Upgrading PowerCenter

This article is meant to introduce the reader to some generic preconditions as well as some general recommendations and pitfalls for upgrading PowerCenter.

It is not meant to give a general order of steps to be executed (a “check list”). Too many of these steps depend on the actual environments and customer situations; it would be unprofessional to claim that such a simple check list existed which would prove helpful in more than 25% of all customer environments.

It is also not meant to be a reference of which option to use in what circumstance. To some extent such reference is available from the software vendor and can be found in two instances, first the manuals (publicly available for download) and second the public discussion and support forums (the so-called “Informatica Network” [INW], accessed via https://network.informatica.com). It is highly recommended to download the manuals (a collection of PDF files) from INW and have them at hand before planning an actual upgrade.

The article assumes basic working knowledge of PowerCenter and a bit of general IT knowledge.

Environment Considerations

This chapter talks a little about the surrounding of some PowerCenter installation.

Requirements for the Software Itself

PowerCenter is not simply installed on some server and used for whatever data warehouse must be filled. It is part of a fairly complex software infrastructure called the “Informatica Platform”.

The software installation consists of at least two parts, namely the server part and the client installation:

- The server installation is available for AIX, Windows, and Linux (not z/Linux). Sun OS / Solaris is no longer supported.
- The graphical clients (Designer, Workflow Manager, Workflow Monitor, and Repository Manager) are available solely for Windows.
- There is also a so-called “Command-Line Client” (a stripped-down server installation) which may be installed on any supported operating system. It can be used, for example, in networks where some enterprise scheduling system runs on servers with a suitable operating system but without a full PowerCenter server installation.[1]

1 The Command-Line Client does not count towards license usage. It can be ordered via a “Shipping Request” from Informatica Global Customer Support (GCS) who will forward such requests to the Shipping department of Informatica.

PowerCenter can be used solely in conjunction with at least one DBMS system which hosts at least two databases (or database schemas, depending on the DBMS and the setup performed by the database administrators). One of these databases is used for the so-called “Informatica Domain” (refer, for example, to the respective blog post hosted by in-factory), the other database(s) is/are used for PowerCenter repositories.

Informatica publishes exact information about the operating systems, hardware platforms, and other software products supported for each version of PowerCenter on INW by means of Excel files. These can be found under the heading Product Availability Matrices (PAMs) and are sorted by product line and version number.

Each PAM lists all OS and DBMS versions supported for the Informatica Platform, PowerCenter, and all other components in this subject area. For example, it is exactly mentioned which versions of the DBMS published by Oracle, Microsoft, IBM, and so on are supported for each version of PowerCenter.

Furthermore the PAM for each version lists the supported Windows versions for the graphical clients.

N.B. It is important to stick to supported operating systems, DBMS versions, and so on. For example, while it may work to install the server part e.g. on Windows 10, at latest after a reboot of the Windows 10 machine the server part will most likely not start (or will work only erratically). Windows 10 is supported only for the graphical clients, not for server installations. **Stick to the PAM. Always.**

Systems around PowerCenter

PowerCenter never exists alone in a software environment. It depends on at least one database software (namely the one used for domain and PowerCenter repositories), and usually the software will read from and write to several or many disparate systems, such as relational databases (Oracle, Sybase, SQL Server, or IBM Db2, to name just a few common DBMS), complex flat files on mainframes, some Hadoop cluster, SAP R/3, and so on.

Which means that each and every system must exist in a version supported by the PowerCenter version to which an upgrade shall be performed (as listed in the respective PAM).

For example, many developers try to install PowerCenter with Oracle Express Edition (XE). This will not work. While the Informatica domain may work with XE, PowerCenter cannot; there are some functional differences between XE and Oracle Enterprise Edition, and these differences keep PowerCenter from working.

In the same way PowerCenter cannot work with the Oracle Instant client. While installing the Oracle Instant client is extremely simple (only one ZIP or TAR archive must be unpacked), PowerCenter needs functionality which is simply not provided by the Oracle Instant client. There is no way to circumvent the installation of the standard Oracle client for PowerCenter.

In short: when installing PowerCenter it is imperative to install the correct client software for the DBMS which stores the PowerCenter repositories. The supported DBMS clients are listed in the respective PAM, too. They must be installed and configured before PowerCenter is installed.

Compatibility between Software Versions

This also refers to all other systems which PowerCenter shall connect to. For example, PowerCenter 10.4 does not work with Oracle 10. There is no “trick” available to convince any supported Oracle driver to talk to an Oracle 10g server.

In the same way an administrator should make sure that the PowerCenter installations (before, during, and after an upgrade) work with supported versions of database clients and DBMS. Everything else may lead to disastrous results.

Separate Runtime Environments

Nowadays almost every PowerCenter customer has more than one PowerCenter environment: one for development, one or more for testing purposes, and one for production use.

This means that upgrades have to be planned or each environment, meaning timing must be planned according to customers’ and users’ needs per environment. While it may be ok to take down the development environment for half a day, most customers are not too happy if the production system must be taken down for a day.

Upgrade Paths and Procedures

In many cases there is no direct way to upgrade a PowerCenter installation to the latest version. For example, there is no direct upgrade from version 9.6.1 to 10.5.2. In such a case an intermediate upgrade cannot be avoided; in the example of upgrading from 9.6.1 to 10.5.2, it will be necessary to first upgrade from 9.6.1 to 10.1.1 and then perform an upgrade from 10.1.1 to 10.5.2.

There is (almost) never a way to avoid such intermediate upgrades. So, if intermediate upgrades are necessary, the best option probably is to set up several virtual machines with DBMS versions supported by both the original and the upgraded version. Which version of which DBMS is supported by which PowerCenter version can be retrieved from the respective PAMs. If in doubt, a support ticket should be opened with GCS, they can definitely clarify which upgrade path is supported with which DBMS version.

In case of such an “upgrade chain”, it is usually not necessary to install e.g. each single version on an AIX machine. Many organisations nowadays have virtualised environments (e.g. utilising VMWare GSX server), and in such environments setting up virtual machines with supported operating systems and DBMS versions usually is much easier than e.g. install different versions on supported AIX boxes.

The biggest catch in executing such multi-step upgrades usually is the code page of the DBMS holding repository databases during an upgrade process. For example, a

repository backup taken from a repository stored on an Oracle instance running with UTF-8 cannot be restored into a database running in MS Windows Latin 1. So, care must be taken to ensure that the necessary code pages for the repository upgrades are supported for each intermediate upgrade.

The next point to keep in mind are users and user groups along with their permissions and privileges. In earlier versions of PowerCenter (until version 8.1.1 Service Pack 5) the users were defined per repository. Since version 8.5 users are taken from the Informatica domain in which the repository service runs. This means that users and user groups must exist in the target domain of an upgrade **before** a PowerCenter repository is upgraded.

Next point is about in-place upgrades and upgrades via parallel installations.

Experience has shown that in-place upgrades go wrong far more often than parallel installations. One simple hardware problem can lead to an installation which cannot be rolled back to the previous version and which cannot be rolled forward to the new version. Meaning this installation is corrupt and cannot be repaired. Trying an in-place upgrade in a production environment is a good way to incur many nights without sleep and finally failing upgrade procedures.
Also parallel installations make it easier to perform all kinds of regression tests.
The big disadvantage of a parallel installation is the need for more hardware for some time, but given how easy it is for an in-place upgrade to fail these costs are negligible (in comparison to a production upgrade failing and getting stuck so that not even a restore would work, and such events can occur).

How to Perform a Repository Upgrade

Informatica mandates only one way to perform a repository upgrade. Experience has shown that there is no safe way to work around this process, so it is highly recommended to stick to this procedure, no matter how time-consuming and superfluous it might look like.

Unfortunately upgrading a repository during a parallel installation is not as straightforward as when performed during an in-place upgrade. Nevertheless this chapter explains the upgrade in parallel installations because this is the safest way to perform an upgrade.

The following process description makes a few assumptions:

- The old and the new version are installed on separate servers (i.e. Informatica nodes).
- Both installations have access to one DBMS which can keep repository databases from both versions.

- The description also assumes that the upgrade process is performed via the Administrator tools of the old and the new version. Command-line upgrades are possible, but in order to understand how to run them the overall process should first be clear (as described below).

The first description assumes that the repository will have a new name after the upgrade. The alternative process for keeping the repository name unchanged will be explained later.

1. In the old version, take a backup of the repository to be upgraded.
2. In the old version, create an empty repository service in the DB instance (resp. DB schema) which is supposed to hold the upgraded repository in the new version. Name this repository service as planned for the new version. Create this repository without contents.
3. In the old version, restore the backup file from step #1 into the repository DB of the new repository service created in step #2. Leave the repository service in Exclusive mode, **do not** switch to Normal mode (otherwise you will have to switch it to Exclusive mode because in Normal mode the next step cannot be executed).
4. After the restore process finished, disable the repository service in the old version.
5. In the old version, remove the repository service. This will leave the DB contents intact.
6. In the new version, create a repository service with the desired name pointing to the DB instance (resp. DB schema) where the restore process in step #3 has saved the repository contents. **Do not** create contents here.
7. When trying to enable the repository service, the Administrator tool of the new version will ask for a repository upgrade. This is almost always mandatory. The only exception is when e.g. upgrading from version 10.5.1 to 10.5.2 because these two versions use the same repository structure, so an upgrade of the table structures and contents is simply superfluous in these rare cases. In all “normal” cases this (first) upgrade of the repository contents will change some table and view structures and some contents of the repository tables.
8. When trying to enable the repository service, the Administrator tool will now indicate that all passwords in the upgraded repository have been encrypted using a different encryption key than the one used by the current (new) domain. It is mandatory to copy the “siteKey” file from the old installation under a new file name (namely siteKey_old) into the same directory where the siteKey file of the new installation is stored.
9. After having copied “siteKey” from the old installation to “siteKey_old” in the new installation, another upgrade must be performed on the repository contents. This time only passwords are decrypted (using the encryption key in “siteKey_old”) and encrypted (using the encryption key in “siteKey”).
10. **Note:** even if the “siteKey” file from the old installation has been copied as “siteKey_old” to the new installation before the repository upgrade is started, the Administrator tool will still complain about the passwords being encrypted

using a different siteKey file. The Administrator tool has not been programmed to “recognise” that the “siteKey_old” file already exists.

11. After the second upgrade of the repository contents has finished, the repository service may now be restarted in Normal mode.

The second description is about upgrading a repository to the same name.

The principal approach is similar to the first process above, but of course there are some important differences in execution. The reason is that it is not possible to have two distinct repository services with the same name within one single domain. This makes steps #6 and #7 above impossible to execute in the same way. In such a case (the repository name shall remain unchanged) the process must be adapted, so here is a complete list of steps for this case.

1. In the old version, take a backup of the repository to be upgraded.
2. In the old version, create an empty repository service in the DB instance (resp. DB schema) which is supposed to hold the upgraded repository in the new version. Give this repository service a name which will not be used in the new software version. Create this repository without contents.
3. In the old version, restore the backup file from step #1 into the repository DB of the new repository service created in step #2. Leave the repository service in Exclusive mode, **do not** switch to Normal mode (otherwise you will have to switch it to Exclusive mode because in Normal mode the next step cannot be executed).
4. After the restore process finished, disable the repository service in the old version.
5. In the old version, remove the repository service. This will leave the DB contents intact.
6. In the new version, create a repository service with the same name as used for the restore process pointing to the DB instance (resp. DB schema) where the restore process in step #3 has saved the repository contents. **Do not** create contents here.
7. When trying to enable the repository service, the Administrator tool of the new version will ask for a repository upgrade. This is almost always mandatory. The only exception is when e.g. upgrading from version 10.5.1 to 10.5.2 because these two versions use the same repository structure, so an upgrade of the table structures and contents is simply superfluous in these rare cases. In all “normal” cases this (first) upgrade of the repository contents will change some table and view structures and some contents of the repository tables.
8. When trying to enable the repository service, the Administrator tool will now indicate that all passwords in the upgraded repository have been encrypted using a different encryption key than the one used by the current (new) domain. It is mandatory to copy the “siteKey” file from the old installation under a new file name (namely siteKey_old) into the same directory where the siteKey file of the new installation is stored.

9. After having copied “siteKey” from the old installation to “siteKey_old” in the new installation, another upgrade must be performed on the repository contents. This time only passwords are decrypted (using the encryption key in “siteKey_old”) and encrypted (using the encryption key in “siteKey”).
10. **Note:** even if the “siteKey” file from the old installation has been copied as “siteKey_old” to the new installation before the repository upgrade is started, the Administrator tool will still complain about the passwords being encrypted using a different siteKey file. The Administrator tool has not been programmed to “recognise” that the “siteKey_old” file already exists.
11. In the new installation, after the second upgrade of the repository contents has finished (which takes only a few seconds), create a new backup file of the upgraded repository. Leave the repository service to run in Exclusive mode.
12. In the new version, delete the repository contents from the repository database.
13. In the new version, disable the repository service.
14. In the new version, create an empty repository service pointing to the same DB instance (resp. DB schema) which held the “intermediate” repository contents during the upgrade process.
15. In the new version, restore the backup file created in step #11 into the database of this newly created repository service.
16. In the new version, the repository service may now be restarted in Normal mode.

As a blog article is simply too short to cover every possible detail of such complex processes, we kindly invite you to contact **in-factory** in order to establish a thorough and well-working upgrade plan!