



HOME

User Manual

DELMIA Process Engineer<sup>®</sup>

**ENOVIA VPM V4 –  
DELMIA Process Engineer Integration  
DELMIA Engineering Hub to  
Manufacturing Hub  
Connection Version V5R20**



# Foreword

This manual provides an introduction to the basic operations and functions of the Engineering Hub to Manufacturing Hub Connection scenario.

While developing these functions we have made every effort to create a clearly organized, easy-to-understand program structure.

A user-friendly interface as well as a clear menu guide will enable you to quickly learn how to operate the program and to get familiar with its functions so that you can carry out your planning tasks in a quick and reliable way.

## **No Liability or Guarantee**

Our programs and manuals have been compiled with great care and to the best of our knowledge. They have also been tested in a production setting. However, we assume no liability and provide no guarantee that the software and related descriptions are free of error or are suitable for special purposes.

DELMIA assumes no liability for any damage that may arise from the use of this software. By using this software, the user acknowledges this exclusion from liability and shall hold DELMIA exempt from all claims.

## **Copyright**

The information in our documents may be copied and distributed for internal purposes provided it is done free of charge and the contents are not altered or distorted.

Any other form of usage, especially the sale on CD-ROM or in any other publication in whole or in part is only permitted after prior written consent by DELMIA.

Some parts of this software are owned by Unigraphics Solutions Inc. and are copyrighted © 2010. All rights reserved.

Some parts of this software are owned by combit® GmbH and are copyrighted. Report-/Print module List and Label® Version 8.0: Copyright combit® GmbH 1991-2010.

Copyright © 2010, International Business Machines Corporation and others. All Rights Reserved.

Copyright © 1999-2010, The Apache Software Foundation. All rights reserved.

## **Modifications**

Moreover, DELMIA retains the right to make modifications and improvements to the product described in this manual at any time without prior notification.

DELMIA and the 3DS logo are registered trademarks of Dassault Systèmes or its subsidiaries, in the United States or other countries.

© 2001-2010 Dassault Systèmes - All rights reserved

# Table of Contents

<b>1. Introduction</b>	<b>1</b>
1.1 How to Use this Manual	1
1.2 Documentation Conventions and Symbols	2
1.3 New Functions in Engineering Hub to Manufacturing Hub Connection	2
<b>2. Prerequisites</b>	<b>3</b>
2.1 Server Side	3
2.2 Client Side	3
<b>3. Altering Settings</b>	<b>4</b>
3.1 ENOVIA VPM V4 Settings	4
3.1.1 Mandatory Settings/Optional Settings	4
3.2 APACHE Settings	5
3.3 Multi CAD Server Settings	5
3.4 3dCOM Settings	6
3.5 General Mapping Information	7
3.6 PPRLoader Settings	8
3.6.1 Configure the PPRDaemon	8
3.6.2 Configure XML Settings	9
3.6.3 Configure LXC Settings	10
3.7 PPRLoader Customization Settings	11
3.7.1 Set Attribute Mappings	11
3.8 Reference Information	12
3.8.1 PPRDaemon	12
3.8.2 Product Structure Mapping	14
3.8.3 Multi-Instancing	15
3.8.4 Resource Support	17
3.8.5 Initial Import	18
3.8.6 Update Protocol and Management	19
3.8.7 Customized/Extended Attribute Mapping	23
3.8.8 Configuration File	27
3.8.9 Partial Product Data Transfer	29
3.8.10 Incremental Update	31

3.8.11 File Transfer via HTTPS, HTTP, or FTP	33
3.8.12 Encryption and Decryption of Registry Settings	37
3.8.13 Configuration	39
3.8.14 Versions	42
3.8.15 Multiple Part Filter on Import	43
3.9 Step-by-Step Setup	49
<b>4. Launching Applications</b>	<b>51</b>
4.1 Interactive Mode	51
4.1.1 3d COM Server	51
4.1.2 XCAD Server	51
4.1.3 ENOVIA VPM V4 Portal/Web browser	51
4.1.4 PPRDaemon	53
4.1.5 Starting Transfer	53
4.2 Batch Mode	55
4.2.1 Start PPRLoader in Batch Mode	55
4.2.2 AT0EXPND (VPM/UNIX Side)	57
4.2.3 Importing Data into Process Engineer (Windows side)	59
<b>Appendix A</b>	<b>62</b>
<b>Appendix B</b>	<b>65</b>
<b>Appendix C</b>	<b>69</b>
<b>List of Figures</b>	<b>74</b>
<b>List of Tables</b>	<b>76</b>
<b>Index</b>	<b>77</b>

# 1. Introduction

This manual provides information on running the Engineering Hub to Manufacturing Hub Connection. It describes the functionality in DELMIA Process Engineer Version E5R15, but also explains important differences between the settings for this version and those prior to and including 5.11.

The customization and setup of a working Engineering Hub to Manufacturing Hub Connection is controlled by a set of registry keys along with some additional information inside a configuration file.

## 1.1 How to Use this Manual

This manual enables you to get familiar with the operation and functions of the Process Engineer. This manual briefly describes:

- Engineering Hub to Manufacturing Hub Connection scenario

The intention of the Engineering Hub to Manufacturing Hub Connection is to close the current gap between the Engineering world (represented by ENOVIA VPM V4/CATIA V5 (Engineering Hub)) and the Manufacturing side (Manufacturing Hub / DELMIA Process Engineer and DELMIA DPM V5). It allows the transfer of information from the ENOVIA VPM V4 to DELMIA Manufacturing Hub:

- Product Structure Information and Attributes
- Configuration Data
- Transformation Matrices and Geometry

There are two ways for the data transfer:

- The Interactive Mode
- The Batch Mode

As part of explaining how to configure, set up, and test the connection, this document describes the required adaptation of the underlying applications PPR Daemon and PPR Loader on the established customer environment and scenario.

This document has both procedural and reference information. It is intended primarily for the administrator setting up the connection, but also contains user information in the testing and appendices sections.



### Note

*When handling the Engineering Hub to Manufacturing Hub Connection, please also refer to the general introduction to Process Engineer in the General Introduction Manual.*



Click [General Introduction, DELMIA V5 PPR Loader for the Manufacturing Hub](#) to access the manual for additional information.

## 1.2 Documentation Conventions and Symbols

The symbols used in this manual are intended to provide you with keys to the contents in an immediately understandable manner.



This symbol is used to introduce key concepts that are covered in the sections immediately following this symbol. As a result, this symbol most frequently appears at the beginning of chapters or sections.



### Note

*This symbol is used to mark notes, which provide you with additional information you need to have for further work. You will either find the Note sign at the beginning of a chapter or in a particular text passage in the chapter. Texts bearing this sign are additionally marked with **Note**. The text is always in italics.*




### Caution

*This symbol indicates that the text that follows describes particular circumstances that you must avoid to avoid potential errors with the operation of the program or harm to data. You will either find the Caution sign at the beginning of a chapter or near a particular text passage in the chapter. Texts that are introduced by this sign are additionally marked with **Caution**. The text is always in italics.*

### Example

This symbol marks examples which serve to illustrate a certain situation.

- 1) This symbol marks the individual operational steps involved in a particular operating instruction. Operating instructions describe operational steps, for example, how to open a menu or execute a function.
- This symbol marks listed subjects. The symbol for listed subjects can be either used to structure a continuous text or to list main subject keywords.
- This symbol marks list inside a bulleted or numbered list.
-  This symbol marks cross reference information that is available in another manual.

## 1.3 New Functions in Engineering Hub to Manufacturing Hub Connection

No new functionality has been added for this release.

## 2. Prerequisites

The list below provides the prerequisites for running the ENOVIA VPM V4 Engineering Hub to Manufacturing Hub Connection.



### Note

*To be compatible DPE5 and V5 (CATIA, 3dCOM) releases must be equal. For VPM only a recommended minimum level has to be provided. The recommended PTF level corresponding to V5 releases can be retrieved in the program directory of the PTF.*

### 2.1 Server Side

- RDBMS (DB/2 or Oracle)
- ENOVIA VPM V4 1.6
- 3dCOM/ENOVIA VPM V4 Portal 1.6 (optional)
- CATIA V5 R20
- Manufacturing Hub-Server (DPE 5.20 or higher)

### 2.2 Client Side

- ENOVIA VPM V4 Client
- DPE5 R20 Client or higher
- Oracle (or DB/2) Client

## 3. Altering Settings

This section provides step-by-step instructions for configuring the settings necessary for the Engineering Hub to Manufacturing Hub connection.

### 3.1 ENOVIA VPM V4 Settings

Activate the XML and CGR files creation mechanism on the ENOVIA VPM V4 Server. You have to set four VPM variables on the UNIX side.

#### 3.1.1 Mandatory Settings/Optional Settings

- The variable (IPD\_EXPORT\_REPOSITORY) gives the directory on the UNIX machine where the exported XML files created by VPM will be stored. To set this variable, open a UNIX window, set the VPM environment and use the command:

Example

```
VPMSettings -a -n IPD_EXPORT_REPOSITORY -v /tmp
```

- The variable (IPD\_INSTANCE\_EXPORT) forces the creation of a second XML file containing VPM instance's name:

```
VPMSettings -a -n IPD_INSTANCE_EXPORT -v YES
```

Those two settings are mandatory to enable the data transfer through the Connection. The next two ones are optional and control the CGR file generation on VPM V4 Server side.

The variable (IPD\_CGR\_OUTPUT\_DIR) gives the directory on the UNIX machine where the exported CGR files created by VPM will be stored. To set this variable, open a UNIX window, set the VPM environment and use the command:

Example

```
VPMSettings -a -n IPD_CGR_OUTPUT_DIR -v /tmp
```

- The variable (CGR\_BATCH\_AUTO\_EXPORT) is here to force the creation of the CGR files automatically on the UNIX machine. You must set this variable to YES, using in the VPM environment the command:

```
VPMSettings -a -n CGR_BATCH_AUTO_EXPORT -v YES
```



#### Note

*To check if your VPM environment is active, type in your UNIX window the command:*

```
echo $VPM
```

*It should send back a path. To launch the VPM environment, go in the directory env under the home of the VPM administrator, and use the two commands:*

```
YOUR.env
```

```
VPMWsUser.sh
```

*(Do not forget the . and the space at the beginning of the command!).*



## 3.2 APACHE Settings

To be able to load files that are stored in DBLFAIX mode in VPM, the repository of these files has to be known to the APACHE server. To do so, the directories where these files reside have to be added through an alias declaration in the `httpd.conf` file of your APACHE installation. For each directory you will have to add a line based on the following format:

```
Alias <full_directory_Path> "<full_directory_Path>"
```

Here is an example of declaration:

```
Alias /home/data/vpm15ga/MDL/ "/home/data/vpm15ga/MDL/"
```

```
Alias /catia_docs/ "/home/vpmadm/DOC/"
```

Please restart your APACHE server after doing this manipulation.

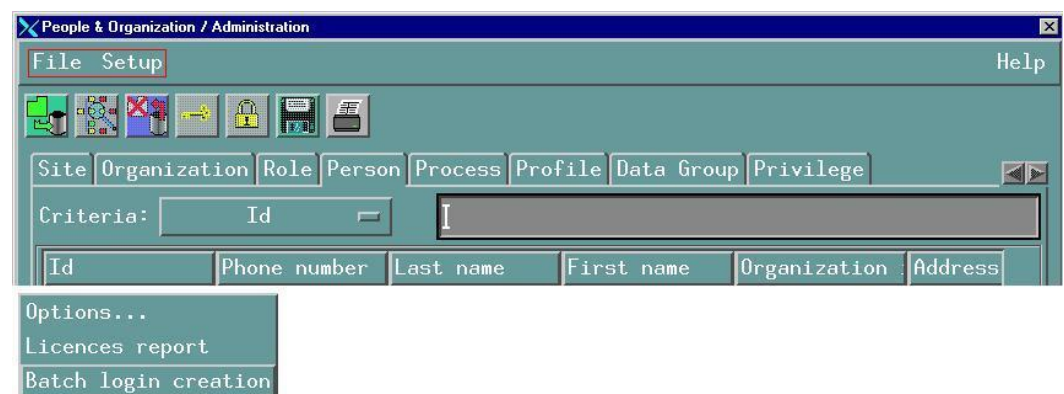
On the Windows machine set the environment variable `HTTP_SERVER` to the IP address of the APACHE server.

## 3.3 Multi CAD Server Settings

A VPM 1.5 PTF1 installation should already be adapted to the MultiCAD. You still need to set the `catia_xc0_multi` variable. Please verify that you define this variable in the `runServerVPM` shell (3dCOM Install/OS/code/command), launched by 3dCOM when starting the communication with VPM. This variable must contain the path + name of a file (that will be created automatically by the XCAD server). For example:

```
export catia_xc0_multi=/tmp/xcomulti.dispatch
```

Since the VPM 1.5 PTF2, a People and Organization file must be created in the home directory of the user that will launch the Multi CAD server. This file will determine which role the user will use to launch Multi CAD server (Security management). To create that file, you must have the proper privilege in VPM and go in the People & Organization panel, and use the command Batch login creation from the Setup Menu:



**Figure 1: People & Organization Panel**

In the displayed panel (cf. after), you will have to define the activity that will use this file. For the Multi CAD server, put `LV0SRVXC` (0 is a zero, not an o). You also have to define the user, password (if you are in Server Authentication Mode) and the role to use. Then you have to define the validity of this file (in the following example, the file is valid until September 18<sup>th</sup> 2002 – 2 PM).



## Note

if you are in Server Authentication Mode, the password to give for the Batch login file is the VPM password (This is the password used when launching VPM if you are in Server Authentication Mode). For DB2 database this password is equal to the UNIX password. For Oracle install, this is the Oracle password.

Client authentication mode panel:

**Figure 2: Client Authentication Mode Panel**

Server authentication mode panel:

**Figure 3: Server Authentication Mode Panel**

The 'User Id' set in the previous panel must be the Id of the user that will launch the Multi CAD server. This user (with the chosen role) will need to have the right to read the models and documents that will be translated to CGR.

## 3.4 3dCOM Settings

When the PPR Loader sends to 3dCOM the list of the projects, 3dCOM is writing temporary files on the NT machine. By default it is trying to write on the directory Models of the E drive. If you don't have an E drive or cannot write on this directory, you will not be able to see the list of projects. In that case, you have to modify the file **defaultCache.properties** of your 3dCOM server (located in the directory install/OS/docs/java). In this file replace the value of:

```
CacheLocation.OS_WINDOWS_NT
```

```
CacheLocation.OS_WINDOWS_95
```

with a proper directory. An example for this variable could be:

```
CacheLocation.OS_WINDOWS_NT=C:\\tmp
```

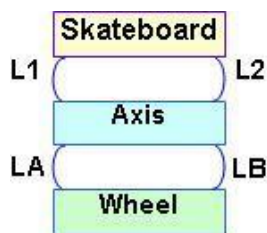
Do not forget also to declare the **catia\_xc0\_multi** variable inside the **runServerVPM** shell (in the directory `install/OS/code/command` of your 3dCOM server install) for XCAD link (*Please refer to the [Multi CAD Server Settings](#) about Multi-CAD server settings*).

## 3.5 General Mapping Information

The current ENOVIA V4 Engineering Hub to DELMIA Manufacturing Hub Connection transfers the ENOVIA VPM V4 reference tree of the product structure to the DELMIA Manufacturing Hub. It maps the ENOVIA VPM V4 product structure into an instance model on Manufacturing Hub side, i.e. for each occurrence of a part reference in the ENOVIA VPM V4 product structure the Connection creates a separate data object in Manufacturing Hub, representing the instance of the object on ENOVIA VPM V4 side (item instance).

- Since all objects on Manufacturing Hub side, have one origin of the very same part reference in ENOVIA VPM V4, all attributes do have the very same values.

### ENOVIA VPM V4 (reference tree)



### DELMIA Manufacturing Hub

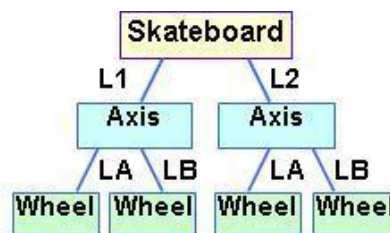


Figure 4: Reference Tree

ENOVIA VPM V4 contains a mechanism of item instances, allowing managing different attributes (beside other features) on different instances of the same Part reference in the product structure.

### ENOVIA VPM V4

(Instance attributes)



### DELMIA Manufacturing Hub

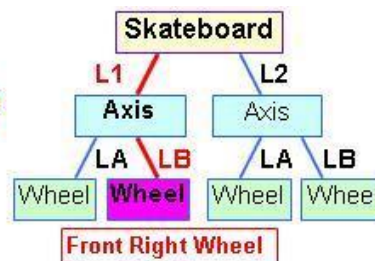


Figure 5: Instance Attributes

To support instance attributes from ENOVIA VPM V4 a second XML file from is created on UNIX side, containing the instance specific attributes of all instances from the product structure, along with their unique instance id (in the context of the selected root part). Like the current XML export file, containing the product structure, the file is fetched, then parsed and imported by PPRLoader.

No additional setting is to be done on PPRLoader/Connection side.

- The name of the instance attribute XML file is the same as the product structure one, but with the prefix “Instance-”. It is generated in the same export directory on ENOVIA VPM UNIX side.

#### Example:

In case the product structure XML file name is “ODT\_TstInstances.xml”, the instance attribute XML file is named “Instance-ODT\_TstInstances.xml”.

- The introduction of a second XML file will result in a second XML parser run for the instance attribute file.

## 3.6 PPRLoader Settings

For the completion of ENOVIA VPM V4 to Manufacturing Hub scenario, several settings and registry keys have to be defined and configured. The following chapters may help you to setup this.

### 3.6.1 Configure the PPRDaemon

From **Tools->Settings->Maintenance Tools**, select the **Current User** tab.

Select VPM.

For **backbonereceiver**, input the name of the machine where the VPM Client is running.

For **backbonesender**, input the name of the machine where the PPRLoader is running.

Scroll down to **installdir**, and enter the directory in which PPRLoader.exe is installed.

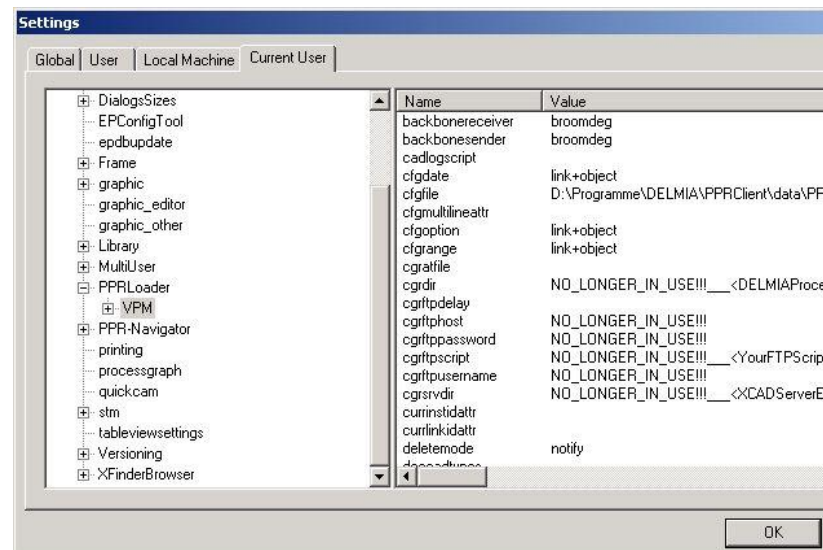


Figure 6: Settings

For information about the details of PPRDaemon’s role, *Please refer to the PPRDaemon.*



#### Note

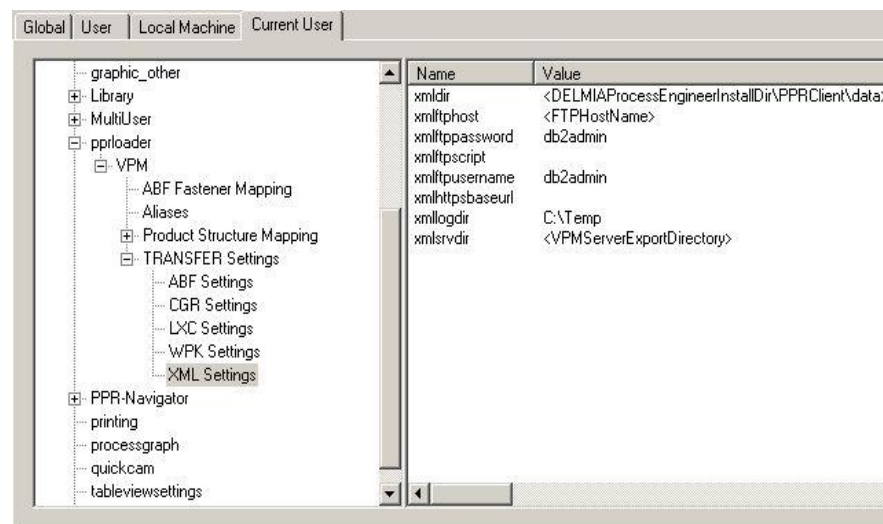
*The automatic login using the –login command line option*  
`pprloader -login $username/$password`

is no longer supported. Please use the SimpleCryptAdmin tool to create an encrypted registry entry. Please refer to the [Encryption and Decryption of Registry Settings](#).

### 3.6.2 Configure XML Settings

On the Current User tab, in the left frame, under VPM, click on the plus sign (+) to expand the tree. Under **TRANSFER Settings**, click on the plus sign (+) to expand the tree, and then select **XML Settings**.

- For **xmldir**, enter the pathname of the directory in which you want the XML files stored.
- For **xmlftphost**, enter the name of the VPM database server.
- For **xmlftppassword**, enter your UNIX password.
- For **xmlftpusername**, enter your username.
- For **xmllogdir**, enter the path to find XML.
- For **xmlsrvdir**, enter the path to find XML on the VPM server.

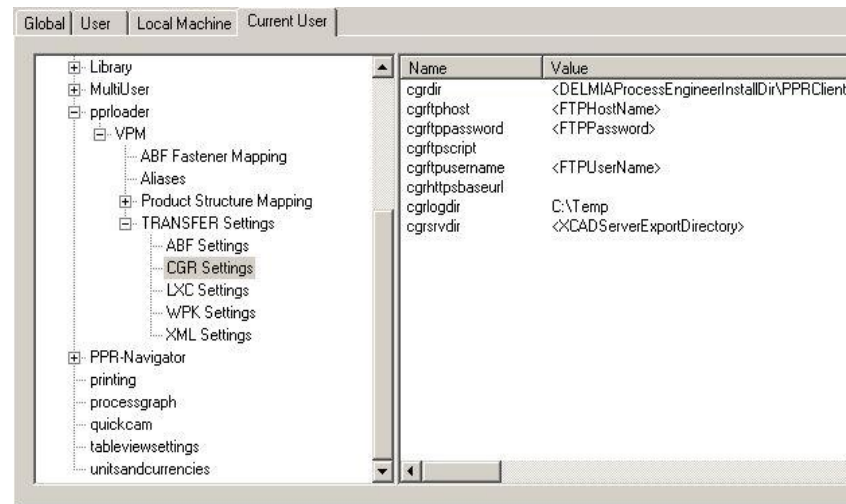


**Figure 7: Configure XML Settings**

For a comprehensive discussion of the FTP file transfer, see [File Transfer via HTTPS, HTTP, or FTP](#). Configure the CGR Settings.

On the Current User tab, in the left frame, select **CGR Settings**.

- For **cgrdir**, enter the pathname of the directory in which you want the CGR files to be stored.
- For **cgrftphost**, enter the name of the VPM database server.
- For **cgrftppassword**, enter your UNIX password.
- For **cgrftpusername**, enter your username.
- For **cgrlogdir**, enter the path to store CGR log files
- For **cgrsrvdir**, enter the path to find CGR files on the VPM server.



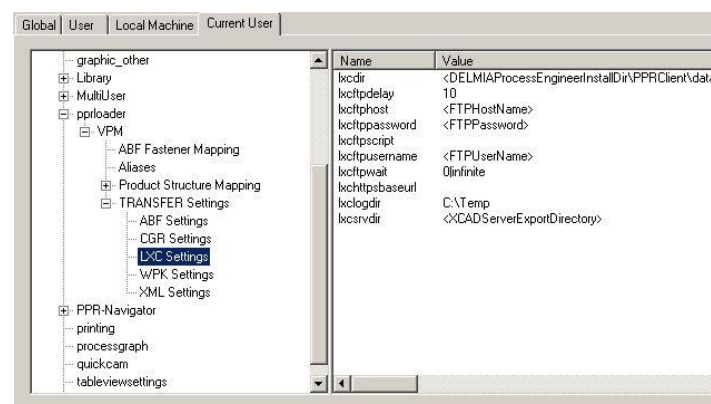
**Figure 8: CGR Settings**

For a comprehensive discussion of FTP file transfer, *Please refer to the [File Transfer via HTTPS, HTTP, or FTP](#).*

### 3.6.3 Configure LXC Settings

On the **Current User** tab, in the left frame, select **LXC Settings**.

- For **lxcdir** enter the pathname of the directory in which you want the LXC files stored.
- For **lxcftpdelay**, enter the delay time for a FTP.
- For **lxcftpshost**, enter the name of the VPM database server.
- For **lxcftpsscript**, enter your UNIX password.
- For **lxcftpusername**, enter your username.
- For **lxcftpwait**, enter the time to wait for FTP.
- For **lxclogdir**, enter the path to store LXC log files.
- For **lxcsrvdir**, enter the path to find LXC files on the VPM server.



**Figure 9: LXC Settings**

For a comprehensive discussion of FTP file transfer, *Please refer to the [File Transfer via HTTPS, HTTP, or FTP](#).*



## 3.7 PPRLoader Customization Settings

### 3.7.1 Set Attribute Mappings

Create the configuration file **vpm.cfg** and save it in the ~\DELMIA\PPRClient\data\pprloader directory. The vpm.cfg follows the format below:

```
[attribute0]
general (name=attribute_2)
context (vpm=C_ORDER;significant=true)
[attribute1]
general (name=attribute_2)
context (vpm=C_CREATE;significant=true)
[attribute1]
general (name=attribute_2)
context (vpm=C_LASTMOD;significant=true)
```

**C\_ORDER**, **C\_CREATE** and **C\_LASTMOD** are the real names of VPM attributes.

Setting **significant equal to true** activates the **update protocol** of the attribute.



#### Note

*The example above is a configuration file valid for all environments. If you want a configuration for a special environment, you have to add the environment type like this:*

```
[attribute1]
general (name=attribute_2)
context (vpm=ENOVIAVPM.PART_LIST.V_status;significant=true)
```

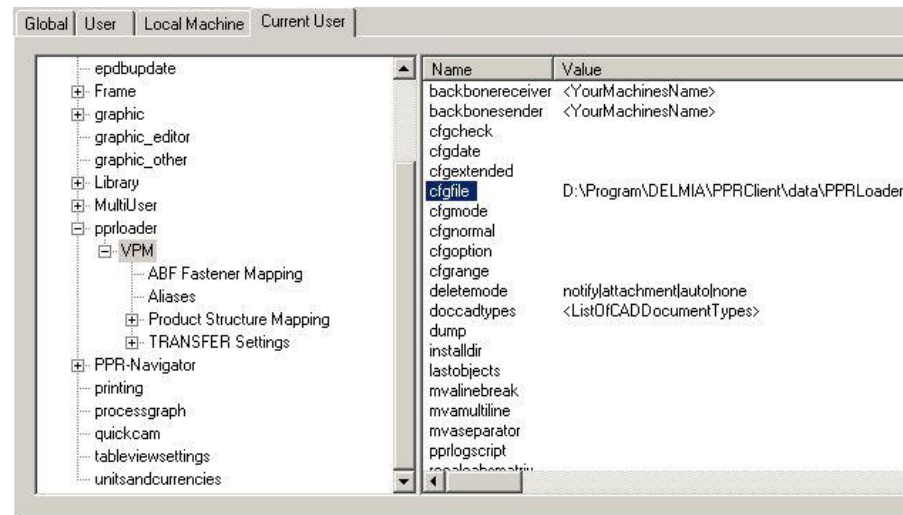
On DELMIA Process Engineer select **Tools->Settings ->Maintenance Tools**, and then select the Current User tab.

Select **VPM** in the left frame.

In the right frame, for **cfgfile**, input the path of the directory in which you have stored the **vpm.cfg** file.

```
[attribute2]
general (name=attribute_2)
context (vpm=C_LASTMOD;significant=true)
```

- If you do not use a config file, the attributes “name” and “nameshort” are taken to be significant by default.
- If you use a config file, you have to explicitly mark “name” and “nameshort” as significant. In this case the above default is no longer valid



**Figure 10: Cfgfile**

For a comprehensive discussion of mapping, *Please refer to the [Customized/Extended Attribute Mapping](#).*

For a general discussion of how to use the configuration manager *Please refer to the Appendix B, [General Information – by Administration Manual](#).*

## 3.8 Reference Information

### 3.8.1 PPRDaemon

The purpose of the PPRDaemon executable (in the `...\PPRClient\program\bin` directory) is to manage all communication between the ENOVIA VPM V4 and the Manufacturing Hub Windows world, except the actual import of the ENOVIA VPM V4 export XML file, which is done through the PPRLoader application. I.e. PPRDaemon is a user-interface-less, Windows 32 application that serves as:

- A “PPRListener,” insofar as it communicates with the ENOVIA VPM V4 Portal (retrieving, analyzing and responding messages)
- A “FTP File Fetcher,” insofar as it transfers all required files from the ENOVIA VPM V4 UNIX server side to the Manufacturing Hub Windows world.

In detail, these steps are:

- Listening to messages from the ENOVIA VPM V4 Portal, sent through the CAT Backbone
- Returning the list of projects to the ENOVIA VPM V4 Portal
- Fetching the ENOVIA VPM V4 export XML file from the UNIX side (ENOVIA VPM V4 export repository)
- Invoking PPRLoader with the appropriate parameters for ENOVIA VPM V4 export XML file and project id

```
pprloader -vpm ExportBOMData... -project $id$...
```





PPRDaemon itself does not import the ENOVIA VPM V4 XML file and PPRLoader is not responsible for the communication to the ENOVIA VPM V4 Portal and the FTP fetches.

### Note

*Like in the previous versions of the Engineering Hub to Manufacturing Hub Connection, PPRLoader can also run in Listener mode, performing all steps of PPRDaemon, but it is recommended not to do so*

When the PPRDaemon invokes PPRLoader to import an ENOVIA VPM V4 XML export file, it creates a new (console window) process. By doing so, the communication to ENOVIA VPM V4 (via the CATBackbone process and the ENOVIA VPM V4 Portal) is now separated from the import, which offers the following benefits:

- No permanent console window/connection functionality is now available on demand
- Asynchronous mode/Parallelization
- Stability/Memory Handling

PPRDaemon is based on a subset of the very same registry keys used by PPRLoader, with one exception. Reusing the registry key avoids additional customization. The only new registry key is:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "installdir"
```

This key might contain a different path, depending on from where PPRDaemon tries to invoke PPRLoader. An example of this key with a different path is:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "installdir"=
"C:\DELMIA\patch0703"
```

It is still possible (e.g., for debug purpose) to run PPRLoader in listener mode. Nevertheless, this is not recommended. Like PPRLoader, PPRDaemon logs all actions and messages into a log file.

The command line, executed from the PPRDaemon process, starting the PPRLoader import process is:

```
$installdir\pprloader -vpm $file -project $id
```

where `$installdir` is either the value from the registry key or the start directory of PPRDaemon itself.

`$file` contains the XML file name, including the local data path.

`$id` specifies the project in which to import the data; that is, the object id of the project selected in the ENOVIA VPM V4 Portal.

The following image shows the interaction and data flow between the ENOVIA VPM V4 Portal, PPRDaemon and PPRLoader.

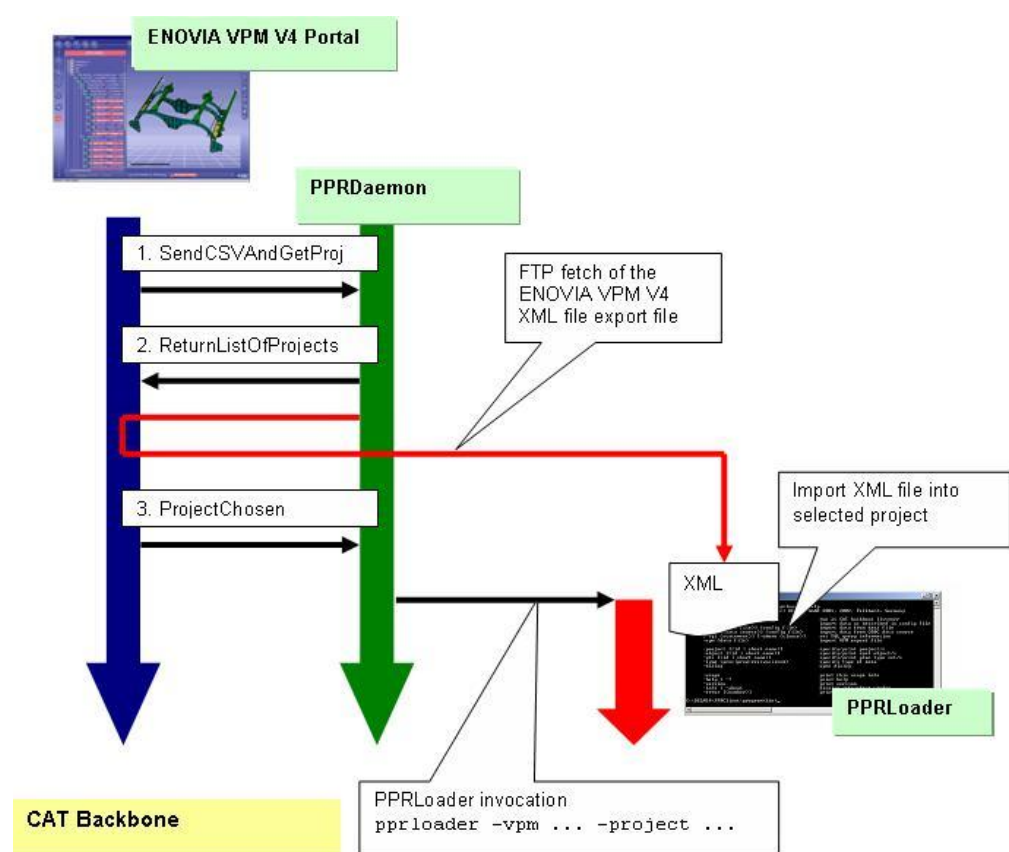


Figure 11: Interaction and Data flow between the ENOVIA VPM V4 Portal, PPRDaemon, and PPRLoader

### 3.8.2 Product Structure Mapping

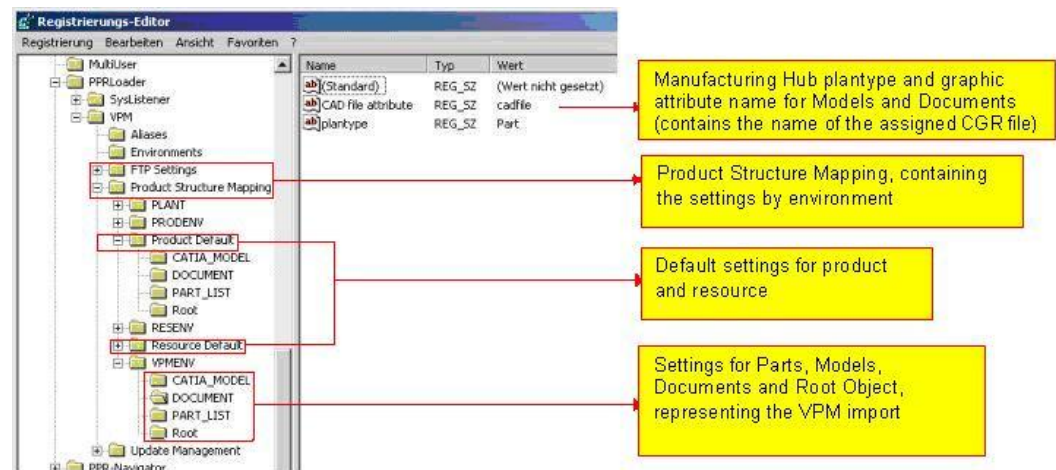
Adapt and modify the Product Structure Mappings under ENOVIAVPM according to your actual ENOVIA VPM V4 scenario. This includes entering/changing the name of the desired Plantypes, corresponding to your DPE PlanTypeSet.

#### 3.8.2.1 Multi-Environment

The Engineering Hub to Manufacturing Hub Connection supports multiple environments from VPM V4. As a result, PPRLoader now allows the individual mapping of VPM types from different environments into different Manufacturing Hub plantypes. Therefore, it is recommended to register all environments from VPM V4 into the corresponding registry keys. Each environment mapping must contain the table name / plantype mapping from VPM to Manufacturing Hub.

For Models and Documents the settings must also include the name of the attribute, in which the CGR file name should be stored. Beside those three settings (for PART\_LIST, CATIA\_MODEL and DOCUMENT), an additional one for the root object must be specified (which actually has no corresponding representation in the VPM product structure).

In addition to the environment settings, default mappings for product and resource ensure the completeness of the data import. In difference to previous versions of PPRLoader, at least the default settings must be included in the registry; there are no hard-coded default values for this. The following figure shows the corresponding keys from the registry:



**Figure 12: Keys from the Registry**

### Note

Please clarify that in the product structure mapping all existing environment definitions must be complete and not empty

## 3.8.3 Multi-Instancing

The Engineering Hub to Manufacturing Hub Connection maps the ENOVIA VPM V4 product structure reference tree into a Manufacturing Hub instance model.

The figure below represents the ENOVIA VPM V4 and Manufacturing Hub product structure.

```
P0
+--link01a--P1
+--link01b--P1
+--link02 --P2
      +--link21 --P1
```

Each Part or Document in the product structure can be identified through a unique ID from ENOVIA VPM V4, containing the VPM environment of the Part, its table name and its OID (object Id). The format in which the ID is stored in Manufacturing Hub is: `VPMEEnvironment.TableName.OID`.

The attribute used to store the Part (or Document) ID is "externalid" on the "ErgoCompBase" object.

### Example

```
P1=VPMENV.PART_LIST.3D3D96FD9DE551A2303030303030303
```

Links have a similar identifier. Subassembly and Part links have "\$EXT" as table name. Links between Parts and Models or Documents are not represented through link objects in VPM. They do not have a specific table name, since they are stored differently in VPM V4. Therefore, to simplify the mapping, links in Manufacturing Hub /Manufacturing Hub contain the table name of the child object, encapsulated within "CLink(" ... ") as the VPM ID. The attribute used to store the link ID is "externalid" on the "SubCompItem" respectively the "Relationship" object.

**Example**

```
link02=VPMENV.CLink(PART_LIST).3D3D96FD9DE551A23D3D96FD9E541B8F
```

ENOVIA VPM V4 allows the storage of attributes specific for an instantiation of a Part.

To allow the same functionality (plus providing things such as code rule inheritance between product and process or different process relations on different product instances), it is necessary to map part references from ENOVIA VPM V4 into part instances in Manufacturing Hub, each of which is identified by a unique instance ID.

To identify part instances from ENOVIA VPM V4 uniquely, it is necessary to assemble the path of link IDs from the root object to the instance. The instance ID of an instance is the path of link IDs from the root to the instance, separated by a '+' character.

The instance id is stored on the attribute "vpm\_instanceid1".

The figure below represents the Manufacturing Hub product structure.

**Manufacturing Hub**

```
P0.1
```

```
  +--link01a--P1.1
```

```
  +--link01b--P1.2
```

```
  +--link02 --P2.1
```

```
      +--link21 --P1.3
```

**Example:**

```
link02+link21=
ENOVIAVPM .CLink(PART_LIST).
3D3D96FD9DE551A23D3D96FD9E541B8F _...+
ENOVIAVPM .CLink(PART_LIST).
3D3D96FD9DE551A23D3D93FE5B28FC2A _...
```

The usage of the attribute "vpm\_instanceid1", used to store the instance ID, can be overwritten through the following registry key:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM]"currinstid"
```

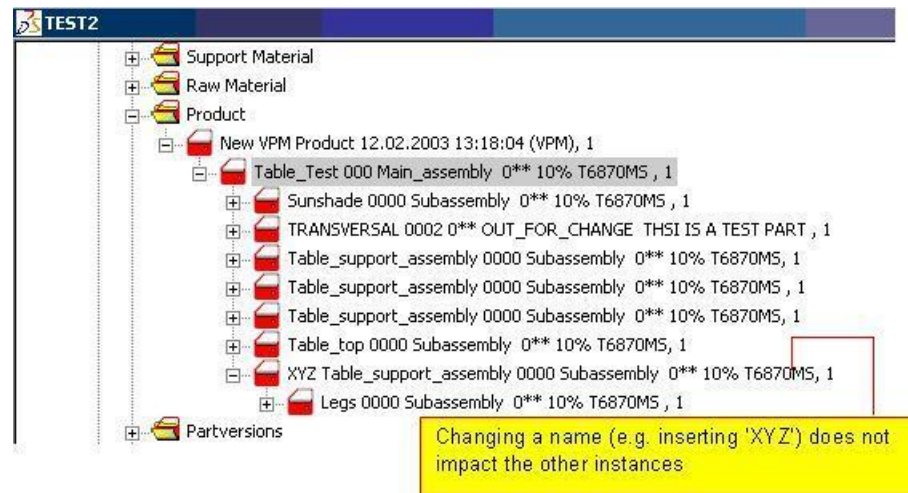
**Example:**

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM]"currinstid"=
"attribute_1"
```

The correct multi instance behavior can be checked in several ways:

Each object/component occurs once, and only once in the product structure. This can be checked by using the "Find Usage" function from the DPE context menu of the selected instance or by script.

- When the name (or any other attribute) is changed on one instance, this has no impact on any other instance in the DPE product structure.



**Figure 13: Changing Name**

If a part is moved from one branch of the product structure to another (especially when the hierarchical level inside the structure is changed), there is no way to re-identify the instances beneath.

### Example

P0.0

```

+--link01a--P1.1
+--link01b--P1.2
+--link02 --P2.1
      +--link21 --P1.3
      +--link21'--P1.?
    
```

All Manufacturing Hub part instances of a given ENOVIA VPM V4 part reference do not know about the other part instances, except that fact that they have an identical ENOVIA VPM V4 ID. They also do not have a common data source (e.g., they do not share a set of common attributes).

## 3.8.4 Resource Support

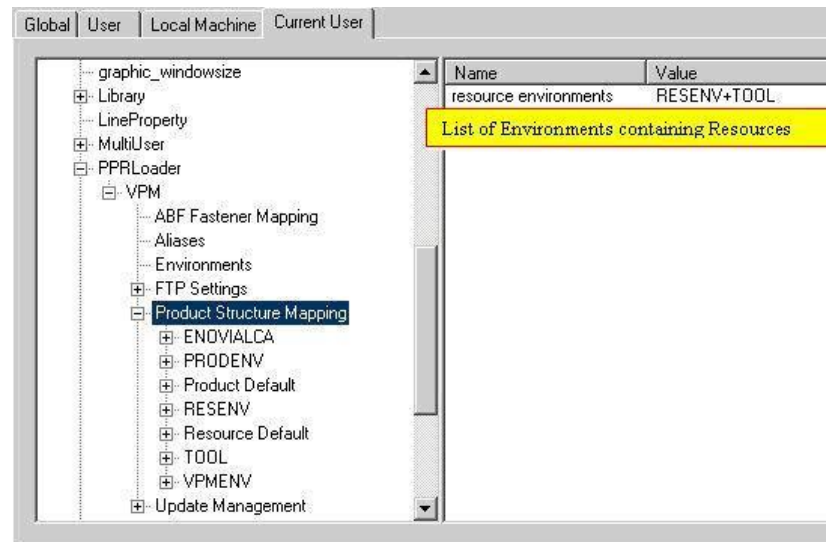
The basic settings for supporting resource environments were already introduced in the previous section.

The handling of resources includes the independent mapping between VPM table names and the corresponding Manufacturing Hub plantypes, along with a new location of the registry key, specifying which environments contain resources and should be treated as such. The relation currently used to link resources to product data in Manufacturing Hub is "plant\_provides\_prod".

### Example

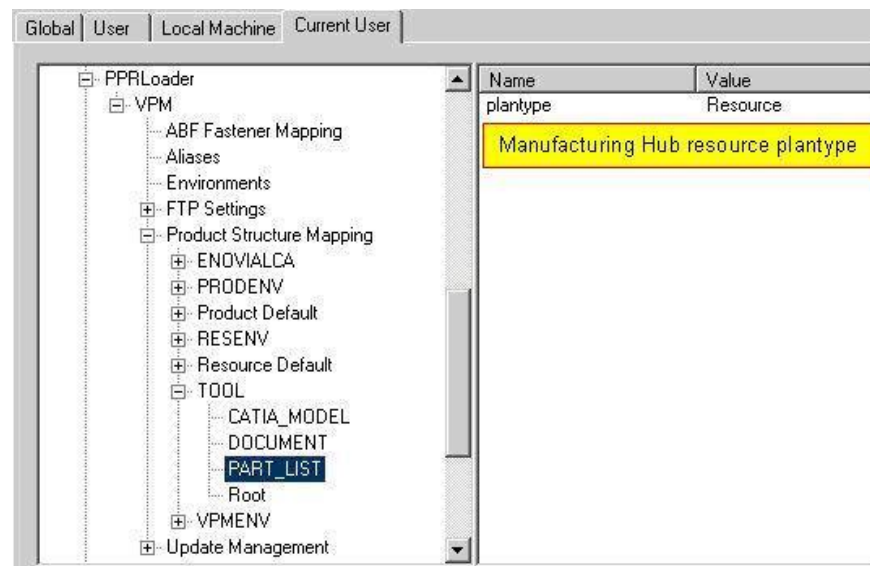
```

[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM\Product
Structure Mapping]"Resource
Environments"="RESENV+TOOLING+PLANT"
    
```



**Figure 14: Resource Environments**

In addition the Plantype setting must contain the appropriate Resource Plantype Names.



**Figure 15: Resource Plantype Names**



### Note

The previous registry key  
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM]"resenvs"  
is out of use.

## 3.8.5 Initial Import

The first time a product is imported into DPE, a Top Level Product Folder will be created in the Project Library. Under this product node, the most upper subassembly representing the top level part will be created.

### Importance of Subassembly node, representing the top level part node

The existence of the Subassembly node representing the top level part indicates whether to perform an update or an initial import (create everything new). I.e. in case of the absence of the top level part node in the Project Li-



library, the import mechanism tries to create a new one, e.g. it starts an “initial import” (that creates a new Top Level Product Folder, a new top level part node and the complete product structure).

The product folder itself can be considered just as a structuring element. Date and time of the import are stored in the name of the product folder. This gives a better orientation to the user when navigating across complex project structures after an import.



### Note

*You can delete the product folder (with option “component, flat”), this does not influence the update mechanism. On the other hand, the update mechanism will not create a new Product Folder since it takes its information only from the node representing the top level part.*

*If you delete the node, representing the top level part in DPE, the update process cannot work correctly. In this case it is recommended to delete the whole product with all components and to start a new initial import.*

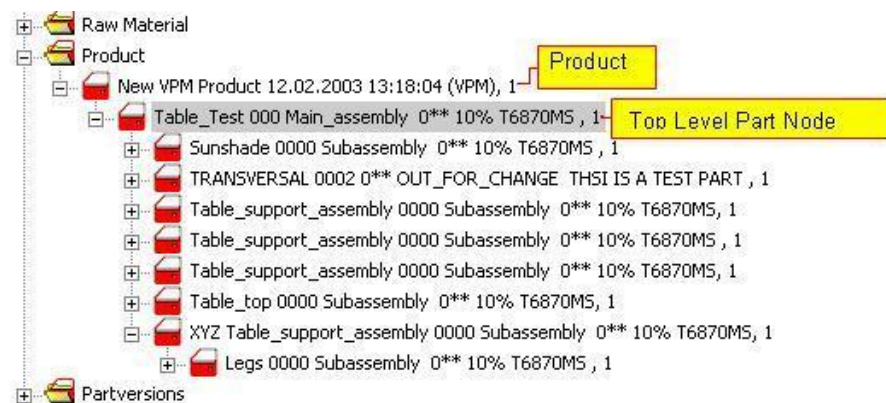


Figure 16: Product Folder

## 3.8.6 Update Protocol and Management

The Update Protocol includes functionalities that determine the status of an object or link after an import/synchronization. The identification of those changes is part of the Engineering Hub to Manufacturing Hub Connection. Beyond that, all additional features such as gathering the modified objects and links (e. g., by script or finder) and notifying the end user (e.g., by changing the bitmap or creating reports) is not under the control of the PPRLoader application.

Update Folders and Log Files, known from previous versions of the ENOVIA VPM V4 scenario, turned out to have several restrictions and limitations. While Update Folders were well accepted by the end user (since this mechanism includes a direct and simple notification of all changes), it is less efficient in terms of performance. Moreover, log files require additional parsing and displaying capabilities of the required data.

The current Update Protocol uses a dedicated attribute “updatestate” to store the status of the last update/synchronization to mark each instance object.

To display the update state of the products imported into DPE, you have to make this attribute visible in the browser and the editor.

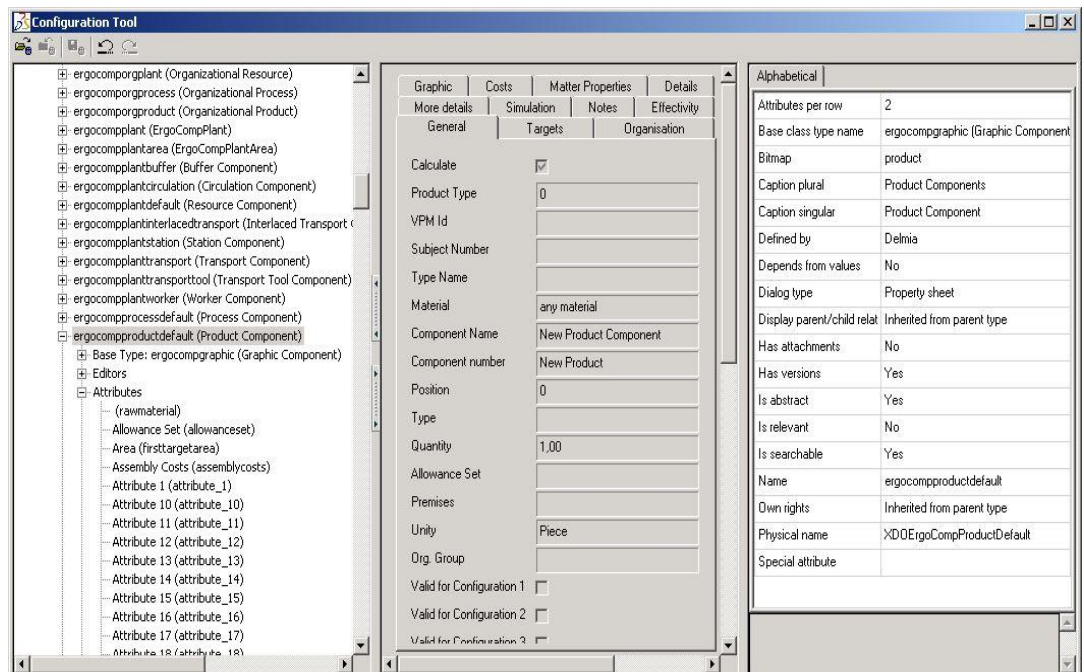
On DELMIA Process Engineer, select

**Tools->Database Utilities->Configuration Manager.**



**Figure 17: Config Manager**

- The Configuration Manager pops up.



**Figure 18: Config Tool**

- 1) In the tree view (left frame) select type “ergocompproductdefault” and expand its attributes.
- 2) Select attribute “updatestate”.

If the type “ergocompproductdefault” has no attribute “updatestate”, go to the Base Type “ergocompbase” and overwrite the attribute “updatestate”.

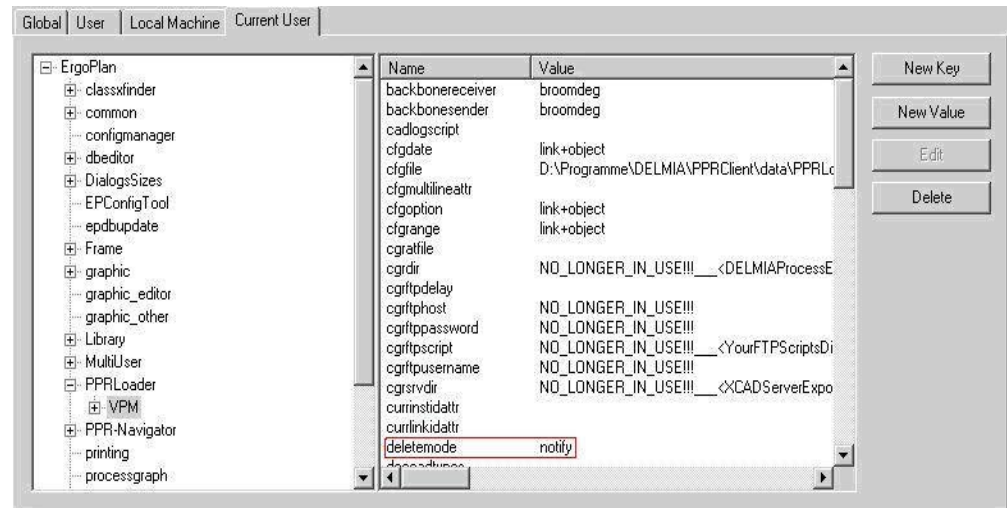


For overwriting attributes and change their properties, *please refer to the Administration Manual*.

- 3) Go to the properties editor to alter the settings:
  - Set “Display in browser” and “Display in editor” to “Yes”
  - Select Group (1) on Page “General”
  - Position in browser/Position in editor: 1010
  - Set Prompt “VPM Update Status”



- 4) Select **Tools -> Settings -> Maintenance Tools**, and then select the **Current User** tab.
- 5) Select VPM in the left frame.
- 6) In the right frame, for `deletemode`, input: `notify`



**Figure 19: Deletemode**

Currently, only string values/attributes are supported. Please see also the registry file template for details.

The Update Protocol supports the following states:

- New (for objects and links, indicated by the value “new”)
- Updated (for objects, indicated by the value “update”)
- Repositioned (only on links, indicated by the value “position”)
- Deleted (indicated by the value “delete” in case the delete mode is set to “notify”)
- Config (for objects and links, indicated by the value “config”)

### 3.8.6.1 Priorities of Update Status

Priority of Update Status for **parts**:

- new + delete > update > position > config

Priority of Update Status for **links**:

- new + delete > position > config

### 3.8.6.2 Significant Attributes

Please note that only significant attributes are checked for modifications to determine the update status “updated”. By default, only the (Manufacturing Hub) attributes “name” and “nameshort” are configured that way (hard-coded).

To set an attribute to significant, use the following syntax in the ENOVIA VPM V4 config file (section):

```
;ENOVIAVPM
[attributeN]
```

```
general (name=...)
context (vpm=...;significant=true)
```

To set all attributes to significant, you can use the following statement in the config section:

```
[config]
defaultcontext=significant=true
```

This applies the context significant to all attributes.

Since the distinction between update status for links and instances is confusing for the end-user and it is actually not required in a multi-instance scenario, the status for the link is also set on the instance – in case the instance update status is empty.

For objects deleted in ENOVIA VPM V4, PPRLoader supports two different modes:

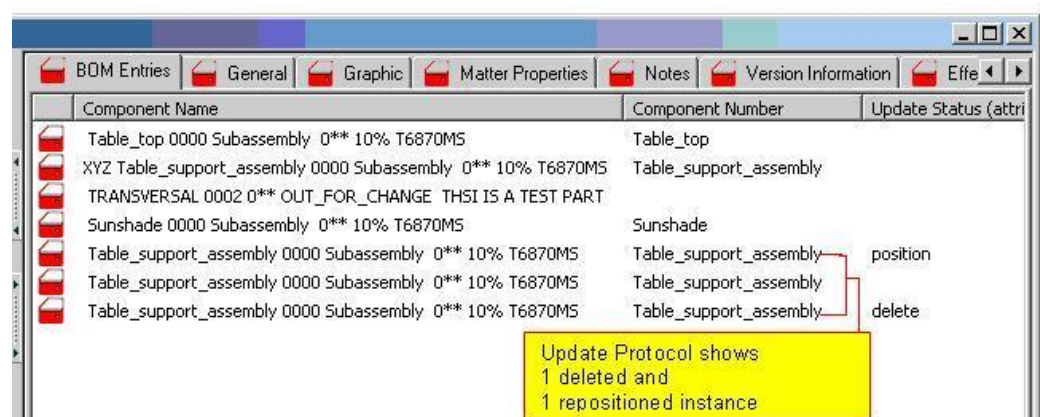
- “auto” deletes automatically all links that are no longer in use. This is the default setting. The corresponding instances stay in the Manufacturing Hub database, to avoid data loss.
- “notify” just marks the deleted/removed objects.
- “none” neither deletes nor notifies the deleted object on DPE side.

Select the mode using the following registry key:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "deletemode"=
"..."
```

To use the notify mode, set the value of this registry key to “notify” (or “0” or “false”). All other values will use the default mode, which is “auto”.

For example: “deletemode”=“notify”



**Figure 20: Update Protocol**

Since the Engineering Hub to Manufacturing Hub Connection cannot compare content, size or modification date of a CGR geometry attached to a part, there’s no dedicated update state for “geometry updated”. Nevertheless, the Connection can be configured so that a comparison of the ENOVIA VPM V4 attribute “C\_LASTMOD” for DOCUMENT objects is performed.

The following section demonstrates as an example the usage of the context token “significant” on those ENOVIA VPM V4 attributes, to setup the required behavior:

```
;VPMENV
[attribute17]
general(name=attribute_28)
context(vpm=VPMENV.DOCCAD.C_LASTMOD;significant=true)
```



### Note

*It is not possible to SET the Manufacturing Hub attributes modification date “modificationdate” or creation date “creationdate” (since they are handled internally by the PPRHub database/the server). A comparison to track modifications therefore makes less sense, since the value from ENOVIA VPM V4 is not reflected in those attributes at all. Nevertheless, if you want to track changes on this attribute, store the ENOVIA VPM V4 value in one of the free default attributes (“attribute\_1” to “attribute\_55”) by adapting the config file and compare the values from ENOVIA VPM V4 and Manufacturing Hub based on this.*

*In addition, setting and comparing configuration attributes (like “effectivity.start”, “tailnumber” ...) does not yield meaningful results. These attributes are treated a totally different way and must be handled separately.*

### 3.8.6.3 Customized Attributes for Updatestate

It is possible to specify a different attribute than the default “updatestate” for the Update Status attribute for links (SubCompltem) and components (ErgoCompBase). When doing so, please ensure that those attributes exist. Even though the status of links is transferred to the child object, it is especially important to verify that the attributes exist.

There are no free attributes (“attribute\_1” to “attribute\_55”) on links. For instance, a setting like:

```
HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "updatestateat
trlink"="
attribute_19"
```

results in a Connection internal use of the links update state, but the state cannot be stored in the Manufacturing Hub database (on the SubCompltem object).

The set of available attributes can be checked in the DPE Configuration Manager Tool.

### 3.8.7 Customized/Extended Attribute Mapping

To use Customized/Extended Attribute Mapping you must create a config file, similar to the regular ones (used for standard PPRLoader imports), that describes the specific mapping of the ENOVIA VPM V4 attributes and the Manufacturing Hub attributes (administered with the Configuration Manager in DELMIA PE). This file has to be defined using the following registry key:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgfile"="..
."
```

The example below illustrates how to use this file. Below is a portion of an ENOVIA VPM V4 export file. The default environments are PRODUCT for parts and DOCDIR for documents.

```
- <Attribute Name="V_user" Type="String">
  <Value Value="APV" />
</Attribute>
- <Attribute Name="V_organization" Type="String">
  <Value Value="ADMIN" />
</Attribute>
- <Attribute Name="V508_AppDomaine" Type="String">
  <Value Value="ADMIN" />
</Attribute>
```

This example shows three attributes: V\_user, V\_organization, and V508\_AppDomaine. Each is defined at the PART\_LIST table inside ENOVIA VPM V5.

Each attribute has an [attribute...] entry in the config file. In our example, the entries could look like this:

```
[attribute0]
general(name=nameshort) (Name of the PPR Hub attributes)
context(vpm=PRODUCT.PART_LIST.V_user)
[attribute1]
general(name=attribute_12)
context(vpm=PRODUCT.PART_LIST.V_organization) (Name of the
ENOVIA VPM V4 attribute including environment and table
name)
)
[attribute2]
general(name=attribute_18)
context(vpm=PRODUCT.PART_LIST.V508_AppDomaine)
```



### Note

*Please note, that there's only one config file for all types and table names. Document attributes will have a different table name prefix (DOCUMENT).*

Also new is the notation using the `general (name...)` format.



### Note

*Please note, that the declaration of the plantype, an attribute mapping is used for, is no longer required, since this information is already specified through the Product Structure Mapping registry keys along with the configuration of the Manufacturing Hub database itself.*

The order of the attributes is of no importance, but keep in mind that the sequence of attributes must be continuous starting with 0 [attribute0].

PPRLoader tries to convert the ENOVIA VPM V4 attributes (always exported as strings) according to the attribute type in Manufacturing Hub. Currently, PPRLoader supports conversion into five different types:

- double/float
- integer
- boolean

- date/time
- string (default)

The type of the Manufacturing Hub attributes is read from the Manufacturing Hub configuration database, so no additional information is required in the config file for the “type” conversion.

### Example

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgfile"=
"D:\DELMIA\d
```

The following text shows a section from a Configuration file used for attribute mapping between ENOVIA VPM V4 Environment VPMENV and the Manufacturing Hub.

```
;ENOVIAVPM
[attribute0]
general(name= attribute_17)
context(vpm=VPMENV.PART_LIST.C_STATUS)
[attribute1]
general(name=attribute_17)
context(vpm=VPMENV.DOCUMENT.C_STATUS)
[attribute2]
general(name=attribute_12)
context(vpm=VPMENV.PART_LIST.S_TYPE)
[attribute3]
general(name=attribute_12)
context(vpm=VPMENV.DOCUMENT.S_TYPE)
```

It is also possible to map attributes from the ENOVIA link to the instance in Manufacturing Hub (since there is only a limited number of free attributes on the subcompitem/link object in DPE). To accomplish this, the environment and type `PRODUCT._CLink_(PART_LIST)` is required, independently from the real environment the link/part belongs to. I.e. there will be no `ENOVIALCA._CLink_(PART_LIST)` entry in the mapping file. In addition, there will be no `PRODUCT._CLink_(DOCUMENT)` contexts.

```
[attribute0]
general(name=attribute_24)
context(vpm=PRODUCT._CLink_(PART_LIST).V_volume_x1)
(Mapping for link attributes start always with
PRODUCT._CLink_(PART_LIST)
)
[attribute1]
general(name=attribute_25)
context(vpm=PRODUCT._CLink_(PART_LIST).V_volume_y1)
[attribute2]
general(name=attribute_26)
```

```
context(vpm=PRODUCT._CLink_(PART_LIST).V_volume_z1)
...
[attribute7]
general(name=attribute_12)
context(vpm=V_nbsubstitute)

[attribute8]
general(name=attribute_13)
context(vpm=V_issubstitute)
```



### Note

*All registry settings under*

*[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM\Environments]*

*are obsolete. The attribute mapping is contained in one and only one configuration file, which has to be registered in:*

*[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM]"cfgfile"="..."*

### 3.8.7.1 Attribute 'logical CompID'

With VPM V4 PTF 33, the VPM V4 export XML file will contain the new VPM V4 attribute "logical CompID" for assembly links ("{\$EXT.\$COMPID\_LOGICAL").

```
ReferenceRef="OSM1://3a4d5a9b82e9c1f56397dda1ebf678a7#fb8986b65226a027d5c18a6aa9225e9d"
Alias="_CLink_(VPMENV.PART_LIST) P1_1122_Config 1 Part --- 10%">
- <Attribute Name="{$EXT.$COID}">
  <Value Value="41419ADEAC403320" />
</Attribute>
- <Attribute Name="{$EXT.$COMPID}">
  <Value Value="41419ADEAC8D7BC5" />
</Attribute>
- <Attribute Name="{$EXT.$COMPID_LOGICAL}">
  <Value Value="41419ADEAC9507C9" />
</Attribute>
- <Attribute Name="{$EXT.$SITE}">
  <Value Value="1" />
</Attribute>
```

In difference to the historical CompID, currently used for re-identification of VPM links in Manufacturing Hub, the logical CompID stays the very same for the entire life cycle of the VPM V4 assembly link, regardless of performed versioning (of parent part) and/or positioning/replacing operations done in VPM V4. The logical ID might have the very same value than the historical ID (e.g. for new links), but need not.

The usage of the new logical CompID will allow a gap-less re-identification of the VPM links in Manufacturing Hub by the Connection, throughout the entire chain of transferred links.

On VPM V4 side the new logical IDs will be initialized with the value of the "old" historical ID. Therefore a smooth upgrade path is given.

#### Limitation

It might be required to perform additional tasks, when upgrading existing data to DPE 5.17 and higher (synchronization of existing IDs).

Since the logical CompID replaces the historical one, the historical ID value is lost.

### 3.8.7.2 Attribute Length

In case you need to visualize the IDs coming from ENOVIA VPM V4 (instance ID, external ID ...), you have to ensure that the displayed length is big enough to keep the entire (string) value of the ID.



The historical instance ID ("pprloaderhistoricalid") and the current instance id of the objects - containing the environment, the table name, two '.' characters and the OID - result in a string of appr. 50 characters length.  
As a reminder, the following formula can be used:

$$L_{id} = D_{VPM} * 50$$

where  $L_{id}$  is the length of the instance ID attribute in characters and  $D_{VPM}$  is the depth of the product structure from ENOVIA VPM V4. The value of 1024 characters used in most L1/L2 test scenarios would e.g. support product structures in ENOVIA VPM V4 up to 20 levels. This length has to be used in the "Extended Properties" page of the corresponding attribute, when switching it to 'visible' in the Configuration Manager Tool of DPE.

### 3.8.8 Configuration File

For reference attributes which are mapped, the mapping description in the configuration file must contain the VPM V4 environment and type information (see [attribute2]). If this is not specified (see [attribute3]), the attribute describes a mapping for instance attributes and in addition serves as a mapping template, valid for *all* VPM V4 environments and types.

```
context(vpm=PRODUCT.PART_LIST.V_order)
```

```
[attribute2]
```

```
general(name=attribute_2)
```

```
context(vpm=PRODUCT.PART_LIST.V_status)
```

VPM V4 environment and type for reference attribute mapping.

```
[attribute3]
```

```
general(name=dbl_attribute_2)
```

```
context(vpm=V_level)
```

Just the attribute name for instance attributes and reference attributes templates, valid for ALL VPM V4 environments and types.

```
[attribute4]
```

```
general(name=dbl_attribute_2; type=Part)
```

```
context(vpm=V_level)
```

The type, indicating that this mapping applies only to this particular plantype.

```
[attribute5]
```

Please note, that the plantype definition for the attribute

```
general(name=attribute_2;) (Type = subassembly)
```

has a different meaning now as in previous releases.

Some possible obstacles and the corresponding warnings are described in the following.

In case the plantype is defined in the attribute definition (either reference or instance attribute), this definition replaces the Plantype, specified through the Product Structure Mapping).

```
[attribute4]
```

```
general(name=attribute_2; type=Part)
```

```
context(vpm=PRODUCT.PART_LIST.V_order)
```



### Caution

*Plantype definition 'Part' for attribute 'attribute\_2' ('PRODUCT.PART\_LIST.V\_order') replaces the registered plantype 'PRODUCT.PART\_LIST' (product structure mapping 'Subassembly'), see [attribute4]*

When an attribute is defined multiple times, the second, ambiguous definition is ignored.

```
[attribute2]
general (name=attribute_2)
context (vpm=PRODUCT.PART_LIST.V_status)
[attribute4]
general (name=attribute_4)
context (vpm=PRODUCT.PART_LIST.V_status)
```



### Caution

*Ambiguous mapping for VPM attribute 'PRODUCT.PART\_LIST.V\_status' ignored ('attribute\_4' <> 'attribute\_2'), see [attribute2] and [attribute4]*

In case an invalid reference attribute is specified (the attribute is not found on the registered plantype), the following warning is displayed

```
[attribute6]
general (name=attribute7)
context (vpm=PRODUCT.PART_LIST.C_created)
```



### Caution

*attribute 'attribute7' not found on plantype 'Subassembly', see [attribute6]*

For invalid instance attributes no warning is dumped during parsing the configuration file, since the corresponding plantype is unknown (it's an instance or reference template attribute mapping!).

```
[attribute7]
general (name=attribute8)
context (vpm=C_created)
```



### Caution

*Instance attribute cannot be checked upfront! ; (plantype is unknown at parsing time)*

In this situation no warning is dumped during parsing the configuration file, since the corresponding plantype is unknown (it's an instance or reference template attribute mapping!).

At the very end of the data transfer, there are also warnings for unused *reference* attribute definitions.

```
[attribute8]
general (name=attribute_9)
context (vpm=PRODUCT.PART_LIST.V_level)
```





### Caution

attribute 'Subassembly::attribute\_9' ('PRODUCT.PART\_LIST.V\_level') not used during import, see [attribute8]

Unused *instance* attributes, do not raise warnings, since it is unknown whether the attribute has been used as a reference template (attribute mapping for reference attribute, independent of VPM V4 environment and type!).

```
[attribute9]
general(name=attribute_10)
context(vpm=V_order)
```

## 3.8.9 Partial Product Data Transfer

With this functionality it is possible to select single sharing objects in a product structure and update. The main objective is to allow the possibility to re-create the context of the extracted instance/partial product structure. The main objective is to allow the possibility to re-create the context of the extracted instance/partial product structure. This will allow splitting an entire product into areas of interest which are updated/synchronized more often than the rest of the product. In ENOVIA VPM V4, it is possible to extract a portion of a complete product, but the ENOVIA VPM V4 Engineering Hub to Manufacturing Hub Connection will not be able to identify the instances, due to the missing part in the path of link ids, which builds the instance id.



### Note

*The application of this functionality is limited to product structures, which are not completely flat.*

*This functionality is only available in batch mode.*

*There is no analogue functionality in the interactive scenario.*

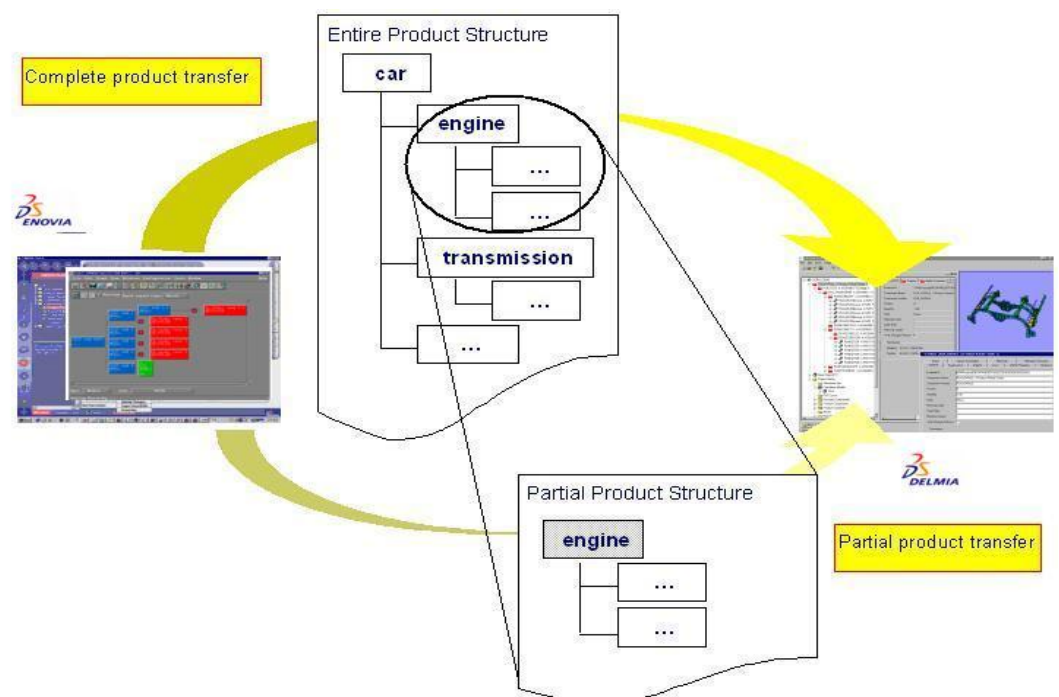


Figure 21: Data Transfer

In difference to ENOVIA VPM V5 Engineering Hub to Manufacturing Hub Connection, the instance identification, i.e. the missing part of the complete instance id, is NOT managed automatically by the Connection. Instead, PPRLoader allows specifying the path of link ids from the required root node to the selected and extracted instance through the command line. This “prefix” value is concatenated with the calculated rest of the instance ids (from the partial structure) and builds-up the complete instance id, which PPRLoader uses to identify, compare and synchronize the product structure in DELMIA Process Engineer.

To specify the heading part of the instance id, the following command line option for PPRLoader is used:

```
pprloader -vpm $ExportBOMDataFile.xml -instance
$instanceIDPrefix \
-project $ProjectIdOrShortName
```

e.g.

```
pprloader -vpm C:\Temp\ExportBOMData-2003-08-17-
14.34.21.xml \
-instance
"VPMENV2.Clink(PART_LIST).41419851B8A5265D41419851B8BE4C6C
"\
-project TK2
```

This command will import the (partial!) product structure, given in the file “ExportBOMData-2003-08-17-14.34.21.xml” in the context of the instance, defined by the id

“VPMENV2.Clink(PART\_LIST).41419851B8A5265D41419851B8BE4C6C”

No check is performed whether the specified XML file is really a valid partial structure of the existing (already transferred) complete structure in Manufacturing Hub.



### Note

*Please note, that it's of course possible to enter a multi-level instance id, such as*

*“VPMENV2.Clink(PART\_LIST).41419851B8A5265D41419851B8BE4C6C+PRO  
ODENV.Clink(PART\_LIST).41419851B89DE55141419851B8E541B8”.*

This functionality is for define and manages the set of required sub-structures of interest, which will allow the complete transfer of data in portions.



### Note

*Please note in addition, that for VPM V4, the command line option “-partial” is ignored, when “-instance” is used (since this already specifies a Partial Update)*

### Example

- Use the two sample files “ExportBOMData-Complete.xml” and “ExportBOMData-Partial.xml”, provided along with this document, store them into the client data directory (... \DELMIA\PPRClient\data)
- Open a command shell (DOS box) and go to the installation directory (... \DELMIA\PPRClient\program\bin)
- Enter the following command:  

```
pprloader -vpm ... \ExportBOMData-Complete.xml -project
VPM
```

- Enter the following command:

```
pprloader -vpm ...\ExportBOMData-Partial.xml -instance
VPMENV1.CLink(PART_LIST).3D3D97698163713C3D3D9769832B17B5
-project VPM
```

### 3.8.10 Incremental Update

To reduce the amount of data transferred between ENOVIA VPM V4 and the DELMIA Manufacturing Hub through the Engineering Hub to Manufacturing Hub Connection, it is now possible to transfer only the data between a past data transfer and the current image of the ENOVIA VPM V4 database.



#### Note

*It is strictly recommended to use this functionality only in batch mode.*

In difference to the full product XML file, the generated XML file will not contain ALL data, but just the modified objects from the ENOVIA VPM V4 database.

The main change in the functionality is the identification of modifications in the ENOVIA VPM V4 database and the identification of instances and objects, independently from the transfer of a structural/hierarchical context. If an incremental update of the product data is performed, the Connection can identify the objects to be updated.

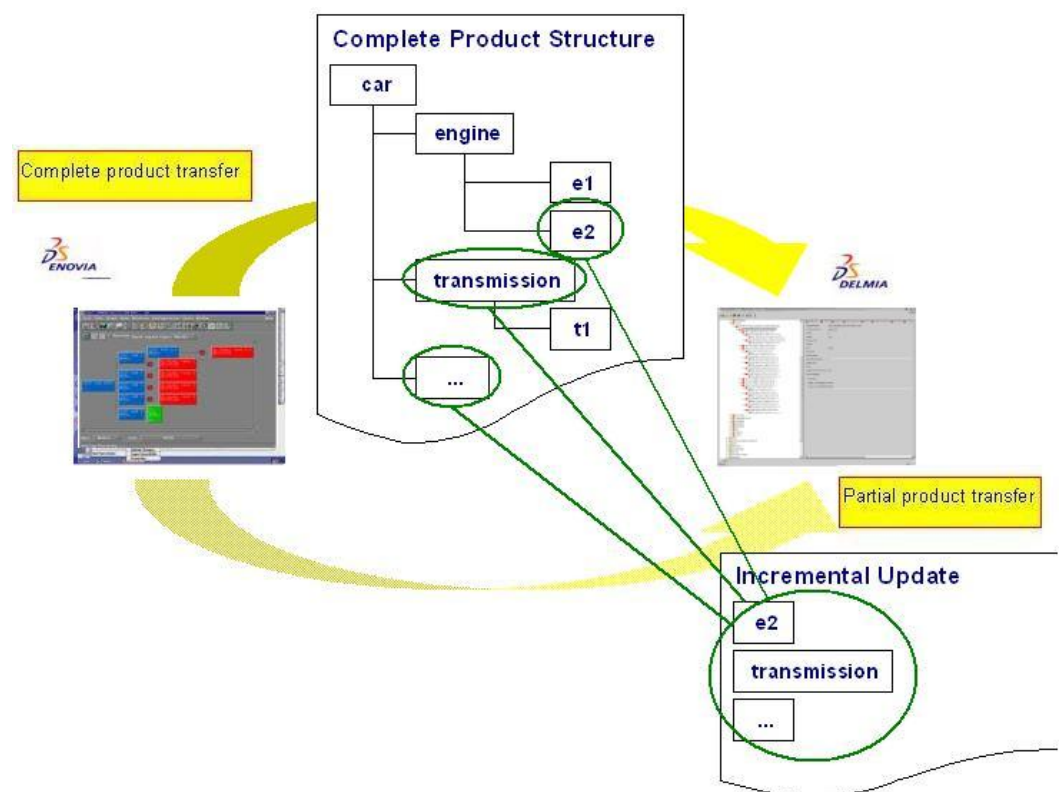
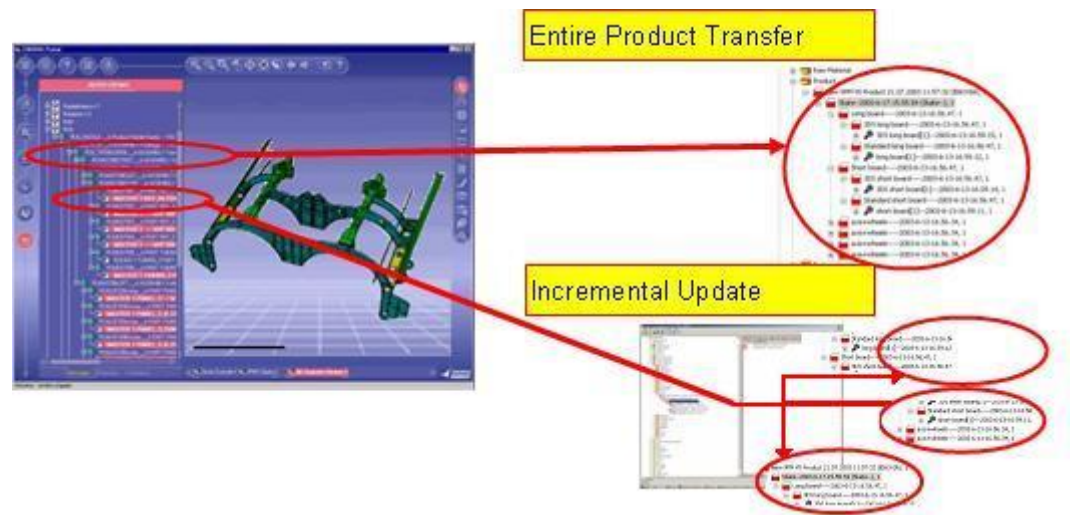


Figure 22: Data Transferred between ENOVIA VPM V4 and the DELMIA Manufacturing Hub through the Engineering Hub

ENOVIA VPM V4 – PRC/product structure

Manufacturing Hub



**Figure 23: ENOVIA VPM V4 – PRC/product structure Manufacturing Hub**

Due to the fact that no structural comparison is done on the product structure, the update will not be able to identify deleted objects.

The following object types from ENOVIA VPM V4 are tracked by their modification date:

- Parts references (Part versions)
- Assembly Relations (Links)
- Documents (Document revisions)
- Options (Specifications)
- Configuration Handlers and Filters
- Configuration Data and Effectivities

An incremental XML file can be identified through the fact, that the contained structure is not complete. In some cases, this identification mechanism fails, since the structure seems complete, even if it is only an incremental file. For this reason please use the incremental update only in batch mode.

To force the PPRLoader application to treat this file as incremental use the -incremental command line option, this will make PPRLoader apply the appropriate synchronization algorithms:

```
pprloader -vpm $IncrementalXMLFile -project $ProjectId -incremental
```

e.g.

```
pprloader -vpm C:\Temp\IncrementalAT0-2.xml -project INC -incremental
```

To force treating as a complete data file, use the -complete option. The options -complete and -incremental are exclusive, i.e. only one of them can be specified.



### Note

*Please note that the option -partial is no longer in use.*

To extract an incremental XML file from the ENOVIA VPM V4 side, please use the AT0EXPND extraction tool, provided along with the regular ENOVIA VPM setup. The syntax to extract a root node in incremental mode is:

```
AT0EXPND -savemode ... -Increment $PreviousExportDate
e.g.
AT0EXPND -savemode ... -Increment 2004-07-01-12.00.00
```

Please see the usage and help of the AT0EXPND for more details.

### 3.8.11 File Transfer via HTTPS, HTTP, or FTP



#### Note

*Only the XML files can be transferred via HTTPS, not the CGR files since HTTPS needs the fully qualified file name. If you want to transfer CGR files too, you have to use FTP.*

Only one connection can be used at one time. The bunch of XML-files has to be stored before sot (= start of transfer) in the same main directory. Subdirectories with different contents are recommended, but a clear and unequivocal nomenclature has to be done. As some of the encrypting methods need a three-time passage of the data between source and drain (the famous Hellman-Diffie-Encoding is described below) the performance couldn't be expected to be max, but it gives a very symmetric communication-design without spreading any keys! In fact the keys may be set individually and perfectly free for every file transfer. We recommend looking into the transfer-logging in order to validate transfer-time and transfer-quality.

Because of transferring the data in a factorized format, or if you prefer, in small well-formed XML-portions like pearls, no serious problem arises by any interruption of data lines. Perfectly transferred file-pearls are fulfilling the XML-Specifications and need not to be transferred again. The factorized or pearl-based-transfer helps to overcome the horizon of impossible transfers for large systems represented in the XML-file-structure. The XML-Validation can be done easily by already existing software, resp. APIs.

The usage of HTTPS instead of FTP requires different customization, as well on ENOVIA side (ensure the data export repository is accessible by an http client) as well as the DELMIA side (register https host and access information instead of the FTP access data). Once this customization is done, there's no difference between the FTP and https data transfer for the end users perspective.

#### Prerequisites:

On the ENOVIA VPM V4 side, it must be ensured, that the data export repository (i.e. the directory where all XML, CGR, ABF fastener and log files - such as lxc and lxc\_log - are stored) is accessible by the HTTP daemon.

On the DELMIA side it must be possible to switch between the current FTP file transfer mechanism and an HTTPS based one (disable FTP, enable https). The required information (HTTPS host, access information, etc.) must be registered, like the analogous information for FTP (e.g. registry settings).

The switch between FTP and HTTPS is part of the customization of the Engineering Hub to Manufacturing Hub Connection.

Once the extraction of the XML product structure export files on ENOVIA VPM V4 Engineering Hub side is completed (independently from whether in interactive scenario or batch mode), the XML reference file and the instance file must



be encrypted on disk. The possibly remaining original files have to be deleted after encryption. In addition to the XML product structure files, also the CGR files, the lxc and lxc\_log files must be stored encrypted on disk. Only exception from this is the lxc\_end file, since it only indicates the completeness of the CGR generation and contains actually no information (empty file, file size 0).

On Manufacturing Hub side, after the file transfer is completed, the required files must be decrypted before being imported into the Manufacturing Hub by the Connection. Once the content of the XML files has been read by the Connection, it must be ensured again, that all directly readable copies of the data files, must be deleted to avoid unsecured access.

The data have to be inserted into the Manufacturing Hub. None of the transfer-files is intended to residue in a repository.

It is obvious, that the additional step of encryption / decryption will have a negative impact on the overall transfer performance. The performance will be better when working in the batch mode on the command line.

### Encryption Methods

If **Hellman-Diffie**-Encoding will be applied, the sender encrypts the data with his private key. After getting the data the receiver encrypts again with his private key. The dataset is now doubly encrypted and must be sent back to the sender. The sender knows his private key and is able to remove his encryption. Remember, the data are still encrypted by the receiver's key. So they may be sent without possible insights of someone else. After second arrival of the data-package the receiver applies his key and starts to use the data.

Advantage: Nobody else knows the applied key except the encrypter, and the key may be changed as often as you want.

Problem: The Encryption-Processes have to be commutable. This has to hold for the Decryption-Processes as well.

If **RSA** (= Rivet, Shamir, Adleman) Encoding will be used, every sender gets the same Public Key. With this key he is not able to decrypt. The receiver is the one and only to use the Private Key and able to decrypt.

Advantage: This system does it with one file transfer from sender to receiver.

Problem: The Public Key must be published to every possible sender. If the Private Key will be detected somehow, every possible sender has to be alerted instantly and prohibited to use the corresponding Public Key from now on, and another Public Key has to be published instead.

PPRDaemon fetches files from the ENOVIA VPM V4 side.

The following files have to be transferred

- The ENOVIA VPM V4 XML export file
- The `.lxc_end` file (which indicates that the XCAD server has completed the generation of the CGR files)
- The CGR files (which allow the geometrical representation of the product in DPE)

In previous versions, the FTP mechanism relied on the direct customization of scripts. The current version reads the required settings from the registry and executes the corresponding HTTPS/HTTP/FTP command directly. The modified approach is reflected by new registry settings. Settings for the secured transfer have to be done on DPE side only.

- 1) securetransfermode: value 0 = FTP; value 1 = HTTPS

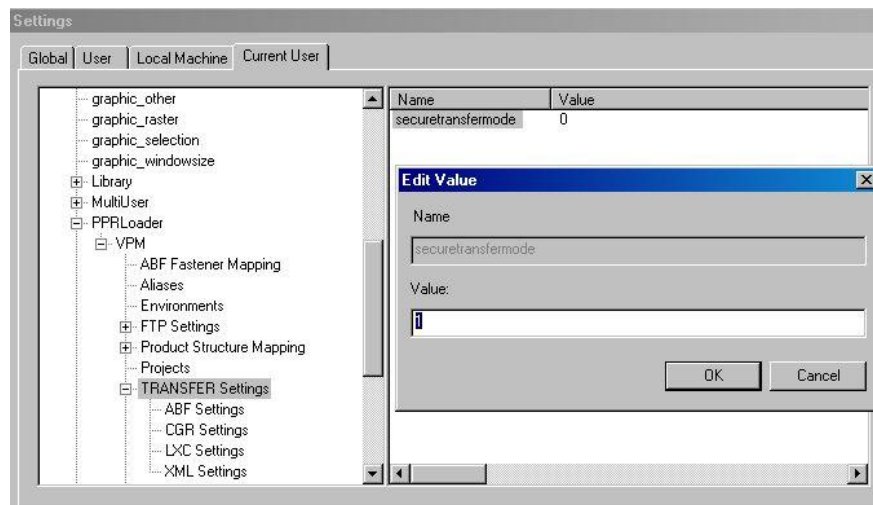


Figure 24: Settings for the Secured Transfer

- 2) xmlhttpsbaseurl: value: HTTPS://sunlcadeg/<shared xml repository>

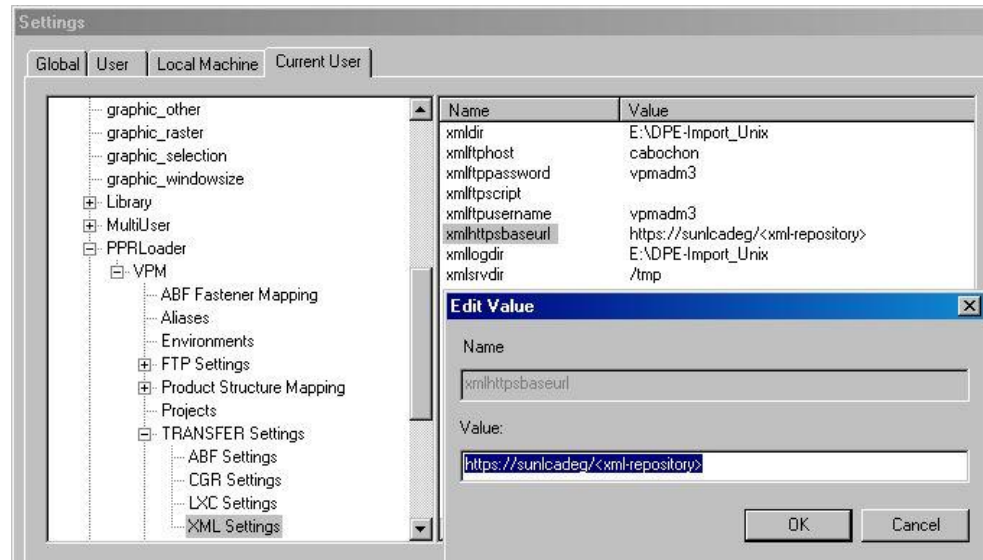
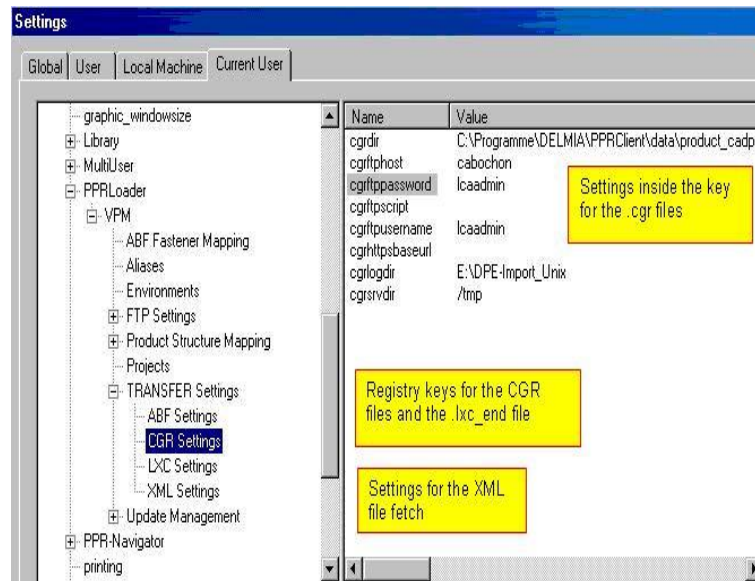


Figure 25: xmlhttpsbaseurl: Value



**Figure 26: Registry Keys**

The registry keys in detail:

- xml/cgr/lxcdir: the directory where the fetched file(s) is/are finally stored
- xml/cgr/lxcftphost: the ENOVIA V4 server host to connect through HTTPS or FTP
- xml/cgr/lxcftppassword and xml/cgr/lxcftpusername: the FTP account information to be used to login to the ENOVIA V4 server
- xml/cgr/lxclogdir: the log directory where temporary and log files are created
- xml/cgr/lxcsvdir: xml/cgr/lxc files are stored in this directory on Unix Server side
- lxcftpwait: the number of tries for the FTP fetch of the .lxc\_end file before passing by. A value "infinite" indicates that PPRLoader tries forever.
- lxcftpdelay: contains the time span in seconds between two tries for the .lxc\_end file

The registry keys lxcftpdelay and lxcftpwait do not have any counterpart in the registry key for the XML file and the CGR files.

As soon as the lxc\_end file is created on the ENOVIA VPM V4 server side, the cgr files can be transferred to the DELMIA Process Engineer side. I.e. these settings permit to control indirectly the cgr transfer.

As in the previous version, the registry keys and settings regarding the .lxc\_end file are a little bit misleading, since they are still named lxc. The .lxc file itself, as well as the .lxc\_log file from the DMU Utilities are not used by PPRLoader, a fetch of those files via FTP is not required.



### Note

*XMLDIR, XMLLOGDIR, LXCLOGDIR, LXCDIR must be equal.*

*CGRDIR must be equal to the Global/graphic\_editor/cadpath value in order to visualize the CGRs in DPE, e.g. PPRClient\data\product\_cadpath*



### 3.8.12 Encryption and Decryption of Registry Settings

The security features since R15 also include passwords set by the Engineering Hub to the Manufacturing Hub connection. To match customers' security requirements, all the password registry settings must be stored encrypted in registry and the transfer of the passwords has to be done encrypted.

- The user has to enter the registry settings for the passwords which have to be secure. To be able to enter them the administrator has to use the administration tool "SimpleCryptAdmin" available for this action.  
Handling the password to get access to the Manufacturing Hub Server: Doesn't matter if there is an update from an earlier release to release 15 or if the release is newly created. At first the administrator has to update/enter the corresponding registry key value. If it is a release update, the password is not encrypted with the encryption algorithm valid from release 15 on. If it is new, the password has to be entered anyway.

```
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader]
```

- 1) Change the value of the entry "password" so that the password can be read in readable format.

➤ Now the administration tool

```
"SimpleCryptAdmin" -ek [0|1] RegistryKey  
[AES|3DES|DES|RC4|RC2]
```

can be used to encrypt the value with the new encrypting algorithm.



#### Note

*"pprloader -login user/password" is NOT supported any more!*

- In case of a release update to R15 and higher, the passwords used before have to be encrypted in the following manner. Otherwise this part is handled by the given registry file newpprloader.reg:

For the **migration** to release 15 and higher the registry key "FTP settings" has to be replaced by "TRANSFER Settings" in registry directly (see at picture below where this is already replaced).

The same action, i.e. replace "ftp" by "transfer" within the strings of the registry entries, has to be done for all strings under registry key

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\  
TRANSFER Settings\CGR Settings]
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\  
TRANSFER Settings\LXC Settings]
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\  
TRANSFER Settings\XML Settings]
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\  
TRANSFER Settings\ABF Settings]
```

In case of a new installation of DPE R16 use the administration tool

```
"SimpleCryptAdmin" -ek [0|1] RegistryKey  
[AES|3DES|DES|RC4|RC2]
```

to encrypt the value with the new encrypting algorithm four times:

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\  
TRANSFER Settings\CGR Settings]
```

```
"cgrftppassword"="<Password>"
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\LXC Settings]
```

```
"lxcftppassword"="<Password>"
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\xML Settings]
```

```
"xmlftppassword"="<Password>"
```

```
[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\ABF Settings]
```

```
"abfftppassword"="<Password>"
```

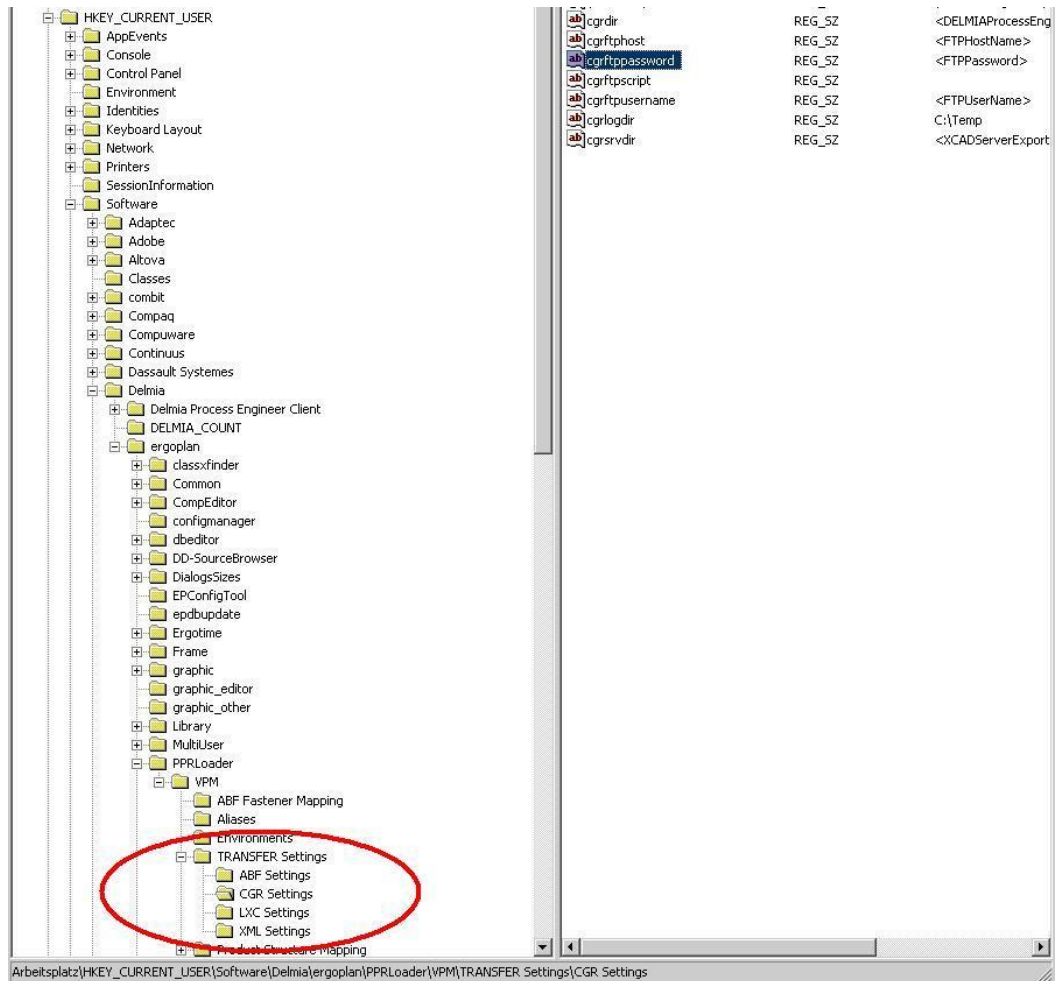


Figure 27: Transfer Settings

### Generally

To decrypt the secure registry settings for the designated registry keys the entered key values have to be handled by a component called Session Data Manager. This is a uniform module handling decrypting of the key values in a uniform way. Once the key values are encrypted, they are written to the registry where they can be seen only in the encrypted format. Afterwards if the key values are needed by the program, the encrypted key values are taken from the registry and decrypted by the Session Data Manager module. So they can be treated as usual. Beside the decryption the Session Data Manager module also treats the write/read actions to the registry. Thus all requests done to the registry are handled by the Session Data Manager module. The transfer of the password to get access to the Manufacturing Hub Server will also be done in an encrypted form, what means that a hash function is used to

encrypt this password before the transportation to the server. On server side the originally entered password located in the data base of the server was also encrypted by this hash function. Thus the encrypted passwords can be compared.

The transfer of the passwords used to get information via FTP/HTTPS is encrypted within the HTTPS protocol. The web server containing the data which is the target of the HTTPS protocol uses its own algorithm to encrypt the password after unpacking it from HTTPS protocol.



### Note

*Encrypted data (e.g. passwords) are valid only on the machine where they are created. They cannot be decrypted on a different machine*

### Example

Usage

"SimpleCryptAdmin" -ek [0|1] RegistryKey [AES|3DES|DES|RC4|RC2]

-ek:            Encrypt the value of the given registry key using the given encryption algorithm

0:    machine key container would be used for encryption

1:    user key container would be used for encryption

RegistryKey:    Name of the registry key

Remark: Choose "0" and "3DES", the RegistryKey has to be typed in without brackets and string name included.

### Example

```
SimpleCryptAdmin -ek 0
```

```
HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\xmltransferpassword 3DES
```

or alternatively:

```
SimpleCryptAdmin -ekv 0
```

```
HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\xmltransferpassword <klartext password> 3DES
```

### Test Verification

- The password found in registry differs from the clear text password. Thus the encryption was done.
- Check if the data transfer via the HTTPS connection was successful. Thus decryption was done and workflow as usual.

## 3.8.13 Configuration

To support properly configuration data from ENOVIA VPM V4, use Standard Code Rules for the desired project. For a detailed overview on how configuration data is mapped to DELMIA Process Engineer, *please refer to the [Appendix A](#)* for details.

PPRLoader supports two different ways to store configuration data, either:

- On the links
- On the instances themselves (this ensures the support for drag and drop to an MBoM and product–process configuration inheritance mechanism).

In ENOVIA VPM V4 all configuration data is related to the link, but the Connection can be customized to handle it a different way.

Basically, links (=Manufacturing Hub SubCompltems) and objects (=Manufacturing Hub ErgoCompBase, e.g. ErgoCompProductDefault) contain four configuration attributes:

- `effectivity.begin` // start date effectivity '31-01-2003'
- `effectivity.end` // end date effectivity '31-12-2003'
- `tailnumber` // range effectivity e.g. "1-5", "10-20"
- `coderulestring` // options, e.g. DIESEL+SUNROOF/-4DOOR

Define the way to store configuration data for each type of configuration individually, using the following registry keys:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgoption"="..."
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgdate"="..."
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgrange"="..."
```

A value "link" keeps the configuration data on the link/SubCompltem. It is stored in the attributes of the link (by default, this should be seen in the Property Page "Usage Data", sometimes in "Entry-Effectivity"). In contrast, a setting of "object" stores the configuration data on the object/ErgoCompBase (this should be seen in the Property Page "Effectivity" of the object). To support both, please use the value "link+object", which stores the configuration data on both the link and the object. When nothing is set, the default behavior is to store the configuration data on both ("link+object").

When nothing is specified (empty key), the default behavior is "link".

The behavior can be set separately for the configuration data types. The valid values for those registry keys are shown in the following example:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgoption"="link"
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgdate"="link"
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgrange"="object"
```

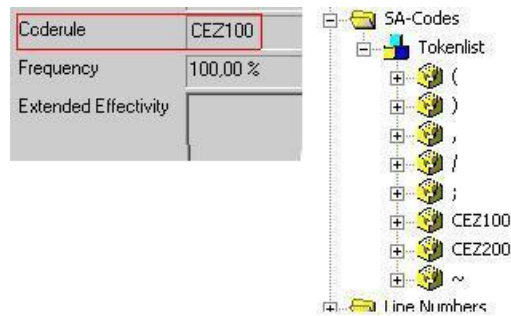
In this case, options (i.e., code rules) are stored on links (SubCompltems) and on instances (ErgoCompBase) of the product structure (this is the default). Date effectivities are stored on the links. Ranges (i.e. tailnumbers) are stored only on the objects themselves.

### 3.8.13.1 Code Rule Modes

The following overview describes the various combinations of the project's coderule mode and the used configuration attributes and their impacts.

#### Project Coderule Mode "Standard", using regular configuration attributes

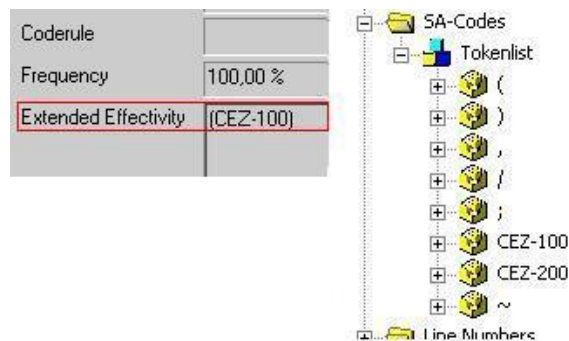
(cfgnormal="1", cfgextended="0", similar default for ENOVIA VPM V4 data)



**Figure 28: Tokens are generated, but special characters are removed**

**Project Coderule Mode „Extended“, using extended effectivity attributes**

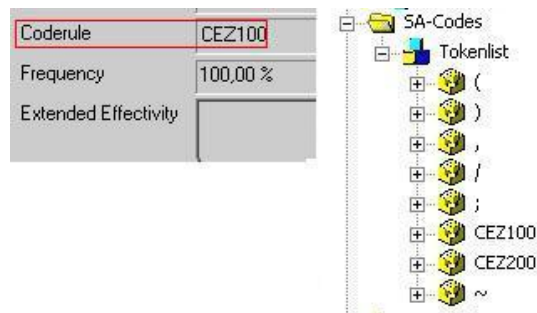
(cfgnormal="0", cfgextended="1", default for ENOVIA VPM V5 data)



**Figure 29: Tokens are generated, special characters are kept**

**Project Coderule-Mode „Extended“, using regular configuration attributes**

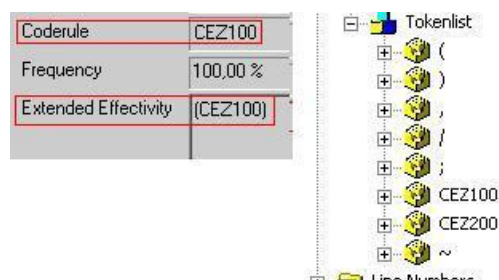
(cfgnormal="1", cfgextended="0")



**Figure 30: Tokens are generated, but special characters are removed**

**Project Coderule-Mode "Standard", using both, extended effectivity and regular configuration attributes**

(cfgnormal="1", cfgextended="1")



**Figure 31: Tokens are generated, but special characters are removed**

I.e. the setting “cfgnormal” rules over “cfgextended”.

### Project Coderule-Mode "Standard", using extended effectivity attributes

(cfgnormal="0", cfgextended="1")

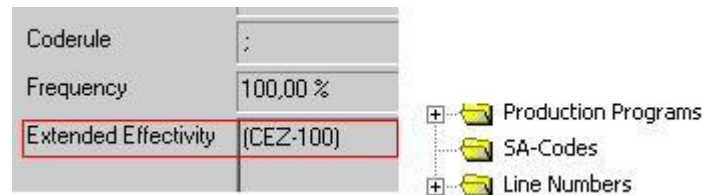


Figure 32: Tokens are not generated

## 3.8.14 Versions

The current ENOVIA VPM V4 to DELMIA Manufacturing Hub Connection allows only maintaining a single snapshot of the VPM product structure in the Manufacturing Hub of DELMIA Process Engineer. By storing the “historical instance id” of an object, the Connection tracks all instances and updates them. No previous states of objects, respectively no versions are stored. All modifications from VPM V4 are directly reflected in the product structure image in Manufacturing Hub.

Since the management of versions is a key feature in VPM V4, it is necessary for the Connection to synchronize and maintain the version related information, transferred from VPM V4.

The support for VPM V4 versions ensures that, if a new version/revision is created in VPM, it is reflected by a new data object in Manufacturing Hub, leaving the previous one unchanged. The new version of the Hub object is used in the current image, but still, the application engineer is able to access previous states of the project and see the corresponding versions.

The following image illustrates the mapping of the versions between ENOVIA VPM and the Manufacturing Hub.

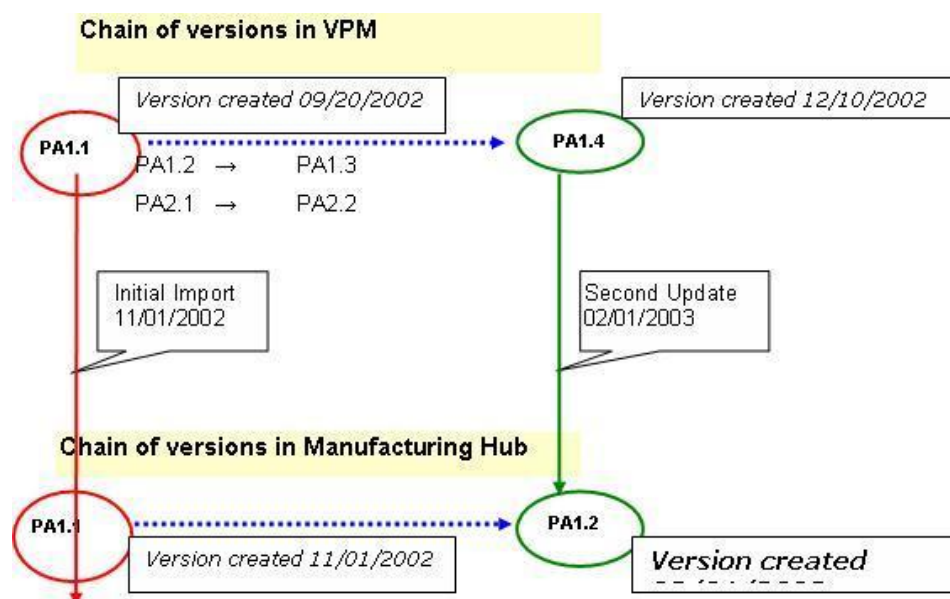


Figure 33: Mapping of the Versions



All versions created and used in the time frame between the two data extractions from VPM to Manufacturing Hub will not appear in Manufacturing Hub. In addition, the version number in Manufacturing Hub will not be the same as in VPM, since the additional object version in Manufacturing Hub is a new version of a Manufacturing Hub object, and not an object mapped from VPM.

This mechanism is not a one-to-one mapping of versions from VPM to Manufacturing Hub. All versions created and used during the time window between two updates/synchronizations from VPM are not reflected by any object in Manufacturing Hub (in the example above, there are no representations of the versions 1.2, 1.3, 2.1 and 2.2 from VPM in Manufacturing Hub).

### Scenarios

- A part in VPM is modified, but no new version is created. This scenario leaves the COID of the part unchanged. After transferring the modified product structure from VPM V4 to Manufacturing Hub, the corresponding part in Manufacturing Hub is updated.
- A new part version is created in VPM V4. If you use the “replace version” or “replace from clipboard” function in VPM V4, a new COID for the new object is created. By checking the “Historical Instance ID” of the part versions, the VPM V4 Engineering Hub to Manufacturing Hub Connection can recognize that the instance of the object is actually the same, but from checking the COID, the Connection can create a new version of the object in Manufacturing Hub. (Update Status “version”).
- Please note that this is not valid for further versions on VPM side. If with further versioning a new link is created on VPM V4 side, the Connection does not recognize the new object as a new version of the object in DELMIA DPE: with the update import the old version is being deleted (updatestate “delete”) and a new object is being imported to DPE side (updatestate “new”). How new versions on VPM V4 side are recognized by the Connection depends also on the exact handling of the objects, e.g. if only the parent part is versioned or both, the parent part and its children.
- In DELMIA Process Engineer the new version does not automatically replace the old version in the EBOM or MBOM views. This has to be done manually by calling the “versions” dialog in the contextual menu of the object.

Please note that the support for VPM V4 versioning requires that the corresponding plantypes have versions enabled:

<input type="checkbox"/> <b>Flags</b>	
Has versions	Yes
Is searchable	No

**Figure 34: Has Versions**

If versioning is disabled for a specific type, the PPRLoader application will dump a critical error (COM Error 80040350): CheckoutNewVersion failed.

## 3.8.15 Multiple Part Filter on Import

With R18 it is possible now to apply import filters during the update/import of the product structure with the ENOVIA VPM V4 Engineering Hub to Manufacturing Hub Connection. This allows different imports (e.g. for different

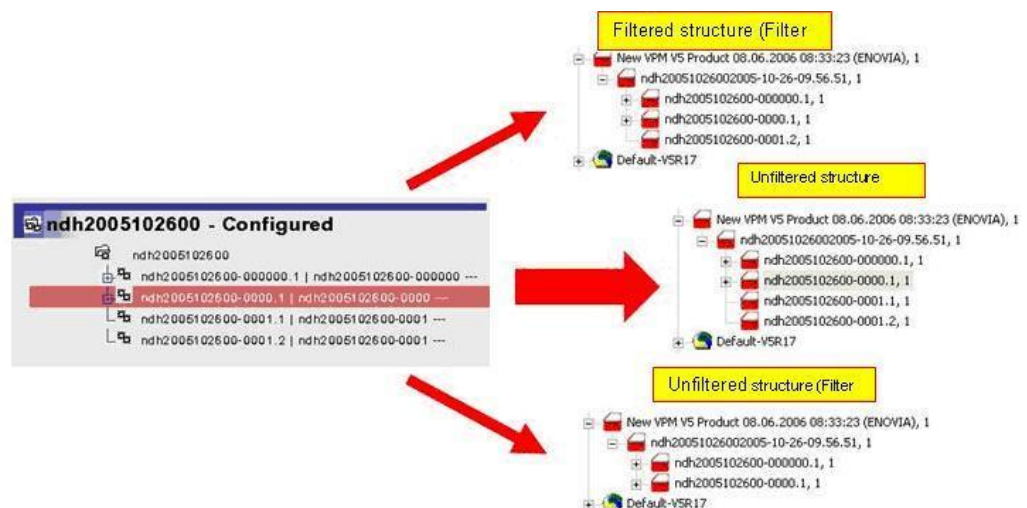
Manufacturing Hub Projects or different partners) by using just one XML export file.

The import filtering uses a filter configuration file, which contains the filter criteria (based upon attribute values) and probably the Manufacturing Hub project references, in case different projects use different filter settings.

During the update/import run of the ENOVIA V4 Engineering Hub to Manufacturing Hub Connection the XML export files to be imported are read as regular. Still, during the parsing of the XML export files, each component read is checked against the corresponding list of filters; components filtered out are marked.

In the second step, the traversal (resp. the initial load) of the product structure such objects filtered out are treated as non-existent. The blockage of one Part by a filter causes its entire sub-structure to be suppressed, even though portions of it might have passed the filter.

Filtering in the context of this document means filter out the data of a specific Part during update/import completely, i.e. this Part is considered as NOT existent in the XML export file.



**Figure 35: Filtering Data**

The capability of import side filtering allows having different product structures (in different Manufacturing Hub projects) for different purposes (such as design, process planning or maintenance and services) with one XML export on the ENOVIA V4 Engineering Hub side. This reduces the number of exports to be performed on the client side, since one XML export file can be re-used to support several Manufacturing Hub projects (or partners) at once.

Filters are defined in a filter configuration file through filter criteria based upon attribute values, either on Part references (reference XML file), Part instances (instance XML file) or Models and Documents (reference XML file).

Filter criteria can either be defined as pass filters (white list) or blocking filters (black list) on attribute values. It is possible to use wildcards (in a limited way). It is also possible to combine several filters.

Additional conditions can be defined on a filter to check only Parts from specific Environments or specific types.

If no filter configuration file is specified or if it is empty or contains no filter criteria, all components are updated/imported. No filter check is performed, this guarantying full compatibility to the previous behavior.

The filter configuration file is registered in the registry, similar to the current attribute mapping configuration file.

Still, in difference to the attribute mapping configuration file, the filter configuration file is XML formatted and is capable of containing different filters for different projects.

A filter configuration template file "import-config.xml" is provided with the regular DPE installation (in ~\PPRClient\data\PPRLoader directory).

During the parsing of the ENOVIA V4 XML export files (reference and instance) each component (incl. Models and Documents) is checked against the list of filters.

In case of components, the entire sub-structure beneath a component blocked-out is suppressed, even though, child components might have passed all filters.

A component which matches a block filter (blacklist) criteria is considered as "out". In case of a pass filter, the component is filtered out, if it does NOT matching the filter criteria.

If a Part, Model or Document is filtered out from the XML reference file, all corresponding instances/MH components are affected. If a Part instance from the XML instance file is filtered out, only this particular component gets blocked. A Part instance is blocked, if the corresponding Part reference has been filtered out, even though the Part instance data itself indicated the opposite.

It's not possible to reset the status of an already filtered "out" component back to "in", i.e. all filters are additive (AND concatenated).

The filter configuration file XML schema describes filters and Projects, which use those filters during update/import.

Components filtered out are logged in the PPRLoader.log file, if tracing is enabled.

### 3.8.15.1 Customization

The filter config file is registered in the registry, similar to the current attribute mapping configuration file.

```
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM]  
"filterfile"="..."
```

A regular filter configuration file is shown in the next image.

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <configuration>
- <global>
  - <properties>
    <pattern wildchar="*" escapechar="\\" />
  </properties>
</global>
- <filters>
  <!-- pass filter for part instances, only status "Approved" and "Released" pass through -->
  - <extendedfilter id="filterA" context="instance" environment="" table="" attribute="V_status" ignorecase="no"
    filtermissing="no">
    - <whitelist>
      <token>Approved</token>
      <token>Released</token>
    </whitelist>
  </extendedfilter>
  <!-- block filter for part instances, admin part instances are filtered out -->
  - <extendedfilter id="filterB" context="instance" environment="" table="" attribute="V_organization"
    ignorecase="yes" filtermissing="no">
    - <blacklist>
      <token>admin</token>
    </blacklist>
  </extendedfilter>
  <!-- pass filter for part references, customizable type "engineering" is allowed -->
  - <extendedfilter id="filterC" context="reference" environment="" table="PART_LIST" attribute="V_type"
    ignorecase="no" filtermissing="no">
    - <whitelist>
      <token>Engineering</token>
      <token>Design</token>
    </whitelist>
  </extendedfilter>
  <!-- block filter for part references, "design" projects are filtered out -->
  - <extendedfilter id="filterD" context="reference" environment="" table="PART_LIST" attribute="V_project"
    ignorecase="yes" filtermissing="no">
    - <blacklist>
      <token>design</token>
    </blacklist>
  </extendedfilter>
  <!-- combined block and pass filter, all states ending with "ed" are allowed (e.g.
    "Approved", "Designed", "Planned"), except "Released" -->
  - <extendedfilter id="filterE" context="reference" environment="" table="" attribute="V_status"
    ignorecase="no" filtermissing="no">
    - <blacklist>
      <token>Released</token>
    </blacklist>
    - <whitelist>
      <token>*ed</token>
    </whitelist>
  </extendedfilter>
</filters>
- <projects>
  - <project id="GPL520A">
    - <filterlist>
      <filter>filterA</filter>
      <filter>filterC</filter>
    </filterlist>
  </project>
  - <project id="GPL520B">
    - <filterlist>
      <filter>filterB</filter>
      <filter>filterD</filter>
    </filterlist>
  </project>
  - <project id="GPL520C">
    - <filterlist>
      <filter>filterE</filter>
    </filterlist>
  </project>
</projects>
</configuration>

```

At the beginning the global properties define wildcard characters and the escape sequence, which can be used for the filter criteria.

```
- <global>
- <properties>
  <pattern wildchar="*" escapechar="\\" />
</properties>
</global>
```

In the next section filters are defined as a set of extended filters. Each extended filter definition contains several attributes. The “id” attribute is the filters name, which is used later on to apply several filters to different projects.

```
- <extendedfilter id="filterA" context="instance" environment="" table="" attribute="V_status" ignorecase="no"
  filtermissing="no">
- <whitelist>
```

The “context” attribute describes the objects source, either “reference” or “instance”.

```
- <extendedfilter id="filterA" context="instance" environment="" table="" attribute="V_status" ignorecase="no"
  filtermissing="no">
- <whitelist>
```

If context is set to “reference” this filter is only applied to the reference file, if the context is “instance” it’s only valid for the instance file. If context is left empty or not specified (or contains any other invalid character string), the filter is applied to both XML export files.

The next three attributes “environment” and “table” can be used to limit each filter to a specific set of objects. They directly refer to the corresponding entries in the ENOVIA V4 XML reference export file.

```
<!-- pass filter for part references, customizable type "engineering" is allowed -->
- <extendedfilter id="filterC" context="reference" environment="" table="PART_LIST" attribute="V_type"
  ignorecase="no" filtermissing="no">
- <whitelist>
```

The following image shows a Part from Environment “VPMENV1”, with table “Parts”.

```
</Instance>
- <Instance Name="(VPMENV1.Parts) Pa 1 part config --- 10% VPMADM ADMIN" Type="CATPDMObject"
  Uuid="01a64b0cd38929cd276507c16c59df04" ReferenceRef="4813c22b5f3d1196f56daab099157edb" Alias
  config --- 10% VPMADM ADMIN">
+ <Attribute Name="PART_LIST.$C0ID">
+ <Attribute Name="PART_LIST.$COMPID">
+ <Attribute Name="PART_LIST.S_PART_NUMBER">
```

These three attributes are optional. Since there’s no Environment, table and type information within the instance XML file, these attributes are only useful on “reference” context.

The attribute “attribute” itself defines which (ENOVIA) attribute is to be checked against the filter criteria.

```
- <extendedfilter id="filterE" context="reference" environment="" table="" attribute="V_status"
  ignorecase="no" filtermissing="no">
- <blacklist>
```

The attributes “ignorecase” and “filtermissing” contain information about case sensitivity (“yes”: no case sensitivity, “no”: lower/upper case is checked) and the behavior, if the filter attribute is not existent at all on a component (“yes”: component is filtered out, if filter criteria attribute is missing, “no”: component may pass, if specified attribute is not found).

```
- <extendedfilter id="filterE" context="reference" environment="" table="" type="" attribute="V_status"
  ignorecase="no" filtermissing="no">
- <blacklist>
```

The next list in the filter configuration file is the list of tokens, separated into “blacklist” (tokens defining block criteria) and “whitelist” (tokens defining pass criteria).

```

ignorecase= no intermissing= no
- <blacklist>
  <token>Released</token>
</blacklist>
- <whitelist>
  <token>*ed</token>
</whitelist>
</extendedfilter>
</filters>

```

In the above example, the checked component can only pass, if the specified attribute has a value ending with “ed”, except “Released”, which defined the component as “out”. So, values as “WIP” and “Integrate” would cause the component to be blocked (string do not end on “ed”), allowed values would be “Approved”, “Planned”, .etc. e.g.).

“Blacklist” and “whitelist” should only occur once for an extended filter, but can contain several token entries:

```

- <whitelist>
  <token>Approved</token>
  <token>Released</token>
</whitelist>

```

The conversion to Manufacturing Hub attribute types from the XML string values is not affected by the filtering, i.e. the filtering will always check and compare string values. With that, it’s e.g. not possible to define range filters on double values.

Errors in the definition of the extended filters like an empty white list will be ignored that is the filter will not be applied. If it is impossible to parse a filter definition (i.e. unknown context, bad XML token, etc.) the importing PPRLoader application will abort the execution.

The “projects” section of the filter config XML file defines the filters, the specific projects use for update/import. Each “project” contains again an “id” attribute, which corresponds to the projects “nameshort” (“Number”) attribute in MH.

```

- <projects>
- <project id="GPL520A">
  - <filterlist>
    <filter>filterA</filter>
    <filter>filterC</filter>
  </filterlist>
</project>

```

The filter list contains the ids of the extended filters, which should be applied to the given Project. If the list contains ids, which do not correspond to one of the extended filters, the entry is ignored.

```

- <projects>
- <project id="GPL520A">
  - <filterlist>
    <filter>filterA</filter>
    <filter>filterC</filter>
  </filterlist>
</project>

```

### 3.8.15.2 Limitations

- The maintenance of the XML based new filter configuration file is up to the user/customer. Only a template is provided with the regular DPE installation.
- A Part or Document that has already been transferred to the Manufacturing Hub via the EH-MH Connection, but is blocked by one of the filter criteria in a later on update/import run (since either the filter itself has changed or a modified attribute value makes the filter suppress this part/document now)



is treated as a regular deleted object. The concrete action upon this delete depends on the given delete mode.

- Filters can only be defined on Parts (Part references and instances), Models and Documents (and the corresponding links). It is not possible to define filters for any other objects resp. content of the data transfer such as Calculation models.
- Filters are additive, i.e. all filters are concatenated by a logical AND. If one filter blocks a particular Part, the match result for all other filter criteria is irrelevant. This is important in case one filter blocks out a Part reference, while a pass filter explicitly allows one of its instances to be transferred. Still, as a result, this instance is skipped during update/import, since it needs to pass ALL filters.
- This also means that the order in which the filter criteria are defined in the filter configuration file is of no interest for the filter result at all.
- Filters are based upon VPM naming. I.e. the attribute names used to define the filters are the VPM database ones, not the mapped attributes from Manufacturing Hub. In difference to the similar functionality for the ENOVIA VPM V5 EH-MH Connection, filters on Manufacturing Hub attributes are not supported. Therefore, the client should not define filters with context "internal".
- The conversion to Manufacturing Hub attribute types from the XML string values is not affected by the filtering. Filtering is always based on string value comparison, i.e. it's e.g. not possible to define range filters on double values (e.g. component is filtered when double attribute is within range -2.8 to +4.2).
- Since VPM V4 does not support fine Grained mapping, the "finegrained-type" attribute from ENOVIA VPM V5 EH-MH Connection feature "Multiple Part Filtering on Import" is not required in the filter configuration file nor evaluated in any way.

## 3.9 Step-by-Step Setup

This section contains a small step-by-step walkthrough to setup and customize the PPRLoader settings in order to support the new Engineering Hub to Manufacturing Hub Connection functionalities.

### Registry Templates

Insert the "newpprloader.reg" registry template file into the registry

### General Settings

Adapt and modify the general Engineering Hub to Manufacturing Hub Connection settings. These include user name, password, backbone sender/receiver, etc.

### Transfer Settings

Adapt and modify the Transfer Settings, which allow the automatic fetch of the XML export file before and the CGR files after importing the ENOVIA VPM V4 Product Structure. The name of those settings should be self-explanatory.

### Product Structure Mapping

Adapt and modify the Product Structure Mappings according to your actual ENOVIA VPM V4 scenario. This includes entering/changing the name of the

desired Plantypes, corresponding to your DPE Plantype Set (*Please refer to the [Multi-Environment](#)*).

### Configuration Behavior

Enter the “cfgoption”, “cfgdate” and “cfgrange” settings to support the required configuration behavior (*Please refer to the [Configuration](#)*

### Extended Attribute Mapping

To support extended attribute mapping (*Please refer to the [Customized/Extended Attribute Mapping](#)*), create a PPRLoader config file and register it in the key:

```
[HKCU\Software\Delmia\ergoplan\PPRLoader\VPM] "cfgfile"="..  
."
```

## 4. Launching Applications

The 3d COM servers (Orbix and APACHE) and the XCAD server are running on the UNIX machine. If they are already running, you don't need to launch them again when you start a test session. The PPRLoader server is running on the NT machine. You can launch it when you log on and use it for all your test sessions.

### 4.1 Interactive Mode

#### 4.1.1 3d COM Server

After a reboot, launch 3d COM server (APACHE + Orbix) server on the UNIX machine. *Please refer to the [pprloader -vpm <data file> \(xml file\)](#) to know if it is needed the re-launch these processes.*

- Orbix - Use the command:

```
/DassaultSystemes/R17/aix_a/code/command/runOrbix
```

- APACHE - Use the command:

```
/apache/bin/apachectl start
```

#### 4.1.2 XCAD Server

Launch the XCAD server on the UNIX machine. Use the documentation **Customizing-MultiCAD.pdf** to do so. The command is, under the VPM environment:

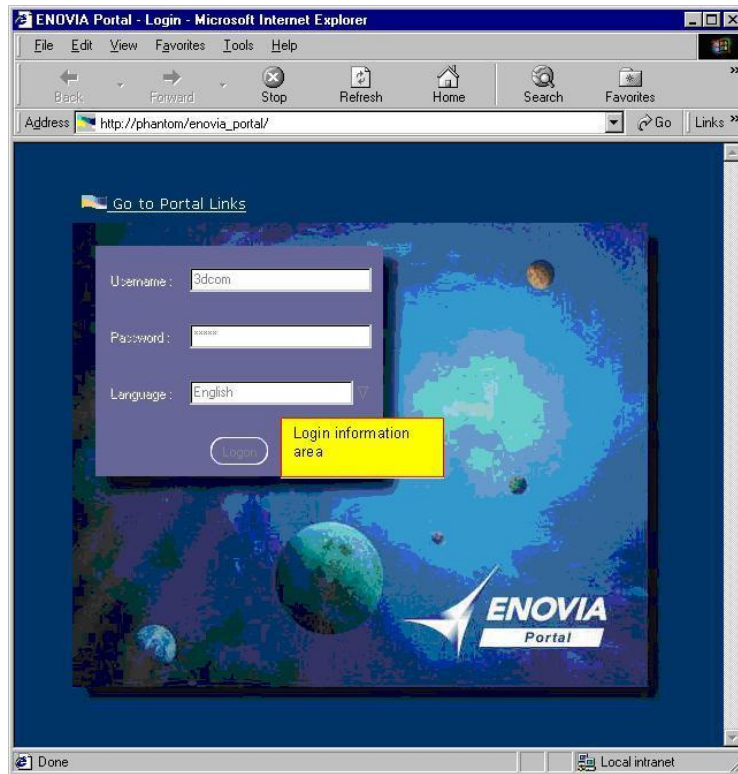
```
VPMStart.sh xcad
```

If a XCAD server is already running on the machine, then the command will fail, which is OK.

If no CGR's are created on the UNIX machine when using the command 'Manufacturing Hub...' of 3d COM, then you should check if the XCAD server is running.

#### 4.1.3 ENOVIA VPM V4 Portal/Web browser

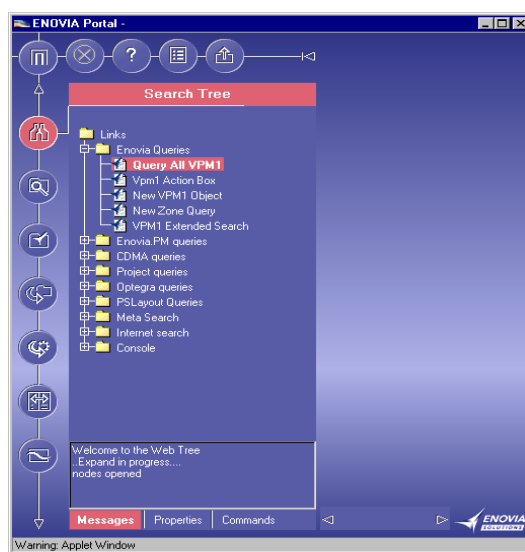
- 1) Launch your web browser (IE 5 or higher) and connect to 3d COM (you should have the 3d COM address to use during the install of 3d COM). If you cannot access to the login panel, then probably the APACHE server is not running. The login screen below appears:



**Figure 36: 3d COM Login panel**

- 2) If in the 3d COM login panel, the login information area is not visible, then probably your Orbix server is not running.
- 3) Define the user and password and click on the Login button.
- 4) After some time few seconds (it can be long – 1 or 2 minutes), the 3d COM window should be displayed.

If it is not the case, you probably have a problem with your settings. In that case, log into the UNIX machine under the user you used to log on 3d COM and remove everything from the directory `$HOME/CATSettings/Server`.



**Figure 37: 3d COM Window**

- 4) Then connect to VPM from 3d COM (using the *Enovia Queries | Query All VPM1* or an already created shortcut).

To be able to connect VPM from 3d COM, a session **must** be open on the UNIX server with the display open to others (use the `xhost +` command on the UNIX server to do so), since VPM will open a panel on this UNIX server.

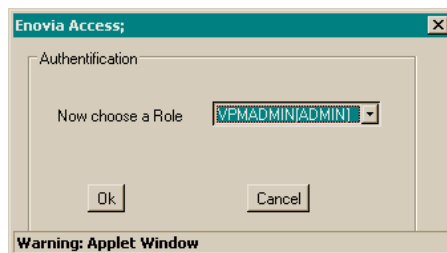
- 5) After few seconds (maximum a minute) a panel is displayed to allow you to select the VPM user & password you will use if you are in Server Authentication Mode.



**Figure 38: Authentication Mode**

(Only if your VPM is in Server Authentication Mode)

- 6) After few seconds (maximum a minute) a panel is displayed to allow you to select the VPM role you will use:



**Figure 39: Enovia Access**

#### 4.1.4 PPRDaemon

Launch the `PPRDaemon.exe`. Check in the Task Manager that it has been started correctly. For testing purposes, you can start `PPRLoader` (`PPRLoader.exe`). When it is running, the `PPRLoader` should answer with:

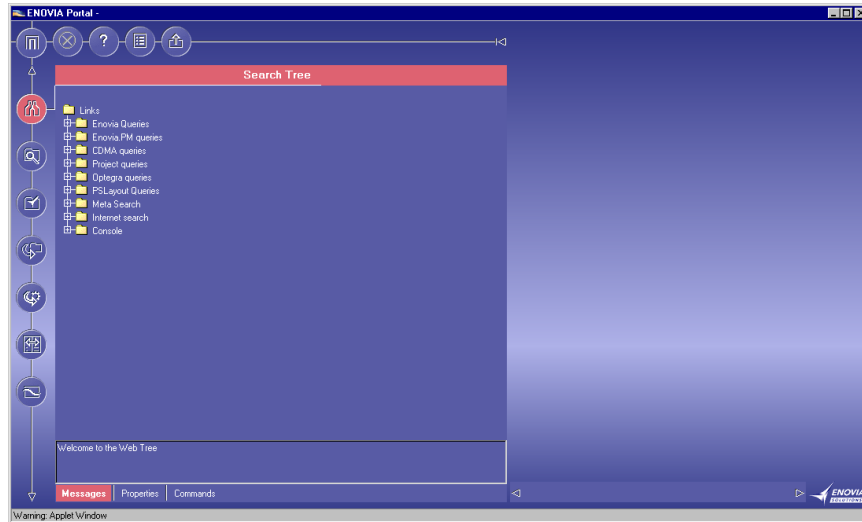
```
Starting CATBackBone listener service
Waiting for messages...
```

Also, the supporting applications `PPRDaemon` and `httpsclient` are now available as 64-bit applications `PPRDaemon64.exe` and `httpsclient64.exe` in the `...\PPRLoader\program\bin64` directory. `PPRDaemon64.exe` invokes the `VPMLoader64.exe` (for interactive product transfer), while the `httpsclient64.exe` is started by the `VPMLoader64.exe`, all those steps similar to the situation on a 32-bit platform

#### 4.1.5 Starting Transfer

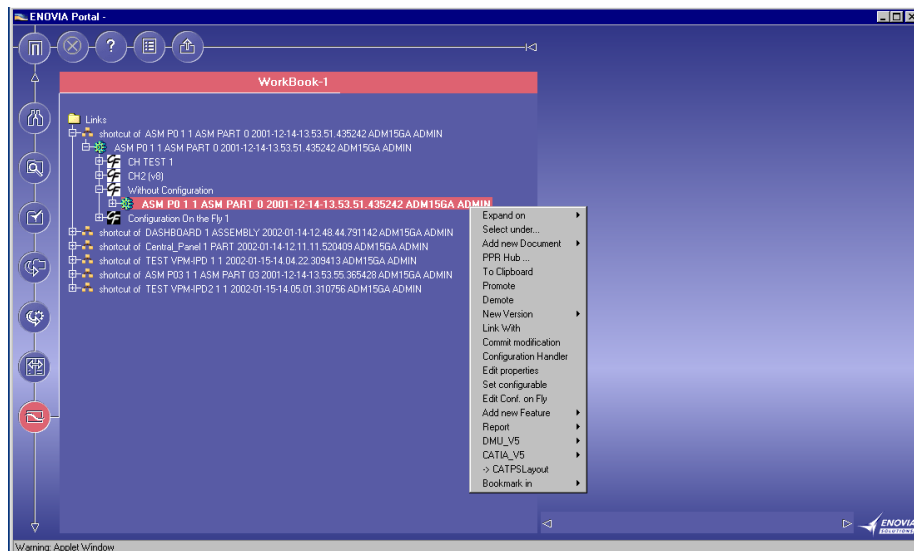
Here is an example of transfer.

- 1) Launch 3d COM and connect to VPM to display a VPM Part. Note that you must transfer a part (and its sub-tree). You cannot transfer a VPM model or a VPM document directly. You have to use its Part to do so.



**Figure 40: Launch 3d COM and Connect to VPM**

- 2) Open the tree that has to be transferred.
- 3) Open the sub tree called 'Without Configuration'.  
At this time the PPR Daemon should be running on the NT machine.
- 4) Select the 1<sup>st</sup> VPM Part (the root part of the structure that will be transferred) of this sub tree and use on it the 'Manufacturing Hub...' command.



**Figure 41: Select the 1<sup>st</sup> VPM Part**

- 5) The list of E5 projects will then be displayed. Please select the project to import the data to and click on the 'Associate' button to start the transfer.

### Note

*The system might ask for the E5 user name and Password.*





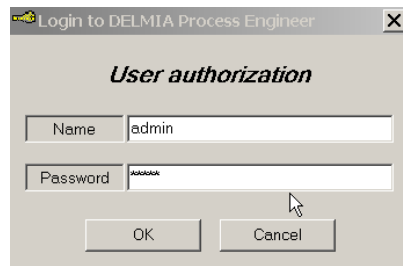


Figure 42: User Authorization



Figure 43: Command Shell Window of PPRLoader

- 6) A command shell window of PPRLoader appears and displays the import messages and output from the data transfer. For fetching the CGR files through FTP, a second command shell window will popup. Once the import is completed, the windows are closed automatically.

## 4.2 Batch Mode

It is possible to exchange product (and resource) structures between VPM & Manufacturing Hub fully in batch mode, without using any 3DCOM.

### 4.2.1 Start PPRLoader in Batch Mode

#### 4.2.1.1 Command Line Application

- 1) In the `...\program\bin` directory of your DELMIA Process Engineer® Client installation you will find the `PPRLoader.exe` application. Go to this path.

#### Example:

`D:\DELMIA\PPRClient\program\bin.`

- 2) Now start the PPR Loader with the prompt `pprloader` and a start parameter. You can find out which parameters are available by entering `-?` or `-help` as a parameter. The complete entry looks as follows:

`D:\DELMIA\PPRClient\program\bin>pprloader -help`

```

D:\DELMIA\PPRClient\program\bin>pprloader -usage
PPRLoader - version 5.13
Copyright © 2001-2003, DELMIA GmbH. All Rights Reserved.
Copyright © 1999, International Business Machines Corporation and others. All Rights Reserved.
Copyright © 1999-2000, The Apache Software Foundation. All rights reserved.
[Feb 09 2004 14:47:58] Usage...
  pprloader
    <config file>                run as CAT backbone listener
    -file [<data file>] [<config file>]  import data as described in config file
    -odbc [<data source>] [<config file>]  import data from data file
    [-sql <statement>] [-where <clause>]  import data from ODBC data source
    -upm <data file>             set SQL query information
    -incremental|partial|complete|reference  import UPM export file
    -abf <data file>             import in incremental/partial/complete/reference mode
    -project [<id ! short name>]  import ABF fastener file
    -object [<id ! short name>]   specify/print project/s
    -pts [<id ! short name>]      specify/print root object/s
    -type <proc|prod|res|wsc|item> specify/print plan type set/s
    -dialog                       specify type of data
                                open dialog
    -log <log file>               specify log file
    -login [<username/password>]  print/set/use login
    -usage                       print this usage info
    -help ! -?                   print help
    -version                     print version
    -info ! -about               display info/about window
    -error [<number>]            print error description

D:\DELMIA\PPRClient\program\bin>_
  
```

Figure 44: Command Shell

#### 4.2.1.2 64-bit PPRLoader

The currently existing PPRLoader/VPMLoader is a 32-bit application. The address space of such an application is technically constrained to a maximum of 4 GB. Depending on the operating system, an even lower value, e.g. a maximum of 2GB virtual address space may be available. To overcome this limitation the importing VPMLoader application (and it's underlying dlls) are migrated to 64-bit technology, thus extending the current memory limitations (2/3GB) tremendously up to 8TB on Windows 64-bit OS.

There is no other change between the 32-bit and the 64-bit PPRLoader rather than the 64-bit executable is named PPRLoader64.exe, i.e. all command line options are the very same and work as before. Same applies to the underlying VPMLoader64.exe (VPMLoader.exe/VPMLoader64.exe is invoked from the PPRLoader.exe/PPRLoader64.exe

- 1) Open the prompt.
  - Click **Start/Programs/Accessories** and then click Prompt. Or
  - Click **Start / Execute...** and enter "cmd" in the dialog that is opening.
  - Confirm the entry.
  - The prompt will open.
- 2) In the ... \program\bin64 directory of your DELMIA Process Engineer® client installation you will find the **PPRLoader64.exe** application.
- 3) Enter this path. Example: D:\DELMIA\PPRClient\program\bin64.

Now start the PPR Loader with the prompt `pprloader64` and a start parameter.

#### 4.2.1.3 pprloader -vpm <data file> (xml file)

Using this type of call of the PPR Loader you can retrieve export files in XML format as they are written by ENOVIA VPM V4 and by ENOVIA VPM V5.

#### 4.2.1.4 pprloader -project and pprloader -object

If these two parameters are used alone, i.e. without any further options, the PPR Loader creates a list of all projects or upper top level objects available, i.e. main nodes (root objects). This is very useful for quickly finding the short

description or ID of a project or of a desired object in order to import into a specific project or to synchronise a structure.

**Example:**

```
pprloader -vpm ...\ ODT_TstPartialDataTransfer_PM5-2003-7-1-11.31.54.xml -project EV5
```

#### 4.2.1.5 pprloader –help or pprloader –?

Using this parameter a list is displayed of all parameters used in the batch mode. .

#### 4.2.1.6 pprloader –info

Using this parameter general information on the PPR Loader and the Support address are displayed ([delmia.de.support@3ds.com](mailto:delmia.de.support@3ds.com)).

#### 4.2.1.7 pprloader ... –incremental

Imports an incremental export file from ENOVIA VPM V4.

**Example:**

```
pprloader -vpm C:\Temp\IncrementalAT0-2.xml -project INC -incremental
```

This option can be used to force incremental mode, in case the automatic detection of an incremental file might fail.

### 4.2.2 AT0EXPND (VPM/UNIX Side)

For exporting VPM data, the official VPM executable AT0EXPND can be used. AT0EXPND allows exporting a complete PSN structure into XML Format file.

To use AT0EXPND exec you have to do the following actions:

#### 4.2.2.1 Init VPM environment

Use the proper shells in `$HOME/env`. For example:

```
YOUR.env
```

```
VPMWsUser.sh
```

#### 4.2.2.2 Launch AT0EXPND

The Mandatory arguments are:

- **caenv** : environment name (example : VPMENV, PRODUIT,...)
- **coid** : COID of the root of the structure to export
- **savemode** : XML in that case

To get the COID, use the sql command:

```
select "$COID" from VPMENV.part_list where S_PART_NUMBER='xxxdfhsdfg' ;
```

VPMENV is the VPM environment name where the Part is located

S\_PART\_NUMBER is the "Part Number" in VPM

### 4.2.2.3 Optional Arguments

- **cgr:** for triggering cgr creation in XCAD server. In that case, the XML and the CGR files will be generated. The XCAD server should have been launched before. (VPMStart.sh xcad)

The main differences concerning the CGR files between Interactive (3dCOM) and Batch mode (AT0EXPND):

- cgr creation is optional
- no lxc & lxc\_end file is generated
- cgr settings inherit from VPM standard XCAD settings. CGR can be generated either inside or outside VPM Vault: AT0EXPND can generate CGR inside or outside VPM Vault, depending on the VPM MULTICAD parameters.
- **file :** output file of the expand
- **user :** vpm P&O mngt
- **pwd :** vpm P&O mngt
- **role :** vpm P&O mngt
- **org:** vpm P&O mngt
- If the user argument is not specified, they will all be read by means of the P&O login file for AT0EXPND.
- If the user & the pwd arguments are specified, the security arguments (Role & Org) do have to be entered.
- The pwd (password) argument is useful in server mode only. It is not used otherwise.
- **-Increment :** Time stamp, since the modifications should be exported for incremental transfer in the format YYYY-MM-DD-HH-MM-SS

### 4.2.2.4 Other Arguments

- **Ch:** Name of the config handler to apply, none by default
- **levels:** Number of levels to expand (-1 by default; negative value = any level)
- **IPDSite:** Name of the IPD Site
- This is only necessary in case of multi IPD instances. With this option the user specifies a name for all the transactions on its IPD instance update through incremental update, so that he will be able to manage multi tracking of changes in the assembly.
- When the user uses the – IPDSite option with a new transaction name as input, a full export will be generated.

For all other optional argument description, you can launch:

- **AT0EXPND -h**

### 4.2.2.5 Using the Shell Script at0expnd.sh

Along with the PPRLoader application, a shell script will be delivered called **at0expnd.sh** (to be used on VPM side) which will make the AT0EXPND User-friendlier.

Example of an at0expnd.sh:

```
AT0EXPND -caenv PRODUIT -coid 41419331C12F54F4 -savemode
XML -file <Directory>

echo 41419331C12F54F4 `ls -atr /tmp/*.xml | head -1` >>
/tmp/at0expnd.tmp

Sort /tmp/at0expnd.tmp > /tmp/at0expnd.log
```

In this example the XML file will be generated in /tmp.

The file at0expnd.log generated in /tmp will contain for each COID the associated XML file.



#### Note

*at0expnd.log is an important file, because it is the file on which Process Engineer will get VPM information about which XML files to import.*

## 4.2.3 Importing Data into Process Engineer (Windows side)

### 4.2.3.1 Batch Parameters via FTP

The vpm2pprbatch directory provided along with the regular PPRLoader script and registry templates contains a list of batches  
(~\DELMIA\PPRClient\data\PPRLoader\):

- **vpm2ppr.bat**: this batch script contains the call of the “vpm2pprbat.vbs” script and

the definition of the arguments passed to it.

In vpm2ppr.bat the following attributes are to be set:

**VPMDIR**: VPM UNIX directory in which the XML file will be localized (must be the same than the one containing the log file):

**DATDIR**: local data directory

**VPMSRV**: VPM Server name

**USR**: FTP username

**PWD**: FTP passwd



#### Note

*vpm2ppr.bat and .vbs should be copied into the DATDIR.*

*DATDIR should be equal to XMLDIR, XMLLOGDIR, LXCLOGDIR, LXCDIR.*

#### Example

```
@set VPMDIR=/tmp
@set
DATDIR=C:\DELMIA\PPRClient\data\PPRLoader\vpm2pprbatch
@set VPMSRV=analepdsy
@set USR=vtest7
```

```
@set PWD=vtest7
```

The VBS script "vpm2pprbat.vbs" itself fetches the log file from VPM UNIX side, scans the entries, tries to identify for each one the related project/root id (from the registry) and generates the appropriate FTP commands and PPRLoader calls in two files to finally fetch and import the XML files.

So, the execution of the VBS script creates 4 files:

- "getvpmlog.ftp": contains the ftp commands to fetch the VPM export log file ("at0expnd.log")
- "getvpmxmls.ftp": contains the ftp commands to fetch the required XML export files
- "pprbatch.bat": batch script with PPRLoader calls (XML file with the related project/root)
- "vpm2pprbat.log": log file

The "pprbatch.bat" batch script contains PPRLoader calls in the following format:

```
pprloader -vpm ./ExportBOMData-...xml -project "$id$..."
```

### Example

```
pprloader -vpm ./ExportBOMData-21-08-2002-16.45.12.xml -project "$id$(0:0-846#0, 168)"
```



### Note

*Instead of the Project ID you can use the shortname of the project.*

Please see the sample files for more details.

## 4.2.3.2 Registry Prerequisites

To be able to import a VPM Product into Process Engineer, it is mandatory to know which DELMIA project is associated with the VPM Product to import, i.e. to have a DELMIA Process Engineer Project vs. VPM Root Part COID mapping.

This information will be set in the register:

### Example

The root product's COID is **41419331C12F54F4** .

It will be associated to the DELMIA Project ID called "**PXS**" (project number/ "shortname")

Windows Registry Editor Version 5.00

```
[HKEY_CURRENT_USER\Software\DELMIA\ergoplan\PPRLoader\VPM\Projects]
```

```
"4141909213FB00D7"="PXT"
```

```
"4141911D2A7D7f96"="FTR"
```

```
"414196E1AEF42C40"="$id$(0:0-24436#0, 168)"
```

```
"41419331C12F54F4"="PXS"
```





### Note

*There might be several products (COIDs) referring the same project. In case you use the project's shortname, instead of the projects id, please ensure it is unique.*

### 4.2.3.3 Auto Commit Settings

On the Manufacturing Hub side, before a data import is done, the registry key "autocommit" can be set to "0", "1" or "N" by the customer. Depending on this registry entry the number of commits to the data base during a data import can be controlled by the customer. So the more commits to the ORACLE data base are performed the faster works the data transfer between Engineering Hub and Manufacturing Hub. But in the same time more memory on the server side is allocated. If the number of the commits is reduced, the performance is worse but the memory usage on server side is not that high.

On the Manufacturing Hub side before a data import is done the registry key [HKEY\_CURRENT\_USER\Software\Delmia\ergoplan\PPRLoader\VPM] "autocommit" can be set to "0", "1" or "N" by the customer whereas "N" is a natural number. By default the "autocommit" is set to "1". Depending on this registry entry the number of commits to the data base during a data import can be controlled by the customer. In detail this means the following:

If the value of the registry key is "0" there will be only one commit to the data base. This causes a minimum of memory usage but a slower performance.

In case the value of the registry key is "1" the commit to the data base will be done for each object which has to be written to the data base. An object in this case is represented by a component or a link.

In case the value of the registry key is "N", i.e. a natural number, the commit to the data base is done after the import of "N" objects. Basically the more commits to the oracle data base are performed the faster works the data transfer between Engineering Hub and Manufacturing Hub. But in the same time more memory on the server side is allocated.

If the number of the commits is reduced the performance is worse but the memory usage on server side is not that high.

[HKEY\_CURRENT\_USER\Software\Delmia\ergoplan\PPRLoader\VPM]

"autocommit"="value = "0", "1" or "N", look at Detailed Specifications (External Behaviour)"

For bigger data transfers the effect can be observed by measuring the time of the transfer duration. Furthermore the Performance Monitor can be used to check the memory usage during the import.

## Appendix A

### Configuration Scenarios – ENOVIA VPM V4 to PPR Manufacturing Hub

#### Some remarks on Configuration Mapping

- There is no “-infinity to date1/range1” in ENOVIA VPM V4.
- A range of “N to infinity” in ENOVIA VPM V4 is transcribed to “N-2147483647” (N-2<sup>31</sup>-1). This is done internally by the DPE Server and cannot be blocked by the PPRLoader.
- Milestones must also include to infinity. A Milestone itself is just a Date or Range.
- Dates and Ranges cannot be mixed in one effectivity.
- Expressions such as “(option1 OR option2) AND (option3 OR option4)” are not possible, they have to be rearranged.
- All scenarios describe the mapping of an effectivity or configuration object (like the dictionary) to its counterpart in Manufacturing Hub (if possible). Additional Manufacturing Hub -specific filtering capabilities (such as Labels, or “-infinity to date1”) are not mentioned.
- Modification effectivities are not handled by PPRLoader.
- The three filter scopes in Manufacturing Hub are concatenated with AND. Therefore, all filter conditions option/code rule AND range/tail number AND date must be valid, to see the link/object in DPE.
- To access a correct view of the complex multiline effectivity in DPE, please display the “vpm\_configuration” attribute, which will give you the correct effectivity expression.
- When using “Standard Coderule Mode” for the project, underscores and other special characters are removed from options/tokens (since they are not allowed).

#### Simple Scenarios

##### Option Effectivity → Code Rule (and Token)

**option1 → option1**

Along with the option effectivity/code rule itself on the link in Manufacturing Hub, a token (option1) has to be created in Manufacturing Hub (similar to create the option along with a category in a dictionary in ENOVIA VPM V4). Currently, tokens are only created when the dictionary is exported, so options defined in a dictionary outside the exported scope are mapped into code rules, but are not handled as tokens.

A possible solution would be to create all options as tokens, not just the ones included in dictionaries.

<b>Date Effectivity</b>	→	<b>Date Effectivity</b>
<b>date1 to date2</b>	→	<b>date1 to date2</b>
<b>date2 to infinity</b>	→	<b>date2 to infinity</b>
<b>Range Effectivity</b>	→	<b>Tailnumber</b>
<b>range1 to range2</b>	→	<b>range1-range2</b>
<b>range2 to infinity</b>	→	<b>range2-infinity</b>
<b>Milestone</b>	→	<b>Date Effectivity/Tailnumber</b>
<b>milestone1 to infinity</b>	→	<b>date1 to infinity/range1-infinity</b>
Milestones are not exported as Milestones, but only as their corresponding Date or Range values. Depending on that, PPRLoader should handle them in the appropriate way.		
<b>Mix from Option and Date or Range Effectivity →</b>		
Mix from Option, Date Effectivity and Tailnumber		
<b>option1 and date1 to date2</b>	→	<b>option1 and date1 to date2</b>
<b>option1 and range1 to range2</b>	→	<b>option1 and range1 to range2</b>
Again, a mix from Date and Range is not possible in the regular Code Rule Mode. The cases “date/range to infinity” were covered before.		

## Complex Scenarios

Complex scenarios cannot be mapped through the Manufacturing Hub-internal regular effectivity mechanism in general. The following examples indicate situations, where you have to use the Extended Effectivity mode. For details on the Extended Effectivity mode and how to define the operator tokens, please see the [Project Library Manual](#).

### Complex Option Effectivity → Code Rule

<b>NOT option1</b>	→	<b>!option1</b>
<b>option1 AND option2</b>	→	<b>option1+option2</b>
<b>option1 OR option2</b>	→	<b>option1/option2</b>
<b>Complex Date Effectivity</b>	→	<b>Code Rule Expression</b>
<b>(option1 AND option2) OR (option3 AND option4)</b>	→	<b>(option1+option2)/(option3+option4)</b>
<b>NOT(option1 AND option2) OR (option3 AND option4)</b>	→	<b>!(option1+option2)/(option3+option4)</b>

### Complex Range Effectivity→ Tailnumber Expression

range1, range2 → range1, range2

range1 to range2 OR range3 to range4

→ range1-range2, range3-range4

**Package from Option and Date or Range Effectivity →  
Mix from Option, Date Effectivity and Tail-  
number**

(option1 AND option2) and date1 to date2 OR

(option3 AND option4) and range1 to infinity

→

(option1 AND option2) and date1 to date2 OR

(option3 AND option4) and range1 to infinity

## Configuration Objects

**Dictionary → No Mapping**

The ENOVIA VPM V4 dictionary object itself has no counterpart in Manufacturing Hub.

**Config Rule → No Mapping**

Config Rules are not be mapped by PPRLoader, since they cannot be handled by Manufacturing Hub.

**Config Matrix → No Mapping**

Config Matrices are not mapped by PPRLoader, since they cannot be handled by Manufacturing Hub.

**Categories → Token Group**

Categories are stored as group descriptions on tokens.

**Config Handler → Calculation Model**

option1 → option1

option2,option2,option3 → option1,option2,option3

Calculation Models in Manufacturing Hub can only contain a list of options/tokens. Neither Date nor Range effectivity can be handled.

## Appendix B

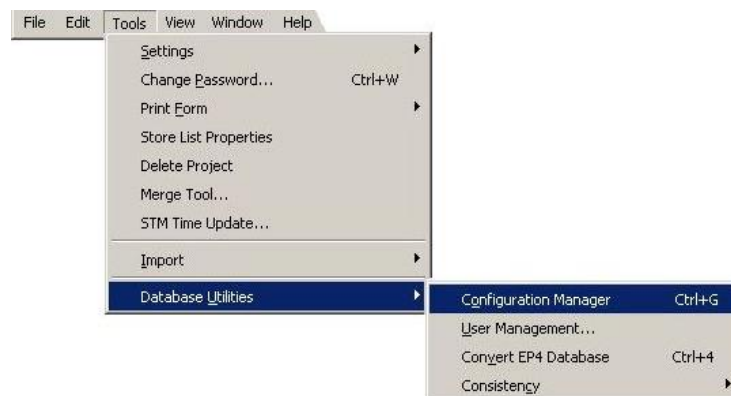
### PPR Manufacturing Hub Configuration and DPE Configuration Manager



Please see also the corresponding sections/chapters in the DPE/Manufacturing Hub manual (especially in the [Administration Manual](#)).

#### General Information – by Administration Manual

The Configuration Manager, which allows the customization of the PPR Manufacturing Hub database, can be accessed via **DPE > Tools > Database Utilities > Configuration Manager**.



**Figure 45: Config Manager**



For instructions on how to display these attributes and change their properties, please refer to the [Administration Manual](#).

The configuration of the attribute “updatestate” is described in the following paragraph.

#### Special Attributes and Bitmap Adaptation for the Update Protocol

The Engineering Hub to Manufacturing Hub Connection Update Protocol stores the status of the last synchronization on each link (SubCompltem) and object (ErgoCompBase) in the dedicated attribute “updatestate”.

To allow a simple notification for the user on changes within this attribute, it is possible to modify the bitmap of an object, based on the update status. This section describes how to set this up.

You can do the configuration in the configuration manager or in the PlanTypeSet. If you configure the attribute “updatestate” in the configuration manager, all plantypes derived from ergocompproductdefault are affected. In the consequence all derived plantypes obtain equal icons. The configuration of the attribute in the PlanTypeSet gives you the possibility to change the bitmaps of selected Plantypes only. This allows a more exact control of the display.



### Note

*You cannot use both ways of configuration at the same time*

### In the PlanTypeSet

The modification of the icon of an object is related to a “Special Attribute” (in our case: updatestate) that has to be selected in the properties of the Plan-type. If it is not there, it has to be overwritten from the basetype, e.g. ergo-comproductdefault or ergocompbase.

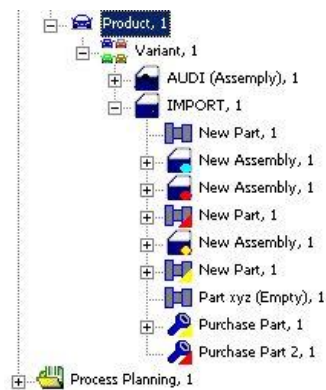
- 1) Open the PlanType Set in the Library. Right-click a product Plantype, e.g. “subassembly”. In the contextual menu select the “Edit” to open the Plantype Editor.
- 2) Select “subassembly”. Since the attribute “updatestate” is missing, go to the basetype “ergocompbase”, select the attribute “updatestate”, open the contextual menu and select “overwrite”.
- 3) Go back to the Plantype “subassembly”. In the list of its attributes there should be now the attribute “updatestate”. Open the contextual menu and create a new value list. Then enter the name, value and sort number (must be unique) for the new value item, which represents one possible status value, set by the Update Protocol mechanism. For each value select a different bitmap. (See table below).
- 4) Select “subassembly and edit the properties. Set “Depends on value” to “Yes” and choose as “special attribute “updatestate” in the combo box.
- 5) Save and close the Plantype Editor.

The values used by the Engineering Hub to Manufacturing Hub Connection Update Protocol for Subassembly are listed in the table below:

**Table 1: Values used by the Engineering Hub to Manufacturing Hub Connection Update Protocol for Subassembly**

Item name	Item Value	Display order	Bitmap
“new”	new	10	
“update”	update	20	
“position”	position	30	
“version”	version	40	
“config”	config	50	
“delete”	delete	60	
EMPTY		70	





**Figure 46: Result of the Configuration within the PlantypeSet**

The different icons represent the corresponding update status on each object, synchronized by the Engineering Hub to Manufacturing Hub Connection.



### Note

*Customized bitmaps can be created using any graphics program (e. g. Paint). The bitmaps must have a size of 18 x 17 pixels (width x height) and must be saved in 256 color mode.*

### In the Configuration Tool

Open the Configuration Manager. Go to type “ergocompproductdefault”. Open the list of its attributes and search for “updatestate”. If it is not there, go to the basetype “ergocompbase”, select the attribute “updatestate” and overwrite it. Go back to “ergocompproductdefault” and select the attribute “updatestate”.

Go to the properties editor. Change the “Prompt” to “VPM Update Status”. Change the control type from Edit to Combobox, picking the corresponding entry from the list.



**Figure 47: Change the Control Type**

- Go back to the tree on the left sight, rightclick “updatestate” and create a new value list with the entries from the table above. *Please refer to the [Administration Manual](#).*
- Now select the type ergocomplantdefault and change its.
- Search the entry “Special attribute”. From the combo box select the attribute update state.

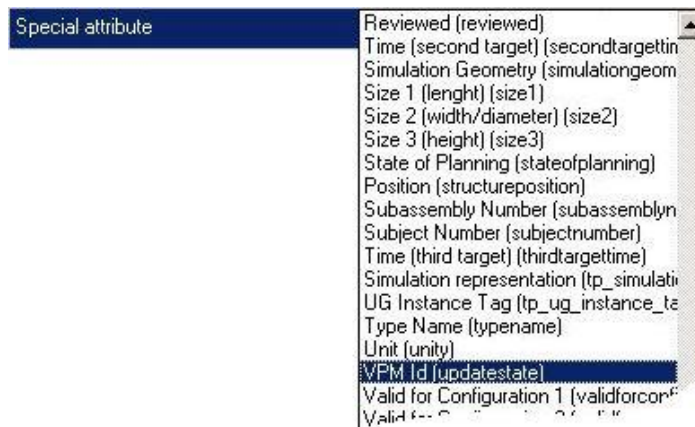


Figure 48: Special Attribute

- Search the entry “Depends on values” and set it to “Yes”



Figure 49: Entry “Depends on Values”

- Save the changes.
- In the PPR Navigator the products are now displayed with different icons. The assigned icons identify the updatestate.

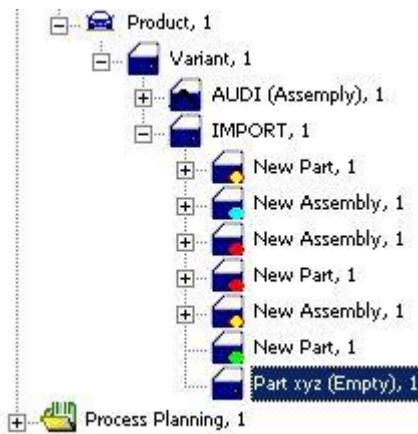


Figure 50: Products Display in PPR Navigator

## Appendix C

### Registry Settings – Overview

KEY	Value (default)	Value (option)	Description
<b>[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader]</b>			
"username"	""		Look at documentation, chapter " <a href="#">PPRLoader Customization Settings</a> ". Enter username of DPE to use the prploader.
"password"	""		Look at documentation, chapter " <a href="#">PPRLoader Customization Settings</a> ". Enter password of DPE to use the prploader.
<b>[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM]</b>			
"backbonesender"	"<YourMachines-Name>"		Look at documentation, chapter " <a href="#">Configure the PPRDaemon</a> ". Enter the name of the machine where the PPRDaemon/PPRLoader is running or rather enter "LOCALHOST" (currently backbonesender and receiver have to be the same machine)
"backbonereceiver"	"<YourMachines-Name>"		Look at documentation, chapter " <a href="#">Configure the PPRDaemon</a> ". Enter "LOCALHOST" (currently backbonesender and receiver have to be the same machine)
"cfgcheck"	"1"	"1 true yes 0 false no"	Enable/disable update protocol check for configuration attributes (effectivity.begin, effectivity.end, tailnumber, coderulestring, extendedeffectivity)
"cfgdate"	"link+comp" for VPM V4 "link" for VPM V5	"all link comp link+comp"	Specifies, if date effectivity is stored on relationship_nodes (link), on ergocompbase (comp) or on both
"cfgfile"	"<DefaultVPMConfig-File>"		Look at documentation, chapter " <a href="#">Customized/Extended Attribute Mapping</a> ". Enter location of Customized/Extended Attribute Mapping configuration file.
"cfgexteff"	"link+comp" for VPM V4 "link" for VPM V5	"all link comp link+comp"	Specifies, if extended effectivity is stored on relationship_nodes (link), on ergocompbase (comp) or on both
"cfgextended"	"0" for VPM V4 "1" for VPM V5		Specifies, whether effectivity is stored in the extendedeffectivity attribute
" <b>cfgmultilineattr</b> "			
"cfgnormal"	"1" for VPM V4 "0" for VPM V5		Specifies, whether effectivity is stored in the « old » configuration attributes
"cfgoption"	"link+comp" for VPM V4 "link" for VPM V5	"all link comp link+comp"	Specifies, if option effectivity is stored on relationship_nodes (link), on ergocompbase (comp) or on both
"cfgrange"	"link+comp" for VPM V4 "link" for VPM V5	"all link comp link+comp"	Specifies, if range effectivity is stored on relationship_nodes (link), on ergocompbase (comp) or on both

KEY	Value (default)	Value (option)	Description
"cgratfile"	""		obsolete
"cgrftpdelay"	""		obsolete
"cgrdir"			obsolete
"cgrsrvdir"			obsolete
"cgrftpghost"			obsolete
"cgrftpscript"			obsolete
"cgrftppassword"			obsolete
"cgrftpusername"			obsolete
"deletemode"	"notify"	"notify auto none"	Look at documentation, chapter "Two modes for deleted objects".
<b>"doccadtypes"</b>			
"dump"	""	"comp link cgr cad attr progress config env id all"	Allows racing of additional information during import
"enuminst"	"ordernumber"		ergocompbase attribute for enumeration of objects
"installdir"		Obsolete for V5	Installation directory of PPRLoader
"longextid"	""		obsolete
"lxcftpghost"			obsolete
"lxcftpscript"			obsolete
"lxcftpusername"			obsolete
"lxcftppassword"			obsolete
"lxcdir"			obsolete
"lxcsrvdir"			obsolete
"mvamultiline"	""	"1 none"	Look at documentation "EH-MH Transfer of Multi Value Attributes". If the MVA value is long and requires more than one line then enter 1 to get several lines. (VPM V5 only)
"mvaseparator"	""	Any separator except "@@" because this is used as default by program.	Look at documentation "EH-MH Transfer of Multi Value Attributes". To use other separators than the default. (VPM V5 only)
"mvalinebreak"	""	"\r\n".	Look at documentation "EH-MH Transfer of Multi Value Attributes". The default MVA line break character is "\r\n". Without any changes every value of the attribute is shown in a separate line. (VPMV5 only)
"prodroot"	""		obsolete
"prodnode"	""		obsolete
"prodleaf"	""		obsolete
"prodmodel"	""		obsolete
"proddoc"	""		obsolete
"prodmodelattr"	""		obsolete

KEY	Value (default)	Value (option)	Description
"proddocattr"	""		obsolete
"propagatedelete-mode"	""		
"recalcabsmatrix"	"1"	"1 true yes 0 false no"	Controls the recalculation of the absolute position matrices by the Manufacturing Hub server
"resdoc"	""		obsolete
"resdocattr"	""		obsolete
"resetupdatestate"	""	1	For incremental update: to reset values like "new" after the next update.
"resleaf"	""		obsolete
"resmodel"	""		obsolete
"resmodelattr"	""		obsolete
"resnode"	""		obsolete
"resroot"	""		obsolete
"stopcondition"			
"stopscript"			
"updatestateattrattachment"	"updatestate"		relationship_plant_provides_prod attribute for update protocol
"updatestateattrlink"	"updatestate"		relationship_nodes attribute for update protocol
"updatestateattrobject"	"updatestate"		ergocompbase attribute for update protocol
"updatestateattrrelation"	"updatestate"		For future use
"useclientcache"	""		
"versions"	"1"	"1 0"	Enables/disables support for VPM V4 versions
"xmlftpghost"			obsolete
"xmlftpscript"			obsolete
"xmlftpusername"			obsolete
"xmlftppassword"			obsolete
"xmlidir"			obsolete
"xmlsrvdir"			obsolete
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\Environments]			
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\Aliases]			
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings]			
"securetransfermode"	"0"	"0 <0 .. ftp   1 .. https>"	Enter 0 for transfer via FTP and enter 1 for transfer via https. See <a href="#">File Transfer via HTTPS, HTTP, or FTP</a>
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\ABF Settings]			
[HKEY_CURRENT_USER\Software\Delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\CGR Settings]			

KEY	Value (default)	Value (option)	Description
<b>[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\LXC Settings]</b>			
<b>[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\WPK Settings]</b>			
<b>[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\TRANSFER Settings\XML Settings]</b>			
"xmlftphost"	"<FTPHostName>"		Look at documentation, chapter " <a href="#">Configure XML Settings</a> ". Enter the name of the VPM database Server.
"xmlftpusername"	"<FTPUserName>"		Look at documentation, chapter " <a href="#">Configure XML Settings</a> ". Enter your username.
"xmlftppassword"	"<FTPPassword>"		Look at documentation, chapter " <a href="#">Configure XML Settings</a> ". Enter your UNIX password
"xmldir"	"<DELMIAProcessEngineerInstall-Dir\PPRClient\data>"		Look at documentation, chapter " <a href="#">Configure XML Settings</a> ". Enter the pathname of the directory in which you want the XML files stored.
"xmlsrvdir"	"<VPMServerExport-Directory>"		Look at documentation, chapter " <a href="#">Configure XML Settings</a> ". Enter the path to find XML on the LCA server.
"xmllogdir"	"C:\\Temp"		Look at documentation, chapter " <a href="#">Configure XML Settings</a> ". Enter the path to find XML.
"xmlhttpsbaseurl"	""		Look at documentation, chapter " <a href="#">Configure XML Settings</a> ".
<b>[HKEY_CURRENT_USER\Software\Delmia\Ergoplan\PPRLoader\VPM\ABF Fastener Mapping]</b>			
"fastener"	""		
"joint"	""		
"jointelement"	""		
"jointbody"	""		
"mode"	"internal external"		
"fileextension"	""		
"cfgfile"	"<DefaultABFConfig-File>"		
"relationship"	""		
<b>[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping]</b>			
"Resource Environments"	"RESENV"	"ENV1+ENV2+..."	Specifies the environment(s) on the VPM side for which relations between a product component and a resource component shall be created
<b>[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping\PRODUCT]</b>			
("PRODUCT" serves as example for other product structure mappings; for any defined environment the keys must be existing and not empty)			
<b>[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping\PRODUCT\CATIA_MODEL]</b>			
"Plantype"	"Part"	Any valid product plan-	"Valid" also includes parent-child constraints resulting from the used plan type set



KEY	Value (default)	Value (option)	Description
		type	
"CAD File Attribute"	"cadfile"		Do not change
<b>[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping\PRODUCT\DOCUMENT]</b>			
"Plantype"	"Part"	Any valid product plan-type	"Valid" also includes parent-child constraints resulting from the used plan type set
"CAD File Attribute"	"cadfile"		Do not change
"Document Attribute"	"nameshort"		Do not change
<b>[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping\PRODUCT\PART_LIST]</b>			
"Plantype"	"Subassembly"	Any valid product plan-type	"Valid" also includes parent-child constraints resulting from the used plan type set
<b>[HKEY_CURRENT_USER\Software\delmia\ergoplan\PPRLoader\VPM\Product Structure Mapping\PRODUCT\Root]</b>			
"Plantype"	"Product"	Any valid product plan-type	"Valid" also includes parent-child constraints resulting from the used plan type set

# List of Figures

Figure 1: People & Organization Panel .....	5
Figure 2: Client Authentication Mode Panel .....	6
Figure 3: Server Authentication Mode Panel .....	6
Figure 4: Reference Tree .....	7
Figure 5: Instance Attributes .....	7
Figure 6: Settings.....	8
Figure 7: Configure XML Settings .....	9
Figure 8: CGR Settings.....	10
Figure 9: LXC Settings.....	10
Figure 10: Clgfile.....	12
Figure 11: Interaction and Data flow between the ENOVIA VPM V4 Portal, PPRDaemon, and PPRLoader.....	14
Figure 12: Keys from the Registry.....	15
Figure 13: Changing Name .....	17
Figure 14: Resource Environments.....	18
Figure 15: Resource Plantype Names .....	18
Figure 16: Product Folder .....	19
Figure 17: Config Manager .....	20
Figure 18: Config Tool .....	20
Figure 19: Deletemode .....	21
Figure 20: Update Protocol .....	22
Figure 21: Data Transfer.....	29
Figure 22: Data Transferred between ENOVIA VPM V4 and the DELMIA Manufacturing Hub through the Engineering Hub.....	31
Figure 23: ENOVIA VPM V4 – PRC/product structure Manufacturing Hub.....	32
Figure 24: Settings for the Secured Transfer .....	35
Figure 25: xmlhttpsbaseurl: Value.....	35
Figure 26: Registry Keys.....	36
Figure 27: Transfer Settings.....	38
Figure 28: Tokens are generated, but special characters are removed .....	41
Figure 29: Tokens are generated, special characters are kept.....	41
Figure 30: Tokens are generated, but special characters are removed .....	41
Figure 31: Tokens are generated, but special characters are removed .....	41
Figure 32: Tokens are not generated .....	42
Figure 33: Mapping of the Versions .....	42
Figure 34: Has Versions .....	43

Figure 35: Filtering Data .....	44
Figure 36: 3d COM Login panel .....	52
Figure 37: 3d COM Window .....	52
Figure 38: Authentication Mode .....	53
Figure 39: Enovia Access .....	53
Figure 40: Launch 3d COM and Connect to VPM .....	54
Figure 41: Select the 1st VPM Part.....	54
Figure 42: User Authorization .....	55
Figure 43: Command Shell Window of PPRLoader.....	55
Figure 44: Command Shell .....	56
Figure 45: Config Manager .....	65
Figure 46: Result of the Configuration within the PlantypeSet.....	67
Figure 47: Change the Control Type .....	67
Figure 48: Special Attribute.....	68
Figure 49: Entry “Depends on Values“ .....	68
Figure 50: Products Display in PPR Navigator .....	68

# List of Tables

Table 1: Values used by the Engineering Hub to Manufacturing Hub Connection Update Protocol for Subassembly .....	66
--	----

# Index

## A

Attribute Mapping .....	23
Auto commit Settings.....	62

## B

Batch mode	
pprloader -partial.....	58
Batch Mode .....	56
AT0EXPND.....	58
pprloader -help.....	58
shell	
at0expnd.sh.....	60
Start PPRLoader.....	56
Batch Parameters	
FTP.....	60

## C

Code Rules .....	40
<i>Config Handler</i> .....	65
Config Matrices .....	65
Config Rules.....	65
Configuration	
CGR Settings.....	10
LXC Settings.....	10
PPRDaemon.....	8
XML Settings .....	9
Configuration Behavior .....	51

## D

dictionary object .....	65
DPE Configuration Manager.....	66

## E

Effectivity .....	63
ENOVIA VPM V4	
dictionary object.....	65
ENOVIA VPM V4 Portal .....	13, 52
ErgoCompBase .....	15
ergocompproductdefault.....	68
Extended Attribute .....	51
externalid.....	15

## F

FTP Settings .....	50
--------------------	----

## G

General Settings.....	9
-----------------------	---

## H

HTTP .....	33
HTTPS .....	33

## I

Incremental Update .....	31
Initial Import.....	18

## L

logical CompID .....	26
----------------------	----

## M

Multi Instancing .....	15
------------------------	----

## N

Nonliability.....	ii
-------------------	----

## O

Object ID (OID).....	15
----------------------	----

## P

PPRDaemon .....	12, 13, 54
Configuration.....	8
PPRLoader.....	54
Customization Settings.....	11
PRC .....	19
Prerequisites	
3dCOM.....	3
Manufacturing Hub-Server .....	3
Prerequisites	
Registry .....	61
Priority of Update Status.....	21
Product Structure .....	51
Product Structure Mapping .....	21

## R

Registry Prerequisites .....	61
Registry Settings	
Encryption and Decryption.....	37
Registry Template .....	50

## S

Server	
3d COM server.....	52
3d COM server	

APACHE .....	52
Orbix .....	52
XCAD Server .....	52
Settings	
3dCOM	
catia_xc0_multi variable .....	7
runServerVPM file .....	7
3dCOM .....	6
APACHE .....	5
ENOVIA VPM V4 .....	4
Multi CAD Server .....	5
PPRLoader .....	8
Setup	
Step-by-Step .....	50
shell .....	60
Special attribute	
updatestate .....	67
Special Attribute .....	67
Start the Transfer .....	54
<b>T</b>	
Tailnumber .....	64
Token Group .....	65
Top Level Product Folder .....	18
Transfer Settings .....	50

## U

Update protocol .....	19
Update Protocol .....	11, 19
updatestate .....	19
Update Protocol Script .....	51
Update Status .....	21
Update Status for links .....	21

## V

Versions .....	43
VPM environment .....	4, 58
VPM Environment .....	4, 52
VPM V4 Portal .....	52

## X

XCAD Server .....	52
XCAD settings .....	59
XML file .....	12, 13, 14, 23, 50, 58, 60, 61
XML File .....	4