Oracle® Agile

Engineering Data Management

Agile e6.1.2.2 Architecture Guide

Part No. E27833-01

April 2012

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 ready, 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

Agile e6 System Architecture Overview

The Agile e6 system architecture is called a 3-Tier architecture, which contains the following parts:

- Client - The client is responsible for the presentation logic.
- Application Server - The application server process is responsible for the business logic.
- Database - The database server takes care of the physical storage of all data.

Three types of clients are available, serving the different needs of casual users and power users.

1. Java Client
2. Web Client
3. Windows Client

Note For more information about the Clients please see the chapter *Clients*. Please be aware there are more client side components which service specific client side functionalities (e.g. Workflow-Editor, Fileserver-Client).

With each client process launched, an Agile EDM server process is started in parallel. The Agile EDM server process provides the core PLM functionalities, like Item Management, BOM Management, Document Management, and Change Management, etc. User data (stored in the Oracle database) is accessed by the application server process (The connection to the database is realized with the OCI protocol. This is regarded as standard and not described any further in this document.). This ensures that the clients do not require a direct database connection.

Some responsibilities of the application server process have been assigned to dedicated services, being able to service several client processes in parallel. These include for example the File Management Service “FMS”, the Business Services and Technical Services.

The Business Services provide PLM functionalities for Workflow Management, Product Configurator and Permission Manager.

Technical Services encompass the following:

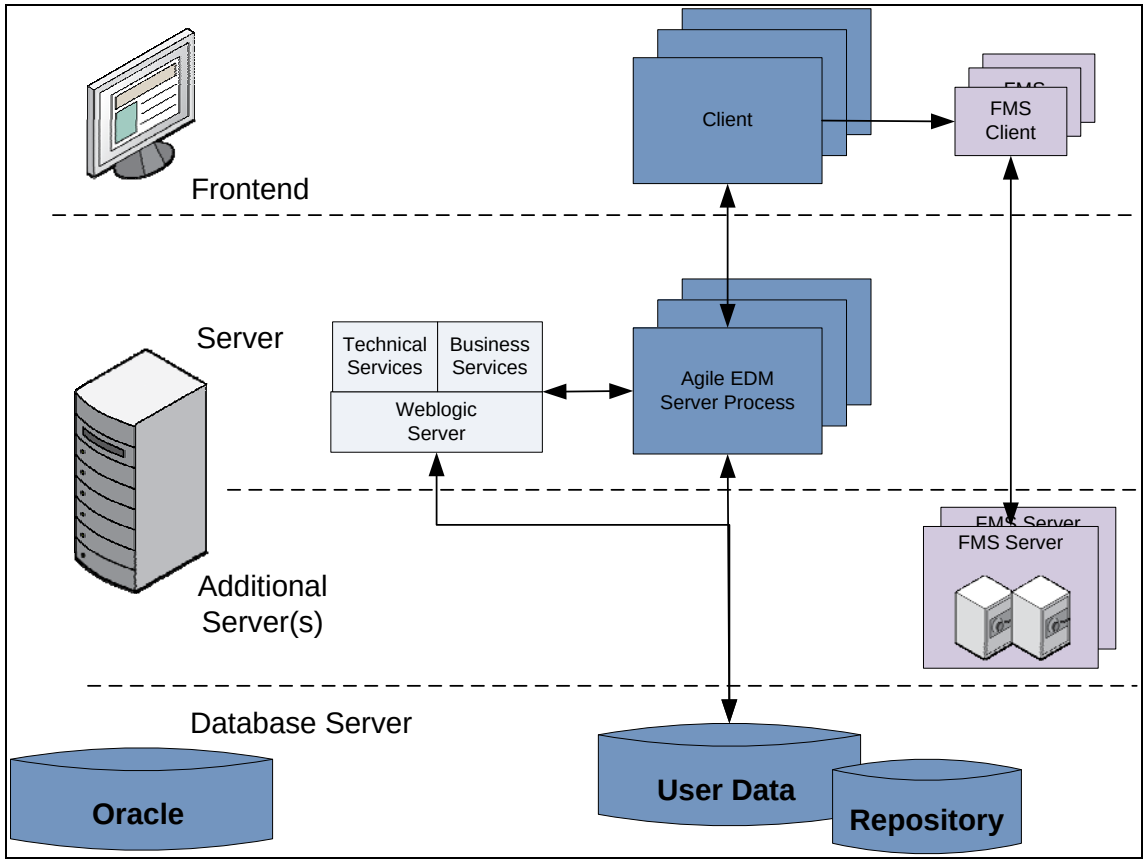
- Java Client WebStart deployment - responsible to provide a centralized Java Client installable deployable via WebStart technology.

- Java Client https support - to enable an encrypted client server communication for the Java Client.
- Web Presentation Service - to run the Web client.
- Web-Fileservice - to provide file access via http/https.
- Administration Client - provides the Front-End to administrate the application.

The Business Services as well as the Technical Services run on top of the Oracle WebLogic Application Server.

Note There are additional components beside File Management Service, which are not shown in the Architecture Overview.

The following diagram gives an overview of the main services which are used in the architecture of Agile e6:



Server-side Components

- File Management Services
- ViewServer – (external) component of AutoVue Viewer used view and redline documents (Office documents, 2D/3D-CAD Models)
- LDAP Server – (external) component (e.g. Oracle Identity Management Suite) to provide centralized store for managing user/password
- Batch Client – component to run PLM batch processes
- Java Daemon
- FMS Daemon
- PLM-API Proxy

Refer to the Chapter 2: Server-side Components for more details.

Client Side Components

- Workflow Editor - to model and view workflows
- Office Suite – to check-in/check-out documents from/to Microsoft Office

Refer to the Chapter 3: Clients for more details.

Object Oriented Repository

Agile e6 uses an object-oriented repository. It is unique in that it stores the complete metadata which defines the application with respect to:

- Object model
- User interface
- Business logic

The Business Logic consists of:

- Lifecycle definitions
- Workflow processes
- Consistency Checks

- Automation Scripts (Decision Tables and Scripting Language LogiView)

  Owing to the repository based approach of Agile e6, each application server process interprets the repository during initialization and can thus dynamically reflect any changes applied to the metadata. Modifications of the User Interface and/or adaptation of the business logic can be carried out instantly.

  The repository ensures the separation of system description and physical structure. Because all metadata is stored in the database, deployment and upgrade processes are simplified.

Chapter 2

Application Server Components

In general, the Agile e6 application server components can reside on the same server where the Agile EDM server processes are executed (this is recommended for Business Services) or can reside on any other computer in the network (especially for the File Management Service FMS).

Due to this additional separation of server components and services, Agile e6 features a distributed architecture.

The entire communication is based on TCP/IP. Only unprivileged ports (above 1024) are used. Once a TCP/IP connection is established the port will not be changed dynamically.

Business Services

The Business Services run on top of WebLogic application server and provide the following capabilities:

- **Permission Manager** - The permission manager is responsible to manage all user-role assignments and the resulting permissions. It provides user authentication and authorization checks for both the Agile e6 Core server and other components of the business services.

  An optional caching mechanism allows having fast access to data like user jobs and roles.

- **Workflow** - The Workflow Engine is used to execute activity-based business workflows.

- **Product Configurator** - The Configurator Engine, responsible for calculating valid product configuration based on logical rules.

File Management Services

File Management Service (FMS)

Files are stored on any server in the network under control of the File Management Service (FMS). The File Management Service manages documents (referred to as “Files”) in vaults, thus implementing the “check-in” and “check-out” functionality provided by the Document Management System in Agile e6.

The File Management Service consists of an FMS Client and a FMS Server. An FMS client communicates with the corresponding FMS server to check-in and check-out files.

Detailed descriptions about the access from the different clients can be found further on in this document in the respective client communication chapters. In case of the Java Client, the FMS Client is embedded and is not required to be installed separately. In case of the Web Client, the FMS Client is executed on the Web Server and does not require any installation on the front-end.

The FMS Server is installed on one or more server computers. Each FMS Server can manage one or more vaults.

- If the FMS Server is installed on a Windows platform, this server must be NTFS based. FAT does not work.
- The vaults managed by a FMS Server have to be created on local hard discs of the computer where the corresponding FMS Server is running or on a SAN. The SAN has to be configured in a way that IO from the file and the database server are separated (separated IO channel).
- The FMS Client communicates with the Agile e6 application server process using sockets.
- The FMS Server and the FMS Client communicate with Remote Procedure Calls (RPC) and sockets. The FMS Server does not communicate directly with the Agile e6 application server process.

In a very simple configuration, there is only one FMS Server with a single vault (e.g. when Agile e6 is installed on a single workgroup server or on a laptop).

In large installations of Agile e6, there may be FMS Servers running on several computers, each FMS Server managing multiple vaults.

Installations with remote users that connect to an Agile e6 Server through a Wide Area Network (WAN), e.g. an external office that is connected to the headquarters, would usually be limited when accessing files by a small network bandwidth. To improve performance in such a configuration, a FMS Server and one or more vaults can also be installed at the remote location (even though the Agile e6 server and the database are running at the headquarters). Files can be checked-in and checked-out into these vaults by any FMS Client that is able to communicate with this FMS Server.

The Distributed File Management (DFM) module controls the transfer and replication of document files between several distributed sites. Redundant storage of files is the basic working principle of this module. The aim of this data redundancy is to reduce network transfers in conjunction with the use of local vaults for file check-in and check-out processes. These measures can result in a significant reduction of transfer volume costs and increase the overall network performance. Files stored at distributed sites can be accessed at all times.

The Distributed File Management (DFM) module provides two methods for the replication of data:

- Data can be replicated by means of scheduled batch processes that are run automatically at regular intervals.

- Alternatively, specific files can be replicated online through user interaction.

The main advantage of automatic data replication is that file transfers can be initiated at off-peak times (e.g. at night), when this additional network traffic does not affect the overall network performance.

Web File Services

In an internet environment it is necessary to provide a file storage access via HTTP/HTTPS. The Web File Services are providing a scalable access to the file vaults which are managed by the Core File Server.

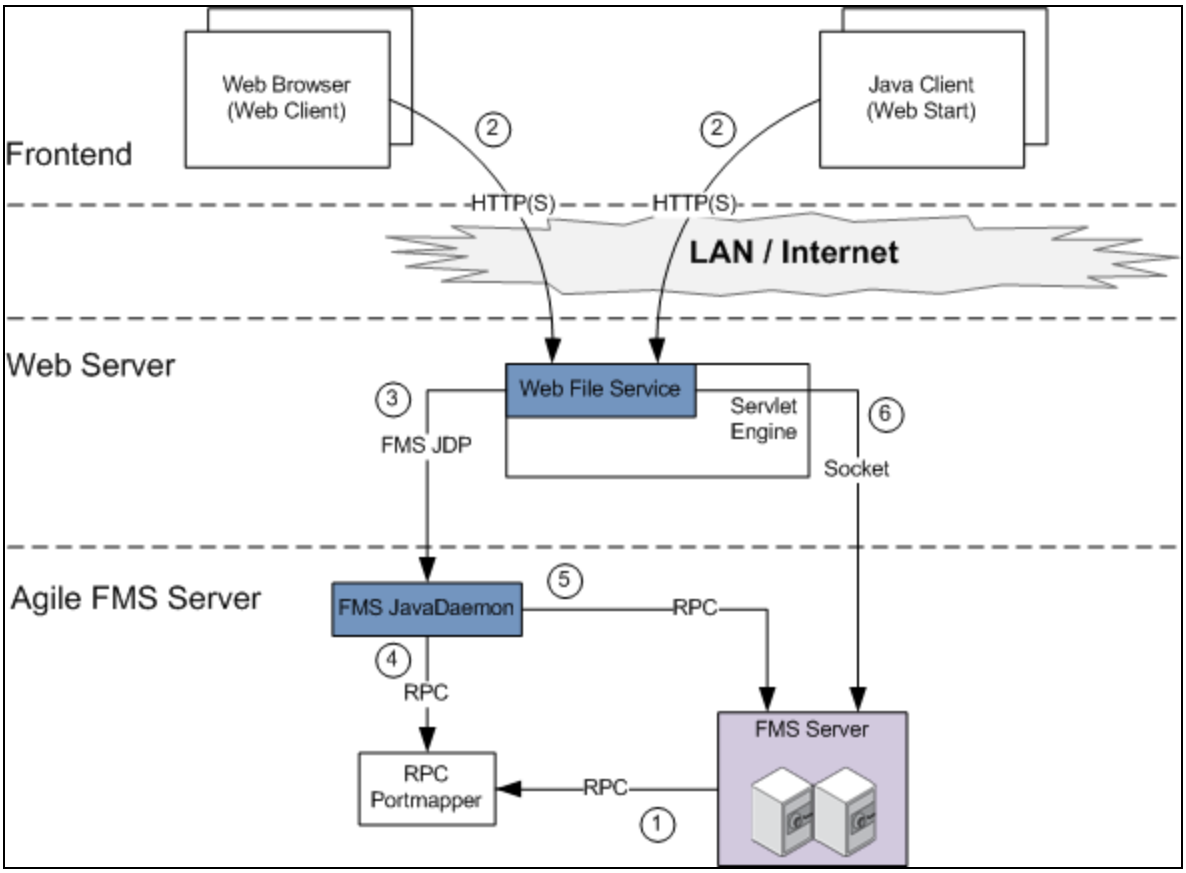
Prerequisites:

- Required hardware - All supported platforms for PLM
- Required Software - WebLogic Server / TOMCAT6 for DFM locations
- Required Knowhow - Administration of a WebLogic Server / TOMCAT6 for DFM locations

---

Note For further information about how to enable secure socket layer support in WebLogic/TOMCAT, please see respective WebLogic/TOMCAT documentation.

The following picture shows the communication lines when using WebFile Service.



1. The FMS Server (file server) registers itself by the RPC Portmapper (1)

2. An EDM Client (Web-Client or Java-Client) contacts the Web-File-Service to check-in or check-out a file. (2)

3. The Web-File-Services uses the configured FMS-Java-Daemon to get the transfer socket to the FMS Server. (3)

4. The FMS-Java-Daemon requests the connect information for the FMS Server from the RPC Portmapper. (4)

5. The FMS-Java-Daemon contacts the FMS Server and requests a transfer socket to upload or download the file.(5)

6. The Web-File-Service uses the transfer socket to upload or download the file. (6)

The following table shows the connections used. Refer to the *Agile e6 Administrator Guide* for more information.

Module	Port Configurable	Description
FMS	yes	HTTP(S) access to the Web-File-Service (2)
FMS	yes	FMS-Java-Daemon with the FMS-Java-Daemon Protocol (3)
FMS	no	RPC Portmapper (1), (4), (5)
FMS	no	FMS communication socket for file transfer (6)

FMS Java Daemon

The purpose of the FMS JavaDaemon is to get rid of the necessity to deal with native libraries in WebLogic context when using native FMS.

Java Daemon

The Java Daemon is contacted by the Java Client in order to connect to the Agile EDM server process. Java Daemon initiates the startup of the Agile EDM server process and routes the connect parameters to the Java Client.

In addition it is Java Daemon's responsibility to shut down server processes if the Java Client is closed by user. Therefore the Java Daemon checks on a regular base, if the Java Client is still alive. If the Java Client is no longer alive it will shut down the corresponding server process.

PLM-API Proxy

In order to allow Java Client to connect via http it is necessary to translate the native RPC calls normally send between Java Client and Agile EDM Server. PLM-API Proxy provides the gateway to translate http calls into RPC calls and vice versa.

View Server (AutoVue)

Note We recommend using the AutoVue Server as a separate server.

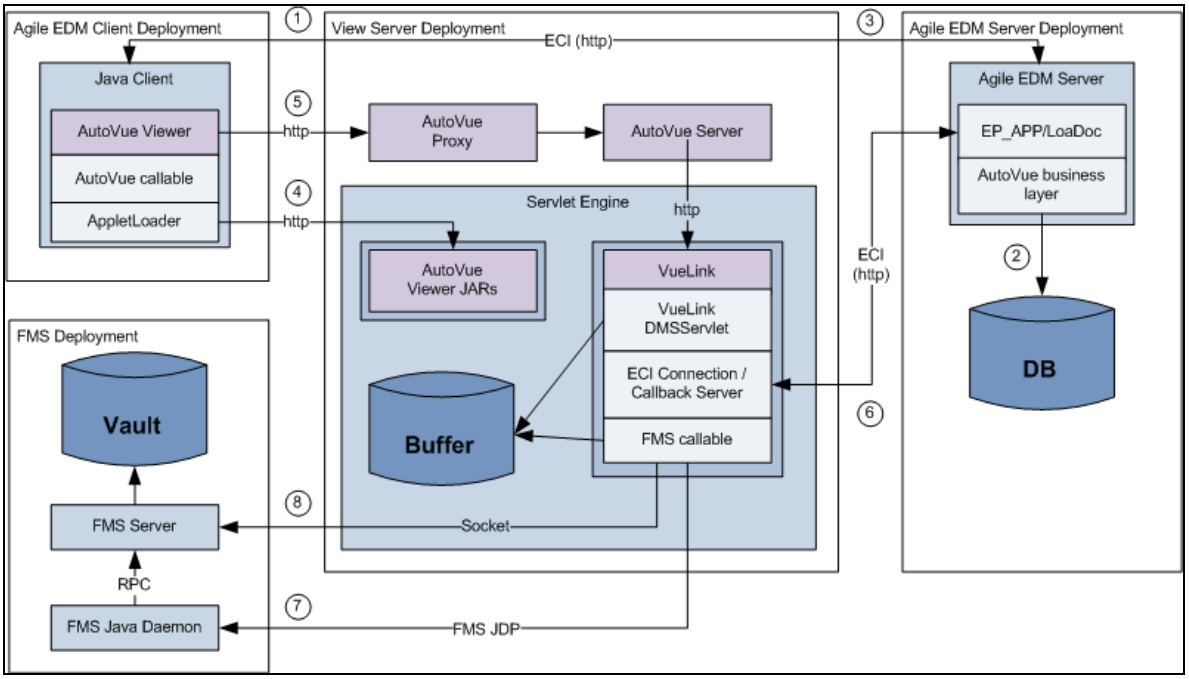
The AutoVue integration simplifies the loading and viewing of Enterprise Documents such as CAD, PDF, Office documents, etc., within the Agile e6 system (only applicable with the Java Client). The advantage of the AutoVue integration is that you do not require any corresponding authoring systems to be installed on a machine.

The enterprise documents can be edited in the Viewer, thus creating and attaching different types of a markup file (e.g. text, note, or attachment). Also, the AutoVue integration supports the feature to add a header/footer and watermark during printing or to show a permanent watermark in viewing mode.

The View Server Deployment contains the AutoVue Server, AutoVue Proxy, and the VueLink integration. The VueLink integration provides the metadata and the files to the AutoVue server.

The Java Client includes the AutoVue integration plugin, which downloads the AutoVue Viewer from the View Server Deployment, and shows the AutoVue Viewer inner frame or outer frame on the client machine.

The following picture shows the communication between the involved components.



1. The user requests to view/markup a document (1)

2. The business logic determines the view which should be used to view the document. In this case, the AutoVue viewer should be used and the system stores the information to view the document into the database. (2)

3. The EDM Server calls a client callback to view the file with the AutoVue viewer. (3)

4. The AutoVue viewer plug-in downloads the AutoVue viewer (once per session) from the AutoVue installation. (4)

5. The AutoVue viewer plug-in calls the AutoVue viewer to load the document and the AutoVue viewer contacts the AutoVue server to request the viewing data for the document to view. The AutoVue viewer can use HTTP or a plain socket to contact the AutoVue server. (5)

6. The VueLink DMS servlet connects to the same EDM server instance which is used by the Java-Client and retrieves the data to view the document. (6)

7. If the AutoVue server does not have the files already cached the VueLink DMS servlet uses the FMS Java Daemon to contact the FMS server. (7)

8. The file transfer is done through a direct socket connection. (8)

The following table shows the connections used. Refer to the *Agile e6 Administrator Guide* for more information.

Module	Port Configurable	Description
ECI	yes	PLM Java Daemon
ECI	yes	ECI communication socket
FMS	Yes	FMS Java Daemon
FMS	no	RPC Portmapper
FMS	no	FMS communication socket, the port range depends on the max. Number of concurrent users
AutoVue	yes	HTTP communication between the AutoVue viewer and the AutoVue Proxy
AutoVue	yes	Socket communication between the AutoVue Proxy and the AutoVue Server
AutoVue	yes	HTTP communication between the AutoVue Server and the VueLink DMS Servlet
AutoVue	yes	HTTP download of the AutoVue viewer

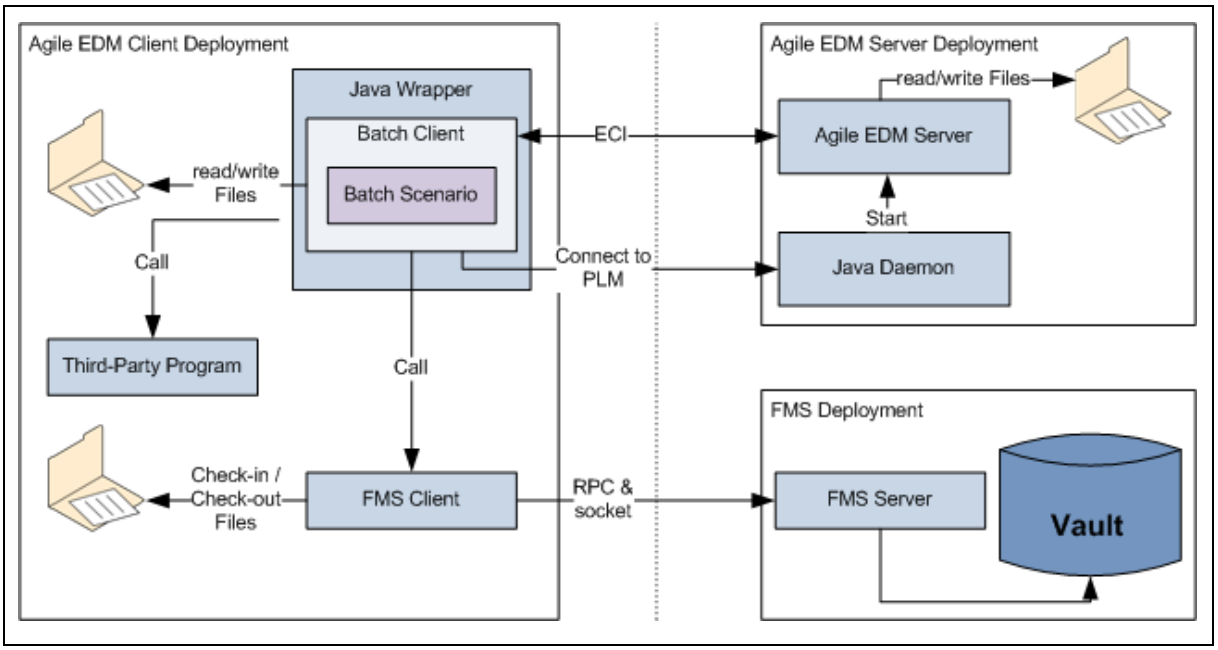
Batch Client

The Batch Client replaces the old batch solution via EDB_BATCH. The Batch Client is in fact a server solution to run batch scenarios.

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

The Batch Client supports two 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.



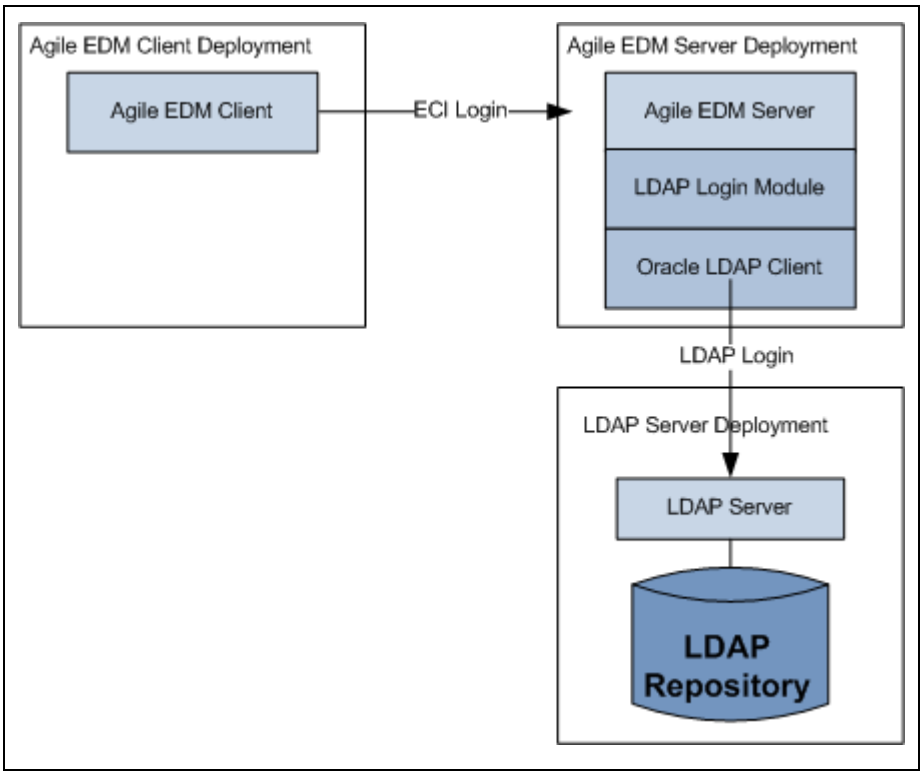
The following table shows the connections used. Refer to the *Agile e6 Administrator Guide* for more information.

Module	Port Configurable	Description
ECI	yes	PLM Java Daemon
ECI	yes	ECI communication socket
FMS	Yes	FMS Java Daemon
FMS	no	RPC Portmapper
FMS	no	FMS communication socket, the port range depends on the max. Number of concurrent users

LDAP

Many companies are using an LDAP server (Oracle Internet Directory, OpenLDAP, Microsoft Active Directory, or any other LDAPv3 server) to manage their users.

The LDAP integration allows authenticating Agile EDM users with their credentials from the LDAP server. The integration supports name mapping and multi domains.



- The Agile EDM Client sends a login request to the Agile EDM Server. The user uses the PLM user account with the LDAP user account password.
- The Agile EDM Server maps the PLM user name to the LDAP user name and contacts the configured LDAP server to execute the login request.

It is possible to configure more than one LDAP server. In the user configuration, the user account can be linked to a dedicated LDAP server and/or BaseDN.

The following table shows the connections used. Refer to the e6 Administrator Guide for more information.

Module	Port Configurable	Description
ECI	yes	ECI communication socket
LDAP	Yes	Connection to the LDAP repository

Chapter 3

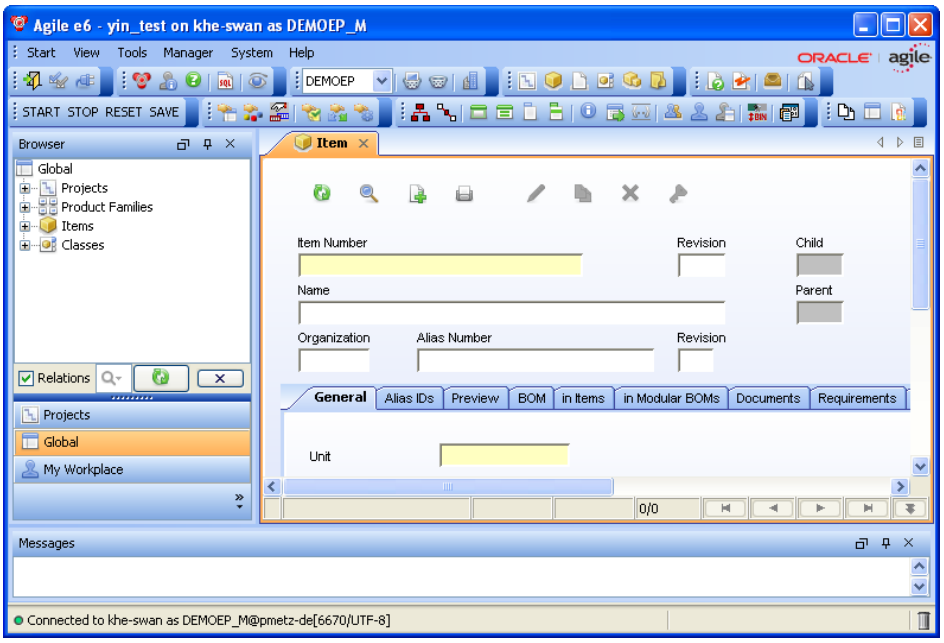
Application Clients

Agile e6 system provides three types of clients, serving different needs of casual users and power users.

- Java Client
- Web Client
- Windows Client

Java Client

The Java Client is the standard client for Agile e6. It enables access to all functions of Agile e6 (depending on the client platform). The Java Client is suitable for user and also customizers (except for the mask generation – this can only be done in the Windows Client) and offers only limited support for Administrators.



In combination with Oracle Sun's Java Web Start technology (see http://java.sun.com/products/javawebstart/), the deployment of the Java Client can be reduced dramatically.

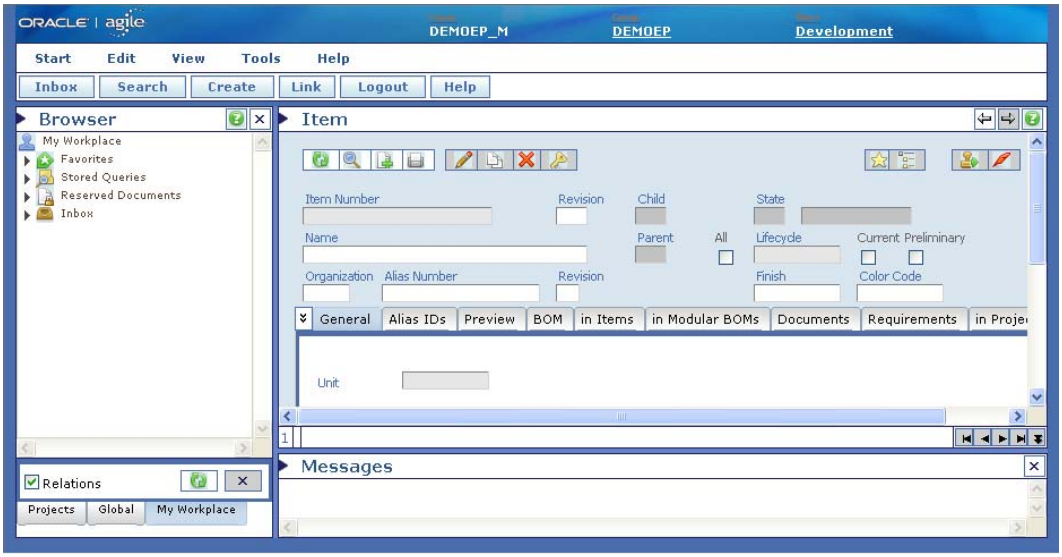
The Java Client does fully support M-CAD, E-CAD, EDA integrations, Office Suite features,

and most other types of integrations.

The user interface of the Java Client is dynamically defined by the metadata in the repository.

Web Client

The Web Client can be used by casual users on all front-ends. It requires one of the common web browsers (for details please see the platform list about supported web browsers). It enables access to most functions of Agile e6 (which are not Windows-specific). The Web Client is not suitable for Customizers and Administrators.



The Web Client uses DHTML (a combination of HTML and JavaScript) because the Web Client is a Web Presentation Service (in this document we refer to it as the Web Client) on the web server. It does not require a local installation that can be accessed by a web browser.

The most important limitations compared to the Java Client include:

- Slightly different concept of the integrated Explorer window (similar functionality can be achieved, but the customization is partially separate).
- No support for significant fields in lists (that means all columns in a list will scroll horizontally).
- No support for drag & drop from/to the desktop.
- Modal dialogs may not be supported completely (however the cases that are not supported have been reduced dramatically, allowing the Web Client to be used in most scenarios).

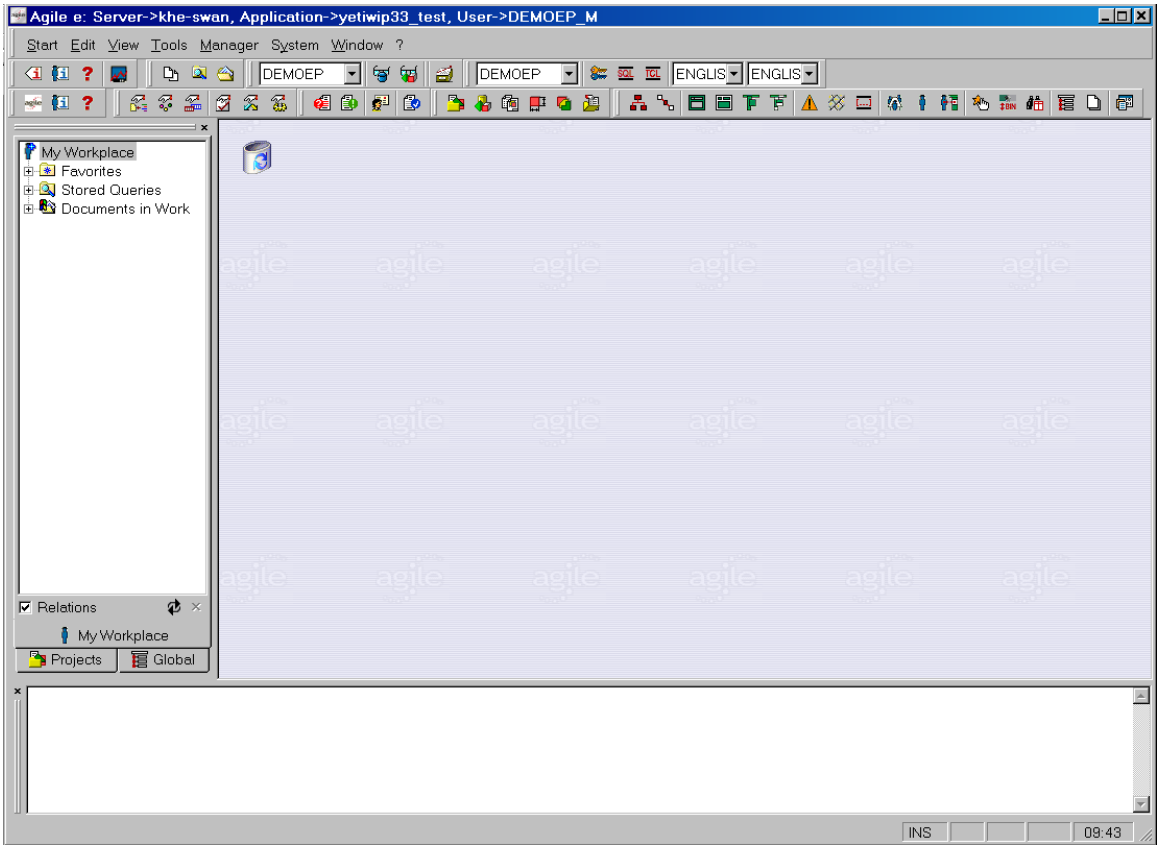
The Web Client does not support M-CAD, E-CAD, EDA, AutoVue integrations, and the Office Suite features. However, the Web Client can be used with integrations that are processed on

the server, e.g. the SAP Link.

The user interface of the Web Client is dynamically defined by the metadata in the repository, except for some specific components, including the integrated Browser windows, the Search Panel and the Wizards.

Windows Client

The Windows Client enables access to most functions of Agile e6. Administrators and Customizers can use the Windows Client.



The Windows Client allows users to drag & drop between Agile e6 and other Windows applications (e.g. Windows Explorer, Microsoft Excel, etc.).

The Windows Client fully supports available M-CAD, E-CAD, EDA integrations, and most other types of integrations (e.g. SAP Link).

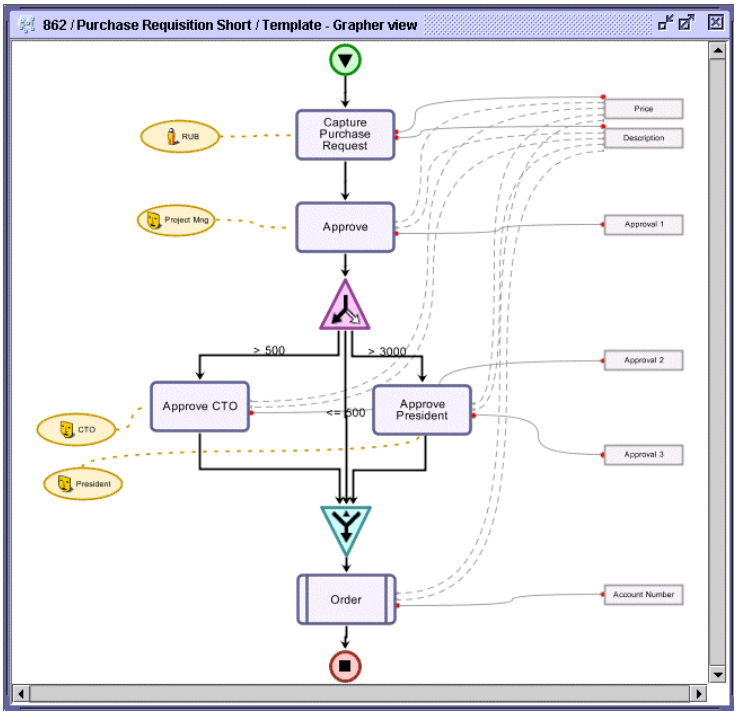
The user interface of the Windows Client is dynamically defined by the metadata in the repository.

Client Components

Workflow Editor

Agile e6 includes a workflow solution that allows automating business processes.

The definition of a workflow process consists of the activities, the resources responsible for the execution of the activities, and the routing. Workflow processes are graphically defined with the Workflow Editor, as depicted in the following figure:

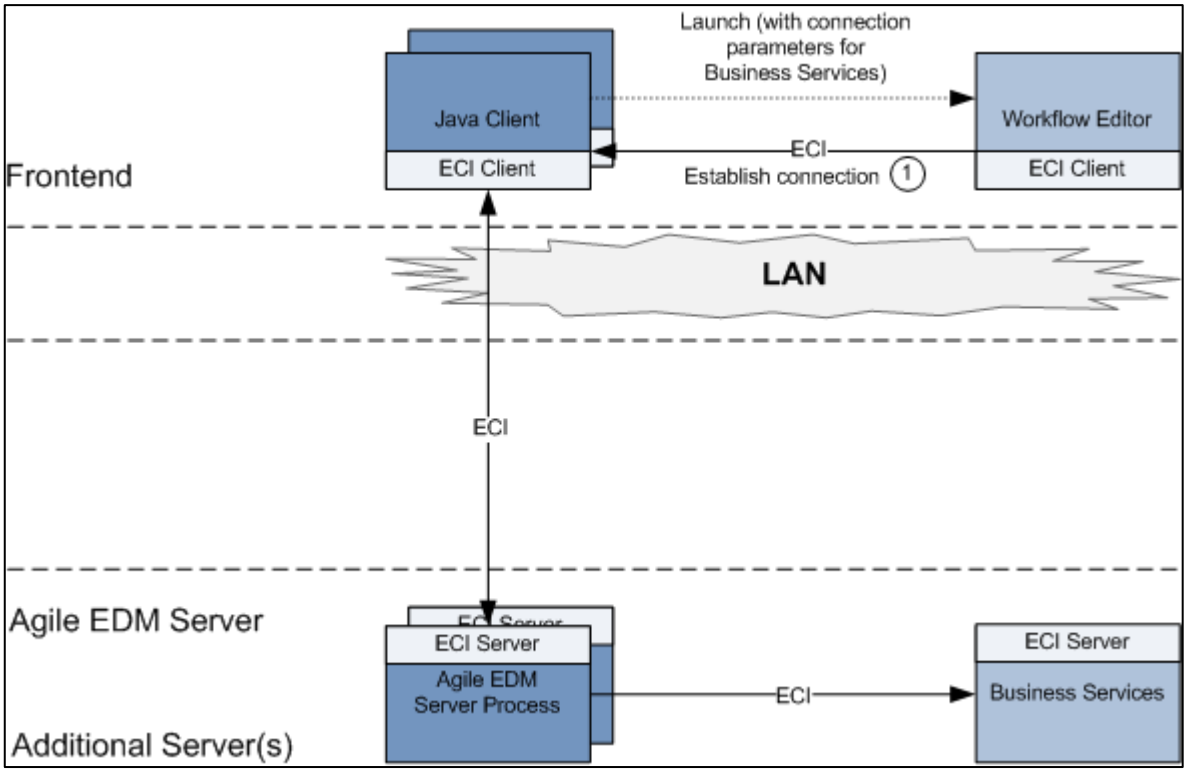


The Workflow Editor communicates with the Business Services, which include the Workflow Engine and the Permission Manager. All data related to a workflow process are stored in the database, thus ensuring the integrity of the system.

The Workflow Editor can be launched from the Windows and the Java Client. To start the Workflow Editor from Windows, an extra Java process is started.

Line of Communication when Using the Workflow Editor – Java Client

The communication between the Java Client, the Workflow Editor and the Agile e6 application server process is depicted in the following figure:



The following steps are executed when the user launches the Workflow Editor from the Java Client:

- The Java Client launches the Workflow Editor. Parameters with information about the Business Services are passed.
- The Workflow Editor connects to the given Business Services through the Java Client and Agile e6.1 Server.

The following table contains the details about the relevant lines of communication:

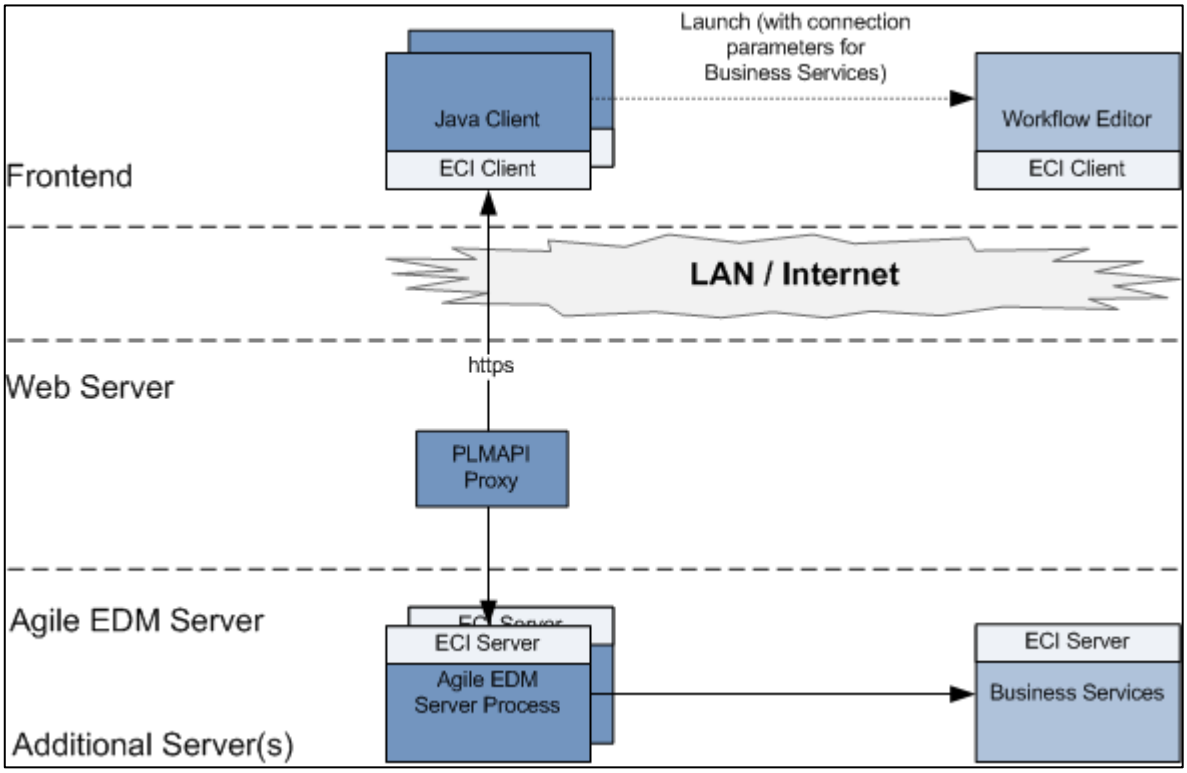
Line of Communication	Type	Port or Range of Ports
(1)	Application specific protocol based on Sockets	Port 19997 by default (can be configured during installation)

With HTTP Support

The Workflow Editor is a component in the Java client that connects to the business service through an ECI protocol.

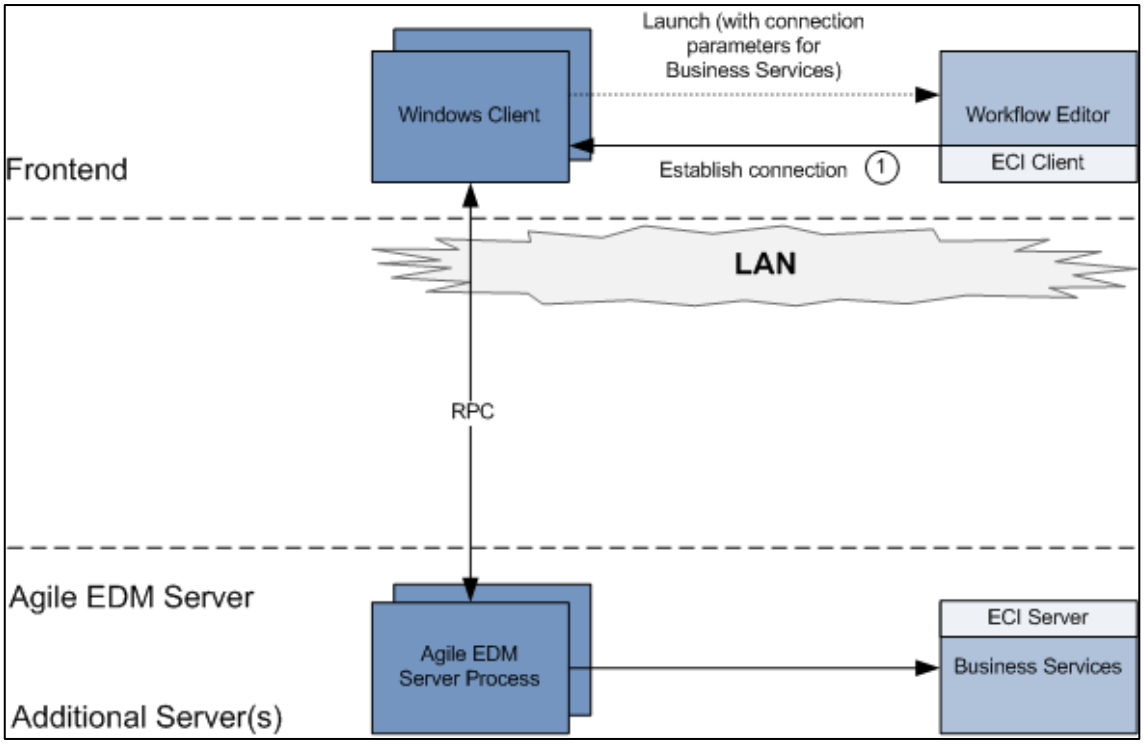
Once the Java Client is communicating in a WAN scenario, the firewall restrictions also apply

for the Workflow Editor, thus the Workflow also needs to communicate through the PLMAPI proxy as described for the Java client in "Lines of Communication with HTTP support".



Line of Communication when using the Workflow Editor – Windows Client

The following figure depicts the lines of communication when using the Workflow editor:



The following steps are executed when the user launches the Workflow Editor from the Windows Client:

- Windows Client launches the Workflow Editor.
- The Workflow Editor connects to the given Business Services (communication line (1)).

The following table contains the details about the relevant lines of communication:

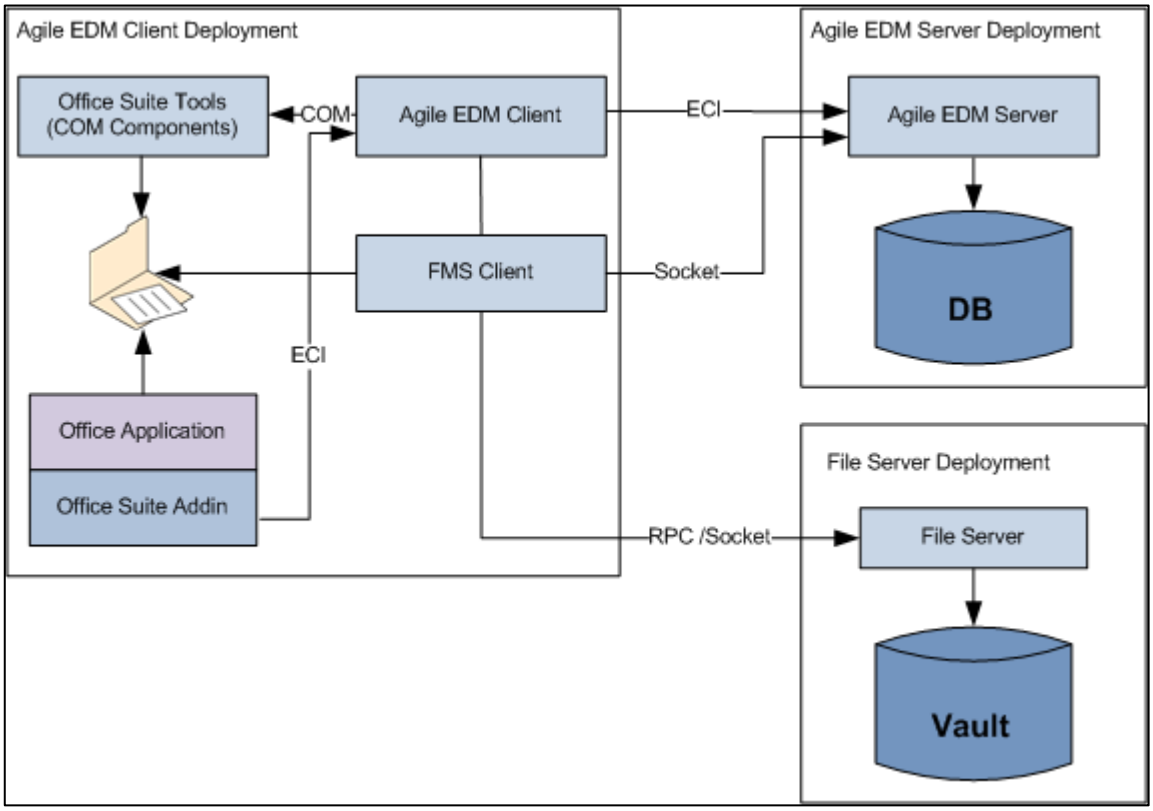
Line of Communication	Type	Port or Range of Ports
(1)	Application specific protocol based on Sockets	Port 19997 by default (can be configured during installation).

Office Suite

The Office Suite module supports all user needs for managing Office documents. This includes all types of text documents, graphic data, and multimedia files.

In addition to the MS Office applications (Word, Excel, PowerPoint, Visio, 2003 and later), a direct integration is also available. The exchange of document properties with Agile EDM Server can be defined in the Office Suite configuration. Thus, the properties information is automatically entered into the fields of the document mask when a document is saved. When

editing a document it is also possible to display the content of fields in the document properties and thus insert the information into the document using MS Office features



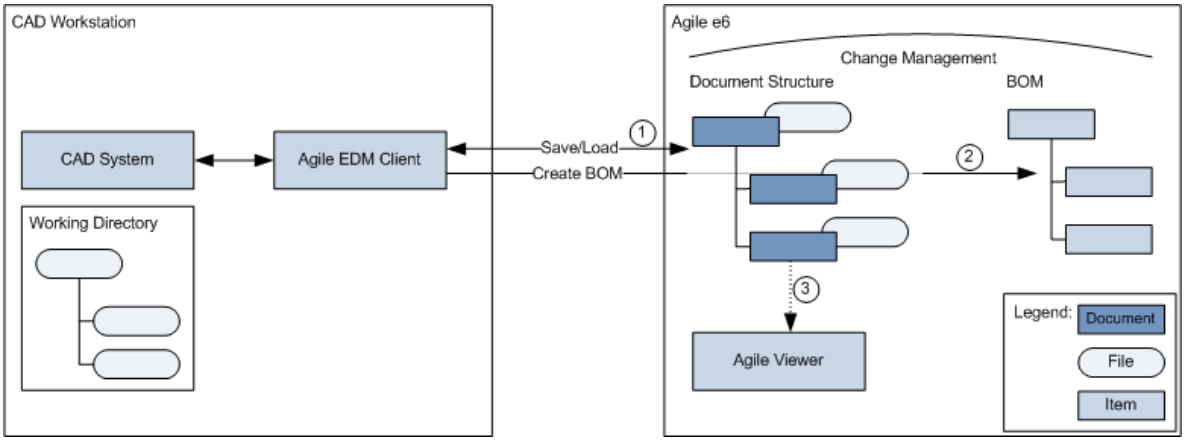
The Office Suite provides the following:

- **The Office Suite Tools** - These are a set of COM components which are accessible via the Microsoft COM interface. These tools are used to exchange the document properties between the Agile EDM server and the Office document.
- **The Office Suite Addin** – It integrates the Office Suite Toolbar/Ribbon into the Office application. With the Addin the user can execute operations like Load / Save the Office document from / into the Agile EDM system.

Note The Office Suite is supported for the Java Client and Windows Client and uses the standard mechanisms to access the File Server.

CAD Integrations

Agile e6 CAD Integrations provides data and process integration between CAD applications and Agile e6. They allow CAD designers and engineers to capture and control the data representing a primary source of the product record. The integrations work from within the familiar CAD system user interface, providing commands which allow the user to interact with PLM to manage the CAD data.



Primary use cases:

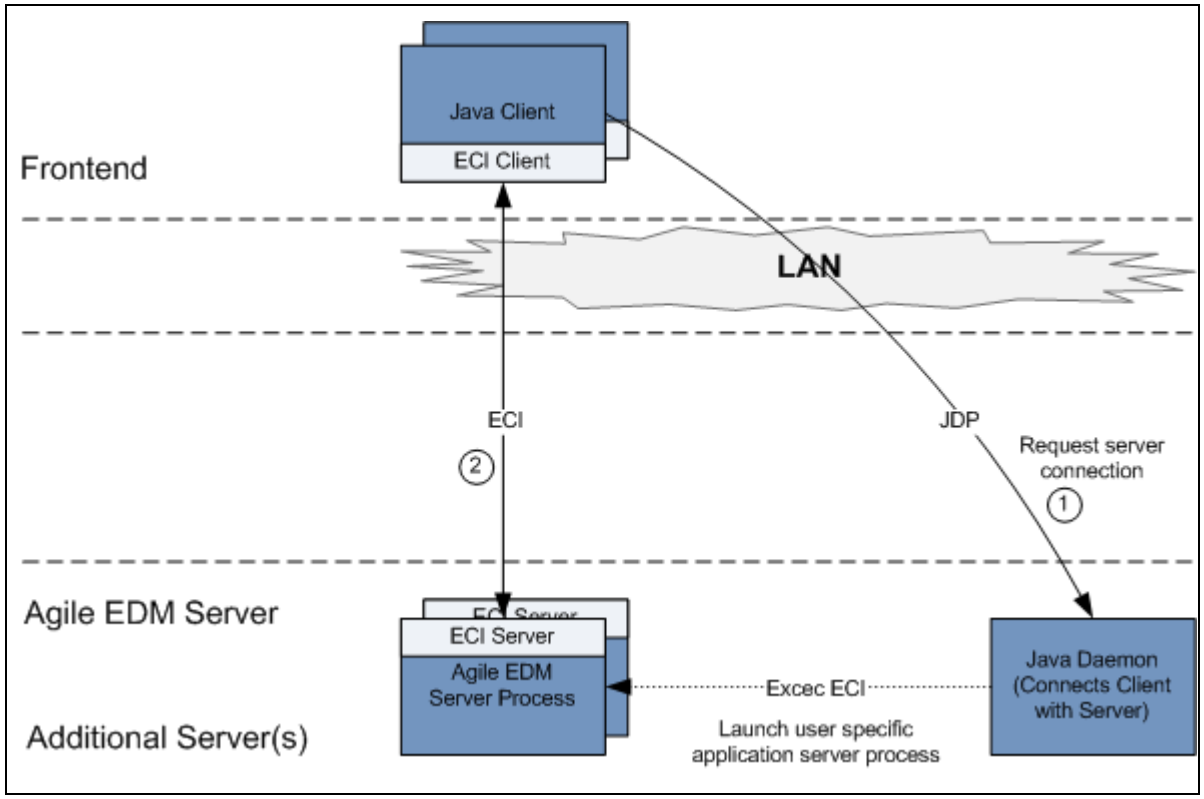
- Save and Load CAD files
- Create BOM
- View CAD files (optional)

Chapter 4

Java Client Communication

Line of Communication when Launching the Java Client

The lines of communication when launching the Java Client are depicted in the following figure:



The following steps are executed when you launch the Java Client:

1. The Java Client connects to the Java Daemon, requesting to launch a user specific Application Server Process and to return the connection information for this process (communication line (1)).

2. The Java Client connects to the given Application Server Process and starts communication (communication line (2)).

The communication line between the Java Client and the Application Server Process utilizes the ECI (Enterprise Communication Interface) protocol. This is same as that used by an M-

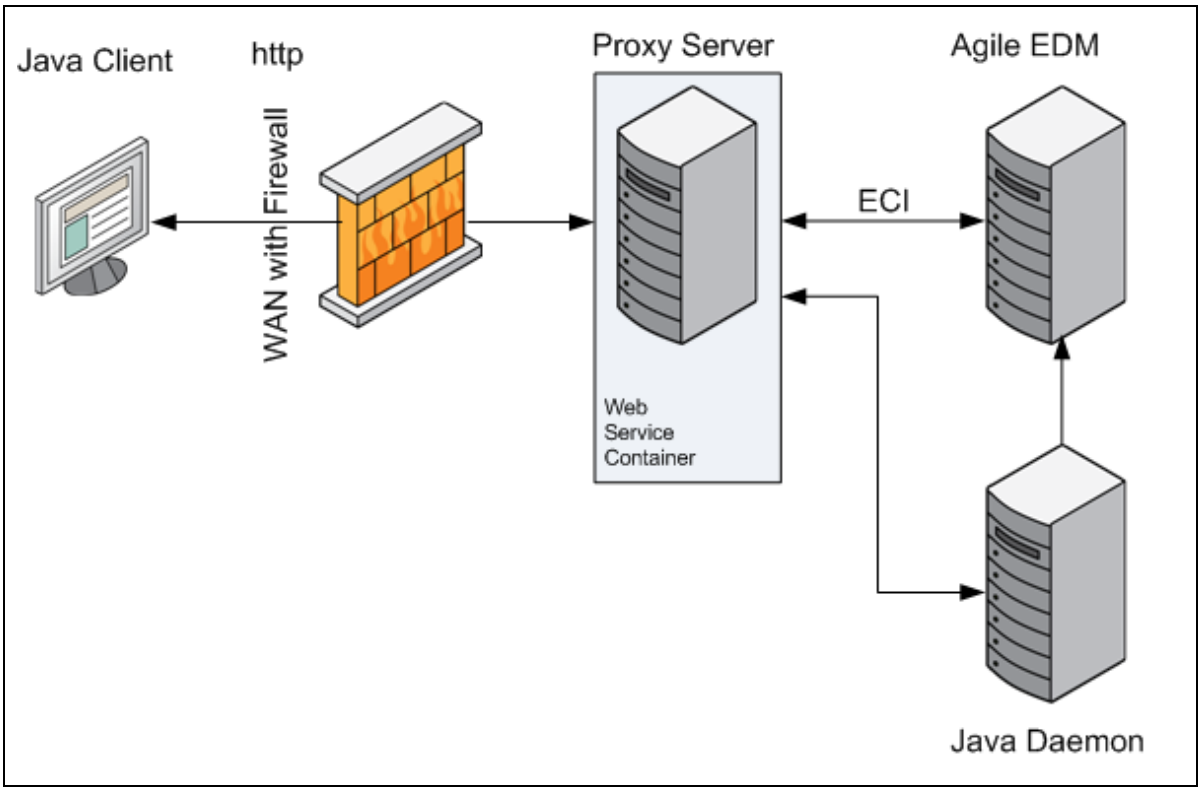
CAD system (for example) to communicate with an Agile e6 client on the front-end.

The following table describes about the relevant lines of communication:

Line of Communication	Type	Port or Range of Ports
(1)	Application specific protocol based on Sockets	Port 16077 by default (can be configured during installation)
(2)	Application specific protocol based on Sockets	One unprivileged port per Application Server Process from range as configured for Java Daemon (5000 to 5999 by default)

Firewall Friendliness

The new Firewall Friendliness allows the communication to Agile e6 application server through firewalls (via http). This is accomplished by using ECI via Web Service with SOAP communicating through http.



The Java Client sends an HTTP requests to the proxy server. First, the proxy server connects with the Java daemon to get the address for the Agile e6 server. Then it connects itself with the Agile e6 server for further calls from the Java Client.

Note The proxy server is built from a web service container (e.g. oc4j) and web services.

Note Using the new Firewall Friendliness can have an influence on the performance of the Java Client.

Line of Communication with HTTP Support

The connection between JavaClient and Agile e6 through HTTP occurs through a component called PLMAPI Proxy, which provides the HTTP communication capability.

The JavaClient basically makes two connections:

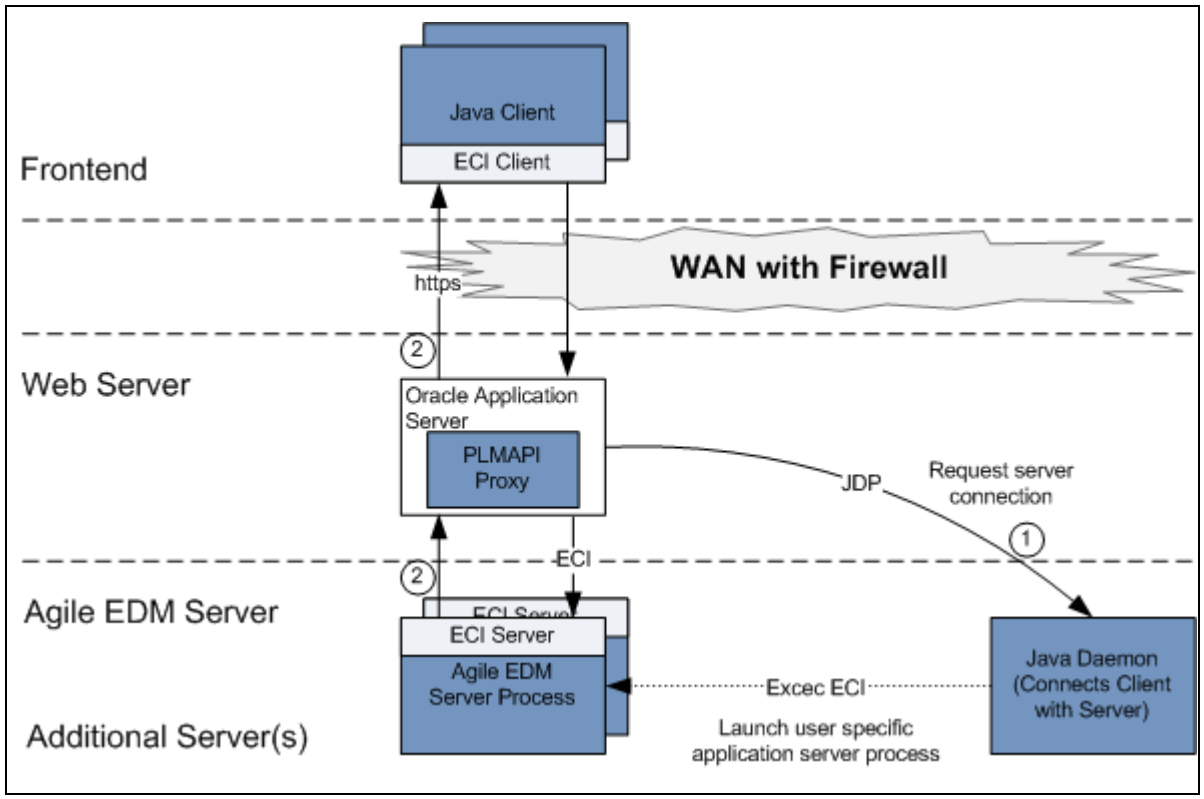
1. With JavaDaemon - To bootstrap the actual communication.

2. Connects to Agile e6 - To start the actually communication with Agile e6 server and then close the connection with JavaDaemon.

The connections are socket based and use a proprietary protocol. In a WAN scenario, this is not possible as there is almost always a firewall which restricts the communication. But as a standard, the firewall does allow HTTP communication through standard port number 80 for HTTP, and 443 for HTTPS.

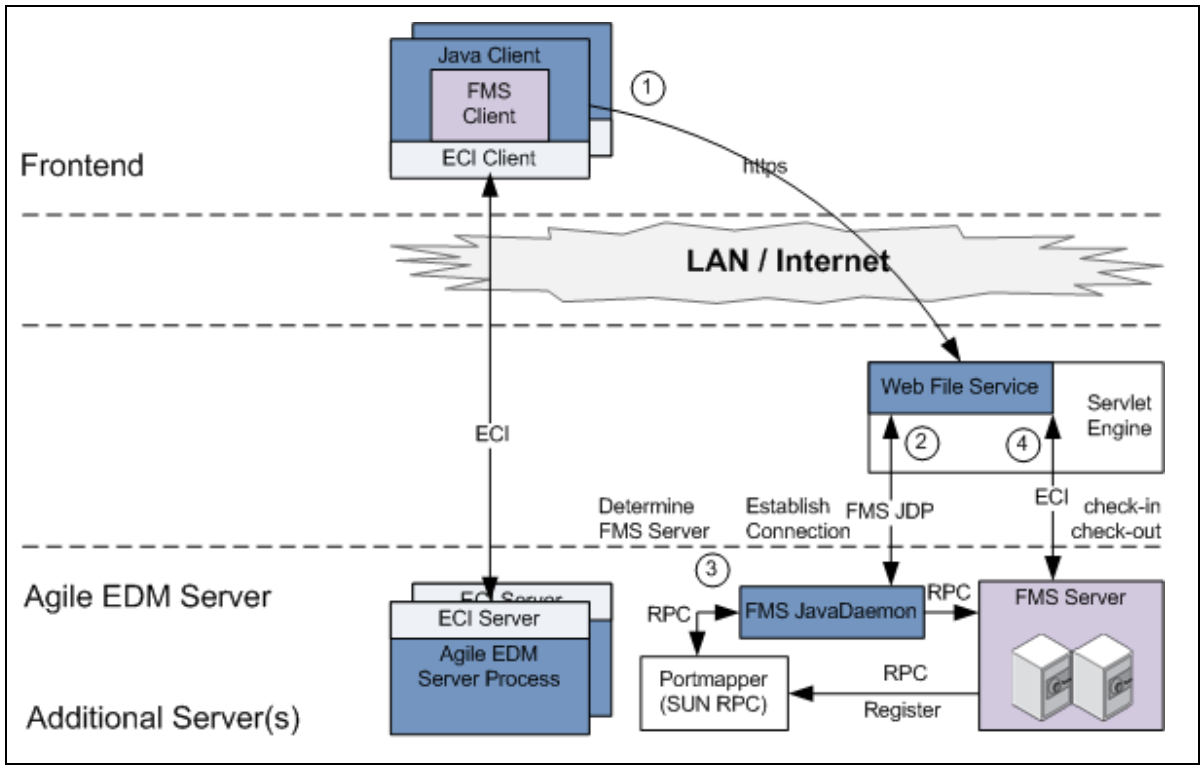
In case of a HTTP communication, there is a web server between Java Client and Agile server components (Agile e6 server and JavaDaemon) inside the firewall. The PLMAPI Proxy should be deployed in this web server.

The Java Client connects to the PLMAPI proxy and the PLMAPI bootstraps the connection for the Java Client by first connecting to Java Daemon, and then to Agile e6 server by using the credentials sent from Java Client. Once the PLMAPI has finished the connection bootstrapping, the Java Client can communicate through PLMAPI Proxy with Agile e6 server.



Line of Communication for File Access (using FMS)

The communication between the Java Client, the Agile e6 application server process, the Web File Service and the FMS Server is depicted in the following figure:



Note In case of the Java Client installation with webstart, the line of communication can either be via the http Client or via the FMS Client.

Note By default, when the Java Client is locally installed, the native FMS Client is also installed and available.

The following steps are executed when the user performs a check-in or a check-out operation in the Java Client:

1. The embedded FMS Client connects to a Web File Service, which is hosted by a servlet engine (communication line (1)). The address of the Web File Service is taken from the configuration of the corresponding vault in the Agile e6 database.

   For a check-out operation, the FMS Client sends the corresponding request parameter to the Web File Server.

   For a check-in operation, the FMS Client sends the corresponding file to the Web File Server.

2. With the information about the FMS Server, the Web File Service connects to the Portmapper to determine the port of the corresponding FMS Server (communication line (2)).

3. The Web File Service can now request the FMS Server to create an individual thread and return the connection parameter (communication line (3)).

4. The file data is transmitted between the Web File Service and the FMS Server thread (communication line (4)). The Web File Service itself returns the call to the embedded FMS Client.

Files are not stored temporarily, but transferred directly between the front-end and the FMS Server.

The following table contains the details about the relevant lines of communication:

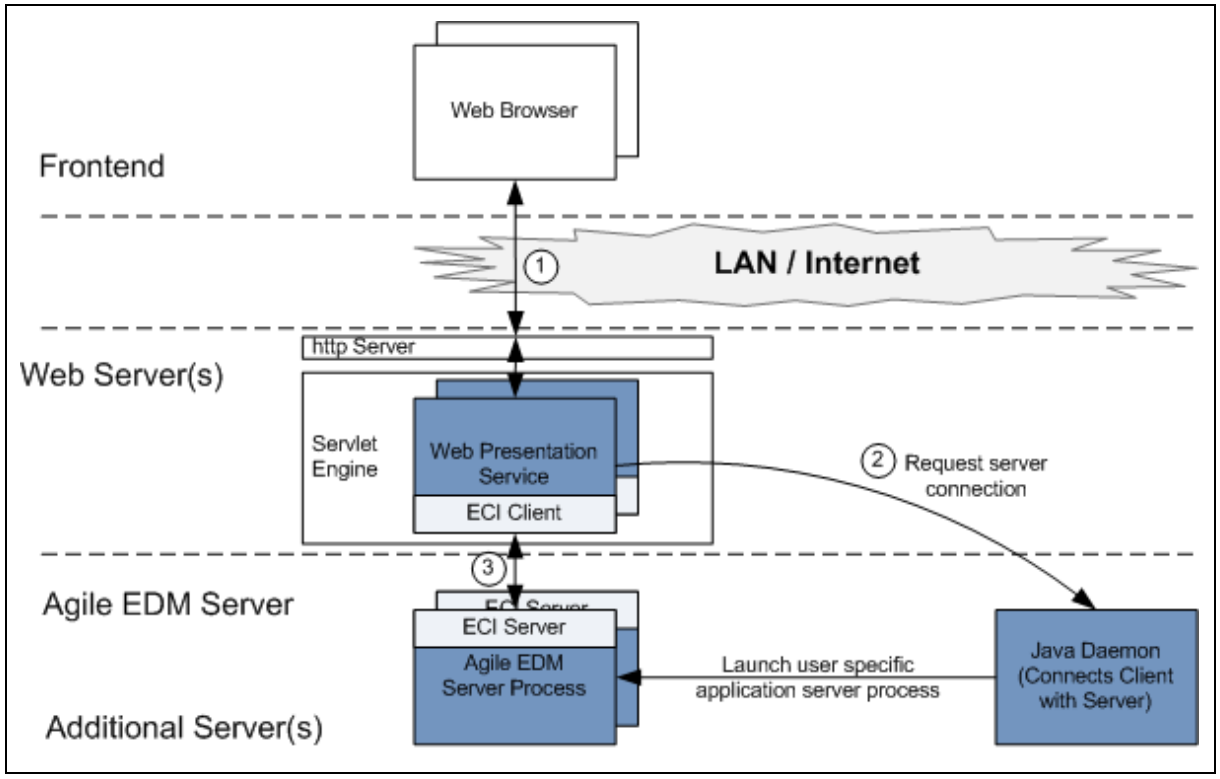
Line of Communication	Type	Port or Range of Ports
(1)	http (or https)	Port 8088 by default (can be configured during installation)
(2)	RPC	Port 111 (used by Sun RPC)
(3)	RPC	One unprivileged port for each file FMS Server (typically only one), assigned by Portmapper – see (2)
(4)	Application specific protocol based on Sockets	One port for each concurrent file upload or download in range from 52517 to 53516

Chapter 5

Web Client Communication

Line of Communication When Launching the Web Client

The lines of communication when starting the Web Client (respectively when accessing the corresponding URL in a web browser) are depicted in the following figure:



The following steps are executed when you launch the Web Client:

1. The web browser sends a request to the Web Presentation Service (communication line (1)) that is hosted by the servlet engine.

   Note It is possible to configure the web browser in a way that it bypasses the http server and communications directly with the servlet engine. This configuration may be configured if the web browser is not used for other (http) services but for the Agile e6 Web Client.

2. The Web Presentation Service connects the Java Daemon, requesting to launch a user specific Application Server Process and to return the connection information for this process (communication line (2)).

3. The Web Presentation Service connects to the given Application Server Process via ECI (communication line (3)).

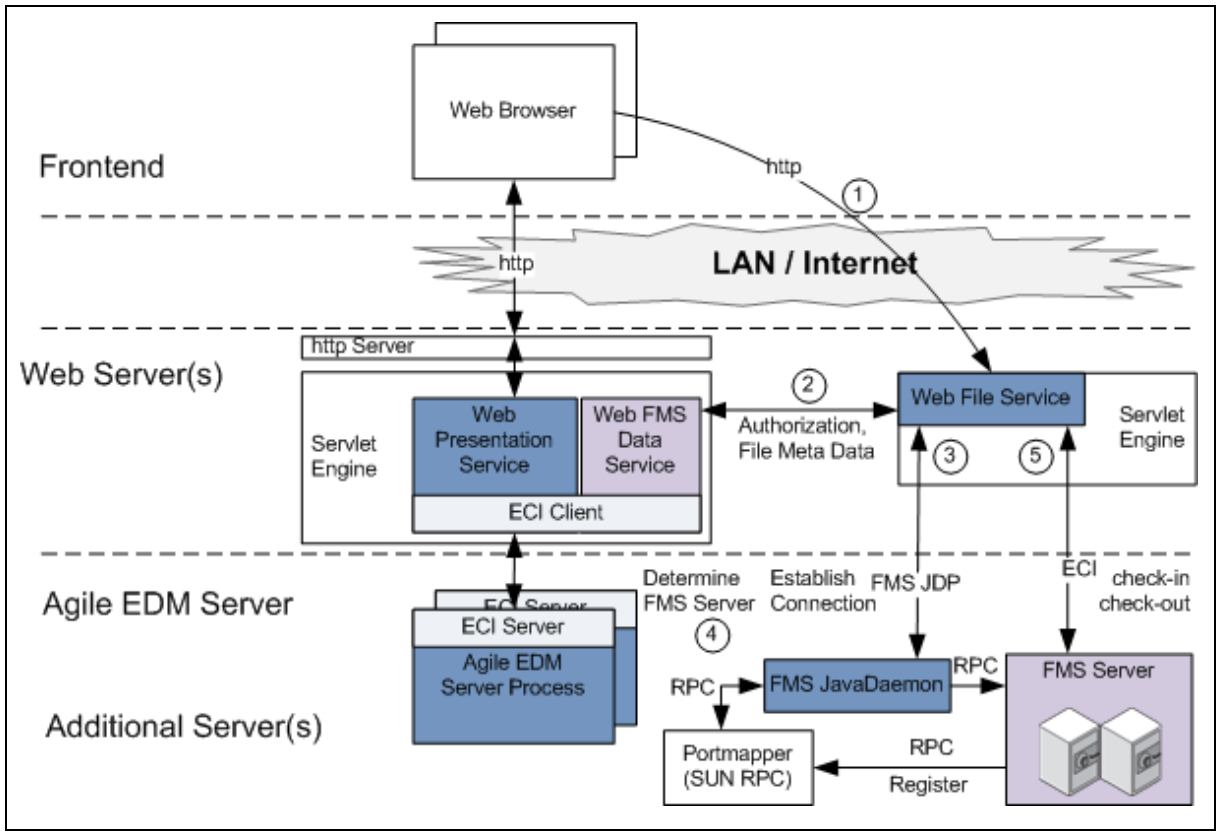
The Web Presentation Service will process requests from the web browser by translating these into ECI calls to the Application Server Process. The Web Presentation Service is also responsible for the rendering of lists and forms into DHTML (dynamic HTML - HTML enriched with JavaScript). The JavaScript in the HTML pages realizes consistency checks and interactive controls within the Web Client.

The following table contains the details about the relevant lines of communication:

Line of Communication	Type	Port or Range of Ports
(1)	http (or https)	Port 8088 by default (can be configured during installation)
(2)	Application specific protocol based on Sockets	Port 16077 by default (can be configured during installation)
(3)	Application specific protocol based on Sockets	One unprivileged port per Application Server Process from range as configured for Java Daemon (5000 to 5999 by default)

Line of Communication in Case of File Access (using FMS)

The communication between the Web Client, the Agile e6 application server process, the web services and the FMS Server is depicted in the following figure:



The following steps are executed when you perform a check-in or a check-out operation in the Web Client:

1. The web browser sends a request to the Web Presentation Service. This request is recognized as a file check-in or check-out operation and as a result the web-browser is re-directed to the Web File Service (communication line (1)). The address of the Web File Service is taken from the configuration of the corresponding vault in the Agile e6 database.

   For a check-out operation, the web browser sends the corresponding request parameter to the Web File Server.

   For a check-in operation, the web browser sends the corresponding file to the Web File Server.

   Note The Web Presentation Service and the Web File Service can be hosted by the same servlet engine. For performance reasons it is recommended to use two separate servlet engines.

   The web browser can be configured that it bypasses the http server and communications directly with the servlet engine. This configuration may be configured if the web browser is not used for other (http) services but the Agile e6 Web Client.

2. To avoid a misuse of the Web File Service, the re-directed request contains an authorization ticket and the address of the Web FMS Data Service. This service is hosted by the same servlet engine as the Web Presentation Service and they share a single ECI connection to the Application Server Process. The Web File Service provides the authorization ticket and in return receives required file metadata (communication line (2)).

3. With the RPC-number of the FMS Server, the Web File Service connects to the Portmapper to determine the port of the corresponding FMS Server (communication line (3)).

4. Now the Web File Service can request the FMS Server to initiate an individual thread and return the connection parameter (communication line (4)).

5. The file data are transmitted between the Web File Service and the FMS Server thread (communication line (5)). The Web File Service itself returns the call to the web browser.

Files are not temporarily stored on the Web Server, but transferred directly between the web browser and the FMS Server. This avoids possible security issues (files that resided temporarily on the web server are accessible to all users) and increases performance especially in the case of large files, because only a minimum of file write/read operations are performed.

The following table contains the details about the relevant lines of communication:

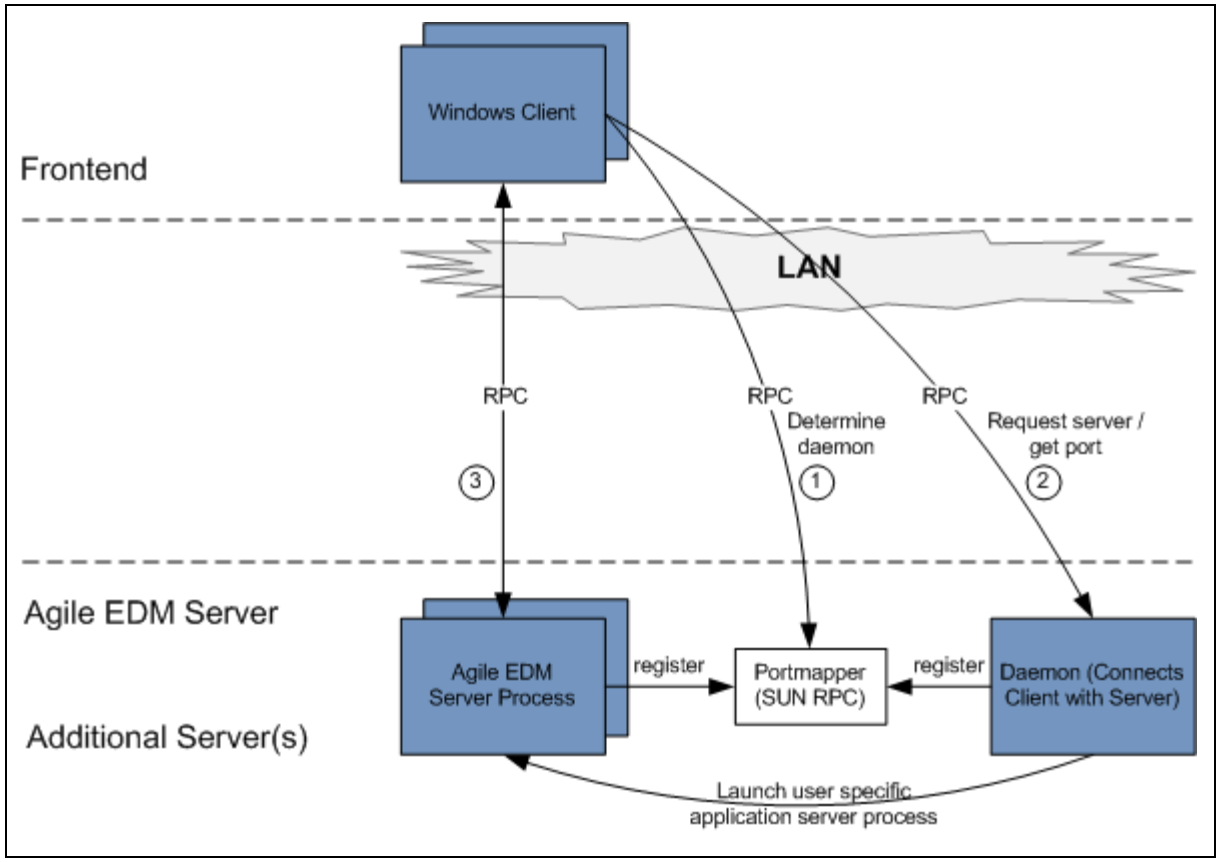
Line of Communication	Type	Port or Range of Ports
(1)	http (or https)	Port 8088 by default (can be configured during installation)
(2)	http (or https)	Port 8088 by default (can be configured during installation)
(3)	RPC	Port 111 (used by Sun RPC)
(4)	RPC	One unprivileged port for each file FMS Server (typically only one), assigned by Portmapper – see (3)
(5)	Application specific protocol based on Sockets	One port for each concurrent file upload or download in range from 52517 to 53516.

Chapter 6

Windows Client Communication

Line of Communication when Launching the Windows Client

The following figure depicts the lines of communication between the Windows Client and the Application Server Process:



The following steps are executed when you launch the Windows Client:

1. The Windows Client connects to the Portmapper to get the address of the required Daemon (communication line (1)). The Daemon had registered at the Portmapper initially and is therefore known to the Portmapper.

2. The Windows Client connects the Daemon, requesting to launch a user specific Application Server Process and to return the connection information for this process (communication line (2)).

3. The Windows Client connects to the given Application Server Process and starts communication (communication line (3)).

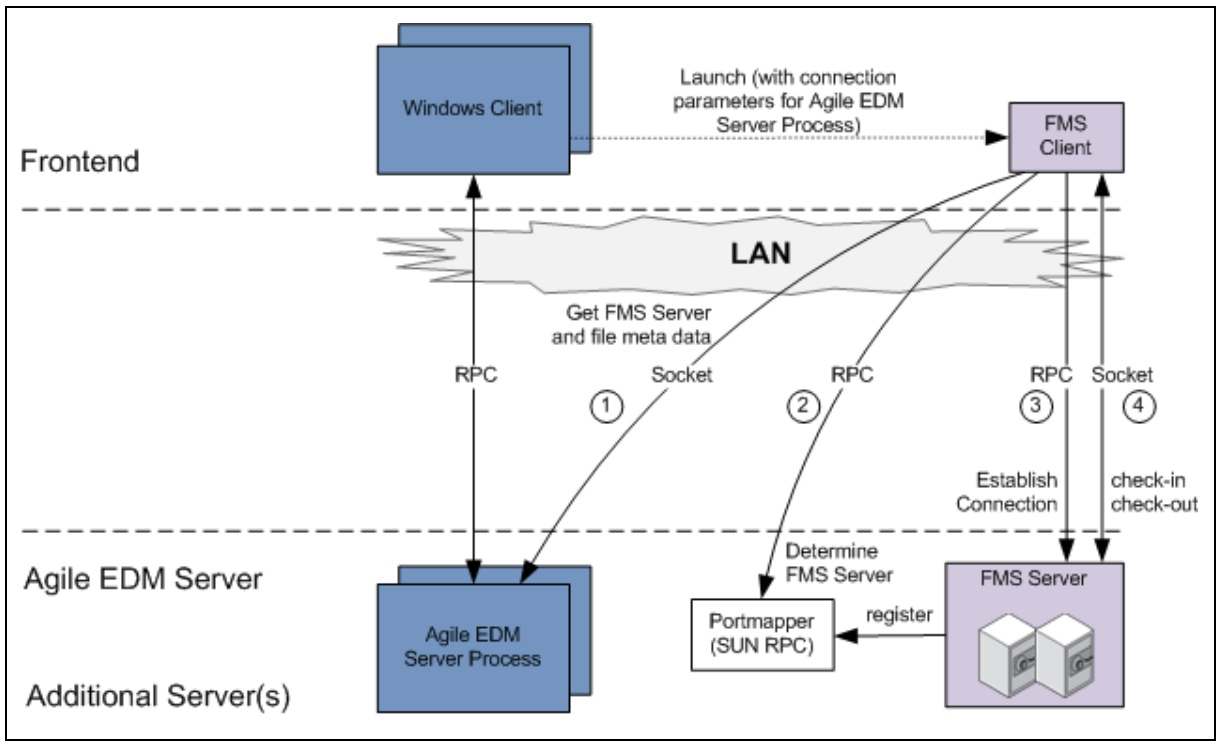
The communication line between the Windows Client and the Application Server Process uses a proprietary protocol that transfers only a minimum amount of data.

The following table describes about the lines of communication between the Windows Client and the Application Server Process:

Line of Communication	Type	Port or Range of Ports
(1)	RPC	Port 111 (used by Sun RPC)
(2)	RPC	A single unprivileged port, assigned by Portmapper – see (2)
(3)	RPC	One unprivileged port for each Application Server Process.

Line of Communication in Case of File Access (using FMS)

The following figure depicts the lines of communication for the File Access using FMS:



The following steps are executed when the user performs a check-in or a check-out operation in the Windows Client:

1. Windows Client launches the FMS Client. Parameters about the current Application Server Process are passed.

2. FMS Client connects to the given Application Server Process to get the corresponding FMS Server data and file Metadata (communication line (1)).

3. With the information about the FMS Server, the FMS Client connects to the Portmapper to determine the port of the corresponding FMS Server (communication line (2)).

   The FMS Client can now request the FMS Server to spawn an individual thread and return the connection parameter (communication line (3)).

4. The file data is directly transmitted between the FMS Client and the FMS Server thread (communication line (4)).

5. The FMS Client terminates after the given timeout period.

The following table describes about the lines of communication for the File Access using FMS:

Line of Communication	Type	Port or Range of Ports
(1)	Application specific protocol based on Sockets	One port for each concurrent file upload or download in range from 51516 to 52515
(2)	RPC	Port 111 (used by Sun RPC)
(3)	RPC	One unprivileged port for each file FMS Server (typically only one), assigned by Portmapper – see (2)
(4)	Application specific protocol based on Sockets	One port for each concurrent file upload or download in range from 52517 to 53516

Chapter 7

Web Services Interface

Agile e6 Web Services expose a subset of the Product Lifecycle Management (PLM) functionalities of the Agile e6 Application. These services support functionalities provided by the PLM modules in the Agile e6 application, such as Item Management, Project Management and many other functions of Agile e6.

Implementation of Agile e6 Web Services adheres to the following principles:

- Well defined, standards based discoverable Interface.
- Java based Web Services Framework using Oracle WebLogic.
- Modularized Agile e6 Schema (XSD) and WSDL for easy maintenance.
- Standards-based WSDL to ensure compatibility across various clients (.NET, Java, and BPEL).
- Bulk APIs wherever applicable for better performance.

Agile e6 Web Services Framework

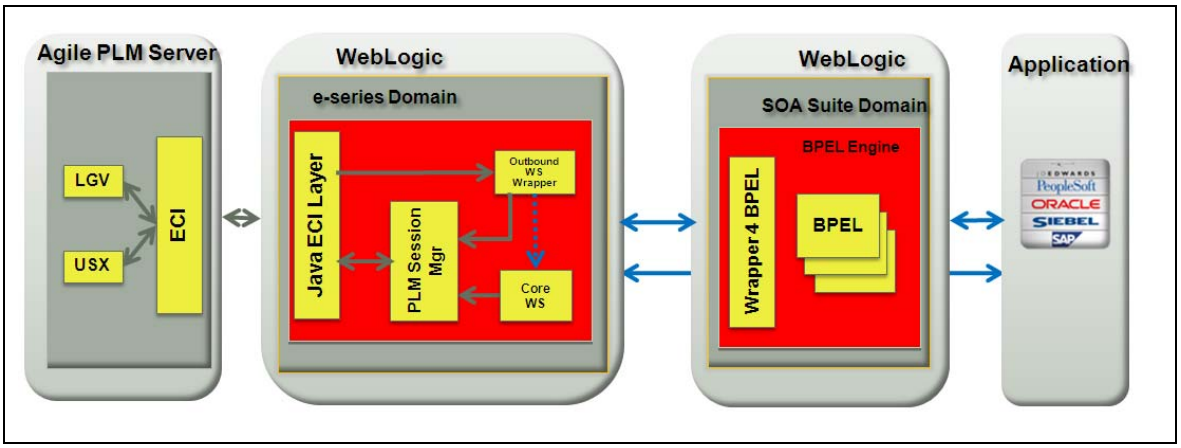
The Web Services Framework is an additional layer on top of Agile e6, which supports inbound and outbound communication based on standard Web Services technology.

The Agile e6 Web Services Framework:

1. Provides the means to call External Web Services from inside Agile e6 LogiView Procedures (outbound direction).
2. Allows the external applications (Web Service Clients) to call the Agile e6 APIs through Agile e6 Web Services.

The Web Service Framework comes with a set of predefined *core* web services, which, out of the box, support the most common integration scenarios like *create PLM object* or *get PLM object*.

Components of Agile e6 Web Services Framework



Agile e6 Web Services are deployed on an Agile e6 WebLogic application domain. The Agile e6 Web Services Framework comprises of the following:

- Web Service Wrapper – to support the outbound web service calls from LogiView Procedures.
- Core Web Services – to support the inbound web service calls mapped into the ECI-API calls.

The Web Services Wrapper Interface

Each wrapper has to implement the *WebServiceWrapper* interface, which prescribes the following method:

```
StringList callWebService(WrapperContext context, CallableParam args)
```

The context contains all relevant information for the wrapper to perform the call. The wrapper then transforms the arguments to an XML payload for the outbound web service and finally makes the call.

If it is an asynchronous web service, the wrapper returns a string list with the correlation ID. Otherwise, it transforms the XML payload returned by the web service into a string list that is expected by the calling Agile e6 server process.

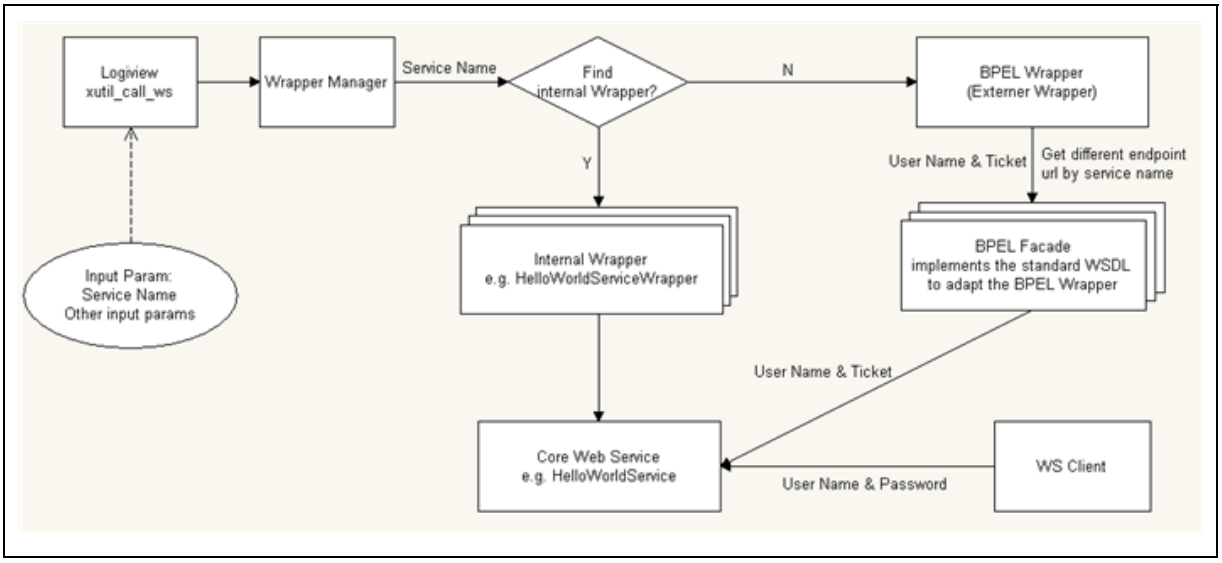
In case the wrapper is implemented in BPEL (or in any other external web service language), you require a special wrapper called the *ExternalWrapper*. This wrapper delegates the call to the respective external web service wrapper.

The BPEL Facade

For the outbound calls, you are required to use the *ExternalWrapper* to call an external web service or a BPEL process. However, you need to first implement an interface to adapt the

External Wrapper. This interface is called as *BPEL facade*.

Normally, this facade should be a BPEL process that you implement. In this facade, you can invoke an external web service or BPEL process and create their business logic. All the facades should implement a standard WSDL. The External wrapper uses this standard WSDL to generate the proxy. Then it can use different endpoints to call different BPEL facades.



Endpoint Configurations for the External Wrapper

All the endpoints for the External wrapper are defined in the properties file **ExternalWrapper.properties.** This file resides in the *APP-INF/classes* directory alongside the *application.properties* file for the Web Services.

The Agile e6 PLM Session Handling

Every call of an Agile e6 Core Web Service needs an Agile e6 PLM server instance. It is very important to limit the number of Agile e6 server instances to reduce the resource loading on the server. This is handled by the following mechanisms:

- HTTP Session – The web service first tries to find an Agile e6 server instance assigned to the HTTP session of the current web service call. If it is found, the web service call uses the existing Agile e6 server instance.

- PLM Ticket – A PLM ticket is returned in the response of a core web service operation. This ticket can be used to access the same Agile e6 server instance that created the ticket.

  While authenticating a web service call, if a ticket is passed instead of the password, the session manager uses the Agile e6 server instance, even if no Agile e6 server is assigned to the HTTP session.

The PLM Ticket mechanism is the only way to let the calls to different core web services, such as Metadata or BusinessObject web services use the same Agile e6 server instance.

A web service client should use the PLM ticket as soon as it has called the first operation.

This is the only way to share an Agile e6 session between two different Core Services.

To free an Agile e6 server instance assigned to a web service session, the client calls one of the *closeSession* operations (every core web service provides this function) with the PLM ticket as the *password*. This shuts down the Agile e6 server instance and frees up the server resources.

The Agile e6 PLM Session Manager

The Agile e6 PLM Session Manager lets you manage the PLM Session objects which are used to keep the existing connections and user contexts to the Agile e6 server.

The key to an existing PLM Session object is the **session ID**, which is generated by PLM Session Manager.

The **PLM Session** provides connection to Agile e6 server.

To retrieve a PLM Session, *PlmTicket* is provided. When a new PLM Session is created, the PlmTicket is set to the PLM Session, which is then set into the SOAP message to the client side.

The life cycle of PLM Session is as same as the given *HttpSession*. The timeout of an *HttpSession* is specified in **web.xml** with the following information:

```
<session-config>
            <session-timeout><time-in-second></session-timeout>
</session-config>
```

The PLM Ticket

A Response contains a string that can be used in subsequent calls. This string is called the *PLM Ticket*. The ticket gives the caller access to the same Agile e6 instance (PLM server) that was used in the last request. The ticket remains valid only as long as the Agile e6 instance is running. After obtaining a ticket, the client code needs to configure the port by setting the ticket string as password.

The PLM Ticket improves the web services performance and simplifies the session management, If different Web Services are used in a use-case flow, which is very likely, the ticket returned by the responses of one service operation (say *Configuration.setUserContext)* is used as a password when client makes a call for another service operation (say *BusinessObject.getObjects)*.

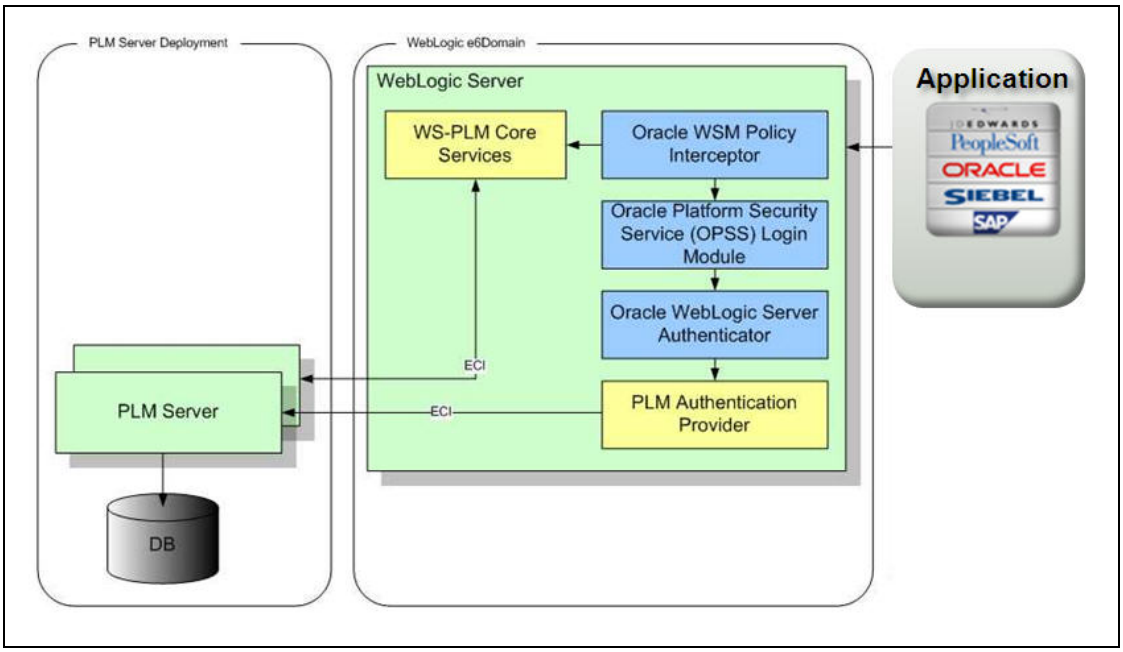
The ticket sharing among different client ports eliminates the need for the server to start new Agile e6 sessions, which would result in new Agile e6 servers being started.

Agile e6 PLM Authentication Provider

The Agile e6 *PLM Authentication Provider* is required to authenticate the incoming Web Requests. It is also used for providing the *PLM user name* and *PLM Password* to the web services to connect to the Agile e6 application.

The authentication provider is called by the *WebLogic Security Framework*. The *Security Realm* of the domain has to be configured to use the Agile e6 PLM Authentication Provider.

The following diagram depicts the authentication mechanism:



The PLM authentication provider uses the standard mechanism of the WebLogic server and can be configured with the WebLogic Administration Console.

The Authentication Provider is installed at the time of Agile e6 application installation. During this installation, the PLM User and PLM password are set to an empty value. These cannot be set with the batch installer; hence, they have to be added manually with the WebLogic Console.

For complete details on Agile e6 application installation, including the preliminary settings of the authentication provider, refer to the *Agile e6 Server Installation Guides*.

Agile e6 Web Services Authentication and Performance

In the implementations where scalability is critical, a lightweight context management facility for

authentication is available and its use is recommended. With this facility, authentication is managed using a combination of *user credentials* and a `sessionID` token (the standard HTTP session ID maintained by the web container):

- When user credentials are presented in the SOAP header of a Web Service request, formal authentication is performed prior to the application execution of the Web Service operation. If the authentication succeeds, the operation proceeds and a special `SessionID` token is placed in the SOAP header of the Web Service reply.
- Whenever the `sessionID` is included by the client in subsequent Web Service requests, that `sessionID` will be used to restore cached session information, thus bypassing the substantially more time consuming process of re-executing the authentication.

Note When presented with both the `sessionID` and a valid set of user credentials, an attempt is made to use the `sessionID` before resorting to the user credentials and re-authentication. As expected, the session that is being tracked by the `sessionID` is subject to expiration and other security checks.

The facility is a distinct alternative to the basic authentication standard described by Web Services Security. Using the `UserName` token as provided in WS-Security, while fully supported as part of Agile e6 WSI Basic Profile compliance, will not yield the same benefit as using the higher-performance session optimization facility provided by the Agile e6 implementation.

Bulk Processing of Requests

Most of the Agile e6 web service operations support bulk processing of requests. The operations with one request input object and one response output object can be configured to process multiple or bulk requests and corresponding responses.

Handling the Bulk Requests

A bulk request contains a list of requests for the non-bulk operation. These requests are executed one by one, and each response is stored in the result list of the bulk response object.

All requests contained in the bulk request list are executed in sequence using the order of the list.

The bulk requests can be configured to either stop on the first failure, or to continue regardless of a failing request.

If a request fails with a response FAILURE, the loop will be aborted if the *stopOnFailure* member of the bulk request is set. If *stopOnFailure* is not requested, the status of the bulk response is set to *PARTIAL_SUCCESS*. This requires you to look into each response in the list to check the individual status.

However, if a request fails with a *PlmFault* (or any other exception), the bulk processing is aborted and the list of requests processed until the fault occurred is returned. Additionally, the causing fault is returned in the bulk response.

If the bulk request does not contain any request, a WARNING response is returned.