



How Products Become Profits™

Agile® e6.0

## **Agile e6.0**

Upgrade Tool 3.0

## **Copyrights and Trademarks**

Copyright © 1992-2005 Agile Software Corporation. All rights reserved.

You shall not create any derivative works of this publication nor shall any part of this publication be copied, reproduced, distributed, published, licensed, sold, stored in a retrieval system or transmitted in any form or by any means: electronic, mechanical, photocopying, or otherwise, without the prior written consent of Agile Software Corporation, 6373 San Ignacio Avenue, San Jose, California 95119-1200 U.S.A.; Telephone 408.284.4000, Facsimile 408.284.4002, or <<http://www.agile.com/>>.

The material in this document is for information only and is subject to change without notice. While reasonable efforts have been made in the preparation of this document to ensure its accuracy, Agile Software Corporation assumes no liability resulting from errors or omissions in this document or from the use of the information contained herein. Agile Software Corporation reserves the right to make changes in the product design without reservation and without notification to its users.

Agile Software is a registered trademark and Agile, Agile Product Collaboration, Agile Product Cost Management, Agile Product Service & Improvement, Agile Program Execution, Agile Product Interchange, AgileMD, and the Agile Logo are trademarks of Agile Software Corporation in the U.S. and/or other countries. Guaranteed Business Results is a service mark of Agile Software Corporation. All other brands or product names are trademarks or registered trademarks of their respective holders.

Java and Solaris are registered trademarks of Sun Corporation.

Microsoft, Microsoft Windows, Microsoft Word, Microsoft Excel, Internet Explorer and SQL Server are registered trademarks of Microsoft Corporation.

Oracle and Oracle8i are registered trademarks of Oracle Corporation.

### **NOTICE OF RESTRICTED RIGHTS:**

The Software is a “commercial item,” as that term is defined at 48 C.F.R. 2.101 (OCT 1995), consisting of “commercial computer software” and “commercial computer software documentation” as such terms are used in 48 C.F.R. 12.212 (SEPT 1995) and when provided to the U. S. Government, is provided (a) for acquisition by or on behalf of civilian agencies, consistent with the policy set forth in 48 C.F.R. 12.212; or (b) for acquisition by or on behalf of units of the Department of Defense, consistent with the policies set forth in 48 C.F.R. 227.7202-1 (JUN 1995) and 227.7202-4 (JUN 1995).

**July 27, 2005**

# CONTENTS

---

<b>Chapter 1 Introduction</b>	<b>1</b>
Overview	1
Architecture	1
Upgrade tool	2
Control and Log files in XML	2
GUI	3
Upgrade Process	3
Preactions on original production environment	4
Customization Upgrade	4
Test	6
Transfer Data from production system	6
Special Upgrade: Classification	7
Adapt original production environment	7
Upgrade Pool Concept	7
Migration concept	8
Take Over Data from the production system	9
<b>Chapter 2 Installation</b>	<b>10</b>
Prerequisites	10
Installing the Upgrade Tool	10
<b>Chapter 3 Configuration of Upgrade Tool</b>	<b>12</b>
Check database settings	12
Check database settings in Oracle	12
Prepare environments	14
Prepare reference environments	14
Define Database Connections	15
Define Source Master/Target Master/Customer Connections	16
Define Production DB	18
Define Parameters	19
Configure control and log files	20
Control and log files for comparing/updating repository tables	20
Control and log file for Synchronizing repository	21
Configure special.xml for synchronizing repository	23
Special Upgrade: Classification	26
Configuration file for Special replace.xml	27
Configure take over data from the production system	27

<b>Chapter 4 Migration</b>	<b>29</b>
Performing Customizing Upgrade	29
Step Run preaction scripts	29
Step DTV-upgrade	30
Adapt physic. table definition acc. to DataView table definition	31
Step Run before-common-scripts	33
Step EDB-UPD	33
Special Upgrade: Classification	33
Transfer data from production system	34
Define reference tables	34
Perform transfer	35
Synchronize repository	35
Script Run after takeover	36
Special Upgrade: Classification	36
Upgrade classification lists only	36
Upgrade complete classification	37
<b>Chapter 5 Annex</b>	<b>38</b>
Convert XML files to HTML	38
Directory structure	38
Shell scripts / Log files	39
SQL scripts	41
Folders Contents “upgrade/conf” and “upgrade/conf/template”	43
Contents of the folder “upgrade/conf”	43
Contents of the folder “upgrade/conf/template”	45
Configuration parameters	45
Migration Rules	46
Standard Rules for Delete	46
Standard Rules for Update	46
Standard Rules for Insert	47
Special Rules	47

# Chapter 1

## Introduction

---

*Part I provides an introduction and an overview to the Agile e6 Upgrade Tool. It includes the following chapters:*

- Overview of Upgrade Tool
  - Architecture
  - Upgrade process
  - Additional Documentation
- 

### **Overview**

The Upgrade tool allows a direct upgrade to Agile e6.0 from earlier releases (CADIM/EDB 2.3.2 or higher, axalant 2000 SPx, Eigner PLM 5.x ).

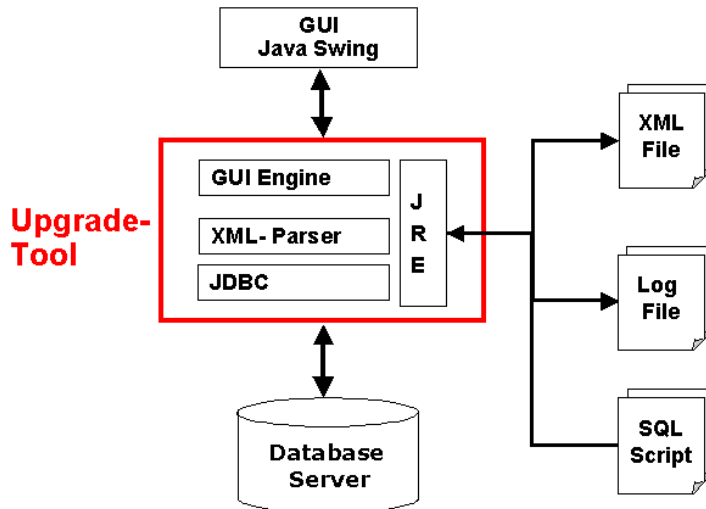
The Upgrade Tool addresses experienced project engineers and PLM administrators with customizing and database experience. Do not use the tool without the necessary knowledge. Read the complete manual in order to get all necessary information. Do not attach to or even change the production system. Always work on a copy of the production database dump. Avoid working on production computers to exclude any influence on the system. Never insert database connections of production database users in any configuration file or script except for exporting the dump or a source for copying tables (Production Database).

Do not use the software in any situation where significant damage to property or business could occur from a software error. In no event will AGILE or any other party who has been involved in the creation, production or delivery of the software be liable for special, direct, indirect, incidental, punitive or consequential damages, including loss of profits, revenue, business, goodwill, data or computer programs or inability to use the software, however caused and regardless of the theory of liability even if AGILE or such other party has been advised of the possibility of such damages.

### **Architecture**

The Upgrade Tool comprises the following components

- The upgrade tool itself
- Control and Log files in XML
- GUI



## Upgrade tool

The upgrade tool is implemented in Java. It accesses the databases directly using a JDBC connection. Through JRE to XML and log files and command scripts which call SQL scripts are executed.

### Control and Log files in XML

The configuration of all upgrade steps is stored in a set of xml control files. When executing a step, a log files is created in XML containing the error messages.

A set of control and log files is defined for each upgrade step. The location of the files is stored in the main configuration file `upgrade\conf\ApplicationParameter.xml`.

The log and error files are mainly in XML format and can be found in the directory `Upgrade/log/...`

For all actions (see **Action** tab) that compare table content (Create files) and change repository information (Perform Insert, Update, Delete) a common set of control and log files is used.

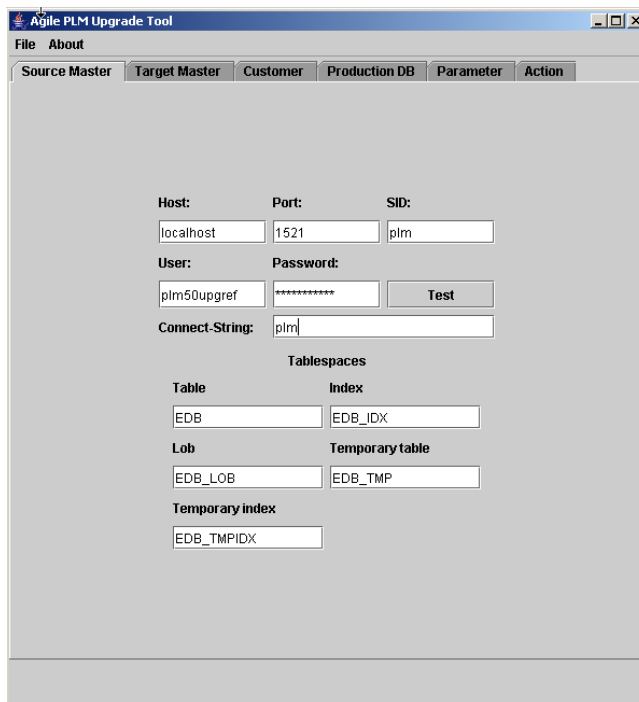
You can convert the XML files created as log and configuration files to HTML and view them with a browser. The batch file `xml2html.bat` creates HTML files for insert, update and delete for each table. This function is available on Windows platforms only.

Note: You need memory for approximately sixth times the XML file size!

See Annex for a description on converting XML files to HTML.

## GUI

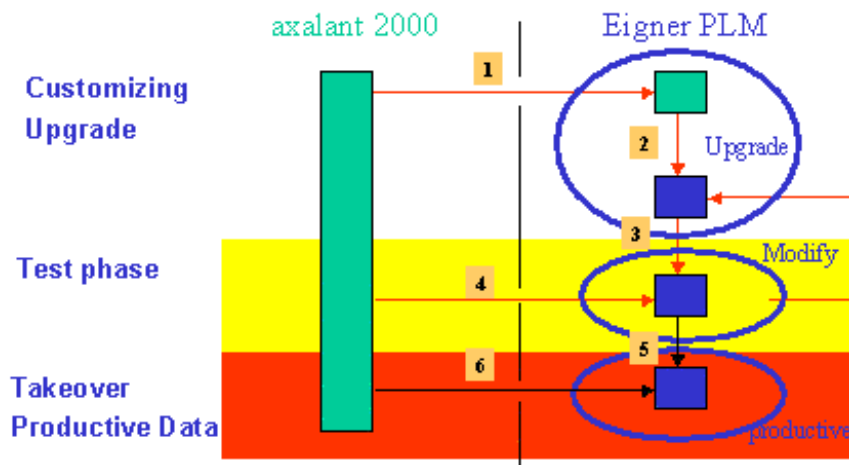
The user can execute the migration steps easily from the user interface of the Upgrade Tool.



## Upgrade Process

To upgrade from an earlier release to Agile e6, you carry out the following steps:

- Preactions  
where you make a copy of your original production environment
- Performing Customizing upgrade  
Customization and configuration stored in the database is updated to Agile e6.0.
- Test phase
- Taking over your production data to finish the process.



## Preactions on original production environment

Make a copy of your production database dump . Do not attach your production system. Always work on a copy of your data.

Start the upgrade on this copy. The minimum passing time will be 4-5 hours (depending on the system, main parameter is memory!).

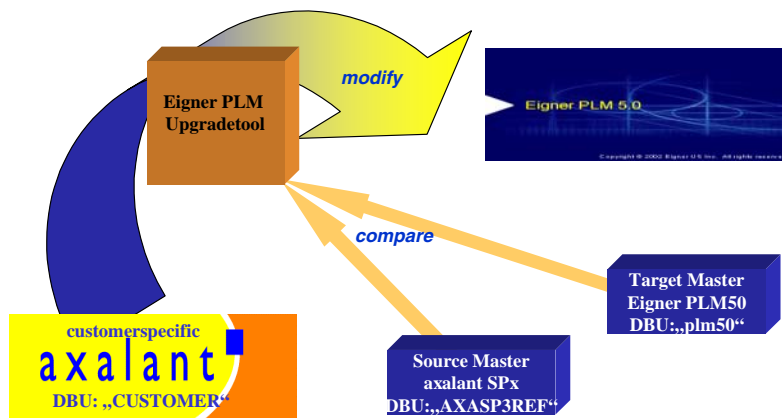
The Upgrade Tool will create a dump on which you can run Agile e6.0. This dump is not error free. You have to check all functionalities and clear out the errors caused by setting up the upgrade tool.

## Customization Upgrade

The Upgrade Tool opens three database connections:

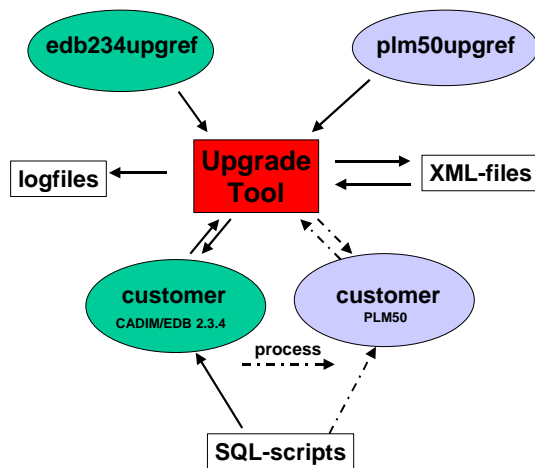
- ❑ Source Master (*Source reference*):  
A CADIM/EDB / axalant 2000 / Eigner PLM dump with Agile e6.0 "n" table structure. The necessary SQL scripts have been executed to adjust the DataView tables. LogiView standard models have been deleted so that standard LogiView models will be completely reloaded. This is already done in reference dumps delivered with this upgrade tool.





- ❑ Target Master (*PLM reference*):  
This is a standard Agile e6.0 dump.
- ❑ Customer (*CUSTOMER*):

The customer dump has the version of the Source Master at the beginning of the upgrade process. At the end it has the version of the Target Master.



The Upgrade Tool selects each row from the Source and the Target Master dump, compares the data sets from both dumps to identify the differences and checks if the customer has modified these data.

The upgrade action (Insert, Update, Delete) is determined for each record and the information is stored in a set of XML files. The migration rules are listed in the Annex.

The Upgrade Tool is only able to compare tables with the same table structure. Therefore the DataView tables in the reference dumps (edb234upgref ...) have Agile e6.0 structure. The

customer dump will be formatted during the upgrade (execution of SQL scripts and step synchronize repository).

The customization upgrade is generally split into two steps:

- ❑ Comparison of the data sets from the different dumps and storing the changes in an XML file for the three possible operations: delete, insert and upgrade (e.g.: dtvdel.xml, dtvins.xml, dtvupg.xml).
- ❑ The Upgrade Tool reads the XML information and performs the corresponding SQL-statements. After this step the changes are available in the Agile e6.0 dump.

## Test

Take over the data from the production system for a first test and let the user test all functionalities, maybe during training. If errors occur, remove them via customizing. Not everything might be done automatically.

If testing does not raise any error you can plan to change to productivity. Shut down your CADIM/EDB, axalant or Eigner PLM system. Take over your data from the production system (and the files!) again and Agile e6.0 is your production system.

## Transfer Data from production system

All tables containing production data like document and item master data are copied from the production system to your Agile e6.0 installation. They will be adapted so that you can work on this data within Agile e6.0.

After finishing the test of the new custom-specific Agile e6.0 functionality, the customer can go live with the new version.

During the test period a lot of new data is created in the production system. This data must be copied again from the old production system into the new environment and adapted to the new Agile e6.0 table structure. This step is called Takeover Production data.

The Upgrade Tool supports the following actions:

- ❑ Definition of a list of reference tables containing production data.
- ❑ Dropping the reference tables in the customer environment and copy the table from the production system.
- ❑ Adapting the table structure to Agile e6.0.
- ❑ Post-actions like migrate production classification data.

For this step the tool opens two database connections:

- ❑ Production Database

This database connection is used as source for the tables copied into the new Agile e6.0 environment. No changes are made in the production database.

- ❑ Customer

Connection to the new production environment. The reference tables will be dropped. This becomes the new production Agile e6.0 environment.

## Special Upgrade: Classification

If you use the Classification module, the following steps are necessary in addition to the standard upgrade process.

- Adapt original production environment
- Upgrade Pool Concept
- Migration concept
- Take Over Data from the production system

### Adapt original production environment

Since Agile e5.0 UIC and GIC  $\leq 1000$  are reserved for the standard development. Existing users or groups use such C\_IC must be migrated to an higher Value. This migration must be executed in your production system before you start any other upgrade activity.

This update can be very time consuming ! In a big customer dump it takes 1h/6 users. To solve the time conflict, breakdown the update into different sub sets and try to run it in parallel .

1. To do this Adapt the following statement in the sql script

```
INSERT INTO TEMP_U (OLD_U) SELECT C_IC FROM T_USER a
WHERE
C_IC > 200 AND C_IC < 1000 AND C_NAME NOT LIKE 'EDB%' AND
C_NAME NOT LIKE 'DEMOEP%';
```

2. To execute the UIC/GIC Migration execute the

```
PC          RUN update_uic.cmd
UNIX       run update_uic.sh
```

### Upgrade Pool Concept

With Agile e5.1 a new classification concept (Pool concept) has been introduced. If you are upgrading from CADIM and axalant to Agile e6.0, you need to adapt the classification data from the ATT concept to the new pool concept. The following table shows a comparison between the old ATT Concept and the new Pool concept:

Overview ATT concept (Old)	Overview new Pool concept
<ul style="list-style-type: none"> <li><input type="checkbox"/> Attributes are defined class specific in the ATT concept</li> <li><input type="checkbox"/> Domain values for an attribute are defined in static menus <ul style="list-style-type: none"> <li><input type="checkbox"/> No release procedures and status management for classes and attributes</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><input type="checkbox"/> Attributes can be defined class independent</li> <li><input type="checkbox"/> Pool attributes can be assigned to more then one class</li> <li><input type="checkbox"/> Domain values for a pool attribute can be stored in special domain table</li> <li><input type="checkbox"/> For every class can be specified which domain value is valid</li> </ul>

Note: Migration of changes in the attribute inheritance must be executed for all version

## Migration concept

The migration includes

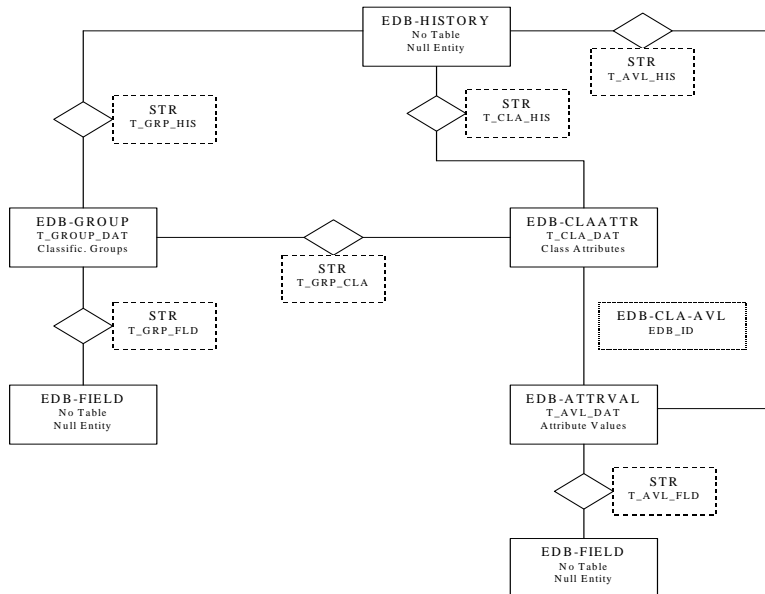
- Merge of Attribute definition

Attributes are considered as identical if the following values are identical

- C\_LETTER
- C\_TITLE
- C\_TYPE

If you have defined C-Letter and C\_TITLE as multi language fields(Standard beginning with axalant 2000) you define with the Parameter DB language which language is used as basic for the merge. (see installation and configuration manual for more information how to define parameters)

- Initial Load of the attribute value pool including the activation of the attributes for special classes
- Update classification lists
- Update used field name
- Set of attribute ATT\_VAL\_REF in the classification Lists
- Update field and Mask definition (if you have defined own forms for classes)



If possible do not create or modify basic definition of classes and attributes between customization upgrade and take over data from production system.

This influences the migration steps must be executed after the takeover process.

- No new classes and attributes are created

Only classification list tables must be defined as reference table

- T\_GRP\_ART
  - T\_GRP\_DOC
  - T\_GRP\_ORD
  - T\_GRP\_PRO
- Customers have created new classes and attributes in the production system after the customization upgrade

In addition to the classification list tables classes, attributes and domain values must be copied and migrated

- T\_GROUP\_DAT
- T\_GROUP\_STR
- T\_GRP\_FLD

### **Take Over Data from the production system**

If you created new users and or groups since the date when the dump was exported from production system the following DataView-tables must be migrated !

- T\_USER,
- T\_GROUP,
- T\_GRP\_USR,
- T\_PROFILE
- Related tables of the PLM – person management

Attention: new plm- or axalant-user like EDB-WFL, EDB-DFM, EDB-DDM, EDB-GDM, EDB-EER, DODEKERNEL will be lost. Export these users first with the binary loader (T\_USER, T\_GROUP, T\_GRP\_USR) and reload them after the upgrade. Table T\_DEFAULT should be migrated by the loader (import/overload) otherwise new defaults will be missing.

# Chapter 2

## Installation

### Prerequisites

The Upgrade tool runs on the following software:

- ❑ Server platforms
  - Windows 2003 Server
  - Unix (all platforms supported by Agile e6.0)
- ❑ Memory
  - The upgrade tool needs at least 512 MB of memory
- ❑ Databases
  - Oracle 8.1.7 / 9.2.0.4 or higher (Source database)
  - Oracle 10.1.4 (Target database)

The best performance is reached when installing the upgrade tool on your database server. It is also possible to work on any machine in the LAN. The machine should have at least 1GB memory.

To install the Upgrade Tool you need at least

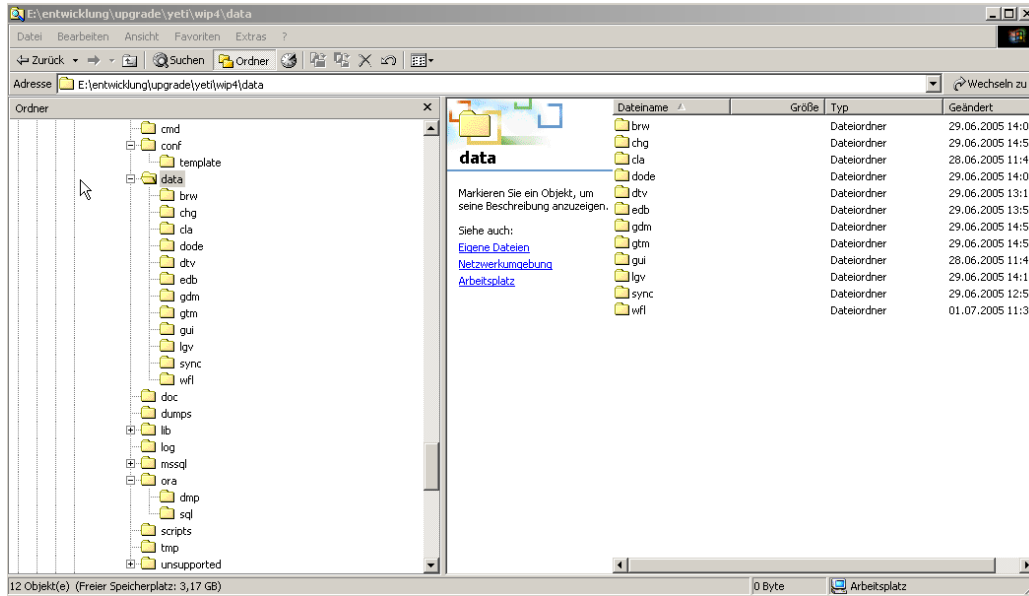
- ❑ 250 MB disk space for the software and generated log and data files

sufficient disc space to store copies of your production database and the reference dumps on your database.

### Installing the Upgrade Tool

1. Extract the software (upgrade.zip) to a folder *Upgrade* on your database server.

The following structure is created:



The file `upgrade\cmd\upg_env.cmd` (PC) or `upgrade/scripts/upg_env.sh` (Unix) contains the environment definition for your upgrade process.

2. Adapt the following environment definitions in `upg_env.sh` / `upg_env.cmd`:
  - **JAVA\_HOME**  
At least JRE 1.4.2 is required by upgrade tool.  
In the standard configuration of the file the JRE of the Agile e6.0 installation is used for that.
  - **ORACLE settings**  
Make that Oracle 10.1.4 environment is set before proceeding with upgrade.

To check the environment for UNIX execute the following commands

**env|grep ORA**

Output should look like this:

```
/usr/oracle> env|grep NLS
NLS_LANG=AMERICAN_AMERICA.WE8ISO8859P15
ORA_NLS33=/usr/oracle/product/10g_db/ocommon/nls/admin/data
/usr/oracle> env|grep ORA
ORACLE_BASE=/usr/oracle
ORACLE_HOME=/usr/oracle/product/10g_db
ORACLE_SID=TITAN
ORACLE_DOC=/usr/oracle/product/10g_db/doc
ORACLE_TERM=xterm
ORA_NLS33=/usr/oracle/product/10g_db/ocommon/nls/admin/data
/usr/oracle>
```

# Chapter 3

## Configuration of Upgrade Tool

*Chapter 3 provides the description of how to configure the upgrade tool . It includes the following chapters:*

- Check Database settings
- Prepare Environments
- Define Database Connections
- Define Parameters
- Configure control and log files
- Configure take over data from the production system
- Special Configuration: Classification

### Check database settings

#### Check database settings in Oracle

The Upgrade Tool needs a well-configured database to provide a good performance. The Oracle standard database settings are not sufficient to run the program within the stated time.

1. Check the Oracle Parameter and verify that at least the following minimum values are set in your database instance:
  - `db_cache_size`  $\geq$  200 000 000 (200MB)
  - `shared_pool_size`  $\geq$  100 000 000 100Mbytes
  - `log_buffer`  $\geq$  163840 3\*64 Kbytes

If database memory consumption is too small, adapt the values.

If you use the server parameter file **spfile** (like in the Agile e6 standard installation), execute the following commands to change the values of the initialization parameters.

- Login into sqlplus as user `sys`  
SQL>ALTER SYSTEM SET <parameter name>=>Value> SCOPE=BOTH

**Note:** Do not change the values of production systems. Make a copy of the initialization file and adapt the values.

- Read the Oracle online manuals and the Oracle10.1 installation manual from Agile in addition.

Oracle needs physical memory. If the system starts swapping or paging, the Oracle performance degrades or causes errors. Examine your free physical memory and prevent the OS from swapping.



Some UNIX systems have maximum values for shared memory. Refer to the installation instructions before changing any value.

**2. Check the SQL Net configuration (Oracle only).**

The network domain is part of different oracle settings. Please check if the domain is consistently used for the following settings:

- Global Database
- Service name
- Listener.ora
- Default domain name

**Global Database name**

- Login as user sys and check the global database name.

The name should contain the network domain.

```
Sqlplus <system>/<db_passowrd>@<db_service>  
SQL>select * from global_name;
```

Example

```
GLOBAL_NAME -----  
PLM.WORLD
```

The example uses the default network domain in world. Also possible are values like agile.agilesoft.com.

Change the global database name login to SQL plus and execute the following commands:

```
SQL>alter database global name plm.agile.agilesoft.com
```

**Service name**

Service name in the SQL net configuration file tnsnames.ora network in the directory \$ORACLE\_HOME/network/admin

The service name must also include the network domain. Please check the setting in the sqlnet configuration file tnsnames.ora

- Change to the directory \$ORACLE\_HOME/network/admin
- Open the file tnsnames ora and check if the service name is fully defined. That means the name contains the same network domain as the global database name.

```
PLM.WORLD =  
(DESCRIPTION =  
  (ADDRESS_LIST =  
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))  
  )  
  (CONNECT_DATA =  
    (SERVICE_NAME = PLM.WORLD)  
  )  
)
```

**listener .ora**

- ❑ Check if the global database name in the section SID\_List of the listener configuration file contains also the same fully qualified global database name.

```
SID_LIST_LISTENER_PLM =  
(SID_LIST =  
(SID_DESC =  
(GLOBAL_DBNAME = PLM.WORLD)  
(SID_NAME = plm)  
)  
)  
  
LISTENER_PLM =  
(DESCRIPTION =  
(ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))  
)
```

**Service name**

Default domain name in SQL net configuration file is sqlnet.ora

The default setting for the network domain in the sqlnet.ora file should be the same.

- ❑ Change to the directory \$ORACLE\_HOME/network/admin and Open the file sqlnet.ora and check the default domain settings.

```
names.default_domain = world
```

## Prepare environments

### Prepare reference environments

The upgrade tool needs the following database environments/users:

Source master	CADIM/EDB, axalant or Agile 5.x reference dump
Target master	Agile e6.0 reference dump
Customer	customer dump

A separate database user is needed for each environment.

1. Download the necessary reference dumps from Agile support website and unzip them in the upgrade/dumps directory.

Note: Do not change the names of the downloaded reference dump files. The dumps cannot be imported automatically if a different name for the dump file is used.

2. Check if the following table spaces (Oracle) or file groups (SQL Server) exist in your database
  - edb\_tmp
  - edb\_tmpidx
  - edb\_lob

If one of them does not exist they have to be created:

Oracle

- Change to the directory *upgrade/ora/sql*
- Adapt file names, paths and file size in the script *cre\_axa\_tbs.sql*
- Login as user *system* to *sqlplus* and execute *cre\_axa\_tbs.sql*
- *sqlplus system/<password>@agile*
- *SQL> @cre\_axa\_tbs.sql*

### 3. Import reference and customer dumps

Note: For importing the dumps, do not change the table space names, because the created table statements on tables containing a blob clause will fail if the original table spaces EDB, EDB\_IDX and EDB\_LOB do not exist

- Copy your customer dump file in the directory *upgrade/dumps*
- Rename the file to *<db\_user>.dmp* where *<db\_user>* is the user name of your customer dump (e.g. *customer.dmp*)
- Run *imp\_dmp.cmd* (PC) or *imp\_dmp.sh* (UNIX) to restore the original and target master databases (oracle and SQL Server) and the customer database (oracle only). To create your customer environment in SQL Server use Backup/Restore functionality of SQL Server.

### 4. Create statistics for all involved database schemas

- Check Language settings  
Because of an Oracle bug the setting for the environment variable *NLS\_LANG* must be *AMERICAN\_AMERICA.WE8ISO8859P15*. Otherwise statistics will not be computed correctly.
- Login as user with *dba* privilege and perform analyzing.  
*sqlplus> EXECUTE DBMS\_STATS.GATHER\_SCHEMA\_STATS ('<schema>', N);*  
N is the sample % for statistics collection (use 100%)

## Define Database Connections

The definition of the database connections is done in four steps:

- Source Master:  
Standard reference dump corresponding to the version of CADIM / axalant / Agile e-series customer dump
- Target Master :  
Standard reference dump for desired target version
- Customer:  
Database environment containing CADIM / axalant / Agile e5.x customer data
- Production Database:  
Database environment containing production CADIM / axalant / Agile e5.x . This connection is used as the source for taking over production data

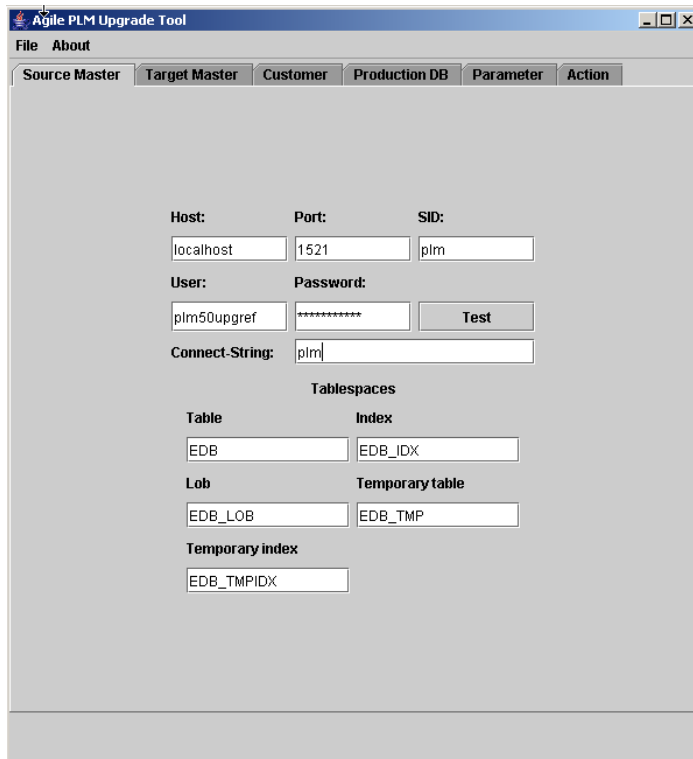
## Define Source Master/Target Master/Customer Connections

### 1. Start Upgrade Tool

**PC:** Run start\_upg.cmd

**UNIX:** Run start\_upg.sh

The following screen will be opened:



The screenshot shows the 'Agile PLM Upgrade Tool' window with a menu bar (File, About) and a tabbed interface. The 'Source Master' tab is selected. The form contains the following fields and sections:

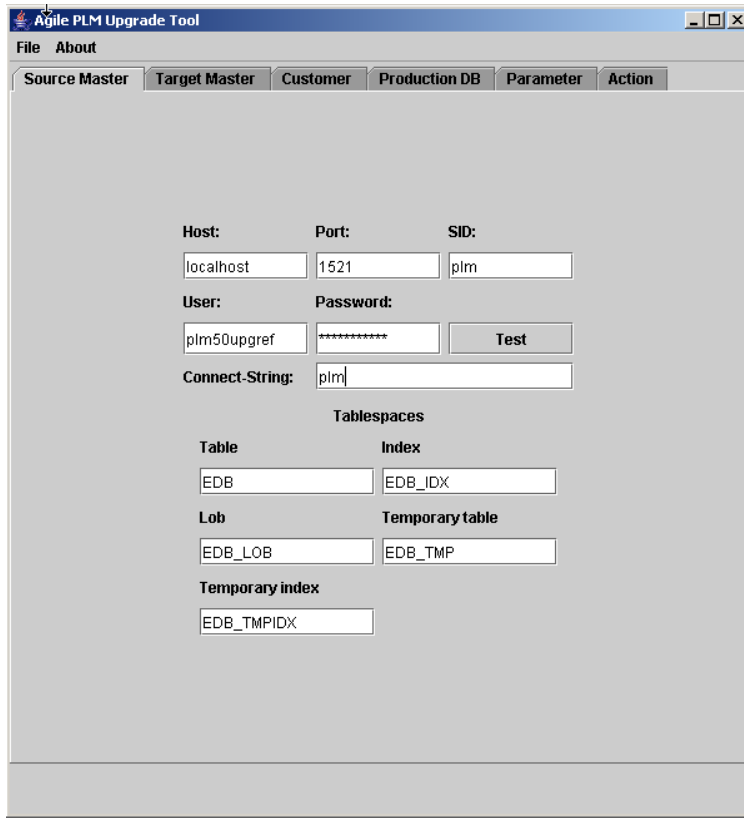
Host:	localhost	Port:	1521	SID:	plm
User:	plm50upgref	Password:	*****	<input type="button" value="Test"/>	
Connect-String:	plm				
<b>Tablespaces</b>					
<b>Table</b>	<b>Index</b>				
EDB	EDB_IDX				
<b>Lob</b>	<b>Temporary table</b>				
EDB_LOB	EDB_TMP				
<b>Temporary index</b>					
EDB_TMPIDX					

### 2. Select the respective tab

- Source Master
- Target Master
- Customer

Important: Make sure not to use a production database dump!

Do not change this connection to a different CADIM /axalant / Eigner PLM dump. The reference dumps are modified. Use the database dumps delivered with the Upgrade Tool. Only the reference dumps have the Agile e6.0 table format but CADIM/axalant/Eigner PLM contents. *Standard LGV models are dropped!*



3. Enter the following information:

Database selection	Select the desired version
Host	host name of database server
Port	port number of Oracle listener (default 1521 ) or SQL Server port number (default 1433)
SID	Oracle_SID (uppercase) or database name for SQL Server (lowercase)
User	database user name
Password	password of database user
Connection String	Service name, which is used to run SQL*PLUS commands on the machine the upgrade tool is installed on. Use fully qualified name including the network domain, f.e. plm.aile.agilesoft.com

Tablespaces:

Note: Name of used tablespaces (Oracle - uppercase) or file groups (SQL Server - lowercase)

Table	Default EDB <sup>*</sup>
-------	--------------------------

Index	Default EDB_IDX*
LOB	Default EDB_LOB*
Temporary table	Default EDB_TMP* (edb on SQL Server until axalant 2000 SP3 )
Temporary index	EDB_TMPIDX* (edb_idx on SQL Server until axalant 2000 Sp3)

4. Test the connection using the Button “TEST

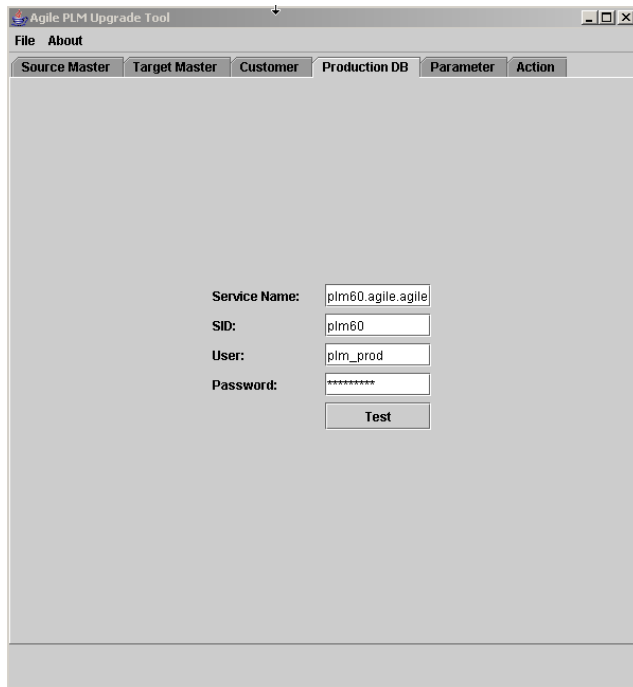
Note: The Java Connection can only be tested in Version 3.0. The Service Connection via SQL Net is not tested with the Version 3.0 of the Upgrade Tool.

**Define Production DB**

This is the database connection to the production system. This connection is used as source for the transfer of production data.

Compared to other connection definition only the service name of the sqlnet connection must be defined (defined in *tnsnames.ora* e.g. agile).

1. Select the tab *Production DB*



2. Make the following entries

Parameter Name	Description
Service Name	Oracle service name including network domain e. g. AGILE.AGILESOFT.COM. Service name must be defined in tnsnames.ora

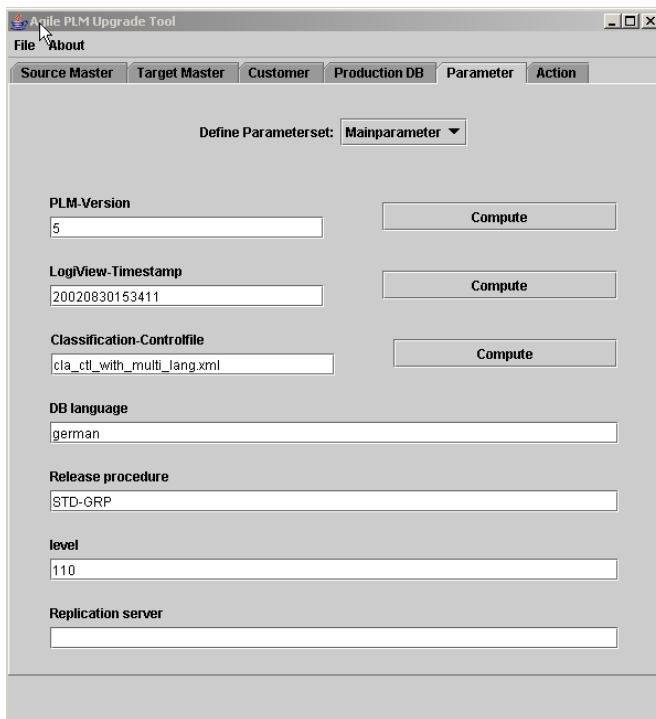
SID	Oracle_SID (uppercase) or database name for SQL Server (lowercase)
User	database user name
Password	password of database user

3. Test the connection using the Button “TEST

## Define Parameters

Additional parameters which are available in the file ApplicationParamter.xml can be viewed and edited.

1. Select the tab *Parameter*



2. Review and correct the entries if necessary and check the following table for valid entries.

With the “Compute” button, the right values can be determined. Always check the computed values.

Parameter name	Description
PLM-Version	<p>The customer dump version Following values are valid:</p> <p>1 = CADIB/EDB 2.3.x 2 = AXALANT SP1 3 = AXALANT SP2 4 = AXALANT SP3 5 = PLM 5.0</p>

Logiview Timestamp	<p>6 = Agile e6.0</p> <p>A Timestamp All logiview items with a change date after this time point will be deleted. You can adapt this value manually. Following values are possible:</p> <p>CADIM/EDB 2.3.2 – 19990329094555 CADIM/EDB 2.3#3 – 19990707174038 CADIM/EDB 2.3#4 – 19990707174038 CADIM/EDB 2.3#5 – 20000329161725 axalant2000 SP1 – 20001109140557 axalant2000 SP2 – 20010723102350 axalant2000 SP3 – 20011113092600 axa2000 SP3 PA1 – 20020808110309 Eigner PLM 5.0 - 20020830153411</p> <p>The version must correspond to your customer dump version</p>
Classification – Controlfile	<p>A file name of the control file for the customer dump in the present case Valid entries are:</p> <p>cla_ctl.xml (used for CADIM/EDB and no multi language fields for classification attributes c_letter and Class)</p> <p>cla_ctl_with_repl.xml used for CADIM/EDB and no multi language fields for classification attributes c_letter and Class, database replication is activated)</p> <p>cla_ctl_with_multi_lang.xml(axalant 2000 or higher ,c_letter_c_class defined as multi language fields)</p> <p>cla_ctl_with_multi_lang_repl.xml (axalant 2000 or higher ,c_letter_c_class defined as multi language fields; database replication is activated)</p>
Database Language	<p>Language for the database dump. This influences the migration of the classification data. Values: German, English Default: German</p>
Level	<p>Status, that is set during classification upgrade for records in the tables t_cla_dat (pool attributes), t_group_dat(classes)</p>
Replication server	<p>Null or a valid name of the database server</p> <p>should be used of an implemented database replication to the environment be migrated</p>

## Configure control and log files

A set of control and log files is defined for each upgrade step. The location of the files is stored in the main configuration file upgrade\conf\ApplicationParameter.xml.

### Control and log files for comparing/updating repository tables

For all actions (see **Action** tab) that compare table content (Create files) and change repository information (Perform Insert, Update, Delete) a common set of control and log files is used.

Parameter Name	Description	Example
Datadictionary	Description of data model, defining the objects and their relationships for upgrade.(all Modules)	upgrade\data\dtv\dtv DD.xml

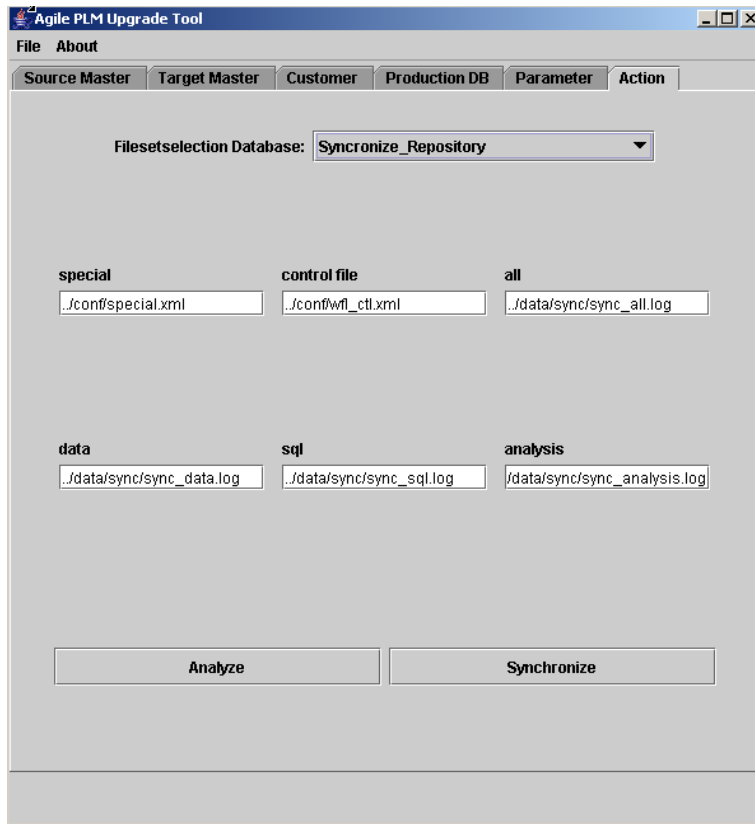


Special.xml	Contains information about a table that must be renamed. For DTV- Upgrade only (DTV Upgrade only)	upgrade\Conf\special.xml
Deletefile	Deleted records in xml format.(all Modules)	upgrade\Data\dtv\dtv_del.xml
Insertfile	New records in xml format.(all Modules)	upgrade\Data\dtv\dtv_ins.xml
Updatefile	Updated records in xml format.(all Modules)	upgrade\Data\dtv\dtv_upd.xml
Errorlog	Errors (all Modules)	upgrade\Data\dtv\dtv_err.xml
Infolog	Information (all modules)	upgrade\Data\dtv\dtv_info.xml
Customizing	All conflicts for specific columns, which can be not handled by the upgrade tool automatically are written to customizing.log (DTV-Upgrade only)	upgrade\Data\dtv\customizing.log
EdbID Replace	Generated during dtv upgrade, define the updates for the new foreign key references (based on new added EDB_ID's) (DTV-upgrade and synchronize Repository)	upgrade\Data\dtv\edb_id_replace.xml

Note: If necessary update the name and the location of the control and logfiles for the actions.

### Control and log file for Synchronizing repository

This step uses a set of specific control and log files. If the step is executed again, the log files will be saved and then overwritten. The saved log files extended with a consecutive number for every version ( e.g. sync\_all001.log — sync\_all002.log — sync\_all001.log)



Parameter Name	Description	Example
sql	All SQL statements for creating and altering database objects are logged. This file is created in step “synchronize repository”.	upgrade\data\sync\sync_sql.log
all	Store all log information. This file is created during step “synchronize repository”.	upgrade\data\sync\sync_all.log
data	Table definition. This file is created during step “synchronize repository”.	upgrade\data\sync\sync_data.log
special	XML file containing special definitions for repository upgrade like move of fields. Default values for the new mandatory columns.	upgrade\conf\special.xml
analysis	Analysis.log store inconsistencies between DataView table definition and physical tables. This file is created in the step ”analyze repository”	upgrade\data\sync\sync_analysis
Control file	Not used for upgrade to Agile e6.0	

Note: If necessary update the name and the location of the control and logfiles for the actions

## Configure special.xml for synchronizing repository

In the step Synchronize Repository, the following functionality is executed:

- Set static and dynamic default values for new mandatory columns or columns changed from null to **not null**
- Rename tables
- Move fields
- Change data type of a field

These functions need a specific configuration. This information is stored in `upgrade\conf\special.xml`. The delivery contains a preconfigured `special.xml` which define standard setting for all expected cases.

Very often the customer dump contains inconsistency, so that in the analyze mode the tool will add entries to the `special.xml` file. You then need to reviewed and adapted the configuration.

If the `special.xml` is damaged, copy a original `special.xml` file from the template directory (`upgrade\conf\template`) into the directory `Upgrade\Conf` and start the synchronization again.

### **Set Static and dynamic default values**

#### Static values

To set a static value the configuration looks like

```
<FieldDefault>
<FieldName>T_TRE_DAT.CUR_FLAG</FieldName>
<FieldType>S</FieldType>
<FieldSize>1</FieldSize>
<DefaultValue>
<Value>n</Value>
</DefaultValue>
</FieldDefault>
```

In this case the column `T_TRE_DAT_CUR_FLAG` is set to 'n'.

#### Dynamic values

The field values can be computed dynamically based on

- a Java function  
preconfigured function to use the number server to set values are available (see example)
- SQL Statement

#### Example:

sql statement is used to compute field value

```
<FieldDefault>
<FieldName>T_CTX_DAT.EDB_SEQ</FieldName>
<FieldType>I</FieldType>
<FieldSize>4</FieldSize>
<DefaultValue>
<Select>DISTINCT (SELECT COUNT(*) FROM T_CTX_DAT T WHERE T.C_ID &lt;=
thisRec.C_ID)*10</Select>
```

```
<Where>C_ID &gt; 0</Where>
</DefaultValue>
</FieldDefault>
```

### Example

Java function is used to compute field values (get a new number from the number server and fill in the value)

```
<FieldDefault>
<FieldName>T_MASTER_DOC.EDB_ID</FieldName>
<FieldType>I</FieldType>
<FieldSize>10</FieldSize>
  <DefaultValue>
    <Function>GetNewEDBID(EDBEDBID)</Function>
  </DefaultValue>
</FieldDefault>
```

### **Rename tables**

Specifies the old and the new name for tables.

If you have used DFM already, the following tables must be renamed (for upgrade from cadim to Agile 6 only).

- T\_EER\_SIT
- T\_EER\_SIT\_STR
- T\_EER\_SIT\_MED

```
<RenameTable>
<TableName>T_EER_SIT</TableName>
<NewTableName>T_DDM_SIT</NewTableName>
</RenameTable>
```

```
<RenameTable>
<TableName>T_EER_SIT_STR</TableName>
<NewTableName>T_DDM_SIT_STR</NewTableName>
</RenameTable>
```

### **Move fields**

This option allows to move a column of a table inclusive stored values to a new location. To move a field you have to specify:

- Source field (<table\_name>.<column\_name>)
- Target field (<table\_name>.<column\_name>)and
- Path (join condition between old and new table)

The sample configuration files show 3 different possibilities to move field values to a new location.

```
<!-- Example transfer from typetable to entitytable. -->
<MoveField>
  <SourceField>T_DOC_DRW.CRE_USER</SourceField>
  <Path>T_DOC_DRW.C_ID_2</Path>
  <Path>T_DOC_DAT.C_ID</Path>
  <DestField>T_DOC_DAT.CAX_CRE_SYSTEM</DestField>
</MoveField>
```

```

    <!-- Example transfer from entitytable to entitytable via
releationtable. -->
    <MoveField>
      <SourceField>T_MASTER_DAT.PART_ID</SourceField>
      <Path>T_MASTER_DAT.C_ID</Path>
      <Path>T_MASTER_DOC.C_ID_1</Path>
      <Path>T_MASTER_DOC.C_ID_2</Path>
      <Path>T_DOC_DAT.C_ID</Path>
      <DestField>T_DOC_DAT.CAX_CRE_SYSTEM</DestField>
    </MoveField>
  <!-- Example transfer in table. -->
  <MoveField>
    <SourceField>T_MASTER_DAT.PART_ID</SourceField>
    <DestField>T_MASTER_DAT.EDB_ICON</DestField>
  </MoveField>
</SpecialCases>

```

If you have a cax interface installed already, please check if one of the columns used to store cax specific information is defined as a document-type-table-column. These columns are now part of the standard axalant data-model and included in the document master table T\_DOC\_DAT.(migration from CADIM to Agile e6.0)

An example configuration file special\_move.xml containing definition of moved fields are stored in the template directory ...\\upgrade\\conf\\template.

### ***Change data type of a field***

The upgrade tool allows changing the type definition of columns. The following type changes are possible:

- ❑ Integer→String

If the value of a column for all records is null then also incompatible data type changes can be executed, for example STRING-->Integer

Cutting a string field is only possible if no record contains a longer value. Please check max length of stored values directly with SQL.

You have to replace “false” by “true” to confirm the change. The type definition “oldType” comes from the database; “newType” is the DataView definition. (T\_FIELD. C\_FORMAT)

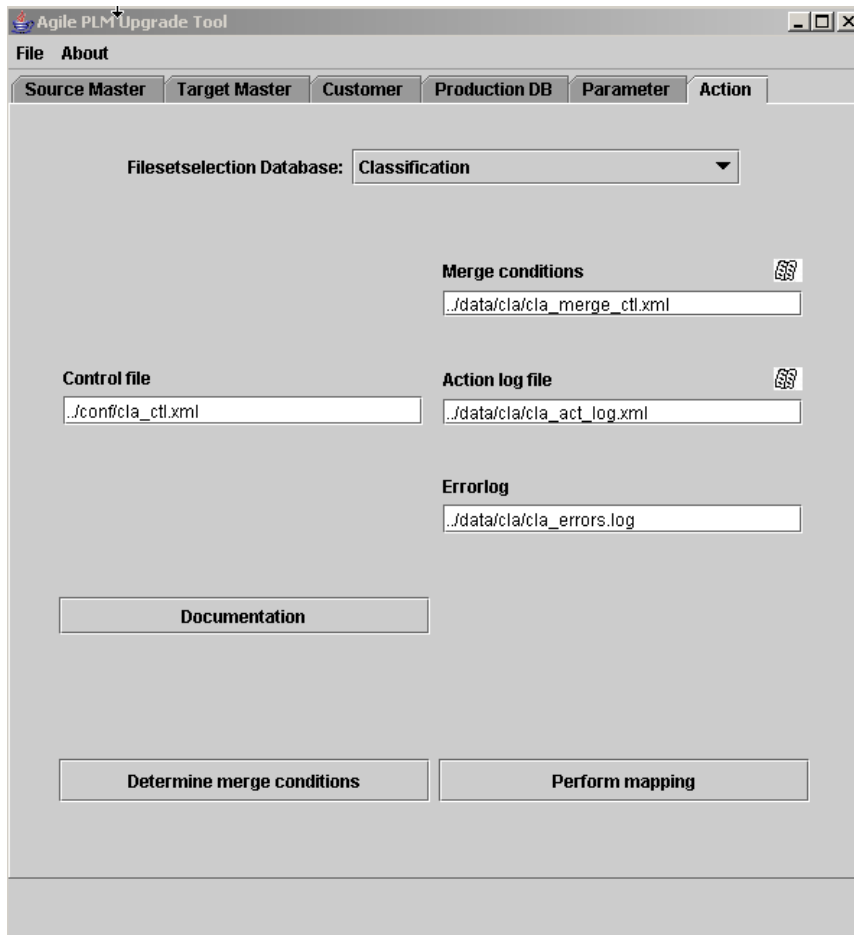
```

FieldChange>
  <FieldName>T_DOC_DAT.FOO</FieldName>
  <ConfirmChange oldType="S80.0" newType="S40.0">>false</ConfirmChange>
</FieldChange>

```

## Special Upgrade: Classification

The following control and log files are used for the classification



Parameter Name	Description	example
Control file	cla_ctl.xml defines the rules for the classification upgrade like data field mapping, merge condition, etc.	upgrade\conf\cla_ctl.xml
Merge conditions	The file merge_ctl.xml is created during the step determine merge conditions and stores the new pool attributes and the original class specific attributes.	upgrade\data\cla\cla_merge_ctl.xml
Action log file	All actions are written to the log file act_log.xml.	upgrade\data\cla\cla_act_log.xml
Errorlog	Is not used yet.	upgrade\data\cla\cla_errors.log

## Configuration file for Special replace.xml

This configuration files contains definition of sub strings in repository columns which should be replace by an other string

The file can be found in the directory ../conf/specialreplace.xml

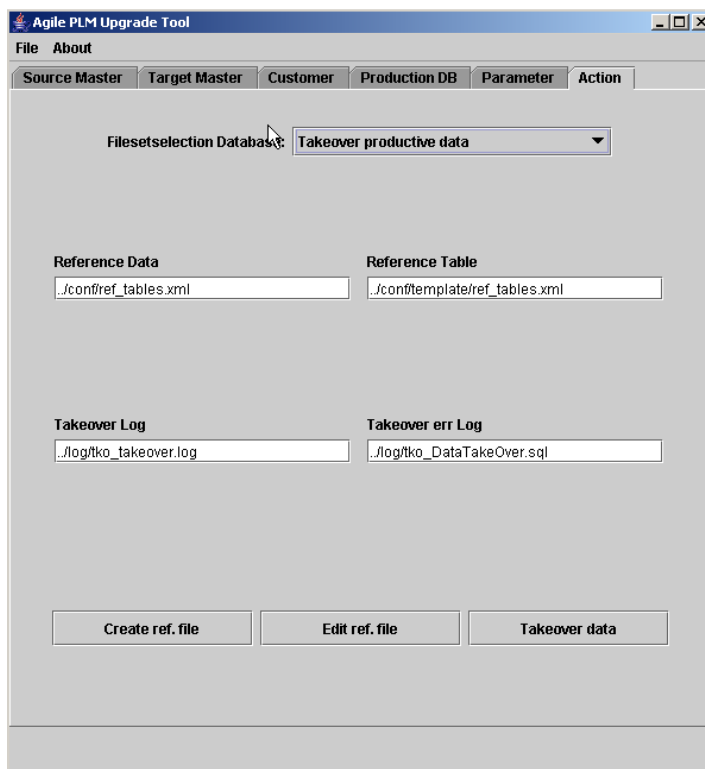
### Example

Update the strings T\_EER\_SIT with T\_DDM\_SIT in LGC Code

```
<?xml version="1.0" encoding="UTF-8"?>
<special>
  <replace>
    <table>LV_DT_PRC</table>
    <field>MAIN</field>
    <example>T_EER_SIT</example>
    <replacewith>T_DDM_SIT</replacewith>
  </replace>
</special>
```

## Configure take over data from the production system

Select the tab Takeover production data and review the configuration and Log files for this step



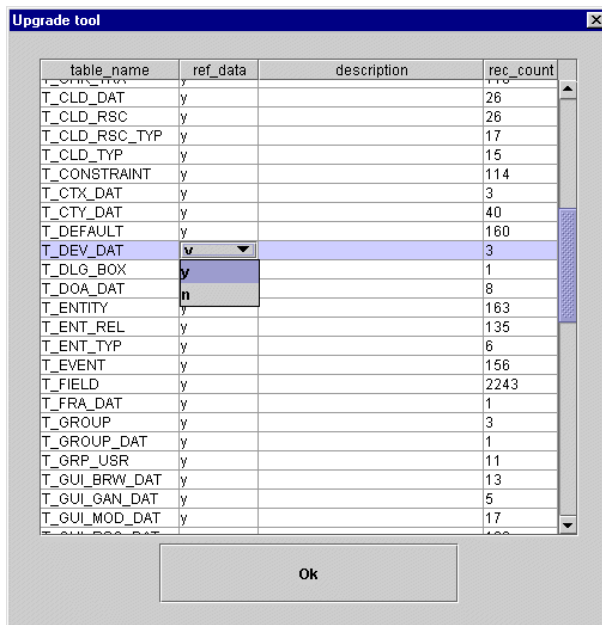
Parameter Name	Description	Example
Reference Data	Data Ref_data.xml defines which tables are proceeded during the step takeover production data (s. chapter “Take over of reference data”).	upgrade\conf\ref_tables.xml
Reference tables	Ref_tables.xml contains the default list of reference tables.	upgrade\con\templates\ref_tables.xml
Takeover Log	Information about all executed commands.	upgrade\log\tko_takeover.log
Takeover err Log	Information about occurred errors.	upgrade\log\tko_DataTakeover.sql

**Define reference tables**

1. Select folder **Takeover** in the Upgrade Tool and press the button **Create Ref File**.

The Upgrade Tool will connect to the production database, identify all tables and synchronize the information with the predefined list (ref\_tables). Only tables with data will be written to the file.

2. To adapt the list, press **Edit ref. file**



For each table you have to define, if it is a reference table. (ref\_data=y.)The called reference tables will be dropped in the customer dump and copied in from the production system using the connection to the production DB.

3. Select OK to save the XML file (*upgrade/data/ref\_data\_tab.xml*).

For detail information which tables should be copied please refer document Overview upgrade process.

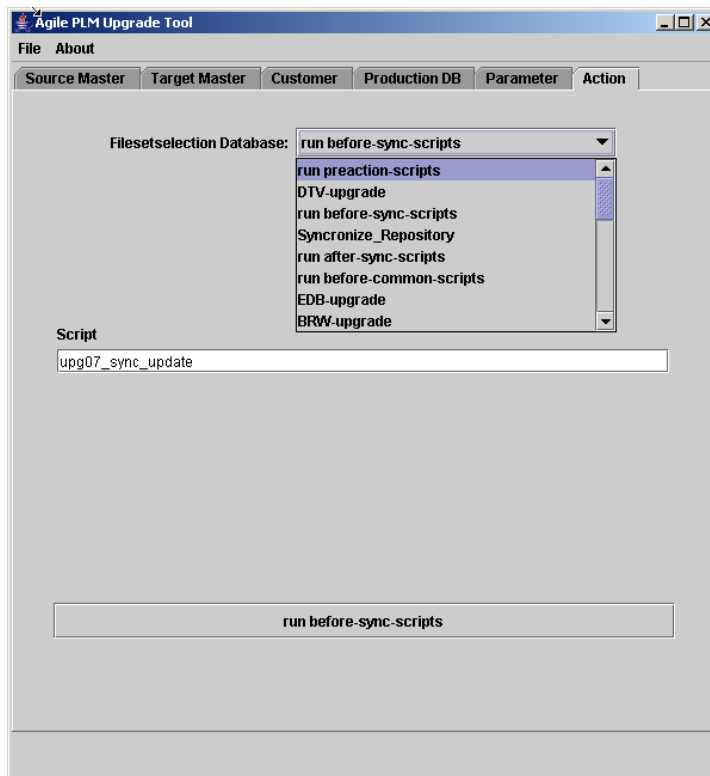


# Chapter 4

## Migration

### Performing Customizing Upgrade

This chapter describes how the Customizing Upgrade is executed interactively. There are different scripts available for selection under the **Action** tab which should be executed in the respective order.



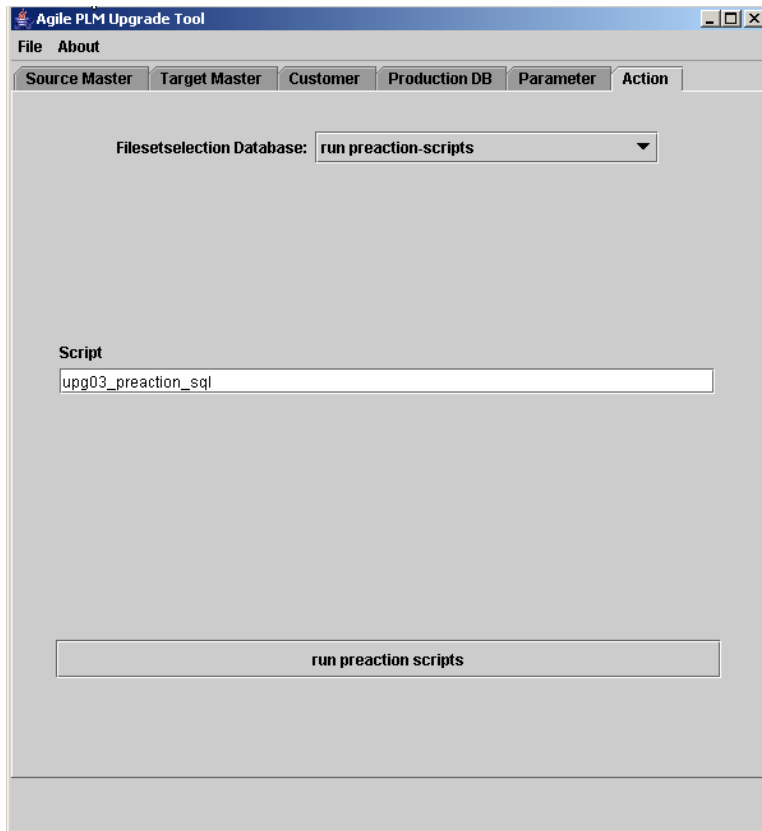
### Step Run preaction scripts

This script must be run before the DataView upgrade is run.

1. Select Run preaction scripts.

The Command script upg03\_preaction\_sql.cmd will be executed. This script executes a set of SQL scripts (depending on the customer dump version). See Annex file for further information.

2. Click the Run Preaction Scripts button



3. Check the created log files that will be stored in the upgrade/log directory.

The following log files are created

- Upg03\_preaction.sql  
This is the main sql script which includes the calls of other sql script.
- Log files are written for each executed sql script.  
The log files use the following naming convention  
upg03\_%<script name>.log

A complete list of SQL files which can be executed can be found in the annex

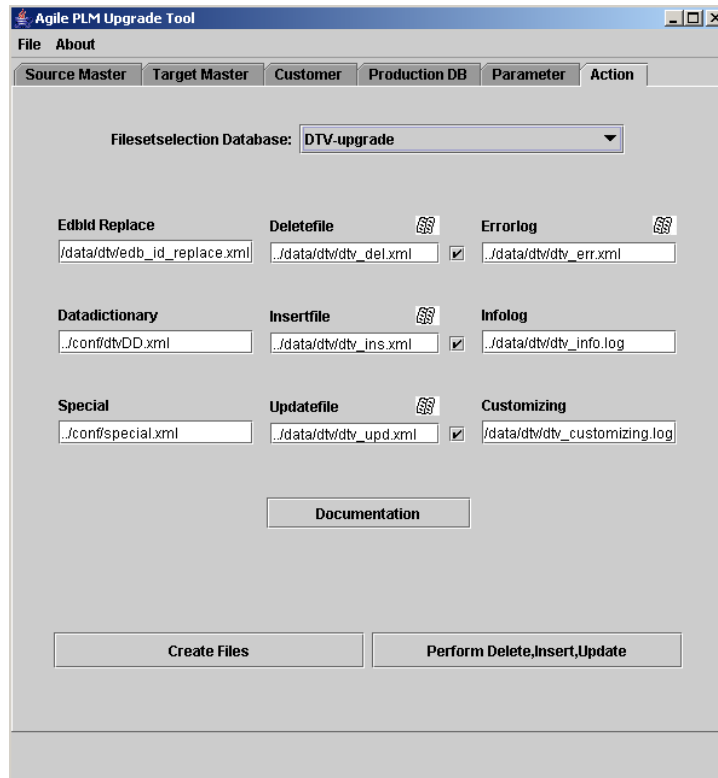
## Step DTV-upgrade

This script updates the DataView repository.

If you want to rename tables, you have to specify these tables in the file upgrade/conf/special.xml first. Refer to the Installation and Configuration Manual for further information.

1. Select DTV-upgrade
2. Click the **Create Files** button.

This step takes around 10 min - 4 hours.



3. Check log files in the directory *upgrade\data\dtv*.

Note: For better readability, it is possible to create HTML files from the XML log files.

4. Click the **Perform Delete, Insert, Update** button.

This step takes about 15 - 40 min.

5. Check the log files in the directory *upgrade\data\dtv*

If an error occurs, check the detail error information in *upgrade\log\errordetail.log* and see the error handling information in the Annex file.

### Adapt physic. table definition acc. to DataView table definition

The table definition in DataView is compared to the physical table structure in the database. SQL statements to create and alter database objects are generated and executed automatically. The adaptation of the physical data structure will consist of the following steps:

#### **Step Run before sync scripts**

1. Select run before-sync scripts.
2. Click the **Run before-sync-scripts** button.

The command script *upg07\_sync\_update.cmd* will be executed. This script executes a sql script.

3. Check the log files *upg07%.log* in then *upgrade\log* directory for possible errors.

**Step Synchronize Repository**

1. Select Synchronize Repository.
2. Click the **Analyze repository** button

If you run the program in the analyze mode you can see which statements the program will perform.

This creates a special file for defaults and special tasks.

If the program terminates the process and releases the connection because of a server error, drop the special.xml file in upgrade\conf directory and copy the preconfigured template from ..\upgrade\conf\template into the ..\upgrade\conf directory. Restart the process.

The data and log files are placed in the upgrade\data\sync directory. A detailed Description of the errors can be found in upgrade\log\errordetail.log

This step takes about 1-6 min.

3. Set defaults and configuration parameters in the file *special.xml* and save the changes.

Note Do not change or delete the default settings.

4. Adapt the following configurations

- Static and dynamic Default values for columns changed from null to not null
- Rename tables
- Move fields
- Change data type of a field

For detail description of the correct configuration, please refer the installation and configuration manual.

5. Click the **Synchronize Repository** button.

This script executes all changes in the database using the information from file special.xml. It takes about 3-20 min.

Data and log files are stored in directory upgrade\data\sync.

Check the file upgrade\log\errordetail.log for possible errors.

**Step after-sync-scripts**

1. Select Run After-sync-scripts

The command script upg08\_postactions will be executed. This script executes a set of sql scripts.

2. Check the results in the log file. All log files will be stored in the upgrade\log directory.

- Upg08\_postactions.sql  
This is the main sql script which includes the calls of other sql script
- Log files for every executed sql script. The log files uses the following naming convention upg08\_%<script name>.log

## Step Run before-common-scripts

1. Select Run before common scripts

The command script `upg09_common_get.cmd` will be executed. This script starts the sql script to save and delete standard LogiView Models.

2. Check log files in the directory **upgrade \log**

- `Upg09_common_get.sql`
- `Sql scripts upgrade\log\upg09%.log`

Note: If an error occurs during LogiView upgrade, restart the script and execute LGV Upgrade again.

## Step EDB-UPD

This step upgrades the Agile e6 Application Repository. It inserts and upgrades data in the application repository of Agile e6. The repository tables are grouped into different upgrade modules.

The following modules are necessary to migrate to Agile e6

- EDB-upgrade
- BRW-upgrade
- DODE-upgrade
- LGV-upgrade (Logiview)
- WFL-upgrade(Workflow),
- CHG-Upgrade (Change Management)
- Classification-upgrade(Classification)
- GDM-upgrade(Office Suite)

Note: This step needs to be execute for each module except Classification-upgrade.

1. Select the module e.g. EDB-upgrade
2. Click the **Create file** button.
3. Click the **Perform Insert, Update, Delete** button

These steps take about 20 min.

4. Check the log and data files found in the directory `upgrade\data\<module short name>`,  
For an error description see the chapter "Handling Of Errors" in the Annex.

## Special Upgrade: Classification

Depending on your source version of the Agile system execute on of the following:

- Migrate classification data from the ATT concept to the new pool concept ( necessary for migration from CADIM and axalant to Agile e6.0)
- Migrate changes in attribute inheritance (migration of all source version to Agile e6.0)

**Migrate class data from the ATT concept to the new pool concept**

1. Check parameters settings for the Classification Upgrade (for detailed information see Installation and Configuration manual)
2. Select **Classification-upgrade** to start the migration.
3. Select **Determine Merge conditions**  
As a result, the file merge\_ctl.xml is created. This file contains so-called merge groups. All attributes mapped onto the same pool attribute are in the same merge group.
4. Select **Execute Perform merge conditions**.  
The file merge\_ctl.xml is read. Based on this information all objects and attributes are created or updated.
5. Check the merge and log files.  
These files are located in upgrade\data\cla

**Migrate changes in attribute inheritance**

1. Select **Attribute Inheritance** to start the migration.
2. Click the button **Perform attribute inheritance to subclasses**.
3. Check log files in upgrade\data\cla

**Transfer data from production system**

After having performed the Customizing Upgrade, you can now start with transferring the reference data from the production system. Reference data are all tables except customized tables.

Identify the reference data for your system and synchronize the versions. For details which tables must be copied, see chapter Overview Upgrade process

The data is transferred with the following steps:

- Define reference table
- Perform transfer
- Synchronize repository
- Run after-takeover upgrade steps
- Upgrade Classification

**Define reference tables**

1. Select folder **Takeover** in the Upgrade Tool and press the button **Create Ref File**.  
The Upgrade Tool will connect to the production database, identify all tables and synchronize the information with the predefined list (ref\_tables). Only tables with data will be written to the file.

Note: Depending on your specific configuration, adapt the preconfigured transfer type.

For detail information on which tables should be copied, refer to the chapter Upgrade process

2. To adapt the list, press **Edit ref. file**

table_name	ref_data	description	rec_count
T_DOC_ORG	y		2
T_DOC_PRS	y		4
T_DOC_STR	y		256889
T_DOC_TXT	y		62876
T_DOC_VER	y		95898
T_DSB_DAT	y	EDB-DISTRIB Di...	147
T_DSB_ELM	y		2
T_DSB_STR	y		2
T_DTO_FNC	n	Classfunctions	2
T_ENTITY	n	Entity Data	363
T_ENT_REL	n		208
T_ENT_TYP	n		23
T_EVENT	n	Event Data	114
T_EWS_DAT	y	EDB-WRK-SOL ...	2
T_FIELD	n	Field Data	5645
T_FILE_DAT	y	File FileServer Fil...	296642
T_FIL_STORE	y		296565
T_FIL_STR	y		48
T_FRA_DAT	y	EDB-FRAME Dra...	527
T_FUNCTION	n	functions	28
T_GROUP	n	Group Data	103
T_GRP_ART	y		18954
T_GRP_SPR	y		69
T_GRP_USR	y		2771

Ok

Note you have to define, if it is a reference table for each table.

3. Select OK to save the XML file (*upgrade/data/ref\_data\_tab.xml*).

### Perform transfer

1. Save your customer Agile e6.0 dump.

Oracle: Run the script *exp\_dmp* in the *upgrade/scripts* directory (*upgrade/cmd* on windows machine).

2. Press the Takeover button.

The tables containing non-repository information are dropped in your customers dump. The tables will be copied from the defined production environment (which is defined in the production DB) into you customer dump.

3. Check the log files in *upgrade\*
  - *Tko:\_takeover.log*

### Synchronize repository

1. Adapt the physical table definition according to the table definition in Data View

The table structure of the newly copied table will be adapted to an Agile e6.0 table structure.

- AFTER TAKEOVER:run before-sync-scripts  
The command script *upg07\_sync\_update.cmd* will be executed.  
Check log files *upg07%.log* in the *upgrade\log* directory

- AFTER TAKEOVER >Synchronize\_Repository.  
Data and log files will be written into the directory upgrade\data\sync  
Check the file upgrade\log\errordetail.log for errors.
- AFTER TAKEOVER > Run after-sync scripts

### **Script Run after takeover**

1. Select Run after takeover scripts

The command script upg15\_prod3\_postaction.cmd will be executed

2. Check log files upg15%.log in the directory in \upgrade\log

The complete list of sql scripts is described in the annex

3. Recreate own customer specific views, packages, procedures and triggers

If you have implemented own views, packages, procedures or triggers please executes the sql scripts to create/replace this objects in the database, so that they are valid.

### **Special Upgrade: Classification**

Depending on your processes you have to execute one of the following possibilities.

- Upgrade classification lists only
- Upgrade complete classification including classes and attributes

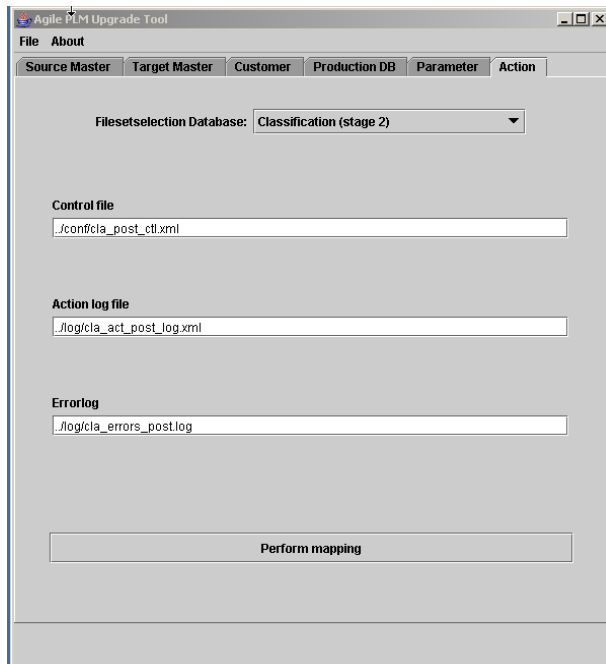
Please see upgrade process overview for more information which option should be used.

### **Upgrade classification lists only**

1. Select Classification (stage 2).

It is required that the attributes are already migrated and so that only the classification lists will be updated.





2. Click **Perform mapping**.

### Upgrade complete classification

If the customer has created new classes and attributes in the production system after the customization upgrade also classes, attributes and domain values must be copied and migrated

#### ***Migrate classification data from the ATT concept to the new pool concept***

Note: Check parameters settings for the classification Upgrade ( for detailed information see Installation and Configuration manual)

1. To start the migration please select action > Fileselection database > Classification
2. Click the Execute Determine Merge conditions button.

As result, the file merge\_ctl. xml is created. This file contains so-called merge groups. All attributes mapped onto the same pool attribute are in the same merge group.

3. Click the Execute Perform merge conditions button.

The file merge\_ctl. xml is read. Based on this information all objects and attributes are created or updated.

4. Check the merge and log files.

These files are located in upgrade\data\cla

#### ***Migrate changes in attribute inheritance***

1. To start the migration, select action Filesetselection Database > Classification attributeinheritance
2. Check log files in upgrade\data\cla

# Chapter 5

## Annex

### Convert XML files to HTML

You can convert the XML files to HTML and view them with a browser. The batch file `xml2html.bat` creates HTML files for insert, update and delete. This function is available on Windows platforms only.

Note: You need memory for approximately sixth times the XML file size!

Run `xml2html` with one of the following parameter values which specify for module the HTML file will be created.

The `html-xml` converter will create one file for insert, update and delete for each table.

The batch job will run several hours. The `jre` allocates 512MB of memory. You can adjust the memory allocation by editing the file

- `Upgrade\cmd\upg_env.cmd` (PC) and
- `Upgrade/scripts/upg_env.sh`

### Directory structure

Directory	Description
<code>cmd</code>	Windows 2000 shell scripts of the upgrade tool
<code>conf</code>	Configuration xml files
<code>conf\template</code>	Some templates of xml configuration files. These files are not used by the upgrade tool. The only Exception is the file <code>ref_tables.xml</code> . It will be read by the tool to recreate <code>/conf/ref_tables.xml</code>
<code>data</code>	This directory contains several subdirectories, each for a module – like BRW, EDB etc. For each module delete, insert and update xml files are created. After performing of these operations on the customer database, error xml file will be written. Additionally html files generated for a module will be saved here. A file <code>customizing.log</code> in this directory contains conflicts caused by customizing of the original dump
<code>data\dtv</code>	Dataview Upgrade files as described above are stored here. Please read carefully the file <code>customizing.log</code> because it contains user-exit conflicts.
<code>data\sync</code>	log files of the synchronize repository upgrade step are stored in this directory
<code>doc</code>	Upgrade tool documentation
<code>dumps</code>	Database dumps can be stored here. Dumps, which are imported / exported by shell scripts have to be stored in this directory
<code>lib</code>	Upgrade tool java executables

log	Log files of all sql scripts and common application log files
mssql\sql	MS Sql Server sql-scripts
ora\sql	Oracle sql-scripts
scripts	Unix / linux shell scripts of the upgrade tool

## Shell scripts / Log files

Here is a short description of all the shell scripts included in upgrade download package.

Shell script	Description	Called SQL scripts
chown_mssql.cmd	A script to change the database owner in MS Sql Server 2000.	--
convert_it.cmd	Convert XML data file for a single table to HTML files. This script is called from xml2html.cmd.	--
exp_dmp.cmd	This script helps you to export oracle database to a dump file.	--
imp_dmp.cmd	This script helps you to import oracle dump files.	--
missing_f.cmd	Create missing file groups in MS SQL Server.	--
preaction_template.cmd	This file is for upgrade internal use only!. It is used as a template for creating the file preaction.cmd, which is needed if one or all upgrade steps are not executed within the standard graphical user interface.	--
start_upg.cmd	Start upgrade user interface.	--
upg01_pre_cleanlog.cmd	Cleanup all log files within current upgrade project.	--
upg02_setup.cmd	Quick check of database connections – this can be performed in the standard user interface too.	--
upg03_preaction_sql.cmd	Run several SQL scripts depending on source, target and customer product versions. Called SQL scripts	source: TRUNC_LVTABS.SQL > 03_PREACTION_SQL.LOG source: GRANT_SELECT_T_CONSTRAINT.SQL source

Shell script	Description	Called SQL scripts
	are saved in the file 03_preaction_sql.log	03_14_GRANT_SELECT_T_CONSTRAINT.LOG source: CRE_REP_EDB.SQL source 03_12_CRE_REP_EDB.LOG target: GRANT_SELECT_T_CONSTRAINT.SQL > 03_13_GRANT_SELECT_T_CONSTRAINT.LOG  CRE_REP_EDB.SQL > 03_01_CRE_REP_EDB.LOG CLEANUP_C_ID_NULL.SQL > 03_03_CLEANUP_C_ID_NULL.LOG ANA_LV.SQL > 03_04_ANA_LV.LOG  <=CADIM: ORA3-4.SQL > 03_05_ORA3-4.LOG <=AXA SP 1: PST10P2TOP3.SQL > 03_06_PST10P2TOP3.LOG <=AXA SP 1: ORA403-404.SQL > 03_07_ORA403-404.LOG <=AXA SP 1: AXASP1_TO_SP2.SQL > 03_08_AXASP1_TO_SP2.LOG <=AXA SP 2: GDM_UPD_MAS_FLD_ORA.SQL > 03_09_GDM_UPD_MAS_FLD_ORA.LOG <=AXA SP 3: DTV405-406.SQL > 03_10_DTV405-406.LOG <=AXA SP 3: UPD_T_SELECTION.SQL > 03_11_UPD_T_SELECTION.LOG
upg04_dtv_get.cmd	Get XML files for the step “DTV-Upgrade” in shell mode.	--
upg05_dtv_update.cmd	Proceed XML files for the step “DTV-Upgrade” in shell mode.	--
upg06_sync_get.cmd	Run the step “Analyze repository” in shell mode.	
upg07_sync_update.cmd	Run the step “Synchronize repository” in shell mode. Called SQL scripts are saved in the file 07_sync_update.log	<= PLM5.x: before_sync.sql > 07_01_before_sync.log
upg08_postaction.cmd	Run some SQL scripts, which are necessary after performing “Synchronize repository” upgrade step. Called SQL scripts are saved in the file 08_postaction.log	cleanup.sql > 08_02_cleanup.log cre_rep_edb.sql > 08_01_cre_rep_edb.log
upg09_commo	Generate XML files for several	del_and_save_lvmodel.sql > 09_01_del_and_save_lvmodel.log

Shell script	Description	Called SQL scripts
n_get.cmd	upgrade steps, one for each common PLM module. Called SQL scripts are saved in the file 09_common_get.log	
upg10_common_update.cmd	Proceed common PLM modules XML files. Called SQL scripts are saved in the file 10_common_update.log	<=PLM5.x: edb_explorerer.sql > 10_edb_explorerer.log <=PLM5.x: levind_in_stalut.sql > 10_levind_in_stalut.log
upg11_cla.cmd	Upgrade PLM Classification.	--
upg13_prod1_takeover.cmd	Start a user interface to proceed with "Takeover production data". Scripts upg14_prod2_rep_update and upg15_prod3_postaction have to be executed after that.	--
upg14_prod2_rep_update.cmd	Synchronize repository (this script includes all necessary preaction and postaction calls).	SQL Files of upg07_sync_update.cmd and upg08_postaction.cmd are called again
upg15_prod3_postaction.cmd	Run SQL scripts that are needed to be executed after takeover step.	get_compile_view.sql > post_prod_compile_view.log get_compile_proc.sql > post_prod_compile_proc.log get_compile_trigger.sql > post_prod_compile_trigger.sql.log get_numvalue.sql > post_prod_numvalue.log
upg_env.cmd	Common upgrade settings, like Java, JRE, Path, etc.	--
xml2drop.cmd	Generates ora/ref_data_tab.par and ora/ref_data_tab_drop.sql files for manually import/export of production tables. You have to configure which tables are relevant for the takeover step before.	--
xml2html.cmd	Converts generated XML files to HTML format for a module. Example: "xml2html.cmd dtv" Or "xml2html.cmd all"	--

## SQL scripts

Here is a short description of SQL scripts delivered with the Agile Upgrade tool. All Script executions creates a log file in the upgrade/log directory which are named like the script itself with a prefix like pre\_cst – which means it is a preaction script and it was executed on the customer dump.

ana_lv.sql	Analyze LogiView Content in the customer dump. Please control the log file of this script as described in the manual.
axasp1_to_sp2.sql	Upgrade axalant sp1 to axalant sp2.
before_sync.sql	This script has to be executed before running the step "Synchronize Repository". It is done by default with the standard upgrade configuration. It prepares the Table T_STA_LUT and drops triggers, because otherwise it's impossible to insert rows in the involved tables.
cleanup.sql	This script cleans up some dump content and is executed automatically after the step "Synchronize Repository".
cleanup_c_id_null.sql	This script cleans up some inconsistencies in the customer dump (like rows with negative C_ID values). It must be executed before DTV-upgrade.
cre_plm_tbs.sql	Creates missing oracle tablespaces.
cre_plm_usr.sql	Create a database user. This script need 2 Parameters: username and password
cre_rep_edb.sql	Create all schema object (tables, views, indexes, Packages, Triggers, Sequences etc.) and insert number server rows, the most of them already exist, so a lot of errors will be logged after executing this script.
del_and_save_lvmodel.sql	Delete standard LogiView content and save customized models with a prefix "SAVE-".
del_dtv.sql	Truncate all internal DTV-tables.
dtv405-406.sql	Preaction script, has to be executed before DTV-upgrade for customer dump version <= axalant SP3.
edb_explorer.sql	Converts DTV explorer to Agile e6.0 EDB-explorer. This step is executed once after common modules upgrade.
gdm_upd_mas fld_ora.sql	Preaction script has to be executed before DTV-upgrade for customer dump version <= axalant SP2.
getoradrop.sql	Get script "dropall.sql" which cleans up a complete database schema.
get_compile_proc.sql	Generates a script to recompile all packages/functions.
get_compile_trigger.sql	Generates a script to recompile all triggers.
get_compile_view.sql	Generates a script to recompile all views.
get_numvalue.sql	This script is executed in the production database and generates a file named "set_numvalue.sql" after takeover step. This file updates number server values in the customer database.

get_rebuildidx.sql	This script generates a file named “ rebuildidx.sql ” to rebuild all indexes in a right tablespace of a schema. It has 5 parameters for tablespaces: EDB EDB_IDX EDB_LOB EDB_TMP EDB_TMPIDX
grant_select_t_constraint.sql	This script grants a select on table T_CONSTRAINT for the customer Database. This permission is needed for constraint conversion.
levind_in_stalut.sql	This script is called automatically after “Synchronize Repository” and converts records in the table T_STA_LUT. It is needed only for upgrades form <= Eigner PLM to >= Agile e6.0.
ora3-4.sql	This preaction-script is called automatically if the customer dump is a CADIM dump.
ora403-404.sql	Preaction script, has to be executed before DTV-upgrade for customer dump version <= axalant SP1.
Pst10P2ToP3.sql	Preaction script, has to be executed before DTV-upgrade for customer dump version <= axalant SP1.
reference_tabs.sql	Is used by shell script del2xml. It creates an oracle parameter file for production data tables export.
trunc_lvtabs.sql	Truncate all LogiView tables. This script is executed on reference dumps only!
update_customers_UIC.sql	This script has to be executed on the customer dump before proceeding with upgrade
upd_t_selection.sql	This script is a workaround for incompatible changes for Table T_SELECTION in Eigner PLM5.0 This script is already executed on all reference dumps delivered with Agile Upgrade Tool. It will be automatically executed id necessary in the step “preaction-scripts”

## Folders Contents “upgrade/conf” and “upgrade/conf/template”

### Contents of the folder “upgrade/conf”

ApplicationParameter.xml	Global application configuration file
brwDD.xml	Configuration file for upgrade module BRW (Explorer)
chgDD.xml	configuration file for upgrade module CHG (Change Management)
wflDD.xml	configuration file for upgrade module WFL (Workflow)
dodeDD.xml	configuration file for upgrade module DODE

	(Print Studio)
dtvDD.xml	configuration file for upgrade module DTV (Dataview Repository)
edbDD.xml	configuration file for upgrade module EDB (Agile PLM configuration)
gdmDD.xml	configuration file for upgrade module GDM (Office integration)
gtmDD.xml	configuration file for upgrade module GTM
lgyDD.xml	configuration file for upgrade module LGV (LogiView)
special.xml	Configuration file for the step “Synchronize Repository”
specialreplace.xml	A sample configuration file for special replace cases
ref_tables.xml	Configuration file for the upgrade step “Takeover production data”
wfl_ctl.xml	Configuration file for Workflow mapping
cla_ctl.xml	Configuration file for Classification Upgrade (Axalant 2000 to PLM5.x)
cla_post_ctl.xml	Configuration file for Classification Upgrade (Axalant 2000 to PLM5.x)
insert.xsl, delete.xsl, update.xsl, upgrade.xsl	XSL-Stylesheet for converting XML control files to HTML, used by xml2html.cmd script
ref_data_tab_drop.xsl	XSL-Stylesheet for generating SQL script, which drops production data tables in the customer dump. This style sheet is used by xml2drop.cmd
ref_data_tab_par.xsl	XSL-Stylesheet for generating table list clause for oracle EXP command, which can be use alternatively to transfer production data tables from production database. This style sheet is used by xml2drop.cmd
cla_stl.xsl	XSL-Stylesheet for Configuration file for Classification Upgrade (Axalant 2000 to PLM5.x), which generates a HTML output of performed mapping operations



dtv_dd.dtd	Document type definition file for module control files
------------	--------------------------------------------------------

### Contents of the folder “upgrade/conf/template”

ApplicationParameter.xml	Upgrade tool settings file with standard values. Copy this file to upgrade/conf to reset the application settings.
cla_ctl_with_multi_lang.xml	Example for the classification control file (with multi-language definition of the attributes).
cla_ctl_with_multi_lang_repl.xml	Example for the classification control file (with multi language definition of the attributes and additional fields for database replication).
cla_ctl_with_repl.xml	Example for the classification control file (with additional fields for database replication).
special_move.xml	Examples for the special case with table field moving.
special_rename.xml	Examples for the special case with table field renaming.
Special.xml	Default template.
specialreplace.xml	Examples for special replace cases.

### Configuration parameters

PLM-Version	The customer dump version Following values are valid: 1 = CADIB/EDB 2.3.x 2 = AXALANT SP1 3 = AXALANT SP2 4 = AXALANT SP3 5 = PLM 5.0 6 = Agile e6.0
Logiview Timestamp	A Timestamp All logiview items with a change date after this time point will be deleted. You can adapt this value manually. Following values are possible: CADIM/EDB 2.3.2 – 19990329094555 CADIM/EDB 2.3#3 – 19990707174038 CADIM/EDB 2.3#4 – 19990707174038

	CADIM/EDB 2.3#5 – 20000329161725 axalant2000 SP1 – 20001109140557 axalant2000 SP2 – 20010723102350 axalant2000 SP3 – 20011113092600 axa2000 SP3 PA1 – 20020808110309 Eigner PLM 5.0 - 20020830153411
Classification – Controlfile	A file name of the control file for the customer dump in the present case Valid entries are: cla_ctl.xml cla_ctl_with_multi_lang.xml cla_ctl_with_multi_lang_repl.xml cla_ctl_with_repl.xml
Database Language	Language for the database dump. This influences the migration of the classification date. Values: German, English Default: German
Level	Status that is set during classification upgrade for records in the tables t_cla_dat (pool attributes), t_group_dat(classes)
Replication server	The valid name of the database server should in case of an implemented database replication to the environment be migrated.

## Migration Rules

Standard rules are available for insert, update und delete and these rules are verified during the comparison of the table contents.

They can be overwritten by special definitions.

### Standard Rules for Delete

Data records deleted in the standard are also deleted in the customer dump.

Customer-specific dump	EignerPLM	CUSTOMER	Action
Selection	-	+	Delete

### Standard Rules for Update

Data records existing in CADIM/EDB or axalant that were deleted in the customer dump are not re-created. Existing data records are updated.

The standard changes overwrite the customer changes.

Customer-specific dump	EignerPLM	CUSTOMER	Action
Selection	+	+	Update

Special rules apply on field level to protect customer-specific changes.

## Standard Rules for Insert

Data records not existing in CADIM/EDB or axalant or in the customer dump are added.

Customer-specific dump	Agile e6.0	CUSTOMER	Action
Selection	-	-	

## Special Rules

Customer changes that will not be overwritten by standard changes are:

- Field defaults and check strings
- Customizings hints for fields containing user exits
- Special handling for mask components
- Replacements of strings ... (-> specialreplace.xml)