Oracle® Agile Engineering Data Management

Installation and Administration Manual for Agile e6.1.1 Batch Client

Part No. E15618-01

August 2009

Copyright and Trademarks

CONTENTS

Preface

The Oracle documentation set includes Adobe® Acrobat™ PDF files. The Oracle Technology Network (OTN) Web site (http://www.oracle.com/technology/documentation/agile.html) contains the latest versions of the Oracle Agile EDM PDF files. You can view or download these manuals from the Web site, or you can ask your Agile administrator if there is an Oracle Documentation folder available on your network from which you can access the documentation (PDF) files.

Note	To read the PDF files, you must use the free Adobe Acrobat Reader™ version 7.0 or later. This program can be downloaded from the Adobe Web site (http://www.adobe.com).

Note	Before calling Agile Support about a problem with an Oracle Agile EDM manual, please have the full part number, which is located on the title page.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Readme

Any last-minute information about Oracle Agile EDM can be found in the Release Notes file on the Oracle Technology Network (OTN) Web site (http://www.oracle.com/technology/documentation/agile_eseries.html)

Agile Training Aids

Go to the Oracle University Web page (http://www.oracle.com/education/chooser/selectcountry_new.html) for more information on Agile Training offerings.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Chapter 1

Introduction

The "old" EDB_BATCH mechanism of the PLM server is no more recommended and the Batch Client replaces the "old" batch solution of the PLM server.

The Batch Client allows to run batch scenarios implemented in Groovy to support the typical batch use cases.

The Batch Client supports two different modes:

- **Scenario Mode** In the scenario mode the Batch Client starts a defined scenario which interacts with the PLM Server to execute the batch use case.
- **ECI Server Mode** In this mode the Batch Client provides an ECI tunnel to the PLM Server and standard features like file access to the File Server.

For detailed information refer to the corresponding use case chapters.

Note	The "old" batch solution is still supported but you must configure the security level (see Settings) to enable it.

Chapter 2

Hardware and Software Requirements

The Batch Client is available for all server platforms of the e6.1.1 release.

An installer for the Batch Client is not available. A manual installation and configuration is necessary.

Chapter 3

Installation and Configuration

Installation

The Batch Client is delivered as a ZIP archive.

The ZIP archive is contained in the *batchclient.zip* installation package located in the ***packages*** folder of the installation medium.

1. Copy the Batch Client installation package (*batchclient.zip*) from the ***packages*** folder of the installation medium into a temporary directory

   Example:

   ```
   copy y:\packages\batchclient.zip c:\temp
   ```

2. Extract the installation package, this unpacks the Batch Client bundle ZIP archive into the ***axalant/bin/java*** sub directory of your temporary directory.

   Example:

   ```
   unzip c:\temp\batchclient.zip -d c:\temp
   ```

3. Extract the Batch Client bundle ZIP archive into the installation directory (Example: C:\Program Files). The Batch Client files are unpacked into the ***batchclient*** directory which is automatically created.

   Example:

   ```
   unzip c:\temp\axalant\bin\java\batchclient.zip -d "c:\Program
   Files\Agile_e6"
   ```

The Batch Client bundle contains the following directories:

```
 -> Batch Client Root        (C:\Program Files\Agile_e6\batchclient)
  -> axalant
     -> bin
       -> hppa-hp-hpux11.23    (HPUX FMS-Client binary)
       -> ia32-linux-sles10    (Linux FMS-Client binary)
       -> intel-ms-nt5.0       (Windows 32 bit FMS-Client binaries)
       -> java                 (Java archives)
       -> rs6000-ibm-aix5.3    (IBM FMS-Client binary)
       -> sparc-sun-solaris10  (Sun FMS-Client binary)
     -> cmd                    (Windows scripts directory)
     -> dmp                    (Example loader files)
     -> ini                    (Configuration files)
     -> scripts                (UNIX scripts directory)
  -> examples                  (Example batch files)
  -> ext
```

```
    -> bin
      -> hppa-hp-hpux11.23    (external binaries)
      -> ia32-linux-sles10    (external binaries)
      -> intel-ms-nt5.0       (external binaries)
      -> java                 (external Java archives)
      -> rs6000-ibm-aix5.3    (external binaries)
      -> sparc-sun-solaris10  (external binaries)
  -> tmp                    (Logging directory)
```

Configuration

The Java environment has to be set to a Java 5 or higher version. For the MS Windows Batch Client adapt the **BatchClient.cmd** shell script.

Example:

```
set JAVA_HOME="C:\Program Files\Java\jre1.5.0_16"
```

For a UNIX environment the adapt the batchClient shell script.

Example:

```
JAVA_HOME=/usr/local/java/jre1.5.0_15
```

Chapter 4

Batch Client Settings

This chapter describes the several settings related to the Java Wrapper framework, the Batch Client in scenario mode and ECI mode.

Batch Client Solution

The Batch Client uses the Java Wrapper as run-time frame work.

Java Wrapper Settings

The Java Wrapper needs a configuration file which is described here.

The Java Wrapper supports two different types of shutdown modes.

1. Simple type, the Java application controlled by the Java Wrapper will be configured and simply killed on shutdown, this mode is used for the Java Daemon
2. Start / Stop type, this mode allows to define a Java application which should be controlled and which Java application should be used to shutdown the controlled application, this mode is used for a TOMCAT for instance.

The Batch Client uses the Start / Stop mode to clean shutdown the Batch Client and the connected PLM server.

Here a short description of the settings which may be adopted to implement a custom batch solution.

This section in the configuration defines the controlled Java application and it command line parameters.

```
# Application parameters.  Add parameters as needed starting from 1
wrapper.app.parameter.1=com.agile.testclient.BatchClientApp
wrapper.app.parameter.2=3
wrapper.app.parameter.3=-p%EP_ROOT%/examples/LgvLoop.properties
wrapper.app.parameter.4=-i%EP_ROOT%/examples/LgvLoop.groovy
wrapper.app.parameter.5=-T%EP_ROOT%/axalant/ini/testclient.properties
```

The next section in the configuration defines the Java application and it command line parameters to shutdown the controller application.

```
# Application parameters to stop the batch client
wrapper.app.parameter.6=com.agile.testclient.BatchClientApp
wrapper.app.parameter.7=true
wrapper.app.parameter.8=3
wrapper.app.parameter.9=-p%EP_ROOT%/examples/LgvLoopStop.properties
wrapper.app.parameter.10=-i%EP_ROOT%/examples/LgvLoopStop.groovy
wrapper.app.parameter.11=-T%EP_ROOT%/axalant/ini/testclient.properties
```

This section within the configuration file defines the MS Windows Service settings. Here the name of the server can be defined and some extended settings for services.

```
#*****************************************************************
# Wrapper NT Service Properties
#*****************************************************************
# WARNING - Do not modify any of these properties when an application
#  using this configuration file has been installed as a service.
#  Please uninstall the service before modifying this section.  The
#  service can then be reinstalled.

# Name of the service
wrapper.ntservice.name=AgilePLM_Batch_Service

# Display name of the service
wrapper.ntservice.displayname=AgilePLM_Batch_Service

# Description of the service
wrapper.ntservice.description=Batch Service for Agile PLM

# Service dependencies.  Add dependencies as needed starting from 1
wrapper.ntservice.dependency.1=

# Mode in which the service is installed.  AUTO_START or DEMAND_START
wrapper.ntservice.starttype=AUTO_START

# Allow the service to interact with the desktop.
wrapper.ntservice.interactive=false
```

Batch Client Properties

The used Batch Client scenario and the corresponding property file are defined as command line parameters in the Java Wrapper configuration file as you can see above.

This configuration file has to be adopted to your environment.

```
#
# ECI connection
#
host=<JAVA-DAEMON-HOSTNAME>
port=<JAVA_DAEMON-PORT>
env=<PLM-APPLICATION>
#
# Directories
```

```
#
varenv.ep_root=<ROOT_DIRECTORY_OF_THE_BATCH_CLIENT>
varenv.axalant_root=<AXALANT_ROOT_DIRECTORY_OF_THE_BATCH_CLIENT>
#
# Platform
#
varenv.EP_MACH=<MACHINE_PLATFORM_NAME>
#
# Security settings
#
KeyStoreFile=cwallet.sso
KeyAlias=C=DE,ST=Baden,L=Karlsruhe,O=Oracle,OU=Agile PLM,CN=PLM
TicketKey=RSA-PUBLIC-
BASE64:D8VyVSBxMfgXcZ8AXhOSZMI6Agh4IVQdU49RgszulDGm+z7dQbSIBYRWbpdfsYgP
s4GmjQL//tVYLdttGvLw6n2uN4/iwLFjGO93PtzGuX7TGqWZQkgXR4pGw7M2KjXMDNN/nIL
9rwlWyRKYLOzHZka1ZMgaopEuwPRmsqoQ2lU=
#
# PLM Client
#
client1=<USER_NAME>,<PASSWORD>,<SCENARIO_CLASS_NAME>
```

The Batch Client bundle provides an example which has to be adopted to your environment.

Secure the user credentials

In the Batch Client properties file the user credentials are stored. To protect the user password it can be encrypted by the **epkeytool** application.

```
epkeytool -encryptpwd -keyStore cwallet.sso -keyAlias
"C=DE,ST=Baden,L=Karlsruhe,O=Oracle,OU=Agile PLM,CN=PLM" -pass
USER_PASSWORD
```

The output is something like this:

```
RSA-PUBLIC-
BASE64:P9joyIBH7ATx6pW5Gxw9RGHxeh4mUWLzEHCmd/o3xhVVmje1mf091cTF+Pum71Px
0da7OTyPkt0M1ubjdL8zdgjaiHx6RczOA+grCgVnNnpAdc=
```

Copy the encrypted password into the scenario properties file as user password.

Example:

```
#
# PLM Client
#
client1=USER_NAME,RSA-PUBLIC-
BASE64:P9joyIBH7ATx6pW5Gxw9RGHxeh4mUWLzEHCmd/o3xhVVmje1mf091cTF+Pum71Px
0da7OTyPkt0M1ubjdL8zdgjaiHx6RczOA+grCgVnNnpAdc=,com.agile.LGVCall
```

Shutdown Batch Client Properties

The shutdown properties are the same as the controlled Batch Client properties for the batch use case. There is only another scenario scripts used.

Batch Client ECI Mode

The configuration of the Batch Client in ECI server mode the configuration is similar to the Batch Client in scenario mode with some small differences which are described here.

The ECI server mode does not need a scenario script but the ECI server settings have to be configured.

Here the additional settings within the Batch Client properties file:

```
#
# ECI Tunnel
#
varenv.EciServer.Port=44444
varenv.EciServer.Encoding=UTF-8
varenv.EciServer.Secure=true
varenv.EciServer.StartupImmediately=true
varenv.EciServer.ConnectionDelay=100
varenv.EciServer.allowHost.1=localhost
```

- The **Port** setting defines the port of the ECI server for client connections
- The **Encoding** should be set to UTF-8
- The **Secure** decides if the ECI should be secured or not. The value **true** is recommended, it ensures that ECI client must authenticate with user/password to access the ECI server.
- The **StartupImmediately** setting defines if the PLM server should started immediately with the start-up of the ECI server or on demand.
- The **ConnectionDelay** defines the minimum delay between two subsequent connection requests to prevent deny of service attacks.
- The **allowHost** list contains all allowed client host names which are accepted for ECI connections.

Additional settings:

```
varenv.EciServer.denyHost.1=<hostname_1>
varenv.EciServer.denyHost.2=<hostname_2>
varenv.EciServer.CallablePackage.1=<custom_callable_package>
```

- The **denyHost** list allows to define client host name which are explicit declined when they try to open an ECI connection.
- The **CallablePackage** list can be used to add custom callable packages

Enable the Old Batch Solution

The "old" batch solution of the PLM server is still available, but due security risks this solution is not recommended.

To enable the EDB_BATCH mechanism the security level has to be decreased to "unrestricted".

The security level can be configured in the <environment>.xml file located in the <ep_root>/init directory.

Example:

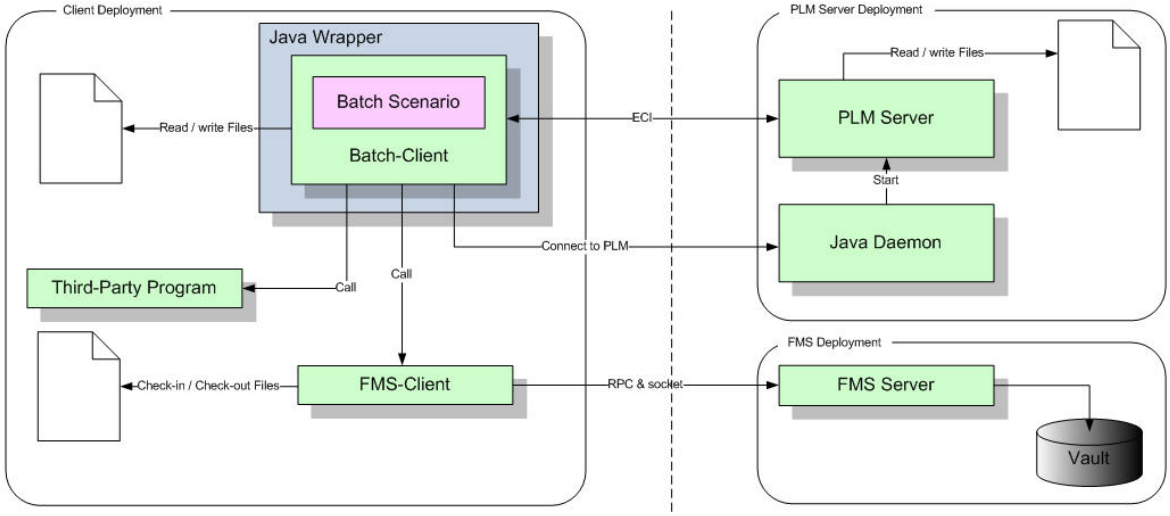
```
  <IPC AbsEciUrl="eci://localhost:19997" SecurityLevel="unrestricted"
TicketTimeout="600">
  </IPC>
```

Note The security level setting controls the security settings of the ECI connection, too. A client needs no authentication to connect to an **unrestricted** ECI server. So ensure that this setting is not effective for PLM servers which allow to be contacted via ECI.

Chapter 5

System Architecture

The Batch Client is a special version of the Java-Client without any UI. The Batch Client uses the Java Daemon to start the PLM Server and communicates via ECI with the PLM Server. The Batch Client supports client side callables which are used by the PLM Server to start client side activities like the startup of an external program or the FMS Client to check-in / check-out files from the File Server. The standard features to exchange files with the PLM Server are as well supported.



The Batch Client runs a batch scenario which is implemented in Groovy to control the batch use case. The most batch use cases are designed as a loop which uses a job table to determine if a job is present or if the process should wait for a new job.

In "old" batch designs that loop is implemented in LogiView and sometimes there are external programs used to implement the sleep functionality to wait some seconds for a new batch job.

That batch job loop has to be moved into the batch control scenario on client side for some reasons.

- The Batch Client provides the "sleep" functionality to wait for a new batch job.
- The Batch Client communicates via ECI with the PLM Server the checks for a new job helps to keep the connection alive.
- The PLM Server is started by the Java Daemon and the Java Daemon communicates with the PLM Server for controlling purposes, if the PLM Server is in a LogoView loop the communication with the Java Daemon is not possible.

The Batch Client uses the ***Java Wrapper*** like the Java Daemon to support to be installed as a service.

Chapter 6

Use Cases

The Batch Client supports two modes

- The scenario mode to implement batch use cases
- The ECI server mode to provide an ECI tunnel and client access like file server access.

Scenario Mode

The scenario mode runs a Groovy script which implements the batch process. This chapter explains how the design and write a script to implement a Batch use case.

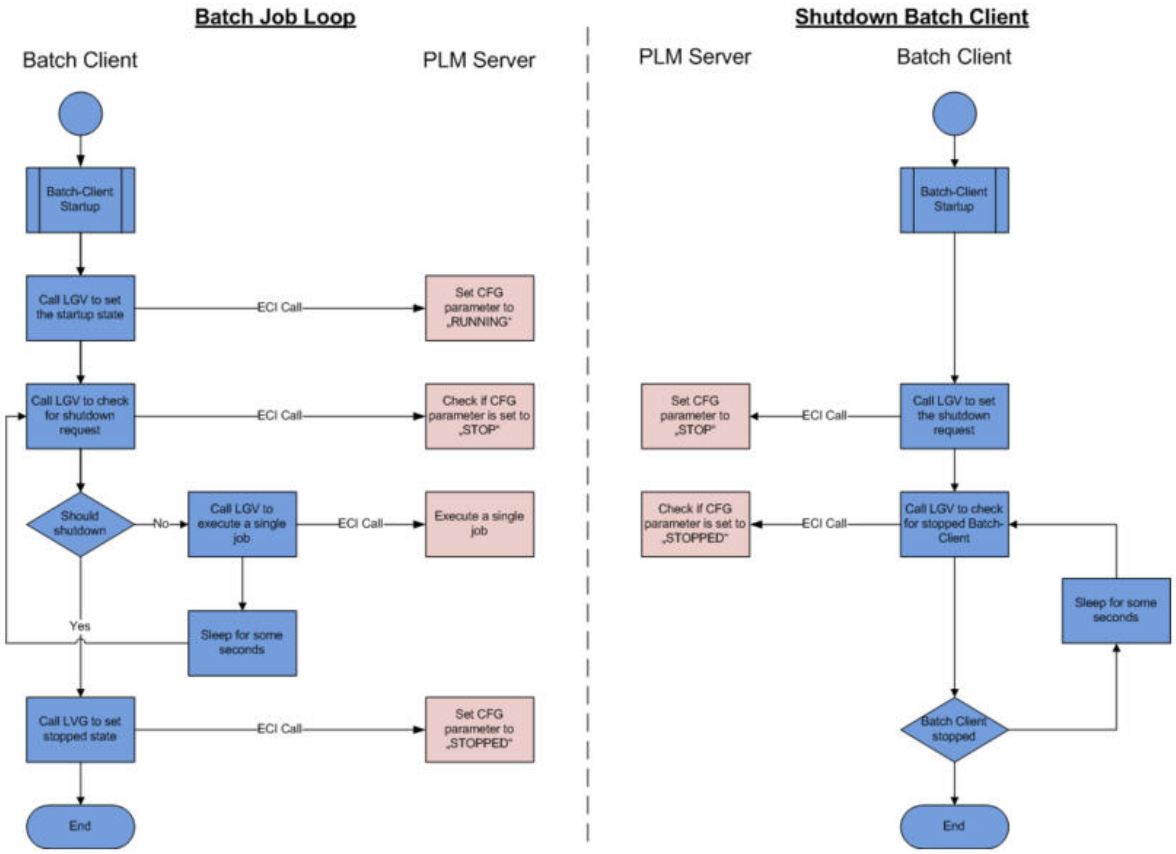
Recommended design of a batch control

Note	With the Batch Client bundle a set of batch control scripts are available in the ***examples*** folder. The LogiView procedures of the examples are available as loader file. In this example the synchronization of the run-time process and the shutdown process is realized by using a PLM configuration parameter within the database. But you can use any information source which is accessible by both processes, for example a shutdown job in the job table, a local file, a record in the database and so on.

With the Batch Client bundle a set of batch control scripts are available in the *examples* folder.

The *LgvLoop.groovy* script can be used as template for each job table based batch use case. The script uses as set of LogiView procedures to check , control and execute batch functionality. Like the TOMCAT Server or the Oracle Application Server the Batch Client provides a run-time process and a shutdown process. In the provided template the Batch Client LogoView backbone uses a configuration parameter to report the current state of the batch process and to control the batch process.

The following picture shows a command flow of the run-time and the shutdown process.



The Java wrapper executes the run-time process during start-up and calls the shutdown process to shutdown the run-time process. The run-time process sets the current state of the batch job by using the corresponding LogiView procedures and executes the batch job with the central LogiView procedure. To detect a shutdown request the run-time progress checks if a shutdown request is available by calling a special LogiView procedure.

In this example there are several LogiView procedures to report the current state of the run-time process:

- **setRunning** called in the start-up phase to report that the batch job is running
- **setStopped** called in the shutdown phase to report that the batch job has been stopped
- **shouldShutdown** to detect a shutdown request
- **executeOperation** to do the real work of the batch job

First the run-time process call **setRunning** during the start-up (you may use another LogiView procedure like **isRunning** to detect if there is already a batch job running to prevent two batch jobs at the same time). The run-time process checks if there is a shutdown request available by calling **shouldShutdown**, if not then the **executeOperation** LogiView procedure is called to execute a batch job. If a shutdown request is available the run-time process sets the "stopped" state by calling **setStopped** and end the run-time scenario.

The shutdown process uses the following LogiView procedures:

- **shutdown** to create a shutdown request
- **isStopped** to detect that the run-time process has been stopped

The shutdown process waits until the run-time process has been stopped before the shutdown scenario ends.

The Java wrapper waits until the shutdown process ends or until the shutdown timeout has been reached. The shutdown timeout can be configured in the Java wrapper configuration file.

Time-Out Value Information

wrapper.jvm_exit.timeout

Number of seconds to allow between the time that the JVM reports that it is stopped and the time that the JVM process actually terminates. 0 means never time out. Defaults to 15 seconds.

In normal operation, the Java side of the Wrapper will execute System.exit when it has completed its JVM shutdown cycle and is ready to exit. When this timeout is triggered, a message like the following will be logged.

```
wrapper  | Shutdown failed: Timed out waiting for the JVM to terminate.
wrapper  | Java Virtual Machine did not exit on request, terminated
```

If the application has registered its own shutdown hook which takes some time to complete, you could experience timeouts waiting for the JVM process to terminate. To avoid this problem, it may be necessary to extend the timeout to give the application's shutdown hook time to execute to completion. Be aware that as a rule, shutdown hooks should always complete almost instantly.

Example:

```
wrapper.jvm_exit.timeout=5
```

Run-time Process (Batch Job Loop)

Here the Groovy script of the *Batch Job Loop*.

```
/**
  * Batchclient example script to demonstrate the basic
  * implementation of a batch service.
  *
  * $Id: LgvLoop.groovy,v 34.2 2009/06/21 15:51:21 schwera1 Exp $
  */

 package com.agile.LgvLoop;

 import com.agile.testclient.*;

 /**
  * Each scenario needs to extend the Scenario class.
  * The Scenario class provides the execution environment.
  */
 public class LgvLoop extends Scenario {

   /**
    * The run() method is called by the framework to start the
```

```
    * scenario.
    */
   public void run() {
     System.out.println("Starting BatchExample/LgvProd");

getTestClient().callNativeUsx("lgv_nosel_run","BatchExample/setRunning");

     while(!shouldShutdown()) {

getTestClient().callNativeUsx("lgv_nosel_run","BatchExample/executeOperat
ion");

       sleepSeconds(10);
     }


getTestClient().callNativeUsx("lgv_nosel_run","BatchExample/setStopped");
     System.out.println("Done BatchExample/LgvProd");
   }

   /**
    * Checks if the batch client should shutdown.
    * The method checks the DTV default "EDB-BATCH-CONTROL" to receive a
shutdown request.
    *
    * @return true if the batch client should shutdown else false
    */
   private boolean shouldShutdown() {
     int result =
getTestClient().callNativeUsx("lgv_nosel_run","BatchExample/shouldShutdow
n");

     return result == 1;
   }
 }
```

The script uses the following methods to implement the run-time process:

- **callNativeUsx** to call LogiView procedures
- **sleepSeconds** to sleep for some seconds to wait
- **println** to print some information into the log file

Therefore the script calls 4 LogiView procedures to set the current process state to execute the batch job and to check for shutdown requests. Here the LogiView procedures used by the batch use case:

BatchExample/setRunning

This LogiView procedure sets the batch process control variable (here the cofiguration parameter *EDB-BATCH-EXAMPLE-CONTROL*) to the value "RUNNING".

```
EDB_BATCH_VALUE = "RUNNING"
update(EDB_BATCH_VALUE)
where("T_CFG_DAT.EDB_ID" = "EDB-BATCH-EXAMPLE-CONTROL")
```

```
exec_update()
```

The configuration parameter *EDB-BATCH-EXAMPLE-CONTROL* can be used to check if the Batch Client is online or not. If your batch process does not allow to run more than one Batch Client you can add another LogiView call to check the current state of the batch process. The Logiview procedure **BatchExample/isRunning** checks if there is already another batch process active so you can prevent to start a new Batch Client by checking the current state and implement your batch script to exit in that case.

BatchExample/shouldShutdown

The LogiView procedure checks the configuration parameter *EDB-BATCH-EXAMPLE-CONTROL* for a shutdown request (you can also use different variables to reflect the current state and to control the batch process). The LogiView reports a shutdown request if the configuration parameter *EDB-BATCH-EXAMPLE-CONTROL* is set to "STOP" or if the configuration parameter is not present in the system.

```
select(EDB_BATCH_VALUE)
where("T_CFG_DAT.EDB_ID" = "EDB-BATCH-EXAMPLE-CONTROL")
RES = exec_select(1)
if (RES == 1
  if (EDB_BATCH_VALUE == "STOP")
    exit()
  endif
else
  exit()
endif
```

BatchExample/executeOperation

That is the LogiView procedure which do the real work. This procedure checks if there is a batch job available and executes that job or it returns that the Batch Client can sleep for some seconds and try it later again. In the example the procedure simply sends a message to the Batch Client that a job is executed. You can use these messages to trace the actions of the batch job in the client trace.

BatchExample/setStopped

The batch process checks in every loop if there is a shutdown request is pending. Before the Batch Client shutdown the LogiView procedure *BatchExample/setStopped* is called to set the current state of the batch process.

All the sample LogiView procedures are available as loader file within the Batch Client Bundle.

Shutdown process (Shutdown Batch Client)

That is the shutdown script used by the Java Wrapper to shutdown the Batch Client.

```
/**
 * Batchclient example script to demonstrate the shutdown process.
 *
 * $Id: LgvLoopStop.groovy,v 34.2 2009/06/21 15:51:21 schwera1 Exp $
 */
```

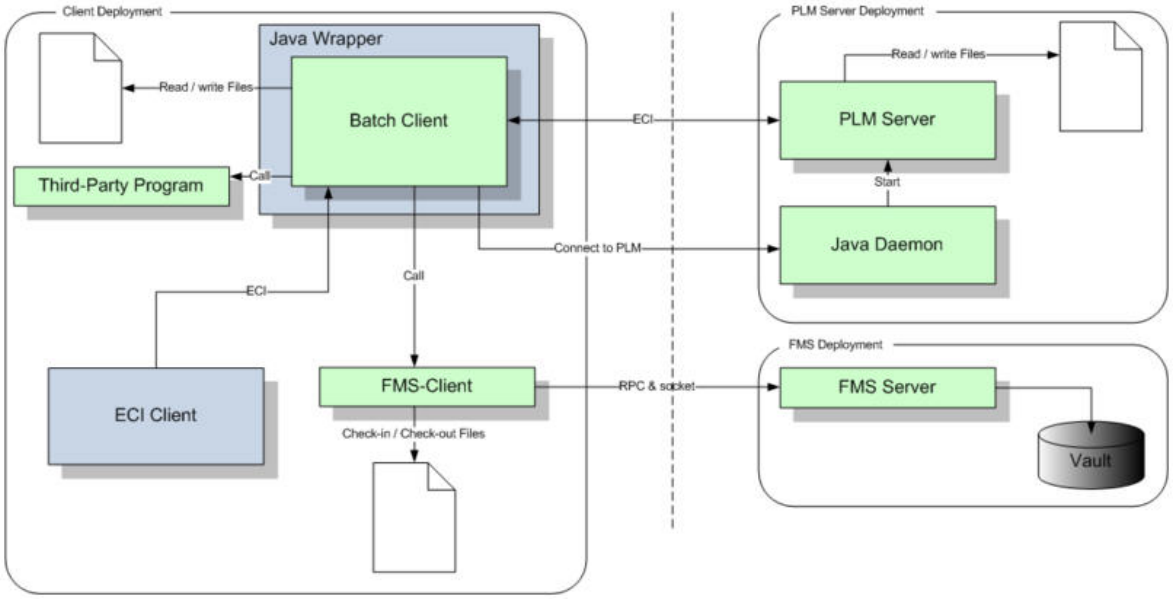
```
package com.agile.LgvLoop;

import com.agile.testclient.*;

/**
 * Each scenario needs to extend the Scenario class.
 * The Scenario class provides the execution environment.
 */
public class LgvLoop extends Scenario {

  /**
   * The run() method is called by the framework to start the
   * scenario.
   */
  public void run() {
    System.out.println("Send shutdown request");

getTestClient().callNativeUsx("lgv_nosel_run","BatchExample/shutdown");

    while(!isStopped()) {
      System.out.println("Wait for stopped batch client");
      sleepSeconds(1);
    }

    System.out.println("Done");
  }

  /**
   * Checks if the batch client has stopped.
   * The method checks the DTV default "EDB-BATCH-CONTROL".
   *
   * @return true if the batch client has stopped else false
   */
  private boolean isStopped() {
    int result =
getTestClient().callNativeUsx("lgv_nosel_run","BatchExample/isStopped")
;

    return result == 1;
  }
}
```

ECI Server Mode

The Batch Client can be used as a ECI Server which tunnels the ECI calls of an ECI client to the PLM Server. Additional to that tunnel functionality the Batch Client provides some callables to start external programs or to interact with the FMS Client for file operations.

So a "simple" ECI client can be enabled to check-in or check-out files from the file vault without an interactive Java Client.

The following picture shows an overview of a Batch Client in ECI server mode.



As well as the Batch Client in scenario mode the ECI mode version uses the Java Wrapper to enable the service features.

The external ECI client can be for example a CAD integration which normal uses a Java Client to gain access to the file within a vault.

The Batch Client allows multi connections from several ECI clients.

To shutdown the Batch Client the ECI client can call the *eci_stb_edb* ECI function.

Chapter 7

References

Office Suite PDF Service

For further information on the Office Suite PDF Service please see chapter 9 “Office Suite – PDF Generator Installation” of the Administration Manual.

AutoVue Offline Metafile Cache Service

The Offline Metafile Caching can be used to pre-cache 2D and 3D files in the AutoVue Server to increase the viewing performance.

The AutoVue Server caches the native files and creates a metafile for the assembly, which is assigned to the base file.

The integration check-in the created metafile into the vault and assigns the file to the viewed document.

On viewing actions the AutoVue Server checks if the native files and the metafile is already present in the local cache and uses them to view the assembly.

In this case the AutoVue Server does not download any file from the PLM system.

If the user views a sub assembly the AutoVue Server does not download the native files if there already present in the local cache. But the AutoVue Server creates a new metafile for that sub assembly and checks in the metafile to the vault and assigns the metafile to the sub assembly document within the PLM system. On the new viewing action on that sub assembly all files are present in the local cache.

Why we check-in the metafile into the vault?

The local cache on the AutoVue Server may be cleaned or crashed. In this case the AutoVue server can download all native files and the metafile from the PLM system and does not need to create a new metafile. The conversion process from the native format to the AutoVue metafile format is a time consuming process.

In which vault the metafile will be checked-in?

The system uses the DFM feature to specify the target vault for metafiles.

The integration sets the following properties:

- File Type: *META*
- File Format: Same as the base file or *NAT* if not set
- Creation System: *AUTOVUE*

Requirements and Assumptions

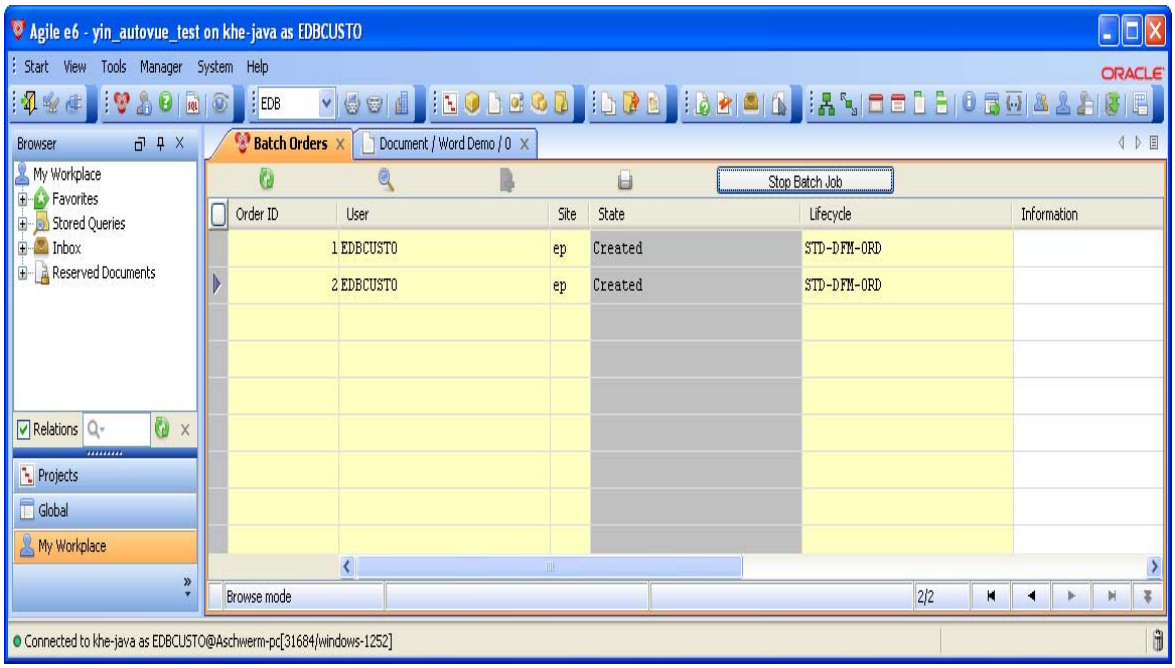
- Batch-Client as service
- JVue bean as control interface of the AutoVue server
- VueLink DMS as data source and used to check-in the meta file
- AutoVue-Batch-Engine implemented in Java (JVue bean control)
- DFM replication is not supported, all files have to be replicated with standard replication feature before the file can be cached on remote locations. The system checks if all files are replicated and sets an error code if not all files are available on the remote location.

Order Queue

The Offline Metafile Cache feature uses an order queue to send the native files and the checked-in metafile (if available) to the AutoVue server (System->AutoVue->Batch Orders)

An order has one or more order elements or order positions which contain the necessary document and file data.

The state shows the current working state of the order.

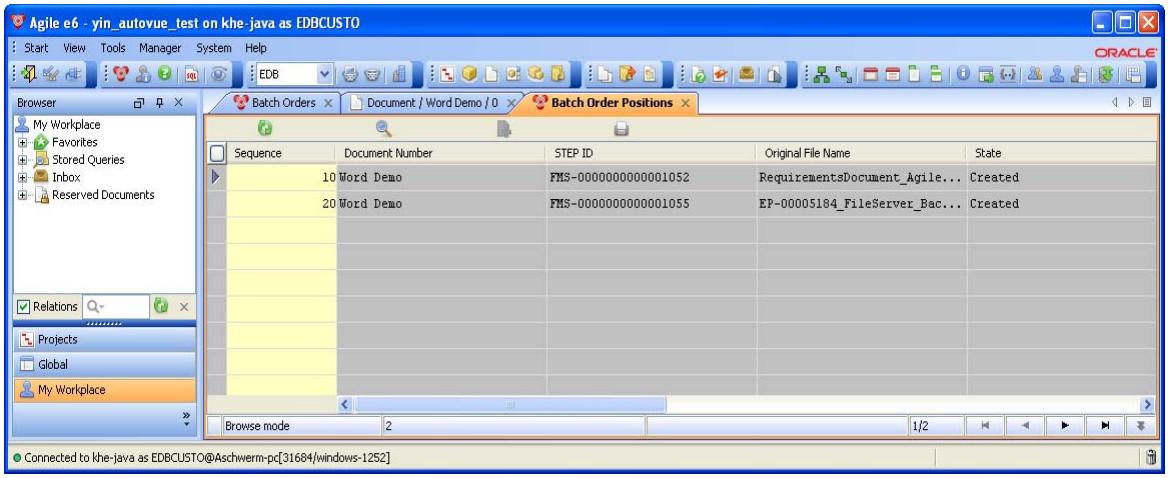


Order Positions

The order positions are assigned to the order and specifies which files should be cached on the AutoVue server.

It is only necessary to add the base file of a 3D assembly. The batch job caches all related native files on the AutoVue server to pre-cache the assembly.

The state shows the current working state of the order position.



Create an Offline Metafile Cache Order

The system provides two userexits to add documents and files to the Offline Metafile Cache.

The userexit **xpvm_bat_doc** creates a new order and adds all assigned files which be viewable with AutoVue as order position.

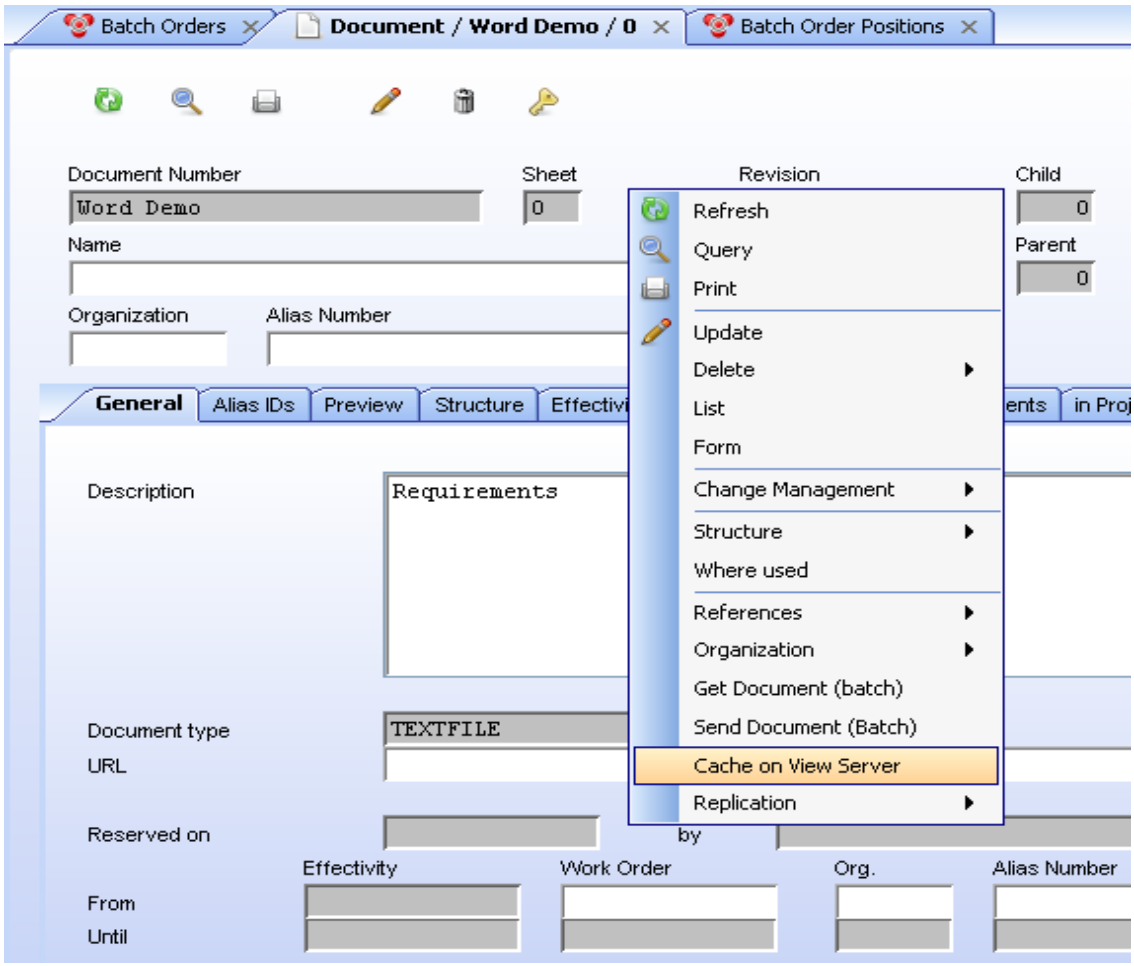
The userexit **xpvm_bat_fil** creates a new order and adds the specified file as order position.

Both userexits are select-menu userexits. **xpvm_bat_doc** can be used on document masks and **xpvm_bat_fil** on document - file relation masks.

It is also possible to create orders and elements via LogiView by creating records in the order and order position masks.

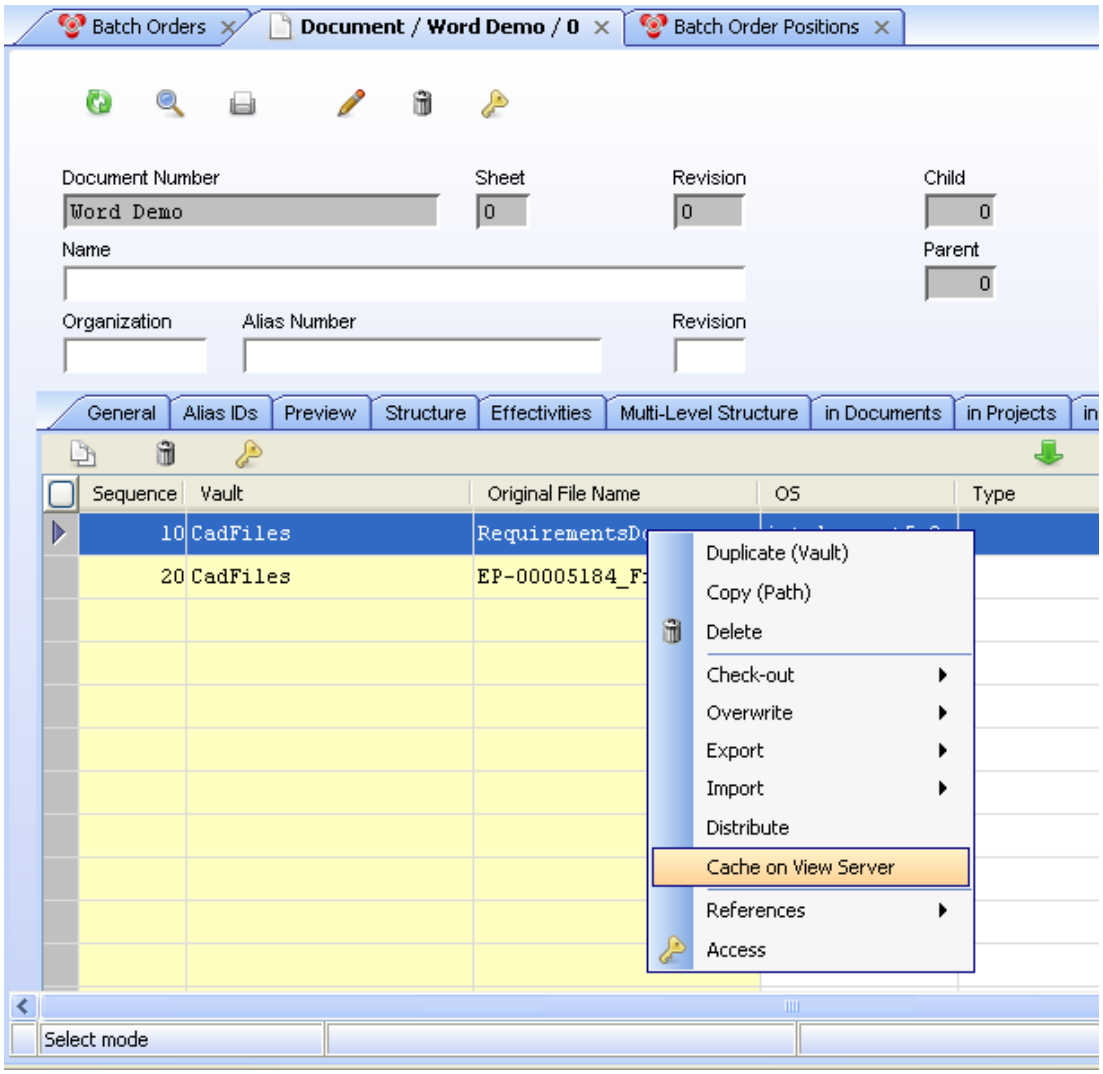
Add all files of a document

In the standard installation the userexit **xpvm_bat_doc** is available on the document master form.



Add a specific file

In the standard installation the userexit **xpvm_bat_fil** is available on the document file sub list.



Batch Process

The Offline Metafile Cache feature uses the Batch-Client to control the caching process.

The main loop is implemented on the Batch-Client to reduce the load on the PLM server machine.

The sleep interval length can be configured in the Batch-Client settings.

The batch process is available on Windows only and should be installed on a machine near to the AutoVue Server to reduce the network traffic.

The batch process can be installed as Windows Service or run as a console program.

Installation

This section describes the manual installation of the Offline Metafile Cache Service.

Extract the installation package

The installation package of the Offline Metafile Cache Service is located on the **package** directory and named **AutoVueBatch.zip**
Extract the package to your installation directory.
In this installation example the installation path is "C:\Program Files\Agile_e6\AutoVueBatch" (the *AutoVueBatch* path is the base path in the package)

```
unzip z:\package\AutoVueBatch.zip -d "c:\Program Files\Agile_e6"
```

The installation package contains the following directories:

```
-> PDF Service Root         (C:\Program Files\Agile_e6\AutoVueBatch)
 -> axalant
    -> cmd                  (scripts directory)
    -> batch                (Batch service files)
    -> bin
      -> intel-ms-nt5.0     (binaries like the FMS-Client)
      -> java               (e6.1 Java archives)
 -> ext
    -> bin
      -> intel-ms-nt5.0     (external binaries)
      -> java               (external Java archives)
 -> tmp                     (Logging directory)
```

Adapt the installation

You need to adapt the start-up script to setup the Java-Runtime and the installation path of the Offline Metafile Cache Service.
The script is located at the **axalant/cmd** sub directory of the installation.

The **vuelink_batch.bat** script contains the following basic configuration settings:

```
set JAVA_HOME=<JAVA_HOME>
set ep_root=<ROOT DIRECTORY OF THE OFFLINE METAFILE CACHE SERVICE>
```

Example:

```
set JAVA_HOME=C:\Program Files\Java\jdk1.5.0_16
set ep_root=C:\Program Files\Agile_e6\AutoVueBatch
```

Adapt the service settings

The **vuelink.properties** file is located at the **axalant/batch** sub directory of the installation.
This file sets the environment variables needed by the Offline Metafile Cache Service.

The following properties must be adapted (the other properties should no be changed):

```
#
# ECI connection
#
host=<HOSTNAME OF THE ECI DAEMON>
port=<PORT OF THE ECI DAEMON>
env=<PLM APPLICATION ENVIRONMENT>
#
# Directories
#
varenv.ep_root=<ROOT DIRECTORY OF THE PDF SERVICE>
varenv.axalant_root=<AXALANT DIRECTORY OF THE PDF SERVICE>
varenv.$TMP=<PDF WORK DIRECTORY OF THE PDF SERVICE>
#
# Environment
#
varenv.EP_DDM_SITE=<DFM SITE>
#
# PLM Client
#
client1=<PLM USER>,<PASSWORD>,com.agile.autovue.VueLinkBatch
```

Example:

```
#
# ECI connection
#
host=khe-plm
port=20001
env=plm_ref
#
# Directories
#
varenv.ep_root=C:/Program Files/Agile_e6/AutoVueBatch
varenv.axalant_root=C:/Program Files/Agile_e6/AutoVueBatch/axalant
varenv.$TMP=C:/Program Files/Agile_e6/AutoVueBatch/tmp
#
# Environment
#
varenv.EP_DDM_SITE=ep
#
# PLM Client
#
client1=DEMOEP_M,not4test,com.agile.autovue.VueLinkBatch
```

Runtime

The Offline Metafile Cache Service uses the same mechanism as the Java-Daemon to install,remove,start,stop the service.

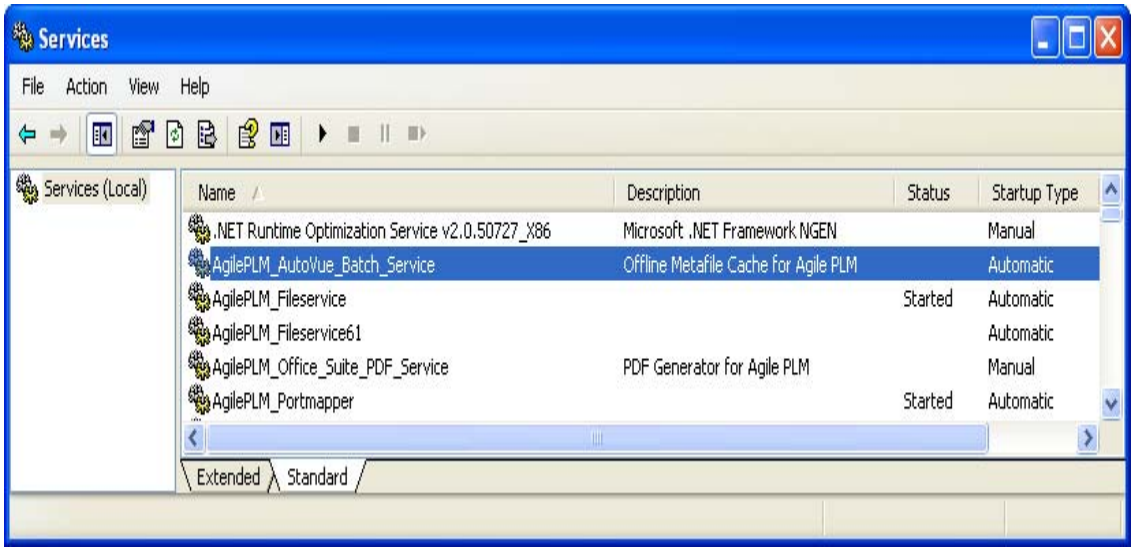
Install as Service

The configuration of the Windows Service registration can be found in the vuelink_wrapper.conf file which is located in the axalant/pdf sub directory of the installation.

```
#***************************************************************
# Wrapper NT Service Properties
#***************************************************************
# WARNING - Do not modify any of these properties when an application
#  using this configuration file has been installed as a service.
#  Please uninstall the service before modifying this section.  The
#  service can then be reinstalled.

# Name of the service
wrapper.ntservice.name=AgilePLM_AutoVue_Batch_Service

# Display name of the service
wrapper.ntservice.displayname=AgilePLM_AutoVue_Batch_Service

# Description of the service
wrapper.ntservice.description=Offline Metafile Cache for Agile PLM

# Service dependencies.  Add dependencies as needed starting from 1
wrapper.ntservice.dependency.1=

# Mode in which the service is installed.  AUTO_START or DEMAND_START
wrapper.ntservice.starttype=AUTO_START

# Allow the service to interact with the desktop.
wrapper.ntservice.interactive=false

wrapper.ntservice.account=.\axalantrt
wrapper.ntservice.password=********
```

To install the Offline Metafile Cache Service as Windows Service use the vuelink_batch.bat command script located in the axalant\cmd sub directory of the installation.

```
vuelink_batch.bat -i
```



Remove service

To install the Offline Metafile Cache Service as Windows Service use the vuelink_batch.bat command script located in the axalant\cmd sub directory of the installation.

```
vuelink_batch.bat -r
```

Run as console application

To run the Offline Metafile Cache Service as console application use the vuelink_batch.bat command script located in the axalant\cmd sub directory of the installation.

```
vuelink_batch.bat -c
```

This page is blank.