Upgrading Service Manager

How to quickly and efficiently upgrade from an older version of Service Manager

Introduction

I had the opportunity to work with a large financial institution to upgrade their Service Manager from SM70 to SM920. This paper describes what we learned while going through the upgrade process and the best practices and procedures for an upgrade based on this experience.

Prerequisites

This document does not replace the upgrade guide. Please follow all steps in the upgrade manual when performing an upgrade.

The upgrade process

Required systems

Each Service Manager upgrade requires three distinct Service Manager instances to be successful. The development instance is used to apply the initial upgrade, perform conflict resolution, fix issues found during and after conflict resolution and then to create the custom upgrade. Creating the custom upgrade is a repetitive process, with fixes done on the development instance and then recreating the custom upgrade until no issues occur when applying that version of the upgrade. Due to this requirement, the development instance has to have sufficient disk space to hold the initial upgrade files as well as all the custom upgrade files, backups and unloads that are required for the upgrade.

The test instance is used to apply the custom upgrade to a fresh copy of the production data. This upgraded instance is made available to stake holders and select end users to test the upgraded system. Any upgrade issue reported against this system will be fixed on the development system and a new custom upgrade will be created and applied. Enhancements and issues not related to the upgrade should not be fixed at this time, but reported for fixing immediately after the upgrade process is finished.

The production instance should not change during the entire course of the upgrade, from the first time a copy is taken for the development system to the final time the custom upgrade is applied.

Process steps of an upgrade

Each successful upgrade starts with a strict tailoring / customization freeze of the production system. Any changes other than changing data records in the production system will increase the risk when applying the custom upgrade.
To assess the time and effort needed for the upgrade and conflict resolution, we strongly recommend running an upgrade assessment through global services. The upgrade assessment will return with information on how many conflicts were found, how many errors, and they will do the basic conflict resolution consisting of Document Engine and Display Application. Basic conflict resolution needs to be done before the detail conflict resolution to determine that basic functionality is available. If issues occur in testing of basic functionality after the basic conflict resolution, these issues have to be fixed before continuing with the other conflict resolution.
Once the upgrade assessment was received, time and effort for the upgrade project need to be planned. Part of that planning will be:

- Will the upgrade be done by in-house personnel only or with the help of HP PSO or HP certified partners?
- Will the customer try to get closer to out-of-box functionality for easier upgrades in the future or is the goal to minimize training effort and stay with the old look-and-feel?

- When will the binary update of the production system take place and when should the final upgrade of the production system happen? All other planning needs to happen with this goal in mind, but the goal must be set realistically based on upgrade complexity and resources available.

In the example of the financial institution from the south east where I had the privilege of assisting with the upgrade, a total of 900 conflicts (roughly 400 of which were due to a mass update of FormatControl and easily resolved) were resolved by a team of 5 engineers within 3 ½ days. This number is not a typical resolution rate and could only be done in this short time because the team was very focused and committed and knew what tailoring was done for what reason.

One of the questions we faced was: Should conflict resolution be done by Object Type or by Module? We started out by Module, but quickly switched to resolving conflict by object type, such as FormatControl, finding that this method is much more efficient. We used the upgrade merge utility to determine what has changed between the old and new records and learned that once you get used to the XML representation of one tailoring table, it is easier to stay with this table until all conflicts are resolved rather than switch between tailoring tables. As a best practice, we recommend resolving conflict by Object Type based on this experience.

After the detail conflict resolution was finished, the custom upgrade is created and the post-upgrade testing phase begins on the test system: Is the end-user functionality as desired? Are there any error messages when performing any function? Did the look-and-feel change so much that end-user training is required? The result of the post-upgrade testing will be changes that need to be done to the development system and another custom upgrade will need to be created and applied to the test system. This cycle repeats until all functionality is available as desired. A note of caution: Some error messages may be caused by tailored functions not being compatible with out-of-box functions any more. To fix this, it may be necessary to change a record that is not part of the upgrade set. To ensure that these records will be pushed to the test system together with the upgrade files, create an unload script record with all changed records that are not part of the upgrade set defined, and create the unload immediately after creating the custom upgrade, then apply the custom upgrade to the test system first, then apply the unload created from the development system. This way, you will not miss a changed record when applying the upgrade to production later.

Once the upgrade is completely finished, most customers will have to perform an IR regen on their production system, since it is recommended to run the upgrade with IR disabled. While the IR regen is running, the system is not available to end users. To minimize the time the IR regen takes, we recommend to

- Run vrir on a copy of production and determine which phrases to add to the stopwords file. This minimizes the size of the IR files and decreases the amount of time the IR regen will run.
- Run multiple IR regens in parallel (test how many parallel runs will give the best results) by running them through sm –util rather than the dbdict utility.
- Ensure to allow for sufficient "down-time" to finish the IR regens after the upgrade.

The final process step is to apply the custom upgrade and the unload with the additional required records to the production system. The allocated downtime for the production upgrade needs to be based on the detailed timing information gathered during the test runs and needs to include the pre-upgrade work that still needs to be done, the upgrade itself, conflict resolution and the IR regen of all IR files that are needed.

Planning stake holder involvement

One of the questions frequently asked is when and how to involve the stake holders, such as end users, CAB, Incident, Problem, and Change Managers, the Configuration Management team etc. Depending on the customer's processes, the stake holders may have feedback on process changes, form changes, or whether the Service Manager upgrade should stay closer to out-of-box or keep the same look-and-feel. There are 5 distinct phases during the upgrade:

1. Upgrade planning

Here the stake holders can have a vote determining whether the upgrade should stay closer to out-of-box or closer to the look-and-feel of the older version. They will have feedback on the upgrade timing of the final production upgrade, too, so that the Service Manager system is not unavailable during a time that the stake holders have to perform vital functions using Service Manager.

2. Upgrade Assessment

   The upgrade assessment is done completely by the Service Manager team with the help of HP or Partner resources.

3. Upgrading the Development system

   The development system upgrade is nearly completely done by the Service Manager team as discussed in these sub-topics:

   - Applying the initial upgrade

     This is done completely by the Service Manager team, possibly with the help of HP or Partner resources.

   - Basic conflict resolution

     This is done completely by the Service Manager team, possibly with the help of HP or Partner resources.

   - Functionality test after basic conflict resolution

     This is done completely by the Service Manager team, possibly with the help of HP or Partner resources.

   - Detail conflict resolution

     This is done by the Service Manager team, possibly with the help of HP or Partner resources. Stake holders may be asked for feedback on specific functionality to help determine whether to use out-of-box or customer functionality. Stake holder involvement here typically is minimal though.

   - Create and apply the custom upgrade on Test

     This is done completely by the Service Manager team, possibly with the help of HP or Partner resources. This step is going to be done multiple times to test the custom upgrade application and to note down timing, issues, unloads required. Each run will be with a new custom upgrade created from the Development system.

4. Testing functionality and look-and-feel on the Test system

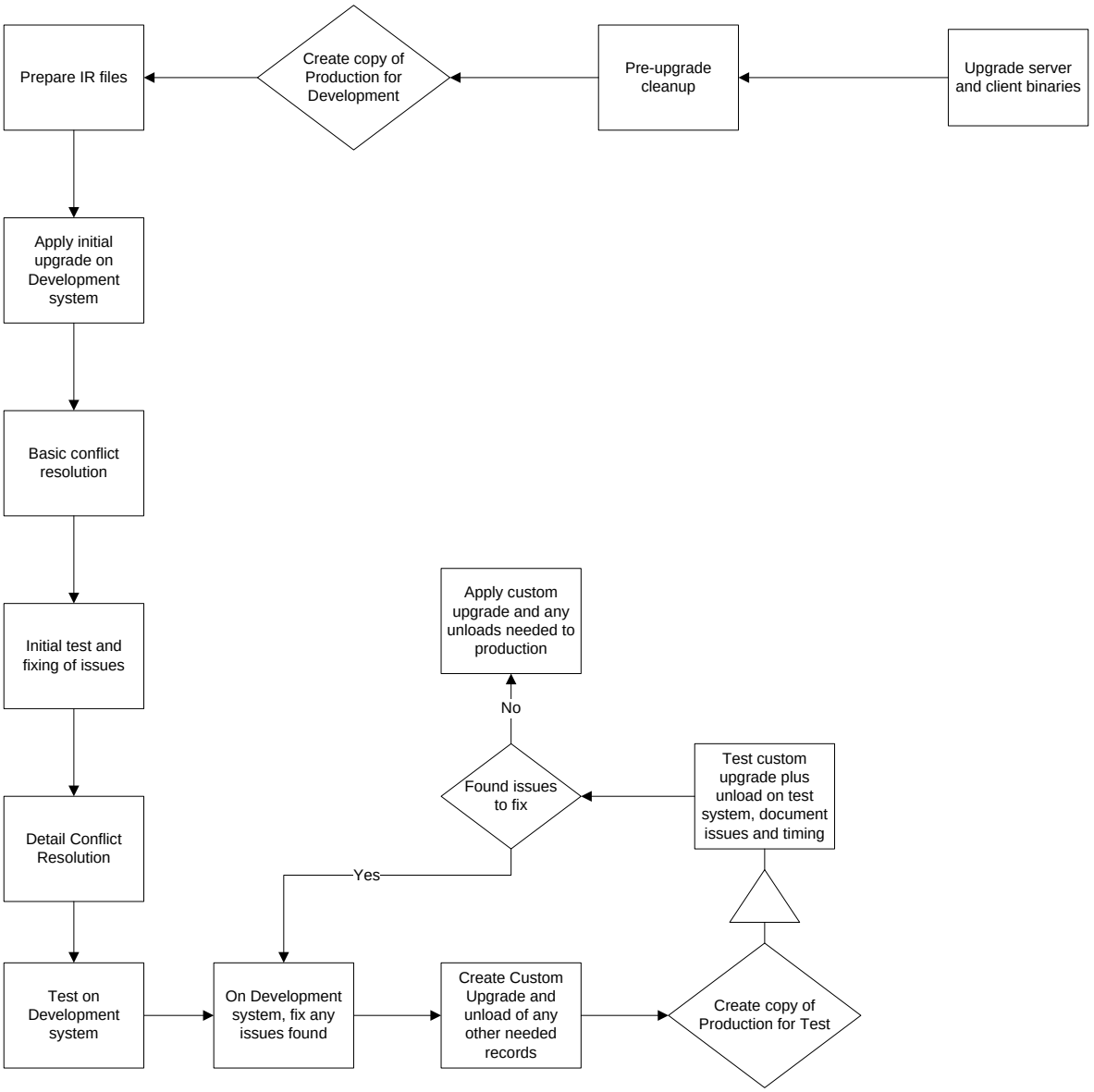
   This is done nearly completely by stake holders and their representatives with assistance from the Service Manager team. It is vitally important that the stake holders test all areas thoroughly and report any issue, question or request to the Service Manager team with as much detail as possible. The test phase is the longest phase of the Service Manager upgrade.

5. Applying the upgrade to the Production system

   This task is done by the Service Manager team, possibly with assistance by HP or Partner resources. The stake holders should have representatives on hand during the upgrade weekend to verify that all functionality works as tested.

Stake holders should be available to the Service Manager upgrade team for questions regarding the process and required functionality, but their main involvement is in testing the system once the custom upgrade is built. Stake holders need to actively sign off on all areas of the system prior to proceeding into the production upgrade.

Upgrade steps – detail



Upgrade server and client binaries

As a best practice, change should be divided into manageable pieces. In the case of the upgrade, there are two distinct pieces: the update of the server and client binaries and the upgrade of the applications. As a best practice, update your binaries to the newer level well before the application upgrade. Doing so will ensure that troubleshooting issues is more efficient, since we can be sure that an issue that surfaces after the binary update must be related to the binaries, and an issue surfacing after the application upgrade must be related to the applications.

Pre-upgrade cleanup

A lot of pre-upgrade cleanup can and should be done on a regular basis on the production system. Files such as mail, msglog, syslog, errorlog should be archived and purged on a regular basis. Eventin and eventout should be checked and cleaned up regularly, as should the schedule table. The module tables, such as incidents, probsummary, cm3r, cm3t, ocmq, ocml, ocmq and their associated activities, attachments, clocks etc. need to be archived regularly according to the companies archival procedure.

All these files to clean up are mentioned in the upgrade guide. But what about the backup copies created when changes were done? Often – as it is best practice to back up before you change – these backup copies stay in the system for a very long time.
Search the dbdict utility for table names starting with SMSQL…. and remove these after making sure that no data is missing. SMSQL… tables are created when a full table copy was necessary for a dbdict change and are usually removed automatically afterwards. If the full table copy failed for any reason, these temporary tables may stay in the system and need to be removed manually after making sure that no data is associated to them. Check the RDBMS to verify that the tables associated with the dbdict record are empty, if they exist.
Search all the tailoring tables such as dbdict, datadict, menu, scripts, triggers, ScriptLibrary, format, formatctrl, link, displayscreen, displayoption, Object, States, and Process for backup copies created for changes that have already been accepted and remove these backup copies.

IR regen preparation

On a copy of the production system run VRIR:
From a command prompt go to the Service Manager server RUN directory and enter the following:
sm -util
vrir
ir.<table name that contains IR key, such as probsummary, for a name of ir.probsummary>
continue these steps until you have checked all IR files.

```
C:\scs\sm920\Server\RUN>sm -util


HP Service Manager Database Exerciser
(Version: 9.20.021 Build: 021)  [04/27/2011 11:45:59]


 add: add        cls: close      cnt: count
 del: delete     dis: display    fnd: find
 get: get        nxt: next       opn: open
 pat: patch      prv: previous   qbe: qbe
 reg: ir regen   rst: reset
 rmv: remove     upd: update    vrir: verify IR

x: EXIT

Enter your choice: vrir


Validate an IR file
(Version: 9.20.021 Build: 021)  [04/27/2011 11:46:05]

Enter fully qualified name for the IR file: ir.probsummary

IR validation succeeded.  See log for details


 add: add        cls: close      cnt: count
 del: delete     dis: display    fnd: find
 get: get        nxt: next       opn: open
 pat: patch      prv: previous   qbe: qbe
 reg: ir regen   rst: reset
 rmv: remove     upd: update    vrir: verify IR

x: EXIT
```

```
Enter your choice: x

C:\scs\sm920\Server\RUN>
```

If you have IR keys on tables that you do not use for IR searches (such as on the help table), remove the IR key from that table instead of having to run the IR regen later.

Analyze the vrir results for words that are frequently indexed but do not contribute to a successful search. In case of a financial institution, these may be numbers 1 (01) through 31 for dates, 2000 through 2011 for years, the words “bank”, “branch”, “account” etc. For an IT provider, these words might be the dates as well, in addition to “computer”, “software” etc.

The following is an excerpt from vrir against an out-of-box system:

```
15388( 15588) 04/27/2011 11:46:10  RTE I Term window is used in 3
Documents.  Hash offset: 27147.  File offset: 5247445
 15388( 15588) 04/27/2011 11:46:10  RTE I Term laptop is used in 9
Documents.  Hash offset: 27247.  File offset: 5247000
 15388( 15588) 04/27/2011 11:46:10  RTE I Term prefer is used in 1
Documents.  Hash offset: 27266.  File offset: 5247363
 15388( 15588) 04/27/2011 11:46:10  RTE I Term incid is used in 15
Documents.  Hash offset: 27304.  File offset: 5247333
 15388( 15588) 04/27/2011 11:46:10  RTE I Term 43 is used in 4 Documents.
Hash offset: 27433.  File offset: 5246848
```

The terms laptop, prefer, incid and 43 can definitely be added to the stopwords list, since they do not add any real value to the searches. Instead of the word laptop, the type of laptop (e.g. HP EliteBook 8530w) should be searched on, as an example.

Once the stopwords file is up to date on the test system, run the IR regens that are needed for searches in parallel by performing the following commands:

```
C:\scs\sm920\Server\RUN>sm -util


HP Service Manager Database Exerciser
(Version: 9.20.021 Build: 021)  [04/27/2011 11:45:59]


 add: add        cls: close      cnt: count
 del: delete     dis: display    fnd: find
 get: get        nxt: next       opn: open
 pat: patch      prv: previous   qbe: qbe
 reg: ir regen   rst: reset
 rmv: remove     upd: update    vrir: verify IR

x: EXIT

Enter your choice: opn
Enter filename: probsummary


add: add        cls: close      cnt: count
 del: delete     dis: display    fnd: find
 get: get        nxt: next       opn: open
 pat: patch      prv: previous   qbe: qbe
 reg: ir regen   rst: reset
```

```
 rmv: remove     upd: update    vrir: verify IR

x: EXIT

Enter your choice: reg

add: add        cls: close      cnt: count
 del: delete     dis: display    fnd: find
 get: get        nxt: next       opn: open
 pat: patch      prv: previous   qbe: qbe
 reg: ir regen   rst: reset
 rmv: remove     upd: update    vrir: verify IR

x: EXIT

Enter your choice: cls
```

Run multiple IR regens parallel by starting the command from several command prompts at the same time to save time.

A knowledge article describes the option on how to run IR regen on a separate system, in case running it in parallel as outlined above still takes too long:

Article: http://support.openview.hp.com/selfsolve/document/KM781779

This article describes the procedure for running IR regens with these steps:

- configure IR asynchronously
- stop irqueue process
- take a backup
- install the backup on a test machine
- run the IR regen on a test system
- copy scirexpert records to production
- start irqueue process again on production.

In Detail, follow these steps to regen IR on test system and promote those changes to production:

Prerequisite: The production system must be configured to run IR updates asynchronously via the ir_asynchronous parameter in sm.ini file.

1. 1. Stop irqueue process on production
2. 2. Take a backup of production (if taking backup should require downtime, be sure to start SM without irqueue (comment it out in the sm.cfg file) and install that backup on a test environment
3. 3. Perform IR regen on the test environment, while production is running without the IR Queue process. IR queries can still be run against the existing IR index. However, all updates will be queued in the irqueue file and not be updated in the IR index at this time.
4. 4. When the IR regen on test is complete, stop the production system and replace the contents of the scirexpert table of production with the contents of this table on the test system
5. 5. Start the production system – including the irqueue background process. Since the irqueue file contains a large amount of records now, the irqueue process will work heavily until these records are processed.

Note: The scirexpert file keeps ir index for all dbdict that have an IR key. These IR indexes can be identified by "filename" field in scirexpert dbdict. When you IR regen only one dbdict on the test system, and the irqueue process is not running, the scirexpert records for the other dbdicts in test and production will not be modified and be identical. So you only need to delete scirexpert records of this filename and replace them with the ones of the test system.

Note: As running IR regen is a time demanding task, it would be a perfect opportunity to tune the IR index. This is described in Diagnostics and Tuning whitepaper. Especially the stop word files can be enhance IR expert performance a lot. However, as this file applies to all IR indexes, this would require ir regen of all dbdicts with an IR key.

Documentation of upgrade decisions and timing

I have added the documentation as an extra essential step of the upgrade process, because – while it is not a requirement to run the upgrade – it is a requirement for a successful and on-time upgrade project. The upgrade process is being run several times on different systems. Each time, every decision and the timing of each sub-step needs to be documented. In a conflict, for example, document whether you used the out-of-box or custom record, or if you merged functionality from both, write down in great detail which function you added to which record and at what place. When resolving issues, write down exactly what the issue was, what was done to fix it in which record, and ensure that the record is either part of the signature set and will be in the upgrade files, or add the record to the unload script utility used to unload all other impacted records to load into the test and production systems.

The timing needs to be documented, to be able to plan the upgrade of the production system precisely, to ensure that everyone needed will be available on time and to ensure that the upgrade does not have to be rolled back because the outage window was exceeded.

The following parameter should be added to the sm.ini during the upgrade process (apply and create upgrade, both):

sessiontimeout:240

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 240 sets the timeout to 4 hours (240 minutes), a period that should be enough for an upgrade phase to complete in a typical scenario. If your upgrade takes longer than 4 hours, adjust this parameter accordingly. Failure to setting this parameter may result in a failed upgrade due to session timeouts.

Basic conflict resolution

As a best practice, conflict resolution is divided into two parts:

- Basic conflict resolution, which includes Display Application, Document Engine and RAD
- Detail conflict resolution, which includes all other tables.

The basic conflict resolution handles the base tailoring engines that determine the workflow within the modules. Issues created here will impact larger areas of the product and need to be addressed first. In the example of the customer I was working with, 161 basic conflicts needed to be addressed (5 RAD applications, 110 displayoptions, 5 displayscreens, 8 Objects, 0 States, and 33 Processes). One common question we are asked is: How long does conflict resolution take per record? We timed the basic conflict resolution on this system and found that on average, resolving the conflicts for each record took a little over 10 minutes.

Initial issue resolution

After basic conflict resolution, any issue found using base functionality in any of the modules needs to be addressed and fixed. This may include modifying items that were not in the signatured and

upgraded items and may include items that are not in the tables that were addressed during basic conflict resolution. For example, if a customer decides to not use the escalate functionality but continue to use the options > relate > open functionality instead, some variables and Processes need to be adjusted to re-allow the old functionality while taking advantage of other new features. Running traces using RTM:3 and debugdbquery:999 is most helpful to find the underlying root cause of any issues encountered in this step.

Examples of work done during initial issue resolution:

New search screen:

The customer had the following issue on the FilterAdvFind search screens when trying to select a different module / table to search on:

Message tablename - <table name> Could not be found:

The reason for this message was that copies of the records in the SearchConfig table were created as <tablename>ORIG. These tables were not found as records in the Object table, causing this error message in the search screen table selection. The error was fixed by removing the backup copies from the SearchConfig table, which was not a signature table during this upgrade.

Opening a change from an incident caused an error message:

When trying to open a change from incident management the message "category not found" was issued.

The out-of-box system tries to open changes from Incident Management using the Default category that the customer had disabled in their implementation, using other category names instead.

To fix the issue in this situation, we changed the pre-RAD expression on process
screlate.cm3r.category from
category in $L.file=nullsub(category in $L.file, "Default")
to
category in $L.file=nullsub(category in $L.file, "RFC")

During the initial issue resolution, form and formatcontrol issues are not yet looked at, just basic functionality, error messages, buttons not working. All other issues are looked at and addressed when conflict resolution was finished.

Detail conflict resolution

In this step, all other conflicts are addressed: formats, formatcontrol, links, menus etc. While using the upgrade merge utility saved a lot of time determining what has changed between the out-of-box and customer version, we found that conflicts on the link records took longest to reconcile, due to the complexity with the main link page and the link line page. While link records took about 20 minutes to reconcile, all other records took between 5 and 10 minutes on average. Please note that the upgrade merge utility can only be used to view, not to actually merge the records in the currently available upgrade utility versions.
Detail conflict resolution will take several days to weeks, depending on how many conflicts exist and how many resources are available to address these conflicts. Even though it is exhausting work, staying focused during this upgrade step is vital and will pay off in a smooth and fast upgrade project.
The following table shows how many records had to be looked at during detail conflict resolution, sorted by Object Type and Module:

Row Labels	Count of Module

datamap	**1**
Request	1
erddef	**2**
Helpdesk	2
eventmap	**7**
Config	7
eventregister	**3**
Config	1
Helpdesk	2
extaccess	**9**
Change	2
Config	3
Helpdesk	3
Problem	1
format	**135**
Base	30
Change	13
Config	9
Helpdesk	51
Problem	12
Request	20
formatctrl	**473**
Base	188
Change	51
Config	54
Helpdesk	83
Problem	10
Request	83
SchedMaint	4
help	**4**
Change	1
Helpdesk	3
info	**2**
Base	2
joindefs	**3**
Config	2
Helpdesk	1
link	**38**
Base	10
Change	5
Config	3
Helpdesk	8
Problem	2
Request	10
menu	**7**
Base	4
Helpdesk	2

Request	1
notification	**18**
Change	1
Helpdesk	10
Problem	2
Request	5
ocmoptions	**3**
Request	3
scmessage	**12**
Base	8
Helpdesk	1
Problem	2
Request	1
ScriptLibrary	**2**
Base	1
Helpdesk	1
scripts	**1**
Change	1
svcCatInterface	**6**
Helpdesk	6
triggers	**1**
Config	1
tzfile	**7**
Base	7
validity	**4**
Change	3
Request	1
Grand Total	**779**

Test of basic functionality and issue resolution

After finishing the detail conflict resolution, all functionality has to be tested by the development team to ensure that no errors occur. Usability and functionality testing are done against the custom upgrade later. This test is to ensure that basic operations: Search, add, update, close / delete work without errors.

For example, if the customer added the functionality of change activities in Service Manager 7.0 and the functionality was added to the out-of-box system afterwards, the customer may have made modifications to their custom forms that were not part of the out-of-box system. To accept the out-of-box activity functionality though, these forms may have to be adjusted to use the out-of-box inputs and variable names and these changes have then to be ported to the production system with the upgrade. It is very important to note, though, that additional tailoring and implementation of tailoring enhancements are not ever to be part of the upgrade process. Adding this additional work lets the upgrade project run out of scope and out of time quickly.

Create Custom upgrade

After the basic testing and issue resolution by the development team, the first custom upgrade needs to be created.

Two common issues when creating the custom upgrade:

1. Make sure that client side load / unload is turned off. Go to Window > Preferences and uncheck the client side load /unload checkbox. Click OK to confirm.
2. A note in the upgrade guide needs to be read and followed to ensure that the upgrade creation creates all files needed:

   Important: You must create a data directory within the upgrade area when doing a custom upgrade

Another recommended setting to do prior to creating the custom upgrade is increase the sessiontimeout parameter in the sm.ini from its default of 30 minutes to a higher number such as 240, as mentioned before, to avoid running into session timeouts, causing the upgrade creation to fail.

Once the custom upgrade creation was run, ensure that you have the following unload files in your custom upgrade directory:

\CustomUpgrade (The content of this folder should be the same for every upgrade):

Detail.log	sqlupgrade.unl	upgdbdct.dta	upgrade.mak
extaccess.unl	transfer.bin	upgrade.inf	upgrade.str
preupg.bin	transfer.log	upgrade.log	upgrade.ver

\CustomUpgrade\data (The content of this folder will differ from upgrade to upgrade. The list here is from a Service Manager 7.00 to Service Manager 9.20 upgrade):

upgradeactivityactions.dta	upgradecm3messages.dta	upgradeerddef.dta
upgradeactivitytype.dta	upgradecm3profile.dta	upgradeeventmap.dta
upgradeAlertDef.dta	upgradecm3category.dta	upgradeventregister.dta
upgradeapplication.dta	upgradecm3catphase.dta	upgradeextaccess.dta
upgradeapplicationfields.dta	upgradecmcontrol.dta	upgradeextactions.dta
upgradeApprovalDef.dta	upgradecode.dta	upgradeformat.dta
upgradeauditspecs.dta	upgradecompany.dta	upgradeformatctrl.dta
upgradecapability.dta	upgradecontractcategory.dta	upgadegloballists.dta
upgradecascadeupd.dta	upgradecounters.dta	upgradehelp.dta
upgradecategory.dta	upgradectenv.dta	upgradeinbox.dta
upgradecirelationshiptype.dta	upgradecurrency.dta	upgradeinfo.dta
upgradecivisualizationadmin.dta	upgradedatamap.dta	upgradejoindefs.dta
upgradecivisualizationcat.dta	upgradeddescript.dta	upgradekmattachments.dta
upgradecivisualizationdecorator.dta	upgradeddmRule.dta	upgradekmdoctype.dta
upgradecivisualizationdevice.dta	upgradedevtype.dta	upgradekmgroup.dta
upgradecivisualizationlabel.dta	upgradedisplayevent.dta	upgradekmhitlisttemplate.dta
upgradecivisualizationline.dta	upgradedisplayoption.dta	upgradekmknowledgebase.dta
upgradecivisualizationrelationship.dta	upgradedisplayscreen.dta	upgradekmmapping.dta
upgradecm3groups.dta	upgradeenclapplication.dta	upgradkmprofile.dta
	upgradeenclapplrev.dta	upgradekmstatus.dta
	upgradeenvironment.dta	upgradekmstopword.dta

upgradeknownerrorcat.dta
upgradeknownerrorphase.dta
upgradelanguage.dta
upgradelink.dta
upgrademacrodef.dta
upgrademenu.dta
upgrademsgclass.dta
upgrademsgtype.dta
upgradenotification.dta
upgradenumber.dta
upgradeObject.dta
upgradeocmoptions.dta
upgradepatcortadmin.dta
upgradepmenv.dta
upgradeproblemtype.dta
upgradeProcess.dta
upgradeproducttype.dta
upgradequerygroups.dta
upgradequerystored.dta
upgradercenv.dta
upgradereport.dta
upgraderootcausecat.dta
upgraderootcausephase.dta
upgraderootcausetaskcat.dta
upgradescmessage.dta
upgradescrelconfig.dta
upgradeScriptLibrary.dta
upgradeScriptPackage.dta
upgradescripts.dta
upgradeSearchConfig.dta
upgradeslacontrol.dta
upgradeslamodulecontrol.dta
upgradeslaprofile.dta
upgradesoatypes.dta
upgradeStates.dta
upgradesubcategory.dta
upgradesubtotals.dta
upgradesvcCatInterface.dta
upgradesvcCatSTatusFieldMap.dta
upgradesvcCatStatusMap.dta
upgradeSYSATTACHMENTS.dta
upgradeSystemEvents.dta
upgradeTodoMap.dta
upgradetriggers.dta
upgradetzfile.dta
upgradeuimcachedimages.dta
upgradeuimcompdefcategories.dta
upgradeuimcompdefcatscdefs.dta
upgradeuimcompdefinitions.dta
upgradeuimevents.dta
upgradeuimorigins.dta
upgradeuimpagecategories.dta
upgradeuimpagecatspages.dta
upgradeuimpages.dta
upgradeuimuserpagecontent.dta
upgradeuimuserpreferences.dta
upgradeupgradereconciliation.dta
upgradevalidity.dta
upgradewizard.dta

Apply custom upgrade

Follow the instructions on how to apply the custom upgrade. It differs from applying the out-of-box upgrade in the following steps:

1. Use all files from the custom upgrade folder, including transfer.bin and preupg.bin
2. When prompted with “Are you going to use this system to create an upgrade?” choose “No” for the answer
3. Use the path to the custom upgrade files (including the \ ) when prompted for the path
4. When asked which version of the object to choose, rather than choosing the recommended “Install the Service Manager version of the object alongside your own (recommended) chose the option to “Replace your version of the object with HP’s version of the object”

For each test run, 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 if you are not available to write down the exact times.

Test custom upgrade

During this step, the stake holders will log in to the test system that the custom upgrade was applied to and test all functionality in detail. They will report any issues (and possible enhancements) to the Service Manager upgrade team. During the test phase, every aspect of the system needs to be tested by the stake holders, including integrations using Web Services, Connect-It, and SCAutomate.

Additionally, ensure that the stake holders test their reporting solutions to see if all operational reports still return valid results.

Apply changes to development system to create new custom upgrade

This step will largely run parallel to the test custom upgrade step. While the stake holders report issues, the Service Manager development / upgrade team will fix these issues on the development system. In regular intervals, a new custom upgrade will be created from the development system and applied to the test system for the stake holders to retest and verify. Once no more issues are found, that final custom upgrade will be used to apply against production. Important: Test the upgrade for timing and note down the exact timing each step takes when applying the last custom upgrade to the test system, since this information is vital for a successful application to production.

Apply custom upgrade to production

The big weekend has arrived to apply the upgrade to the production system. Based on the timing recorded during the application of the custom upgrade to the test system, the upgrade team is available and stake holders are on call for a final test. Since applying a custom upgrade was tested several times, it is very important to stick with the script – do not try to skip or add steps that were not tested prior. The script to stick to was created during the prior custom upgrades.
The cleanup of old data should have been done prior to the upgrade weekend, by creating scheduled purges of data and removing backup copies early on in the process, so that even new copies of production needed for new test systems did not have to do the cleanup repeated every time.
Applying the custom upgrade to the production system then does not differ from applying it to the test system. If anything else on the production system changed, such as the OS was upgraded, a new machine is used, etc. ensure that a proper load test was run prior to going into production. You need to make sure that the JVM options for the Service Manager servlets are optimal for the amount of users you expect to have active at the same time. Refer to the Service Manager sizing guide for guidelines.

Useful Tips and Tricks

Prior to upgrade

- Ensure to remove all backup copies of records from the production system
- If not done so already, regularly purge supporting files that are not required for reporting
- If you plan to change the OS or hardware, run a load test for the average and maximum load you expect to have on the system, adjust performance parameters according to the findings
- Do the update of the binaries a few weeks prior to the application upgrade to be able to troubleshoot any issues piece by piece.

During initial upgrade

- Turn off client side load / unload on the Windows client used to run the upgrade
- Turn off IR (ir_disable:1) on the development system
- Keep the sm.log, upgrade log files and the client messages from the messages window for later troubleshooting
- We found that index problems on the underlying database cause error messages on screen and in the sm.log that need to be fixed immediately, otherwise these issues can compound and get worse. If during the initial upgrade an index problem is found, ensure to fix this issue on the production system as well, otherwise the issue will hinder any upgrade from this point on. These error messages are similar to these:

  RTE E Error: SQL code=1 message=ORA-00001: unique constraint (FALCON.UPGRESULTSM1_P) violated

RTE E Error: SQL code=1 message=ORA-00001: unique constraint (FALCON.NOTIFICATIONM1_P) violated

If you try to update these indexes (alter) on the RDBMS and it fails, it is an indication that the index on the RDBMS is corrupt and they need to be dropped and re-created. The help of a DBA may be necessary in that case.

During conflict resolution

- Make use of the upgrade merge utility to determine what is different between the old and new renamed records. Do not use the merge utility to remove the conflict, but rather use the correct tailoring tool (format control, forms designer, etc.) to resolve the conflict
- Document in detail each decision you made during conflict resolution. If you merged old and new functionality, describe in detail how it was merged, if you choose the old or the new record, document which one you chose and why
- In resolving Process conflicts, it may be necessary to insert a RAD application at a location other than at the end to ensure the application flow is still correct. To do so, enter the following command in the RAD debugger while viewing the Process record in database manager:

  ```
  x rad in $L.file= insert(rad in $L.file,Y,1,{[]})
  ```

  where Y is the position (start counting at 1) where the new RAD application needs to be inserted.
- In resolving link conflicts, if you need to insert a new line, select the line beneath the line to insert and choose the insert line option.
- For formatcontrol conflicts, choose the insert line option with the cursor set in the line beneath the one to insert.
- Do not do additional tailoring while doing conflict resolution, but do make sure to add all records required for issue resolution to a single unload script record that is run immediately after the create custom upgrade. Doing even small amounts of additional tailoring will take additional time in both doing the tailoring, adding the unload, and of course testing the additional change, putting the upgrade process at risk.

During creation of the custom upgrade

- Ensure to have client side load / unload turned off on the windows client used to create the custom upgrade
- Set the sessiontimeout parameter in the sm.ini to 240 or higher and restart the Service Manager server prior to starting the creation of the custom upgrade.
- Create the data directory in your custom upgrade directory to ensure that all files are created
- Ensure that all files were created based on the list in this document. Once all files were created the Windows client will show the message that the upgrade was successfully created.

During application of the custom upgrade

- Note down all timing and all additional steps necessary to have a clear idea on how long each step takes
- Document all conflict resolution steps in detail to be able to just follow pre-written instructions and do not have to spend time researching how to resolve a conflict.

Appendix A – Sample upgrade planning sheet

Proper documentation and timing are crucial to upgrade success. Below is a sample upgrade plan from a customer upgrade:

Pre- Upgrade Task	Duration	Comments/Notes
Email Notifications for Productions Weekend		
Change Request Submitted and Approved		
Release Request Submitted and Approved		
Team Signoff for Production		
Install SC Auto on Production TEC Server		
CSR for Desktop Client Installation		
Install ServiceManager 9.21 to production		
Copy SM 7.02 production to SM 9.21 server		
Install the web-tier and point to SM 9.21 production		
Install TEC component of New SM production server and Test		This is just testing connectivity betweem TEC and ServiceManager
Point Dev/Test Connect.IT to New SM 9.21 box and testing some senarios.		This is to test connectivity between ServiceManager and Connect.IT. Will also outbound email notifications and data loads.

Final Custom Upgrade Task	Duration	Comments/Notes
Email to Sys Admins		**No changes the system except add users inactive users, Group membership additions**
Log in to the AIX Unix and create the customupgrade folder		Note: A folder called "customupgrade" might already exist from the previous custom upgrade build. This should be renamed out of the way and another created. Also create a data folder within the custom upgrade folder
Using chmod command, grant read, write access to the customupgrade folder	10min	chmod -R 4777 customupgrade
Ensure custom upgrade folder is empty.		Rename the current custom upgrade folder
comment out all lines in the sm.cfg and sm.ini		
Disable ir regen		
Stop and Re-start SM 9.21 Dev system with latest changes		
Logon to SM 9.21 Dev		
Make sure that the Client side unload is unchecked		
Logon to SM 9.21 Dev		
Remove upgradetoc dbdict		

Run **smupgrade** in the Service Manager client command box	30min	
1. Service pack		
2. Create an upgrade		
3. De-select all languages		
4. Type SM7 for the current version		
5. Type fully qualified path for the CustomUpgrade Directory : /servicemanager/customupgrade/		
6. Which Patch should be used for the build : SM 92		
7. Take which action : Complete Upgrade Build		
8. No - Internal logging		
9. Start Upgrade Build		
review messages to make sure there is nothing expect duplicate keys for notifications files. Keep copy from sm.log if necessary.	10min	check upgrade, detail and exception logs in CustomUpgrade folder
Verify the number of files in the customupgrade/data folder	10 min	Should be 123 files
Copy custom upgrade folder to the new production SM 9.21 server box	10min	

Apply Custom Upgrade in Production	**Est. Duration**	**Actual Duration**	**Start Time**	**End Time**	**Comments/Notes**
Apply custom upgrade					
Shutdown Production system - all servers			12:01AM	12:30 AM	
Shutdown Reporting Services			12:01AM	12:30 AM	
Backup Production SM 7.02 System and copy to SM 9.21 on	120 min		**12:30 AM**	**2:30AM**	
Reset log and status data					
Start up Production SM 9.21 System in single user mode			2:30 AM		
Cleanup : Ensure that the following record are deleted:					
1. SMSQL* and upgraderesult tables. **Delete also through the dbdict**					Restart SM because of the possible DB loop due to the delete of SMSQL* tables

back-end					
2. datadict					Using database manager - remove all *.xxx* and *yyy*
3. dbdict					
4. format					xxx: 233; yyy: 10 (renamed xxx.guess.release.edit to nnn.guess.release.edit); do not delete nnncommonprob.xxxnosel.user
5. formatctrl					xxx: 72; yyy : 6 (exclude nnncommonprob.xxxnosel.user)
6. links					xxx: 62; yyy:4
7. menu					xxx: 11; yyy:0
8. Process					xxx : 20; yyy:0
9. Objects					
10. State					
11. Scripts Library					xxx:14; yyy:0
12. scripts					xxx: 8
13. triggers					xxx :3
14. validity					
15. notification					xxx:3
16. Delete clocks with date less than 01/01/2010					close.date>'01/01/10'

Verify the list of SQL system table mappings					See page 20 of the Upgrade Guide. Could not find **rootcausephase** and **knownerrorphase** tables
Update the SQL system mapping for the sytemperform table					See page 21 of the Upgrade Guide
Verify that all data policy records have an SQL Base name Value					See page 22 of the Upgrade Guide
Increase the length of fields for Problem management					See page 24 of the Upgrade Guide
Copy the Service Manager application Upgrade Utility					See page 27 of the Upgrade Guide

Prepare sm.ini and sm.cfg files					Update the configuration file (sm.cfg as in page 36) and Initialization files as in page 37
Purge existing upgrade files					Type ***aapm.upgrade.purge** on the SM client command box
Load the application upgrade files					See page 38 of the Upgrade Guide
Turn off background processes					
Manually modify the SQL Mapping on the systemperfom table					Add n1 alias to the NULLTABLE as per Dev system
uncheck client side unload					
Modify sm.ini and sm.cfg configurations					Allow sessiontimeout and ir_disable:1
Run the Application Upgrade: Type **smupgrade**	50min				Follow steps from page 44 of the Upgrade guide
1. Click SERVICE PACK					
2. De-select all the languges					
3. Produce full-qualified path to the custom upgrade directory					
4. Choose to Replace version of the object with HP SM version of the object					
Make sure that the notification file is unique on the ID field					This will happen in the middle of the upgrade run: Sort notifcation record based on the ID field. Visibly identify duplicates and make unique by perfoming a mass update of the duplicates using this query: id in file=id in $file + name in $file. Also OLDSM7RM Request Open Next Phase
View Upgrade Logs					
Post Upgrade					
shut down system					

sm.ini- reverse pre-upgrade changes					
sm.cfg- reverse pre-upgrade changes					
shut down system					
Reload WSDL in HPSM					
Point Business Object to new Reporting DB					
Start Production System					Non Standard ports for testing
Change Incident Environment - Check and uncheck Delay assignment of numbers					
Deleted extra display option for cc.first (gui=7)					
Deleted extra display option for cc.edit.incident (gui=906)					
Run ir regen for the following files: - rootcause - probsummary - incidents - ocmq - ocml - ocmo - cm3r - cm3t - knownerror					
Notify user test team					once initial testing is successful
Customer testing					
Integration Connection and Testing:					
Tivilo TEC					
Connect.IT					
go-no go decision					based on issues found
Go-live complete					
total estimated outage duration					

Actions to be Taken - Backout	Est. Duration	Actual Duration	Description
Re-Start Production system - all servers and interfaces			
Restore SM 7. database	30		**This might not be necessary since no change has been made to the database since the last backup.**
Reset log and status data	5		
Start Production System with normal scripts			

For more information

Please visit the HP Software support Web site at:

www.hp.com/go/hpsoftwaresupport

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

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

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

Note: Most of the support areas require that you register as an HP Passport user and sign in. Many also require an active support contract.

To find more information about support access levels, go to the following URL:

www.hp.com/go/hpsoftwaresupport/new_access_levels

To register for an HP Passport ID, go to the following URL:

www.hp.com/go/hpsoftwaresupport/passport-registration

Technology for better business outcomes