This appendix describes the following topics that are related to the data migration tool.

Field mapping and value mapping

Field mapping defines the mapping between the fields in the source file and target file, as shown in the following figure.

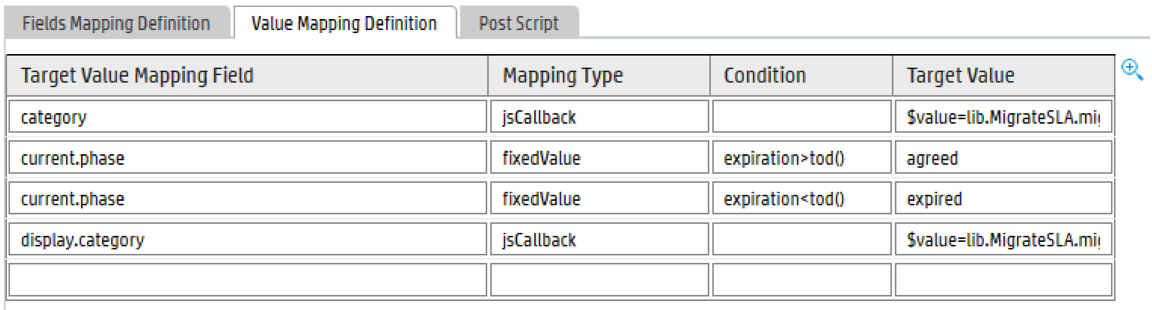


Target Field	Source Field	Comments
agreement.id	agreement.id	
agreements	agreements	
category		
contacts	contacts	
current.phase		
customer	customer	
description	description	
display.category	display.category	
expiration	expiration	
external.support.groups	external.support.groups	

Note:

- The Target field cannot be empty.
- The Source and Target fields must already exist in the table.
- If there is a structure field mapping and you are not in Batch Update mode, both fields must have the exact same structure definition.
- Array of structure and structure fields are not supported when in Batch Update mode.
- The M2 and alias table are not supported in Batch Update mode.
- You cannot use jscallback in Batch Update mode.

Value mapping defines the value mapping for each mapped field.



Target Value Mapping Field	Mapping Type	Condition	Target Value
category	jsCallback		$value=lib.MigrateSLA.mi
current.phase	fixedValue	expiration>tod()	agreed
current.phase	fixedValue	expiration<tod()	expired
display.category	jsCallback		$value=lib.MigrateSLA.mi

Note:

- For the fields that do not have a value mapping defined, the default is direct value copy.
- You can define multiple value mappings for the same field.

The following table describes each column in the Value Mapping Definition tab.

Column	Description
Target value mapping field	The field from target field that needs to define value mapping.

Mapping Type	The following three types are supported:
	1. **fixedValue:** target field value = the fixed value defined in "Target Value" column. The supported fixed value are number, string, datetime, and Boolean. You cannot specify an expression in a fixed table, nor a fixed value for structure and array field type.
	2. **sourceField:** target field value = source field value defined in "Target Value" column.
	3. **jsCallback:** target field value = a JavaScript script to dynamically set the value. Three keywords ($sourceTable, $targetable, $value) can be used if the direct script code is written in the Target Value column:
	◦ $sourceTable, $targetTable reference the field value in the source table field and the target table field, respectively as shown in the following example:
	`var status = $sourceTable["rcStatus"],$targeTable["rcStatus"] = "open"`
	◦ $value is used to specify the current target field value as shown in the following example:
	`$value=vars.$G_user_role retrieves the value from another variable.`
	jsCallback can also support directly calling a JavaScript function defined in Script library, such as the following:
	`lib.UpgradeScript.migrateData($sourceTable,.$targetTable);`

Condition	If the condition is true, then the value mapping is applied during the migration. The condition expression supports the following: 1. Arithmetic operators: >, <, >=, <=, =, ~= 2. Logical operators: or, and 3. The "null" keyword can be used as a null value in the formula.
Target Value	Depending on the Mapping Type setting, Target Value can be a fixed value string, or a source field name or a JavaScript script.

Batch Update mode

When the target file equals the source file, the migration tool executes the update operation on the same table. In addition, an additional SQL batch update mode is automatically enabled to allow the migration tool to directly update the underlying database.

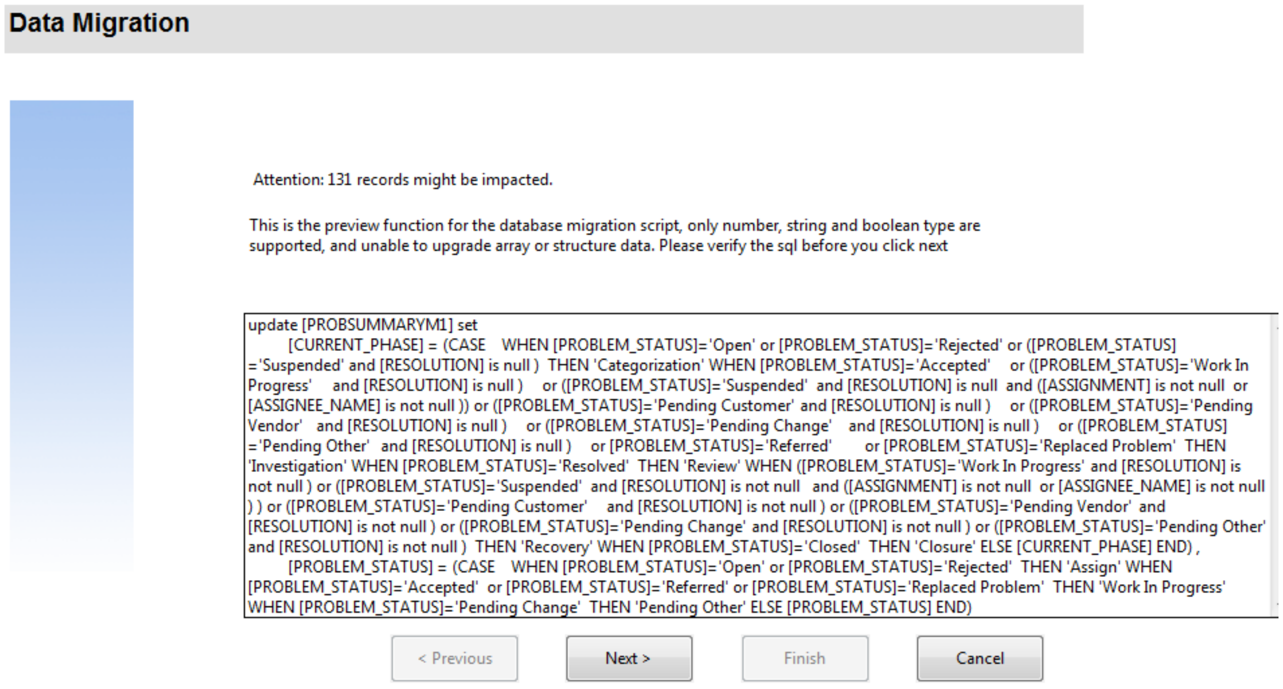
> **Note:** You can run the migration in Batch Update mode to get better performance when the source and target files are the same. The Batch Update mode cannot be used when you migrate data between two different files, such as from Legacy known error to Problem.

> **Note:** Some of the out-of-box migration scripts cannot be run in Batch Update mode. For more information, see "Appendix B: Data migration" on page 154.

Batch update mode can only be enabled when the target file equals the source file. If it is enabled, a batch SQL update is generated in a popup wizard instead of using the Service Manager API to update the data records one by one. After confirmation, the batch SQL update is sent to the underlying database.

If Batch Update mode is enabled, you should note the following:

- You should update the data records directly via the database without triggering Service Manager. Otherwise, updating the data records will trigger additional background process.
- The Filter Condition field should follow the RDBMS SQL grammar for your database. Otherwise, Filter Condition should follow query statement grammar of Service Manager.
- The “Condition” field in the Value Mapping Definition should follow the RDBMS SQL grammar for your database. Otherwise, Filter Condition should follow query statement grammar of Service Manager.



Data Migration

Attention: 131 records might be impacted.

This is the preview function for the database migration script, only number, string and boolean type are supported, and unable to upgrade array or structure data. Please verify the sql before you click next

```
update [PROBSUMMARYM1] set
       [CURRENT_PHASE] = (CASE   WHEN [PROBLEM_STATUS]='Open' or [PROBLEM_STATUS]='Rejected' or ([PROBLEM_STATUS]
='Suspended' and [RESOLUTION] is null )  THEN 'Categorization' WHEN [PROBLEM_STATUS]='Accepted'    or ([PROBLEM_STATUS]='Work In
Progress'    and [RESOLUTION] is null )    or ([PROBLEM_STATUS]='Suspended'  and [RESOLUTION] is null  and ([ASSIGNMENT] is not null  or
[ASSIGNEE_NAME] is not null )) or ([PROBLEM_STATUS]='Pending Customer' and [RESOLUTION] is null )    or ([PROBLEM_STATUS]='Pending
Vendor'   and [RESOLUTION] is null )    or ([PROBLEM_STATUS]='Pending Change'    and [RESOLUTION] is null )    or ([PROBLEM_STATUS]
='Pending Other'   and [RESOLUTION] is null )    or [PROBLEM_STATUS]='Referred'      or [PROBLEM_STATUS]='Replaced Problem'  THEN
'Investigation' WHEN [PROBLEM_STATUS]='Resolved'  THEN 'Review' WHEN ([PROBLEM_STATUS]='Work In Progress' and [RESOLUTION] is
not null ) or ([PROBLEM_STATUS]='Suspended'  and [RESOLUTION] is not null   and ([ASSIGNMENT] is not null  or [ASSIGNEE_NAME] is not null
) ) or ([PROBLEM_STATUS]='Pending Customer'    and [RESOLUTION] is not null ) or ([PROBLEM_STATUS]='Pending Vendor'  and
[RESOLUTION] is not null ) or ([PROBLEM_STATUS]='Pending Change' and [RESOLUTION] is not null ) or ([PROBLEM_STATUS]='Pending Other'
and [RESOLUTION] is not null )  THEN 'Recovery' WHEN [PROBLEM_STATUS]='Closed'  THEN 'Closure' ELSE [CURRENT_PHASE] END) ,
       [PROBLEM_STATUS] = (CASE   WHEN [PROBLEM_STATUS]='Open' or [PROBLEM_STATUS]='Rejected'  THEN 'Assign' WHEN
[PROBLEM_STATUS]='Accepted'  or [PROBLEM_STATUS]='Referred' or [PROBLEM_STATUS]='Replaced Problem'  THEN 'Work In Progress'
WHEN [PROBLEM_STATUS]='Pending Change'  THEN 'Pending Other' ELSE [PROBLEM_STATUS] END)
```

Out-of-box data migration settings

The Data Migration Tool provides the following out-of-box data migration settings.

Problem data migration

Considering that Problem does not need to copy data from two different tables, and that it typically has a large volume of data, we recommend that you run the migration by using the batch update mode (especially for migrating a large volume of closed tickets).

Note: After you migrate the Problem data, remove the “Problem_Library_disabled_by_PDHD” knowledge base as this legacy library is no longer used.

The out-of-box example data migration is based on the following three key fields:

- status
- category
- current.phase

Note: You can add additional fields if needed, but do not remove these three fields.

Problem

Default settings:

Source Table	rootcause
Target Table	rootcause
Filter Condition	category="BPPM"

Field Mapping:

Target Field	Source Field
category	category
current.phase	current.phase
rcStatus	rcStatus

Value Mapping:

Target Value Mapping Field	Mapping Type	Condition	Target Value
category	fixedValue	category="BPPM"	problem
current.phase	fixedValue	(current.phase="Problem Detection, Logging and Categorization" or current.phase="Problem Prioritization and Planning") and rcStatus~="Closed"	Categorization
current.phase	fixedValue	current.phase="Problem Investigation and Diagnosis" and rcStatus~="Closed"	Investigation

Target Value Mapping Field	Mapping Type	Condition	Target Value
current.phase	fixedValue	current.phase="Problem Resolution" and rcStatus~="Closed"	Resolution
current.phase	fixedValue	current.phase="Problem Closure and Review" and rcStatus~="Closed"	Review
current.phase	fixedValue	rcStatus="Closed"	Closure
rcStatus	fixedValue	(current.phase="Problem Detection, Logging and Categorization" or current.phase="Problem Prioritization and Planning") and rcStatus~="Closed" and rcStatus~="Deferred"	Categorize
rcStatus	fixedValue	current.phase="Problem Resolution" and (rcStatus="Pending Vendor" or rcStatus="Pending User")	Pending
rcStatus	fixedValue	(current.phase="Problem Investigation and Diagnosis" or current.phase="Problem Resolution") and rcStatus~="Closed" and rcStatus~="Deferred"	Work In Progress
rcStatus	fixedValue	current.phase~="Problem Closure and Review" and rcStatus="Deferred"	Deferred
rcStatus	fixedValue	current.phase="Problem Closure and Review" and rcStatus~="Closed"	Resolved
rcStatus	fixedValue	rcStatus="Closed"	Closed

Problem Task

Default Settings:

Source Table	rootcausetask
Target Table	rootcausetask
Filter Condition	task.category="Default"

Field Mapping:

Target Field	Source Field
task.category	task.category

current.phase	current.phase
rcStatus	rcStatus

Value Mapping:

Target Value Mapping Field	Mapping Type	Condition	Target Value
task.category	fixedValue		Investigation
current.phase	fixedValue	rcStatus~="Closed"	Active
current.phase	fixedValue	rcStatus="Closed"	Closure
rcStatus	fixedValue	rcStatus="Work In Progress"	Work In Progress
rcStatus	fixedValue	rcStatus="Pending Vendor" or rcStatus="Pending User"	Pending
rcStatus	fixedValue	rcStatus="Closed"	Closed
rcStatus	fixedValue	rcStatus~="Work In Progress" and rcStatus~="Pending Vendor" and rcStatus~="Pending User" and rcStatus~="Closed"	Assigned

Task Planner data migration

In versions of Service Manager earlier than 9.40, Task Planner data is stored in the changeModel and changePlan tables. However, in Service Manager 9.40, Task Planner data is stored in the changePlan table only. Therefore, two data migration settings are provided to migrate Task Planner data from Service Manager 9.40 and earlier versions.

Migrate legacy change model Task Planner data to new Task Planner

Default settings:

Source table	changeModel

Target Table	changePlan
Query	not null(tasks)
Batch Update	false

Field mapping:

Target field	Source field
fileName	
number	id

Value mapping:

Target value mapping field	Mapping type	Condition	Target value
fileName	fixedValue		changeModel

Post script:

```
for(var i=0;i<$sourceTable.tasks.length();i++){
       $targetTable.tasks[i].taskId=$sourceTable.tasks[i].taskId;
       $targetTable.tasks[i].taskCoords=$sourceTable.tasks[i].taskCoords;
       $targetTable.tasks[i].dependentIds=$sourceTable.tasks[i].dependentIds;
       $targetTable.tasks[i].dependentCoords=$sourceTable.tasks[i].dependentCoords;
       $targetTable.tasks[i].taskCategory=$sourceTable.tasks[i].taskCategory;
       $targetTable.tasks[i].taskTemplate=$sourceTable.tasks[i].taskTemplate;
       $targetTable.tasks[i].taskDescription=$sourceTable.tasks[i].taskDescription;
       $targetTable.tasks[i].openInPhase=$sourceTable.tasks[i].openInPhase;
       $targetTable.tasks[i].closeByPhase=$sourceTable.tasks[i].closeByPhase;
       $targetTable.tasks[i].activeCond=$sourceTable.tasks[i].activeCond;
       $targetTable.tasks[i].activeCondXML=$sourceTable.tasks[i].activeCondXML;
       $targetTable.tasks[i].activeCondDesc=$sourceTable.tasks[i].activeCondDesc;
       $targetTable.tasks[i].mandatory=$sourceTable.tasks[i].mandatory;
}
$targetTable.doUpdate();
```

Migrate legacy change instance Task Planner data to new Task Planner

Default settings:

Source table	changePlan

Target Table	changePlan
Query	null(fileName)
Batch Update	true

Field mapping:

Target field	Source field
fileName	fileName

Value mapping:

Target value mapping field	Mapping type	Condition	Target value
fileName	fixedValue		cm3r

Migrate legacy change/task data to new Task Planner

In Service Manager Hybrid, task information is stored in the change's task plan. Change task information must be migrated to the new Task Planner. To do this, use the "Legacy change/task data to new taskplanner" data migration setting in the data migration tool.

Default settings:

Source table	cm3r
Target Table	changePlan
Query	open=true
Batch Update	

Field mapping:

Target field	Source field
fileName	
Number	header,number

Value mapping:

Target value mapping field	Mapping type	Condition	Target value
fileName			

Known Error data migration

Known Error data migration differs from Interaction, Incident, and Problem because Known Error data must be copied from the knownerror file to the rootcause file, and the field mapping is also more extensive than in other modules.

Because it is a data copy, you cannot use the batch update mode. However, knownerror data volume is usually much less than that of Incident or Interaction. Therefore, the normal copy mode should be sufficient for most environments.

> **Note:** After you migrate the knownerror data, remove the "KnownError_Library_disabled_by_PDHD" knowledge base as this legacy library is no longer used.

The Data Migration Tool provides four migration settings, which you should run in their corresponding scenario as described in the following table.

Scenario	Migration setting(s)
Migrating from legacy Known Error	Legacy known error to new known error Legacy known error related record migration Legacy known error attachment to new known error attachment
Migrating from legacy Process Designer Known Error	Legacy Process Designer known error to new Process Designer known error For more information about how to use this migration setting, see "Migrating from legacy Process Designer known error records" on page 167.

Additionally, because knownerror data is copied and the records are re-created in the rootcause file, you must also update the screlation file to correct the relations of known errors. To do this, the out-of-box Process Designer provides another migration script called "Legacy known error related record migration".

> **Note:** Attachments for legacy knownerror records are migrated as attachments for the new known error records.

Default Settings:

Source Table	knownerror
Target Table	rootcause
Filter Condition	true

Field Mappings:

Target Field(Problem)	Source Field(Known Error)
id	id
category	category
assignment	assignment
status	status
logical.name	logical.name
brief.description	brief.description
description	description
root.cause	root.cause
update	update
open	open
open.time	open.time
opened.by	opened.by
update.time	update.time
updated.by	updated.by
close.time	close.time
closed.by	closed.by
reopen.time	reopen.time
reopened.by	reopened.by
priority.code	priority.code
ticket.owner	ticket.owner
severity	severity
sysmodtime	sysmodtime
sysmodcount	sysmodcount
sysmoduser	sysmoduser
assignee.name	assignee.name
subcategory	subcategory

Target Field(Problem)	Source Field(Known Error)
product.type	product.type
problem.type	problem.type
company	company
dump	dump
resolution	resolution
workaround	workaround
incident.category	incident.category
current.phase	current.phase
incident.count	incident.count
initial.impact	initial.impact
future.impact	future.impact
impact	impact
users.affected	users.affected
problem.start.time	problem.start.time
expected.resolution.time	expected.resolution.time
location.type	location.type
frequency	frequency
proposed.solution	proposed.solution
review.notes	review.notes
cause.code	cause.code
affected.ci	matching.ci
ci.device.name	matching.device.name
ci.device.type	matching.device.type
ci.assign.group	matching.assign.group
ci.location	matching.location
affected.companies	affected.companies
affected.ci.count	matching.ci.count

Target Field(Problem)	Source Field(Known Error)
last.task.no	last.task.no
kpf.id	kpf.id
kpf.file	kpf.file
folder	folder
closure.code	closure.code
estimatedCost	estimatedCost
estimatedMandays	estimatedMandays
rootcauseDate	knownerrorDate
solutionDate	solutionDate
affected.item	affected.item
rcStatus	rcStatus
interaction.count	interaction.count
publishWorkaround	publishWorkaround

Value Mappings:

Target Value Mapping Field	Mapping Type	Condition	Target Value
category	fixedValue	true	known error
current.phase	fixedValue	rcStatus~="Closed"	Logging
current.phase	fixedValue	rcStatus="Closed"	Closure
isKnownError	fixedValue	true	true
rcStatus	fixedValue	rcStatus~="Closed"	Open
rcStatus	fixedValue	rcStatus="Closed"	Closed

Migrating from legacy Process Designer known error records

If you upgraded your system from an older Process Designer version, you need to migrate your old Process Designer knownerror records to new Process Designer knownerror records by using the **Legacy PD known error to new PD known error** migration setting. The migration process includes the following procedures:

1. Change the old Known Error record to Problem record, and then create a new Known Error record with a category of "known error."
2. Link the new Known Error record to the Problem record.
3. Copy or do not copy the original relations to the new Known Error record, based on the option you select in the **Post Script** tab of this migration setting, as described in the following table.

Option for relation processing	Description
Skip Copying Relations to New Known Error Records	This option does not copy original relations to new known error records.
Copy Relations to New Known Error Records	This option copies original relations to new known error records.

Service Level Management data migration

As of Service Manager 9.40, the Service Level Management (SLM) module is reimplemented on Process Designer (PD). If you upgraded from an earlier version of Service Manager, you need to migrate your legacy SLM data so that your SLM module can work correctly on Process Designer-based workflows.

The purpose of SLM data migration is to set a category and set a phase for agreement records based on the following rules:

- The Category of an agreement record is populated with the category of one of the targets if all the targets of the agreement have the same category (that is, the **Service Level Category** field in the target record).
- The Category of an agreement record is populated with a value of **Service Level Agreement** if the targets of the agreement have different categories or have an empty category.
- The Phase of an agreement record is populated with “agreed” if the expiration date is later than the current date.
- The Phase of an agreement record is populated with “expired” if the expiration date is earlier than the current date.

For SLM data migration, the **Legacy SLA to new Agreement** migration setting is provided. See the following tables for its details.

Default settings

Source Table	
Source Table	sla
Target Table	sla
Filter Condition	true

Field Mappings

Source Field	Target Field
agreement.id	agreement.id
agreements	agreements
category	
contacts	contacts
current.phase	
customer	customer
description	description
display.category	display.category
expiration	expiration
external.support.groups	external.support.groups

Value Mappings

Target Value Mapping Field	Mapping Type	Condition	Target Value
category	jsCallback		$value=lib.MigrateSLA.migrateSLA (sourceTable['agreement.id'])
current.phase	fixedValue	expiration>tod ()	agreed
current.phase	fixedValue	expiration<tod ()	expired
display.category	jsCallback		$value=lib.MigrateSLA.migrateSLA (sourceTable['agreement.id'])

Related Records data migration

Process Designer-based related records retrieves information that is displayed in the related records columns directly from the screlation table. The table contains a large number of screlation records and the field values are generated based on legacy sccrelconfig settings. Therefore, you must perform data migration in order for the related records section to be displayed correctly.

An out-of-box script ("DataMigrationForRelatedRecords") is available to perform this data migration. To run the script, follow these steps:

1. Open the Script Library and search for the "DataMigrationForRelatedRecords" file.
2. Open the file and uncomment the following line:

   ```
   //migrateScrelations
   ```

3. Execute the script. The data migration is performed automatically.

> **Note:** By default, only active screlation records are migrated (for time-saving purposes). Therefore, if you view inactive (closed) records in the related records section, the wrong field value may be displayed.

To migrate all the screlation records, uncomment the following line, and then execute the script:

```
//var REFRESH_All_SCRELATION_RECORDS = "true";
```

If you have a large number (millions) of screlation records, the script may take hours to update all the records. Therefore, you must ensure that the Windows client session timeout setting is large enough for the script to complete.

Service Catalog data migration

Considering that Service Catalog does not need to copy data from two different tables, and that it typically has a large volume of data, we recommend that you run the migration by using the batch update mode (especially for migrating a large volume of closed records).

The out-of-box example data migration is based on the following two key fields:

- status
- current.phase

Note: You can add additional fields if needed, but do not remove these two fields.

Default settings:

Source Table	svcCatalog
Target Table	svcCatalog
Filter Condition	null(current.phase)

Field Mapping:

Target Field	Source Field
current.phase	
status	

Value Mapping:

Target Value Mapping Field	Mapping Type	Condition	Target Value
current.phase	fixedValue	active="t"	Published
current.phase	fixedValue	null(active) or active="t"	Retired
status	fixedValue	null(active) or active="t"	Retired
status	fixedValue	active="t"	Operational

Re-index the knowledge bases

After you remove the Problem_Library_disabled_by_PDHD and KnownError_Library_disabled_by_ PDHD knowledgebases, you must run a full re-index of the knowledgebases if you use Knowledge Management in a production environment.

Appendix C: The SQL compare utility

The SQL compare utility is an informational tool that compares your existing table fields and unique key information with those of the HPE Service Manager version you are upgrading to, and then reports the new and modified fields and unique keys that will merge into the existing tables. You can use the list of the fields and unique keys produced by the SQL compare utility to determine whether any fields or unique keys in your current system differ from those in the new version. You can also use the report to determine which new fields and unique keys you must add to RDBMS-mapped files if you choose to make the changes manually during the applications upgrade.

After running the SQL compare utility on the table fields, refer to the following table for the result types and the recommended actions.

Result type	Recommended actions
Nonexistent in HPE Service Manager 9.50 fresh installation	This field is added by tailoring. No further action is necessary.
Not to be modified to other field type by the Upgrade Utility	Manually modify this field type according to the matching solution described in "Data type mismatches" on page 75.
Not to be modified to other field database type by the Upgrade Utility	Manually modify this field type according to the matching solution described in "Data type mismatches" on page 75.
Not to be moved to alias table by the Upgrade Utility	Manually modify this field type according to the matching solution described in "Data type mismatches" on page 75.
To be modified by the Upgrade Utility	This field type modification will be done automatically by the Upgrade Utility during the upgrade process. You can also manually modify this field type before the upgrade to improve the upgrade performance.
To be modified by the Upgrade Utility, and the original field name xxxx will be renamed to xxxx.old	This field type modification will be done automatically by the Upgrade Utility. No further action is necessary.
To be added by the Upgrade Utility	This new field will be added automatically by the Upgrade Utility during the upgrade process. You can also manually add this new field manually before the upgrade to improve the upgrade performance.

After running the SQL compare utility on the table unique keys, refer to the following table for the result types and the recommended actions.

Result type	Recommended actions
Nonexistent in HPE Service Manager 9.50 fresh installation	This key normally is added by tailoring. Check this key to see whether it includes any field of added primary key or unique key.
Exist in Old OOB, and will be removed by the Upgrade Utility	This key exists in Old OOB only, but does not exist in Service Manager 9.50. The Upgrade Utility will remove this key automatically. No further action is necessary.

Result type	Recommended actions
To be added by the Upgrade Utility	This key does not exist in Old OOB, but exist in Service Manager 9.50. The Upgrade Utility will add this key automatically. No further action is necessary.
To be ignored by the Upgrade Utility	The field is added as a unique key because Service Manager does not support operations on tables with primary keys when the applications version is earlier than 9.32. You can manually change the key type to primary key after the system upgrade.

> **Note:** If you are going to accept the new DBDICTS and the changes made to the DBDICTS in the upgrade, you do not need to run this utility.

Running SQL Compare

The following SQL Compare files are included when you install the HPE Service Manager Upgrade Utility:

- sqlupgrade.unl
- upgdbdct.dta

SQL Compare returns messages for dbdict mappings that contain new fields. You can update the dbdicts to contain the fields specified by the SQL Compare applications before you begin the applications upgrade.

> **Note:** Run SQL Compare on a development system. Do not run SQL Compare on the post-upgrade system.

> **Note:**
> Before running SQL Compare, load the preupg.bin file using Service Manager Database Manager. See "Step 4: Load the applications upgrade file" on page 64 for more information.

To run the SQL Compare utility, follow these steps:

1. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **SQL Compare Utility**.
2. Select **Run SQL Compare Utility**, and then click **Next**.
3. Type the full path to upgdbdct.dta including the final back slash (\) or forward slash (/), depending

on your operating system. For example, if you copied the files to a temporary directory, the path might be:

Windows: `c:\temp\upgrade\`

Linux: `/tmp/upgrade/`

4. Click the **Next** button.

   SQL Compare returns the following message:

   ```
   Process Complete. Please check for any additional messages.
   ```

   The results of the SQL Compare process are stored in the sqlupgrade table. This table resets each time you run SQL Compare.

To view the SQL compare results of the table fields, follow these steps:

1. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **SQL Compare Utility**.
2. Select **View SQL Field Compare Results**, and then click **Next**.
3. Click **Search**. The results are displayed in a record list.

Each table field difference found by the SQL Compare Utility appears as a separate record in the sqlupgradefield file. This record also lists the new fields that you must add to the database dictionary if you are updating your RDBMS mapped system manually.

The sqlupgradefield record provides the following information for each field you must add or modify if you are updating your RDBMS mapped system manually.

List of sqlupgradefield fields

Field	Description
File Name	The exact file name which delegates the database dictionary.
Field Name	The exact field name to add or modify to the associated database dictionary.
Field Structure	The field structure in the database dictionary.
Field Type	The field type in the database dictionary.
Field Level	The field level in the database dictionary.
Result Type	The actions happened or will happen to this field.
Field Alias Of	If this is an alias field, it contains the name of the primary field that it is an alias of. Otherwise, this field is blank.
Field SQL	The SQL table alias of this field in the database dictionary.

List of sqlupgradefield fields, continued

Field	Description
Table Alias	
Field SQL Name	The SQL name of this field in the database dictionary.
Field SQL Type	The SQL type of this field in the database dictionary.
Field SQL RC	The encoding format of this field in the database dictionary.
Exceptions	The exceptions that occurs when adding or modifying this field.

To view the SQL Compare results of the table unique keys, follow these steps:

1. From the System Navigator, click **System Administration** > **Ongoing Maintenance** > **Upgrade Utility** > **SQL Compare Utility**.
2. Select **View SQL Unique Key Compare Results**, and then click **Next**.
3. Click **Search**. The results are displayed in a record list.

Each table unique key difference found by the SQL Compare Utility appears as a separate record in the sqlupgradekey file. This record also lists the new unique keys that you must add to the database dictionary, if you want to check the possible duplicated record manually.

The sqlupgradekey record provides the following information for each field you must add or modify, if you want to check the possible duplicated record manually.

List of sqlupgradekey fields

Field	Description
File Name	The exact file name which delegates the database dictionary.
Result Type	The actions happened or will happen to this field.
Key Type	The unique key types include primary, unique, and noduplicated.
Key Field	Concatenates all the key fields with the | character.
Exceptions	The exceptions that occurs when adding or modifying this unique key.

Add new fields

For the new fields to perform correctly, they must exist in both the HPE Service Manager database dictionary and the SQL database. If you are updating your RDBMS mapped system manually, you must add them to the SQL database and update the existing Service Manager SQL mapping in the database dictionary. When you update a table in sqlsystemtables, add fields only through the database dictionary. Modifying the SQL mapping damages the file structure of the table.

Determine the correct structure

In most cases, you should add the new field to the descriptor structure. However, sometimes the Structure field contains something other than the word "descriptor". When this occurs, add the new field to the appropriate location.

Action to take with non-descriptor fields:

In this instance	Add the field here
The field resides in another structure	Check the cm3r dbdict. There is a "middle" field of structure type.
The field is an array	If the field is an array, the field name appears twice in the new field list. Check any field of array type. The root field has array type, but the same name field normally has the real type (for example, VARCHAR(60)). Use the first entry to determine the structure where you should add the array. The Structure field in the second entry reflects both the structure for the array (unless it uses the descriptor structure) and the name of the array itself.
The field is part of an array of structures	Check the kmcategory dbdict permission field, which is a structured array field.

Note: When adding fields to an array of structures, add them in the same order as they appear in the sqlupgrade record.

Process the NVARCHAR SQL type

Although the upgrade version uses the VARCHAR SQL type for fields, SQL Compare and Upgrade Utility retain the use of the NVARCHAR SQL type after the upgrade. For example, if the SQL type of a field is NVARCHAR(60) in the current version and the SQL type of this field in the upgrade version should be VARCHAR(60), the SQL type of this field will remain NVARCHAR(60) in the upgraded version.

If the current and updated versions of a field have different length SQL types, SQL Compare and Upgrade Utility use the longer length. For example, if the field SQL Type is NVARCHAR(60) in the current version and if the SQL Type of this field in the upgrade version is VARCHAR(100), the SQL Type of this field will be NVARCHAR(100) in the upgraded version.

If you want to tailor the SQL type of a field from VARCHAR to NVARCHAR and retain the same length, make sure to perform tailoring from the database instead of from the HPE Service Manager database dictionary. Otherwise, the tailoring will be ignored by Service Manager.

Appendix D: Troubleshooting

Before you contact HPE Service Manager Customer Support, try the following troubleshooting instructions to identify and resolve your issues.

Troubleshooting: SQL Compare cannot run for unsupported version

Symptoms

When run SQL Compare, the system displays a message that the version is not supported.

Resolution

SQL Compare will check whether the Service Manager Applications version is supported or not. If the version is earlier than 9.30, you need to firstly upgrade to Service Manager 9.30 or later and then upgrade to Service Manager 9.50. If the version is not a standard release, please contact HPE Service Manager Customer Support.

Troubleshooting: Applying upgrade cannot run for unsupported version

Symptoms

The wizard for applying upgrade cannot continue and the system displays the message of unsupported version.

Resolution

Upgrade Utility will check whether the Service Manager Applications version is supported or not. If the version is earlier than 9.30, you need to firstly upgrade to Service Manager 9.30 or later and then upgrade to Service Manager 9.50. If the version is not a standard release, please contact HPE Service Manager Customer Support.

Troubleshooting: Some data files are missing from the Upgrade Utility package

Symptoms

The upgrade wizard shows that some data files are missing from the Upgrade Utility package.

Resolution

Please check whether the missed data files exist in the extracted directory of the Upgrade Utility or not. If not, please check if you have removed them manually. The best solution is to download the Upgrade Utility and extract it again.

Troubleshooting: The Upgrade Utility appears to stop responding

Symptoms

The Upgrade Utility may appear unresponsive during the upgrade process. This issue may exhibit the following symptoms:

- The Windows Task Manager indicates that the HPE Service Manager client is not responding.
- The Upgrade Utility does not show the progress after you click **Next** to start the upgrade execution.
- The Upgrade Utility appears to stop at a percentage of completion.

Resolution

This is normal and does not indicate a problem with the upgrade. Additionally, the percentage on the status screen does not accurately indicate the actual progress.

Troubleshooting: The client session was terminated during an upgrade

Symptoms

The upgrade failed with "Session no longer valid" error when running in foreground mode. This issue is most likely to occur when you are creating or applying a custom upgrade.

The client session was terminated during an upgrade. The HPE Service Manager client may temporarily lose heartbeat when the Upgrade Utility is running. If the server cannot detect the heartbeat for a certain amount of time, it disconnects the client session.

Resolution

To resolve this issue, restore the database to the latest pre-upgrade state, add `sessiontimeout:1200` and `heartbeatinterval:120` to the sm.ini file, and then restart the upgrade. See also "Step 3: Perform preparatory tasks" on page 61.

> **Note:** We recommend that you set the timeout period to a length of time that is long enough for an

upgrade phase to complete.

Troubleshooting: Unexpected errors during an upgrade

Symptoms

An unexpected error stopped the upgrade, such as the HPE Service Manager server running out of memory, stack overflow, power outage, and network failure.

Resolution

Fix the issue that stopped the upgrade, restore the database to the latest pre-upgrade state, and then restart the upgrade.

Troubleshooting: Upgrade failed with a "Not enough shared memory available" error

Symptoms

The upgrade process ended abruptly when you were applying an upgrade, and the following error message was logged into the sm.log file:

```
E big_alloc: Not enough shared memory available to allocate <number> bytes
```

Resolution

Restore the database to the latest pre-upgrade state, increase the size of the shared memory (see "Step 3: Perform preparatory tasks" on page 61), and then restart the upgrade.

Troubleshooting: Database transaction log full

Symptoms

The process ended abruptly when you were loading data or applying an upgrade. The sm.log file contains error messages that resemble the following:

```
[Microsoft][ODBC SQL Server Driver][SQL Server] The log file for database 'DB_Name'
is full. Back up the transaction log for the database to free up some log space.
(message,add.schedule)
```

```
An error occurred while attemping to add a record (file.load,add.record.0)
```

Resolution

Running an upgrade generates a huge number of transactions and this is likely to make the transaction log full. Restore the database to the latest pre-upgrade state, refer to the documentation for your RDBMS to increase the database transaction log size or enable auto-growth, and then restart the upgrade.

Troubleshooting: Integrations do not work after an applications upgrade

Symptoms

After you upgrade you applications to HPE Service Manager 9.50, certain existing integrations do not work correctly.

Resolution

Certain integration fields that were optional in the old applications version have been made required in the new Service Manager9.50 applications. Existing integrations that do not have those fields populated will become invalid. Therefore, you must remove and re-create these integrations in the Integration Manager. These integrations include the following:

- Release Control integration
- BSM OMi integration

For more information about your specific integration, see the Integrations section in the Service Manager Help Center.

Troubleshooting: Automatic merge fails

Symptoms

The Automatic Merge task fails during the upgrade and you receive the following error message:

```
Failed to unzip <zip file path>. Auto Merge skipped.
```

This error message indicates that automatic merge was skipped because of a failure to unzip the required zip file. The Upgrade Utility skips the Automatic Merge task and continues the upgrade process.

Resolution

To rerun the Automatic Merge task after running the Upgrade Utility, follow these steps:

1. In the **Upgrade\3waymerge\oob** folder, find the zip file named after the version that you selected for the **Base Version** when running the Upgrade Utility.
2. Extract the folder named after the **Base Version** that you previously selected from inside the zip file to the **Upgrade\3waymerge\oob** folder. For example, if you selected **SM9.3PD** for the **Base Version** previously, extract the **SM9.3PD** folder from inside **SM9.3PD.zip**, and then place the extracted folder (**SM9.3PD**) in the **Upgrade\3waymerge\oob** folder.
3. Open the Upgrade Results list and search for records with a type of "Renamed."
4. Select the objects that you want to merge, and click **Mass Auto Merge** from the More Actions list.

Troubleshooting: Unable to check the upgrade results after upgrade

Symptoms

Unable to open and view the upgrade results after upgrade.

Resolution

Run the following command to check the upgrade results:

sm.-bg apm.upgrade.resolve NULL NULL showresult <*object type*> <*object name*>

Run the following commands to revert or choose an upgrade for upgrade results:

sm -bg apm.upgrade.resolve NULL NULL revert <*object type*> <*object name*>

sm -bg apm.upgrade.resolve NULL NULL upgrade <*object type*> <*object name*>

Troubleshooting: Unable to submit service items in SRC

Symptoms

Unable to submit service items in SRC because the mandatory fields are not shown on the UI.

Resolution

Follow these steps:

1. Click **Tailoring** > **SRC Tailoring** > **Checkout Panel**.
2. Check the value of the Company field in SRC custom field configurations. Update the Company value if it is different from the current company that is in use.

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 Upgrade Guide (Service Manager 9.50)

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@hpe.com.

We appreciate your feedback!

Hewlett Packard Enterprise

Java™ COMPATIBLE