

Oracle® Agile

Engineering Data Management

Architecture Guide for Agile e6.1.3.0

Part No. E50963-01

January 2014

Copyright and Trademarks

Copyright © 1995, 2014, Oracle and/or its affiliates*. All rights reserved.*

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

This documentation is in preproduction status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your beta trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

CONTENTS

Copyright and Trademarks iii
Preface vii

Agile e6 System Architecture Overview 1

Object Oriented Repository 4
Installation Scenarios 5
Single Site Installations 5
Multiple Site Installation 7

J2EE Deployment Architecture 9

WebLogic Deployment 9
Remote Site Deployment Structure 9

Application Server Components 11

Business Services 11
File Management Services 11
File Management Service (FMS) 11
Web Fileservices 13
FMS Java Daemon 15
Java Daemon 15
PLM-API Proxy 15
View Server (AutoVue) 15
Batch Client 17
LDAP 18

Application Clients 21

Java Client 21
Web Client 22
Windows Client 23
Client Components 24
Workflow Editor 24
Office Suite 27
CAD Integrations 28

Java Client Communication 31

Line of Communication - Launching the Java Client 31
Firewall Friendliness 32

Line of Communication - HTTP Support 33
Line of Communication - Kerberos Based Single Sign-On 34
Line of Communication - File Access (using FMS) 36
Local Installed Java Client 36
WebStart Deployed Java Client 37
Line of Communication - Distributed File Management (DFM) 38
Line of Communication - Firewall Friendly DFM 40
Web Client Communication 43
Line of Communication - Launching the Web Client 43
Line of Communication - File Access (using FMS) 44
Line of Communication - DFM 46
Windows Client Communication 49
Line of Communication - Launching the Windows Client 49
Line of Communication - File Access (using FMS) 50
Web Services Interface 53
Agile e6 Web Services Framework 53
Components of Agile e6 Web Services Framework 54
The Web Service Wrapper Interface 54
The Agile e6 Session Handling 55
Agile e6 PLM Authentication Provider 57
Agile e6 Web Services Authentication and Performance 57
Single Sign-On with WebLogic and SAML Web Service Authentication 58
File Handling in a DFM Infrastructure 60

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 Application Clients. Please be aware that there are more client side components with service specific client side functionalities (e.g. Workflow Editor, File Server Client).

With each client process launched, an EDM server process is started in parallel. The EDM server process provides the core Agile e6 functionalities, like Item Management, BOM Management, Document Management, and Change Management, etc. User data (stored in the Oracle database) is accessed by the EDM 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 Agile e6 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, 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.

- Core Web Services

  Technologies for building distributed applications.

- StreamingFileServices

  For the DFM Web Service, continaing uplaodFile and downloadFile operations.

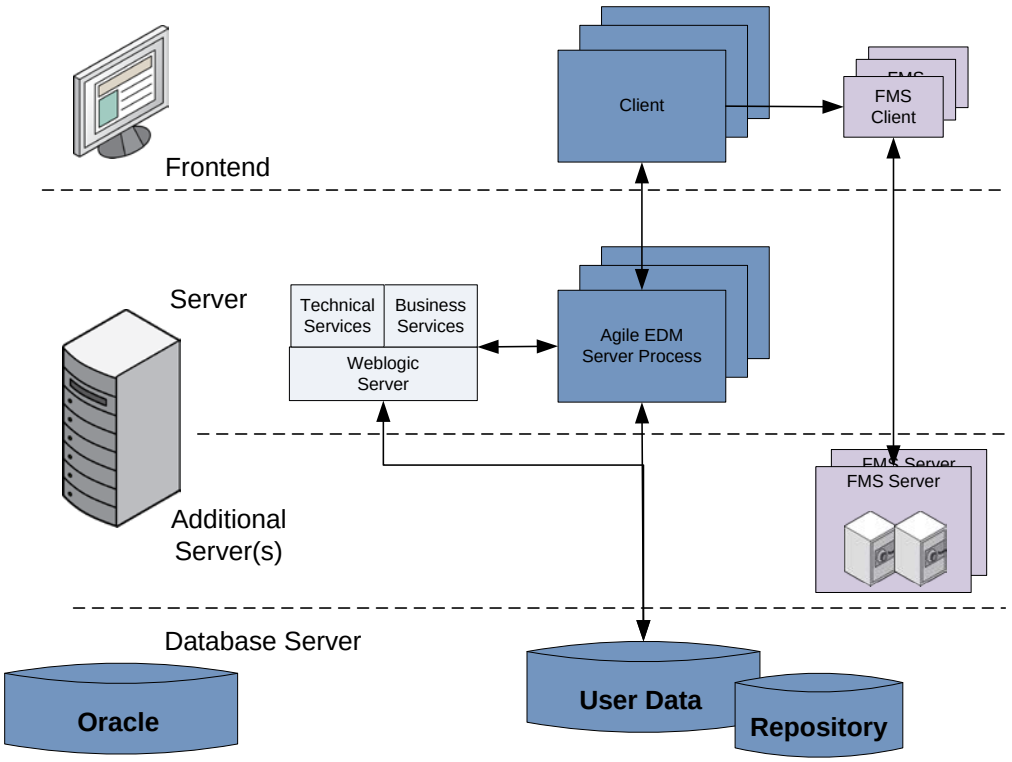
- Administration Client

  Provides the frontend 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 to 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.

- Kerberos Server

  (External) component (e.g. Windows Domain Controller) to provide single sign-on feature.

- Batch Client

  Component to run EDM batch processes.

- Java Daemon

  Component to start an EDM server.

- FMS Daemon

  Component to contact the File Server.

- PLM API Proxy

  Component to support HTTPS connection.

Refer to the Chapter: Application Server 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.

- AutoVue viewer plug-in

- To view various file formats with an integrated viewer.

Refer to the Chapter: Application 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 thus can 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.

Installation Scenarios

Depending on the number and user profiles of involved sites, used Agile e6 components, and the availability requirements for the system, different installation types are possible.

Single Site Installations

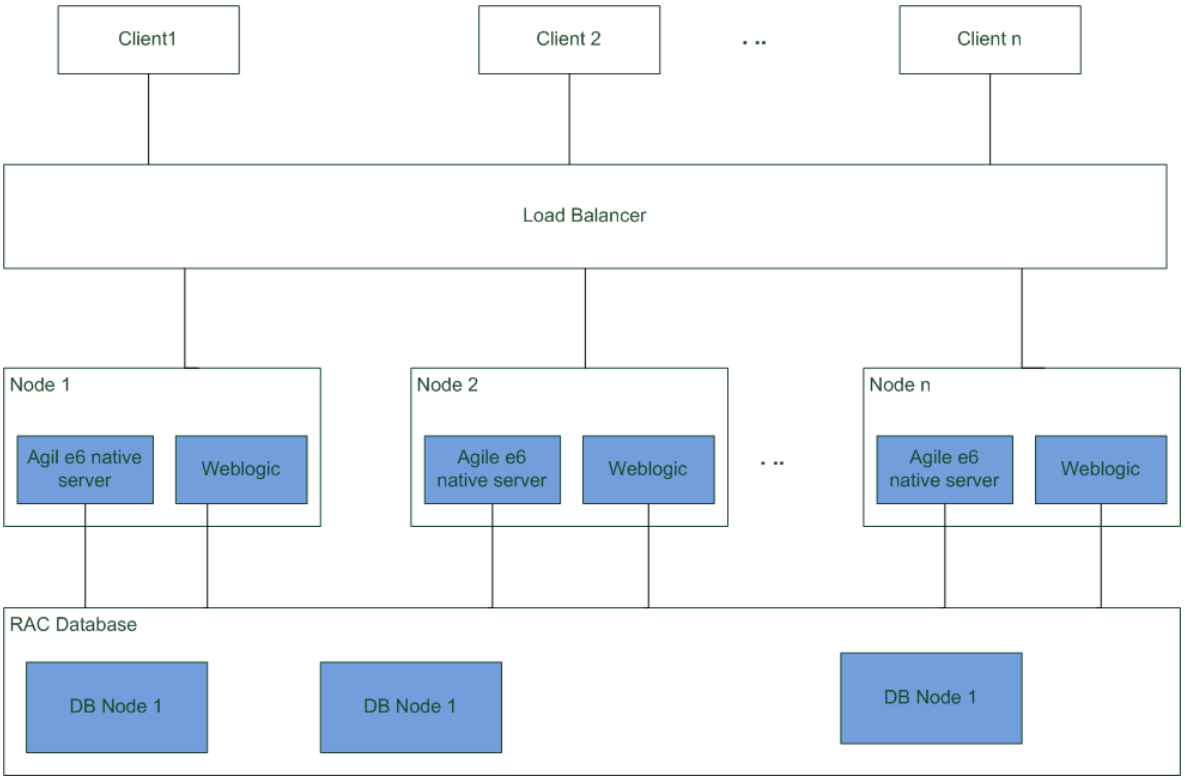
All components are installed on a single site. The different Agile e6 services can be distributed to one or more servers depending on the number of users working with the installation.

The preferred architecture is a set of servers where every server hosts a complete set of all EDM server components that are installed in a cluster, e.g. as NLB cluster where load balancing tools are used to distribute the load to the servers.

Cluster Setup with Load Balancer

The load balancer connects Agile e6 clients to a server node. On this server node the Agile e6 native server processes run as well as the J2EE components deployed in WebLogic application server. If the server node crashes, the user has to re-login. In this case, the load balancer connects the user automatically to one of the running servers. The native server and the WebLogic components open a connection to an Oracle Database. The Oracle Database could be a single or a Real Application Database as shown in the graphic below.

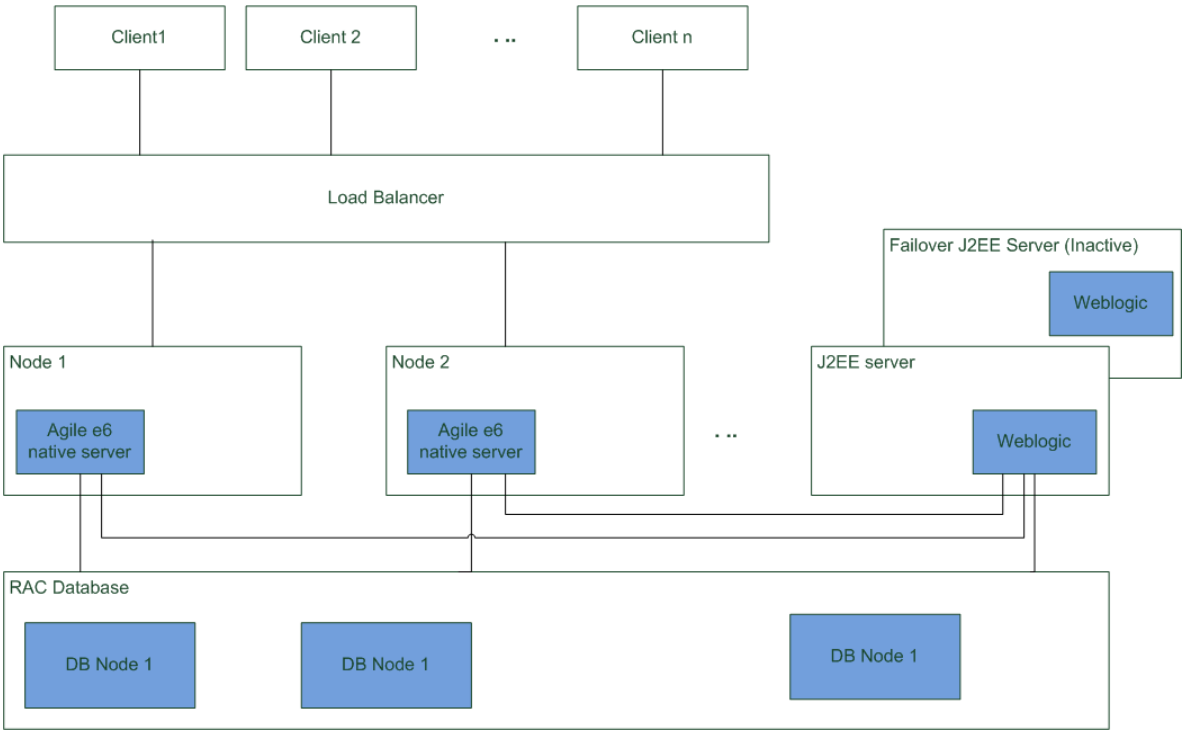
As we have a permission manager running on every node, the content of the related cache will be synchronized over all nodes in the cluster.



Single Weblogic Application Server with Failover Configuration

Especially for UNIX there is a 2nd way to setup Agile e6. There could be only one single J2EE Server on a separate node, hosting the WebLogic with all J2EE components for Agile e6. In addition, there is a set of servers, hosting the Agile e6 components. The native nodes could be installed in a cluster, e.g. as NLB cluster, and load balancing tools can be used to distribute the load to the servers. The database could be a RAC in the scenario, too.

Normally a 2nd WebLogic server on separate nodes will be installed and prepared. The second WebLogic node remains inactive as failover server until the main WebLogic server stops working.



Database High Availability Setup

Native as well as J2EE components of Agile e6 could be configured to connect to a RAC database. The Agile e6 applications connect to a cluster of database instances. If one database node crashes, the session will be automatically transferred to one of the running instances. The current transaction is rolled back, thus a new login for the Agile e6 user is not necessary.

Multiple Site Installation

The following scenarios are possible to support remote sites:

1. Central installation with local clients.

   The central installation with local clients is frequently implemented and allows a simple, centralized management with worldwide access.

   All services, besides the client, are installed on one central site and accessed via network.

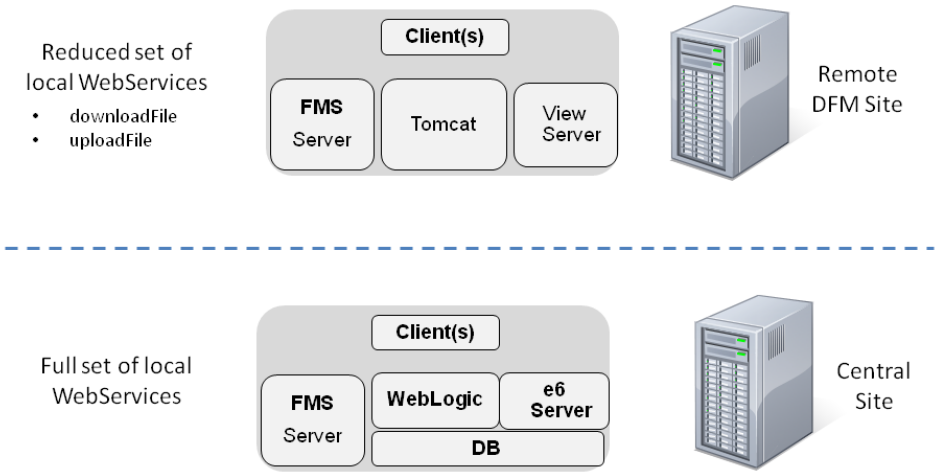
   In addition, a central management of the client via Webstart is possible. The user will connect to a central Webstart URL and download the software in case of updates.

2. Central installation with local clients and distributed file management setup.

Distributed file management means that clients and vaults are installed locally on several sites and one central application server and central installation is available. Files are (partially) distributed/ replicated. This leads to an improved availability of managed files. To support DFM functionality with the Web Services, a set of local Web Services is deployed to a TOMCAT installation on the remote site.

3. Central installation with full remote deployment, including local clients, distributed file management, and decentralized AutoVue server.

4. Depending on the number of local users and network bandwidth, a decentralized View server can be used on sites with DFM setup. The viewer uses the local File Server vaults as source and triggers file replication if necessary.



Chapter 2

J2EE Deployment Architecture

WebLogic Deployment

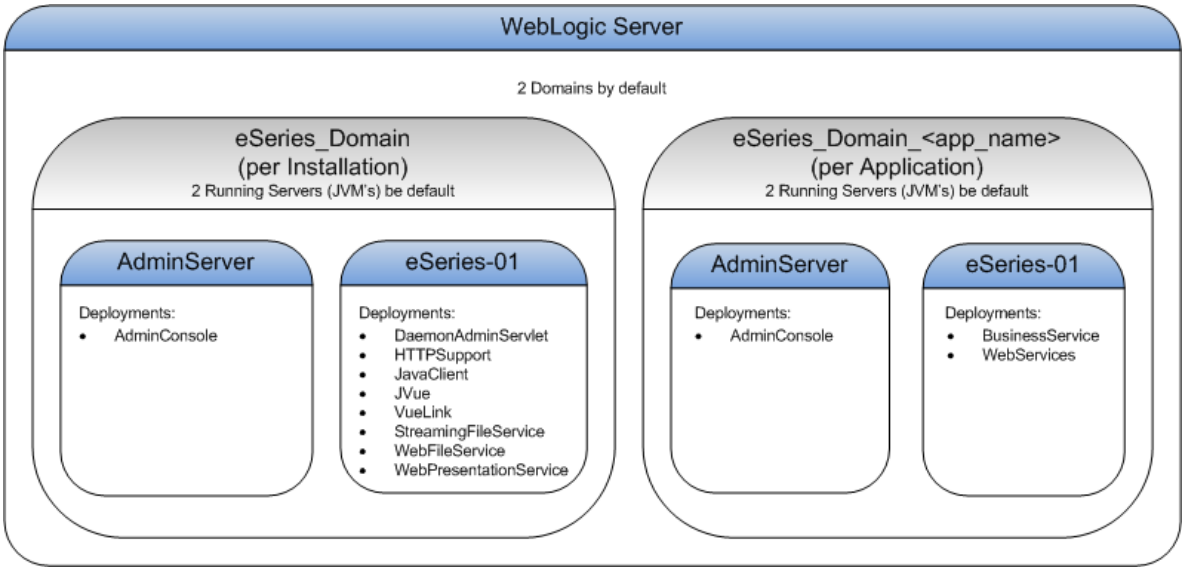
Note The Oracle WebLogic server is mandatory for Agile e6.1.3.0.

There are typically two types of WebLogic domains available as part of an Agile e6 installation. Every domain has two servers (JVMs) running by default.

- Admin Server
- EDM Server

The technical services will be deployed to the so called installation domain. The technical services will be shared by all application of an Agile e6 installation. Agile e6 application related services like Business Services and Web Services run in a separate domain.

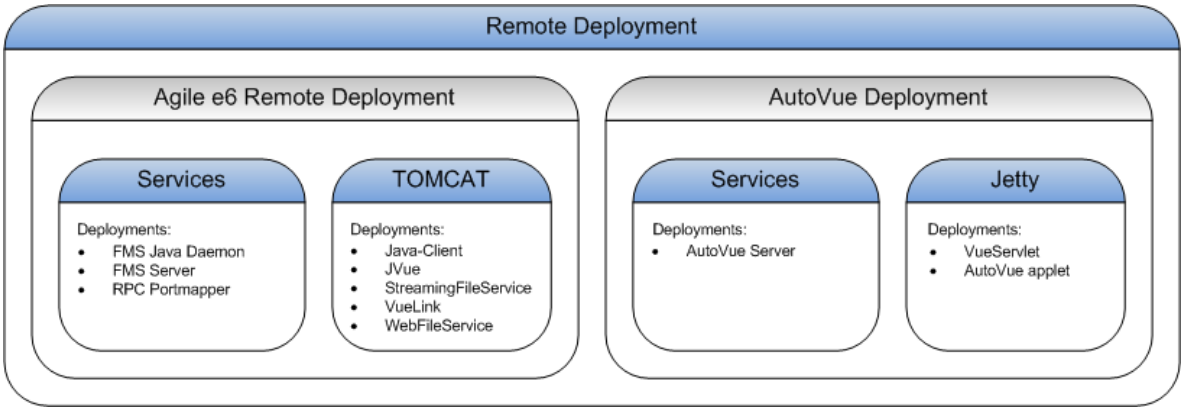
The following graphic shows the detailed WebLogic deployment architecture.



Remote Site Deployment Structure

A limited set of technical J2EE, and native services need to be installed on remote sites with DFM support. The J2EE components run in TOMCAT and jetty, instead of WebLogic.

The following diagram shows the remote site deployment architecture after an Agile e6 installation.



Chapter 3

Application Server Components

In general, the EDM server components can reside on the same server where the 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 (higher 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 an 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.

- Java Client

  The FMS Client is embedded and is not required to be installed separately.

- Web Client

- The FMS Client is executed on the Web Server and does not require any installation on the frontend.

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 an 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 EDM 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).

For 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 EDM 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, an FMS Server and one or more vaults can also be installed at the remote location (even though the EDM 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 in 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 the additional network traffic does not affect the overall network performance.

Web Fileservices

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

Prerequisites:

- Required hardware

  All supported platforms for EDM

- Required Software

  WebLogic Server, or TOMCAT6 for DFM locations

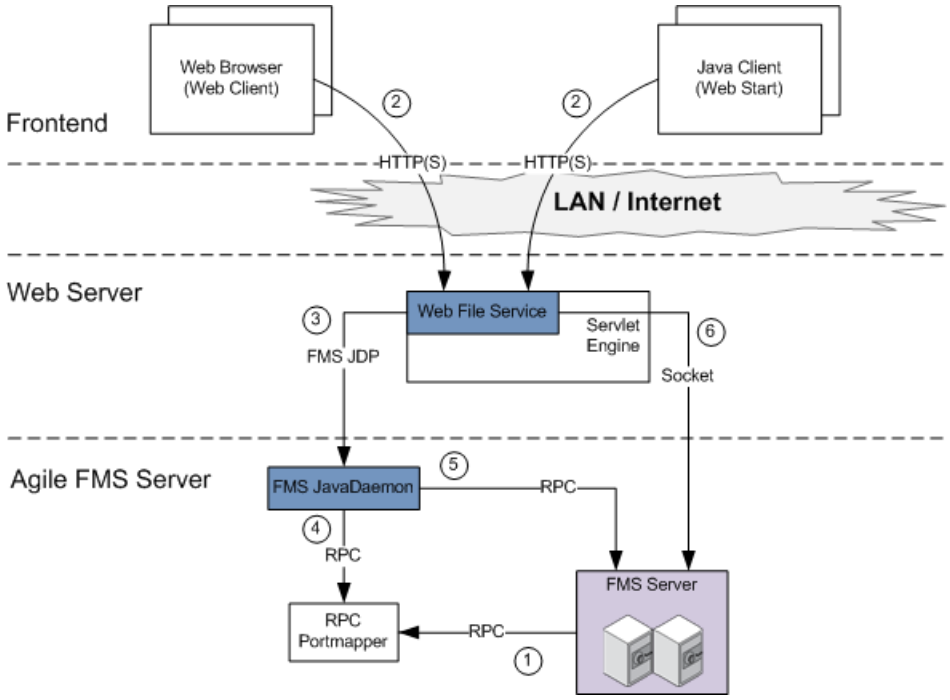
- Required Knowhow

  Administration of a WebLogic Server, or 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 graphic shows the communication lines when using Web Fileservice.



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

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

3. The Web Fileservices uses the configured FMS Java Daemon to get the transfer socket to the FMS Server.

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

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

6. The Web Fileservice uses the transfer socket to upload or download the file.

The following table shows the connections used. For more information refer to the Administration Guide for Agile e6.1.3.0.

Module	Port Configurable	Description
FMS	yes	HTTP(S) access to the Web Fileservice (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 Java Daemon 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 basis, 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.

Note Only applies to the Java Client.

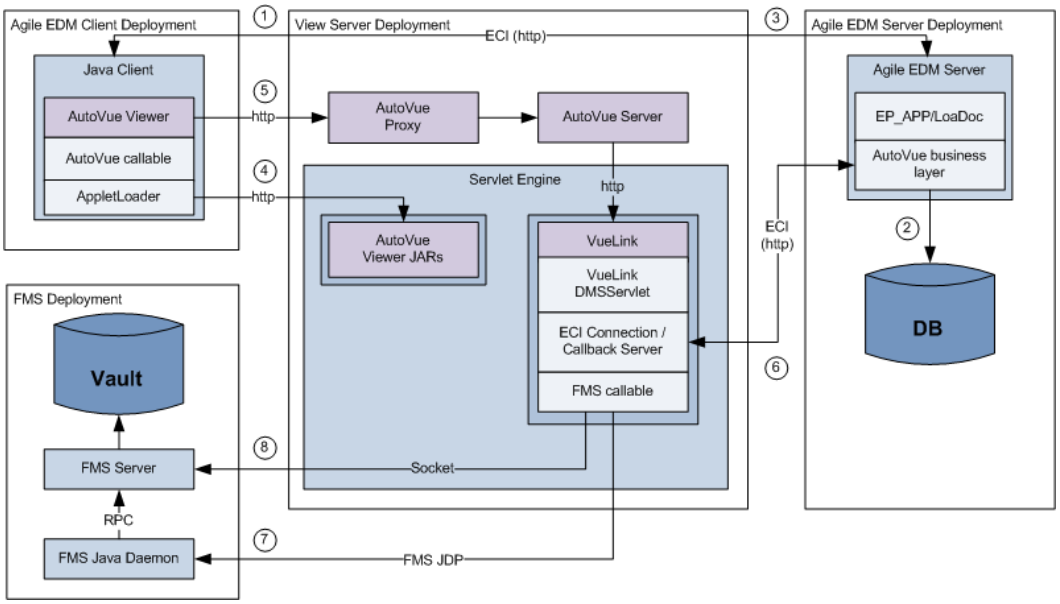
The AutoVue integration has the advantage that you do not require any corresponding authoring systems to be installed on a machine.

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 graphic shows the communication between the involved components.



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

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.

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

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

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.

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.

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.

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

The following table shows the connections used. For more information refer to the

Administration Guide for Agile e6.1.3.0.

Module	Port Configurable	Description
ECI	yes	EDM 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.

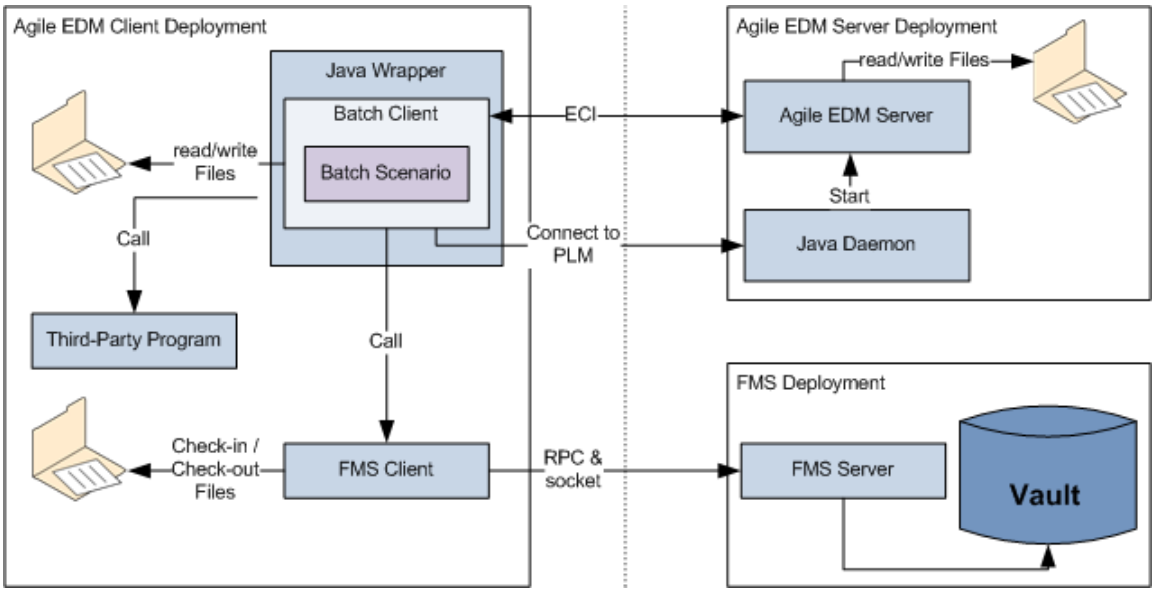
The Batch Client supports two modes:

- Scenario Mode

  In this mode the Batch Client starts a defined scenario which interacts with the EDM Server to execute the batch use case.

- ECI Server Mode

  In this mode the Batch Client provides an ECI tunnel to the EDM Server and standard features like file access to the File Server.



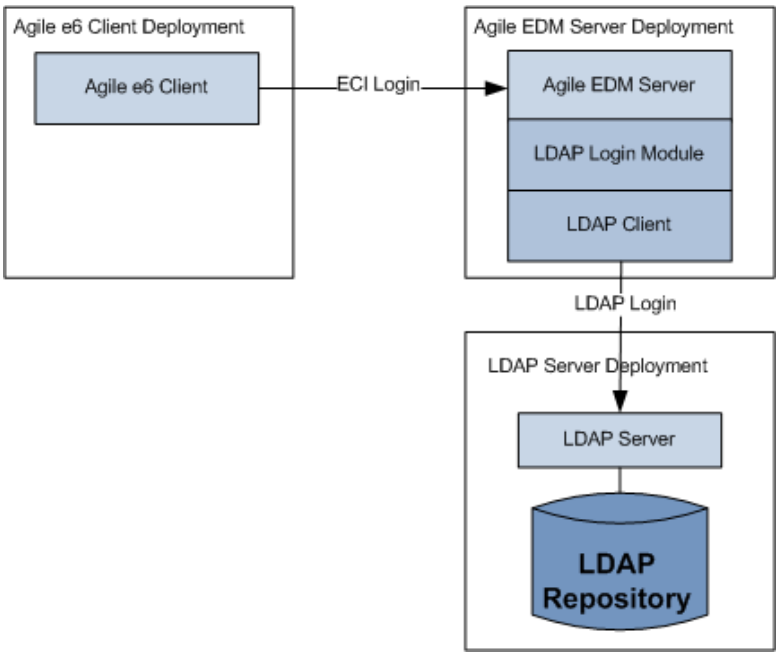
The following table shows the connections used. For more information refer to the Administration Guide for Agile e6.1.3.0.

Module	Port Configurable	Description
ECI	yes	EDM 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 e6 Client sends a login request to the Agile EDM Server. The user uses the Agile e6 user account with the LDAP user account password.
- The Agile EDM Server maps the Agile e6 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. For more information refer to the Security Guide for Agile e6.1.3.0.

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

Chapter 4

Application Clients

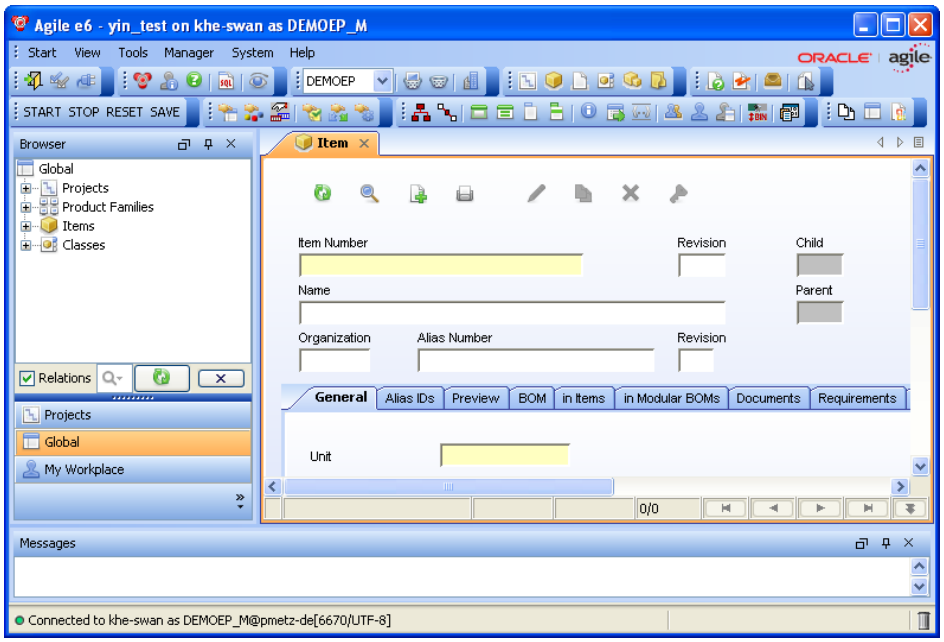
Agile e6 system provides three types of clients, serving different needs of casual users and power users.

- 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 and offers limited support for Administrators.

Note Mask generation can only be done in the Windows Client!



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

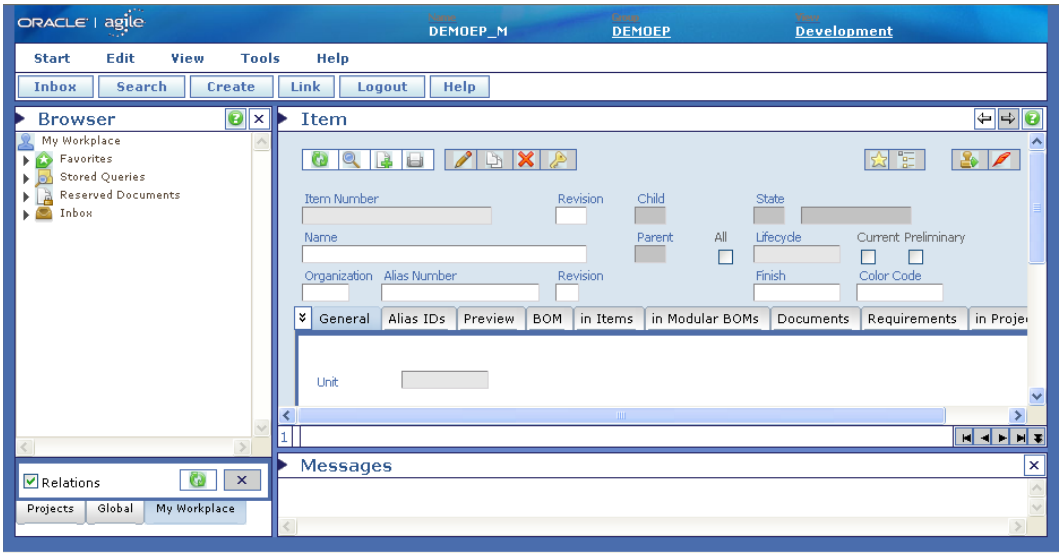
The Java Client fully supports M-CAD, E-CAD, EDA integrations, Office Suite features, and most other types of integrations.

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 frontends. 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).

Note The Web Client is not suitable for Customizers and Administrators.



The Web Client uses DHTML - combination of HTML and JavaScript - because the Web Client is a Web Presentation Service on the web server. It does not require a local installation that can be accessed by a web browser.

Compared to the Java Client, the most important limitations include:

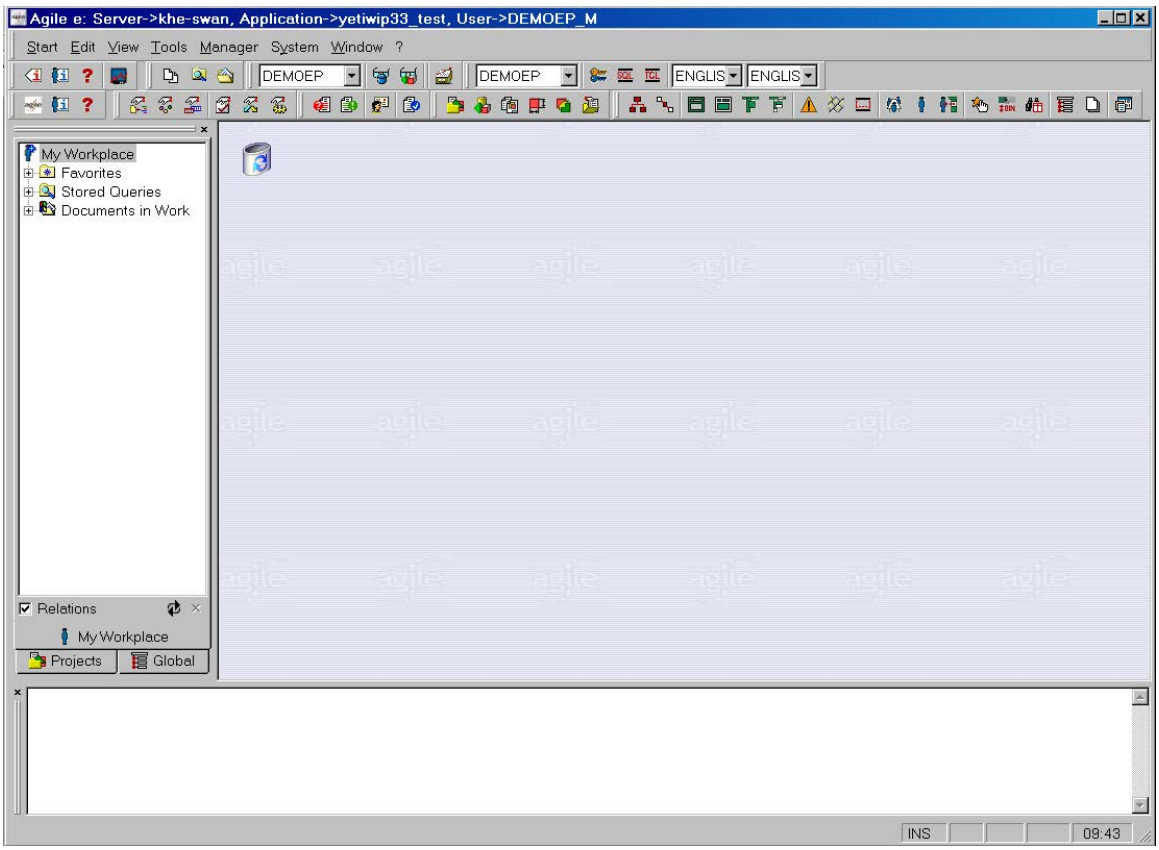
- Slightly different concept of the integrated Explorer window. Similar functionality can be achieved, but the customization is partially separate.
- No support for significant fields in lists. This means all columns in a list will scroll horizontally.
- No support for drag & drop from/to the desktop.
- 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 Wizard.

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).

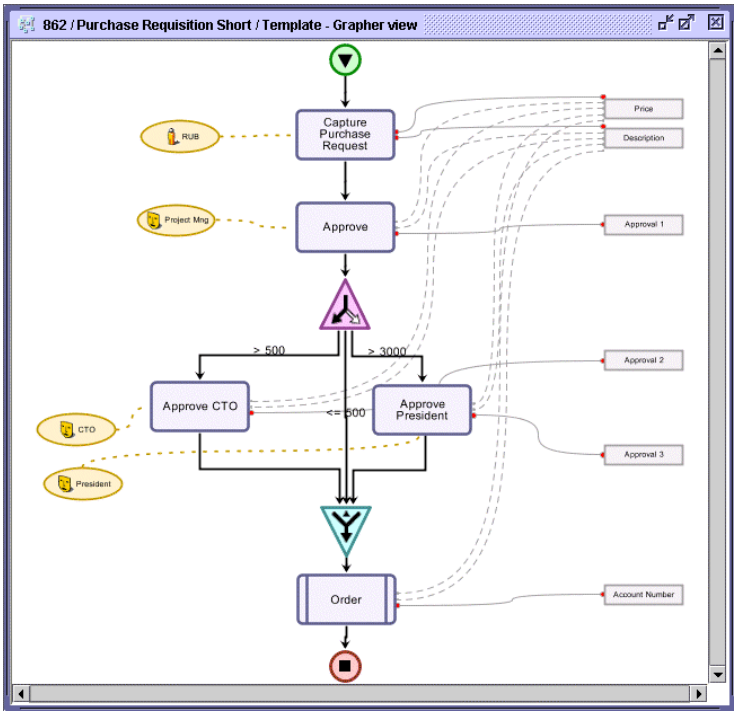
The user interface of the Windows Client is dynamically defined by the metadata in the repository.

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 screenshot:

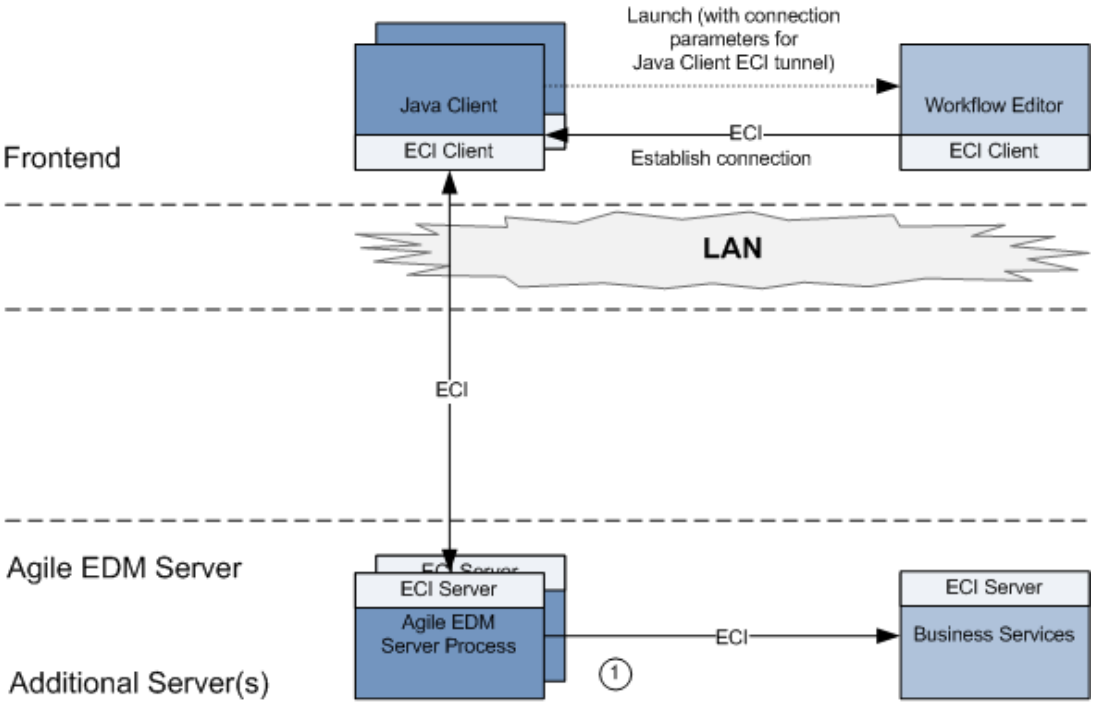


The Workflow Editor communicates with the Business Services, which include the Workflow Engine and the Permission Manager. All data related to a workflow process are stored in the database, thus ensuring the integrity of the system.

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 graphic:



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 Java Client ECI tunnel are passed.
- The Workflow Editor connects to the Business Services through the Java Client and EDM Server.

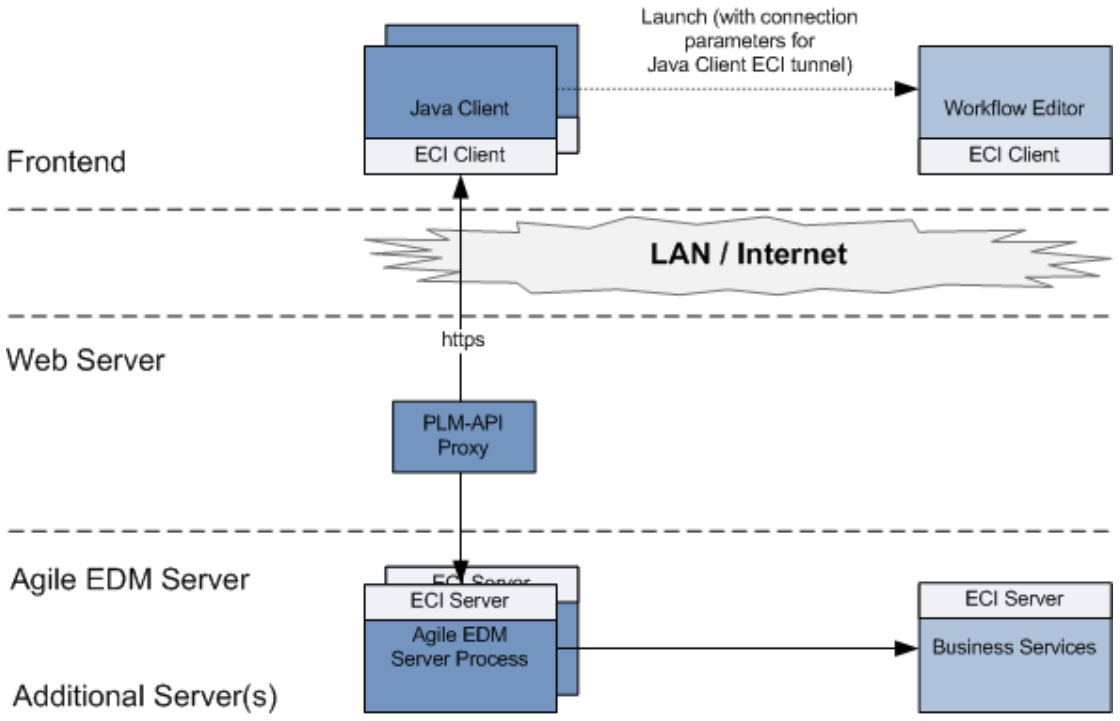
The following table contains details about the relevant lines of communication:

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

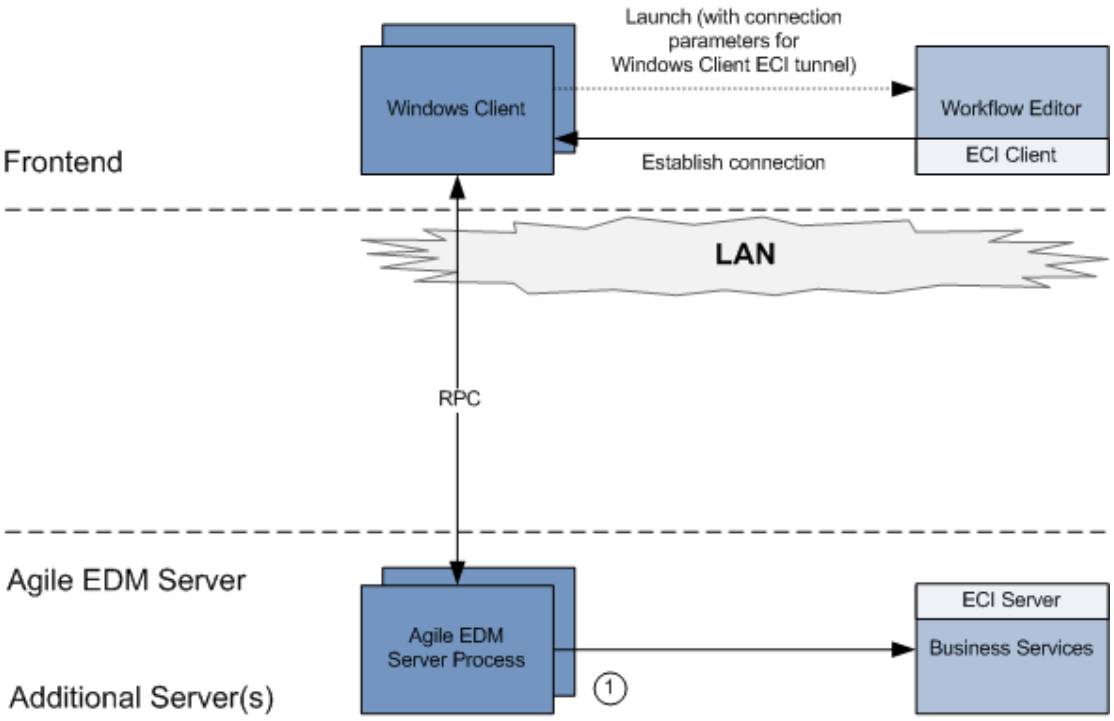
The Workflow Editor is a component in the Java Client that connects to the business service through an ECI protocol.

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 PLM-API 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:

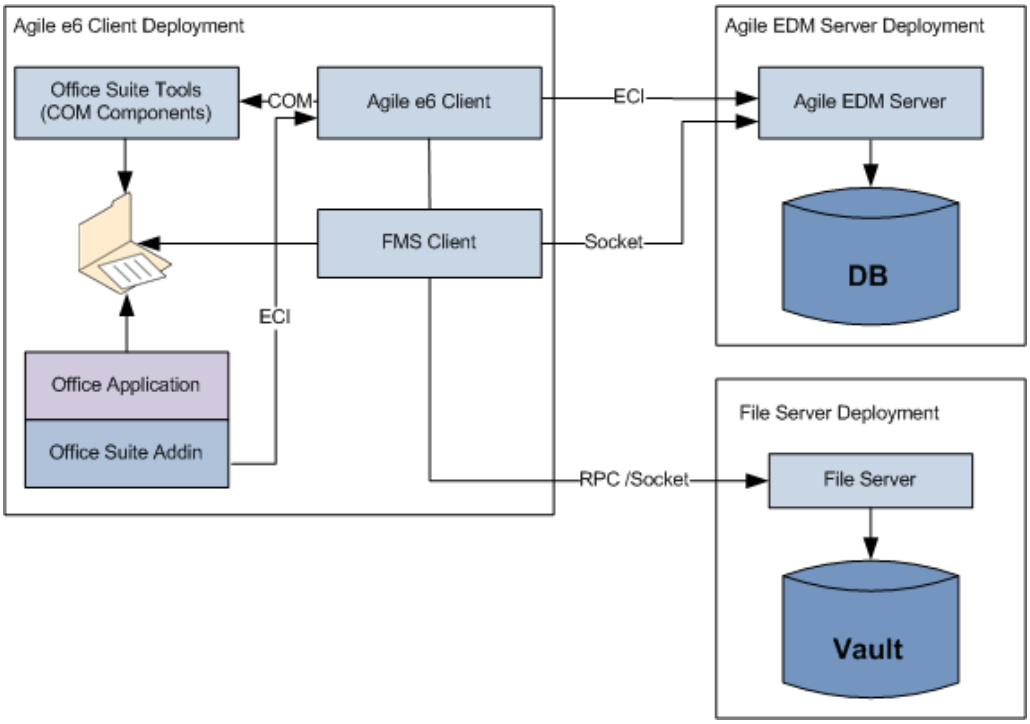
Line of Communication	Type	Port or Range of Ports
(1)	Application specific protocol based on Sockets	Port 19997 by default (can be configured during installation).

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

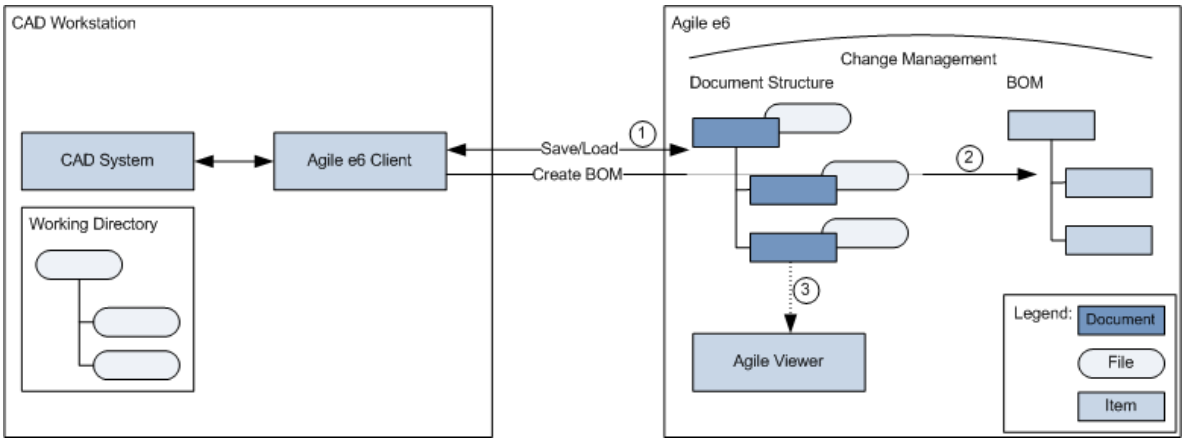
  It integrates the Agile e6 Office Suite Toolbar/Ribbon into the Office application. With the Addin the user can execute operations like Load / Save the Office document from / into the Agile e6 system.

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 EDM to manage the CAD data.



Primary use cases:

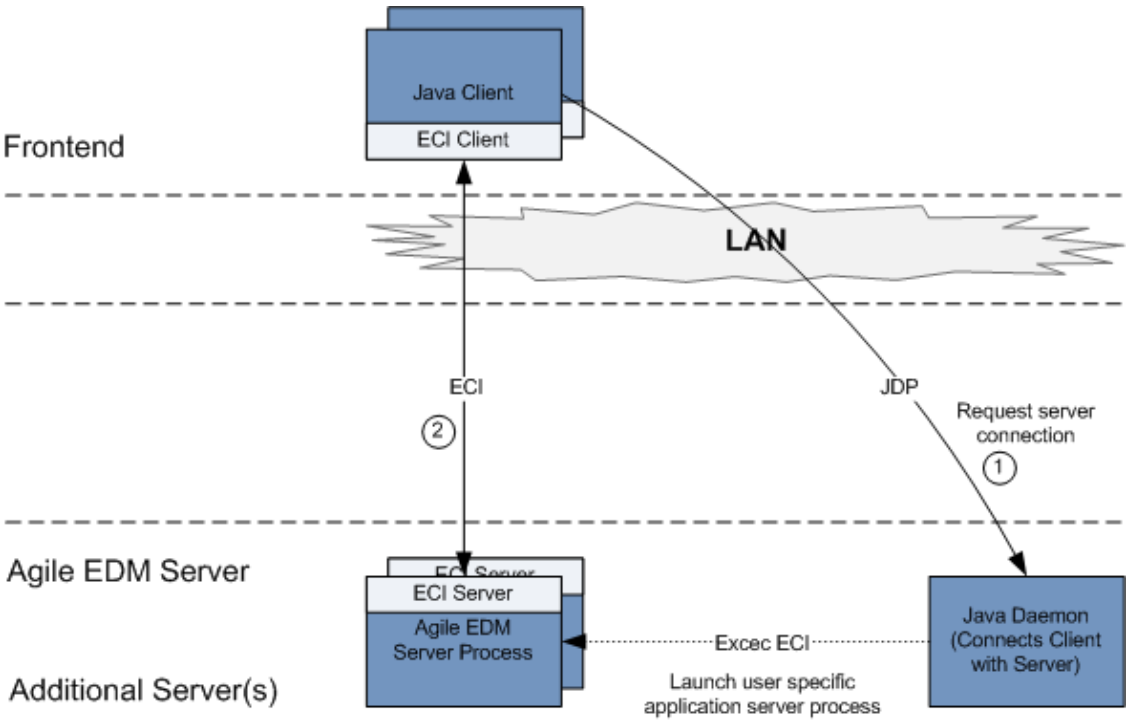
- Save and Load CAD files
- Create BOM
- View CAD files (optional)

Chapter 5

Java Client Communication

Line of Communication - Launching the Java Client

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



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.

2. The Java Client connects to the given Application Server Process and starts communicating.

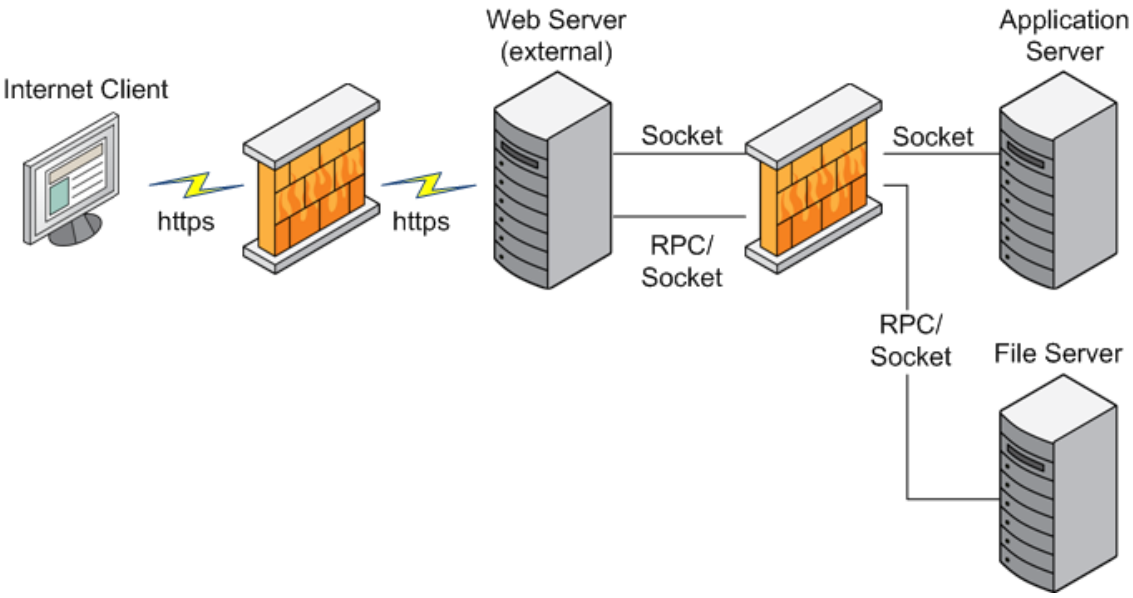
The communication line between the Java Client and the Application Server Process utilizes the ECI (Enterprise Communication Interface) protocol. This is the same as used e.g. by an M-CAD system to communicate with an Agile e6 client on the frontend.

The following table describes 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 EDM Server. Then it connects itself with the EDM Server for further calls from the Java Client.

Note The proxy server is built from a Web Service container (e.g. WebLogic) and Web Services.

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

Line of Communication - HTTP Support

The connection between Java Client and Agile e6 through HTTP occurs through a component called PLM-API Proxy, which provides the HTTP communication capability.

The Java Client makes two connections:

1. With Java Daemon

   To bootstrap the actual communication.

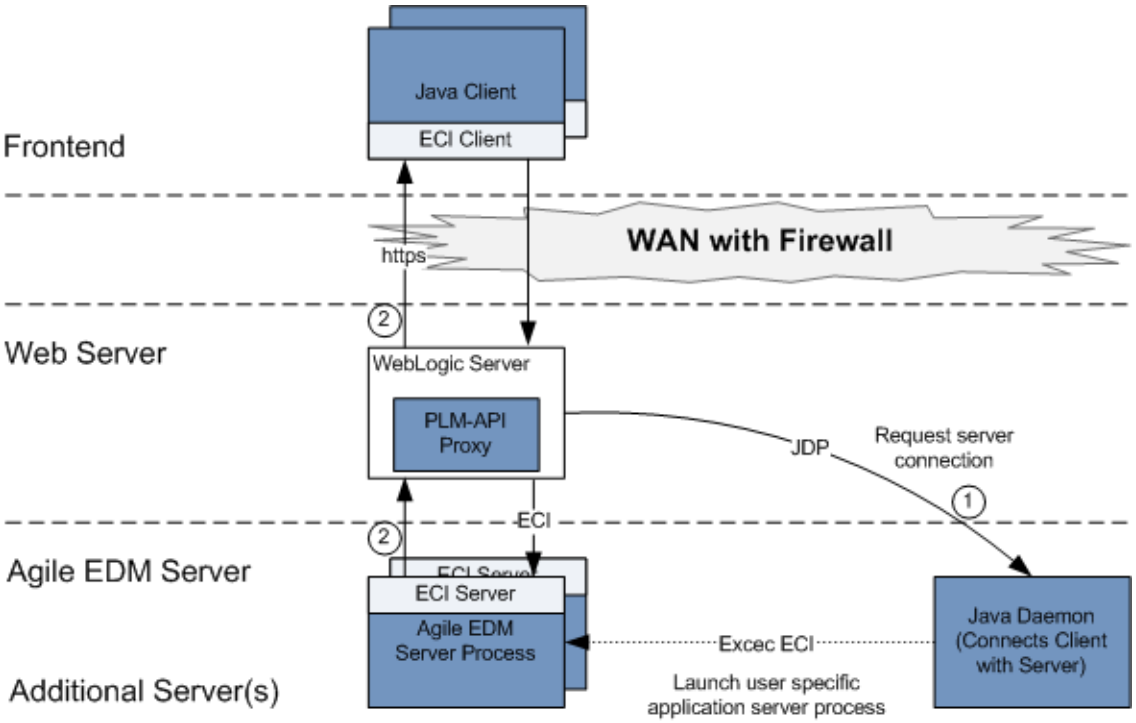
2. Connects to Agile e6

   To start the actuall communication with EDM Server and then close the connection with Java Daemon.

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 (EDM Server and Java Daemon) inside the firewall. The PLM-API Proxy should be deployed in this web server.

The Java Client connects to the PLM-API proxy and the PLM-API bootstraps the connection for the Java Client by first connecting to Java Daemon, and then to EDM Server by using the credentials sent from Java Client. Once the PLM-API has finished the connection bootstrapping, the Java Client can communicate through PLM-API proxy with EDM Server.



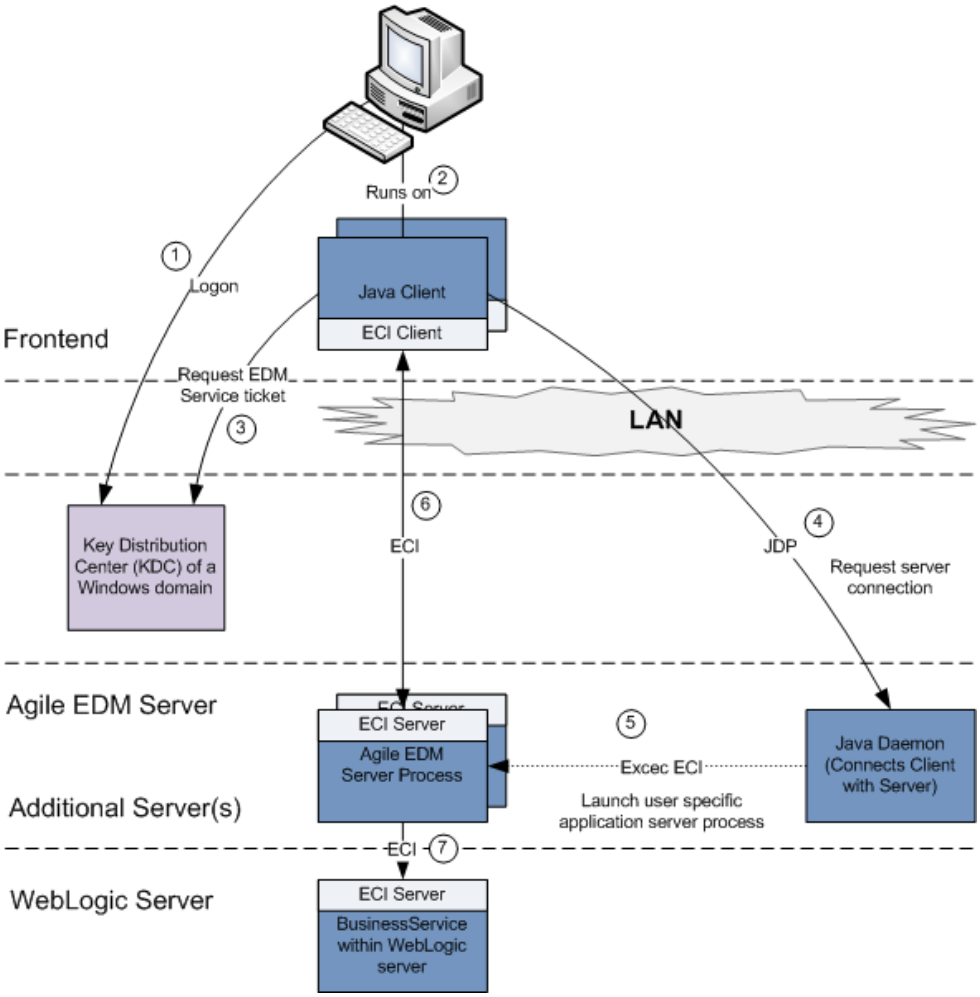
Line of Communication - Kerberos Based Single Sign-On

The Java Client supports Kerberos based single sign-on interface, which allows you to e.g. authenticate on the Windows Operating system level and to open Agile e6 without further authentication dialog.

The following graphic depicts a typical infrastructure consisting of Windows Domain Servers (e.g. Windows Active Directory server), responsible for the identity management, and the Agile e6 client and server applications.

As part of the initial authentication at the Windows Domain Server, the client machine will receive a so-called Ticket Granting Ticket (TGT), which is later used to request a service ticket from the Key Distribution Center (KDC) to start the Agile e6 application.

The service ticket allows the Agile e6 client to start Agile e6 without further authentication.



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

1. The Windows domain login is based on Kerberos and the user authenticates himself during the Windows logon against the Kerberos server. During the login the Kerberos server creates a TGT which is stored on the client machine.

2. The user starts the Java Client

3. The Java Client uses the TGT of the client machine to request a service ticket for Agile EDM.

4. The Java Client connects to the Java Daemon, requesting to launch a user specific Agile EDM server process and to return the connection information for this process

5. The Java Daemon starts the Agile EDM Server process

6. The Java Client connects to the given Agile EDM Server process and starts communication

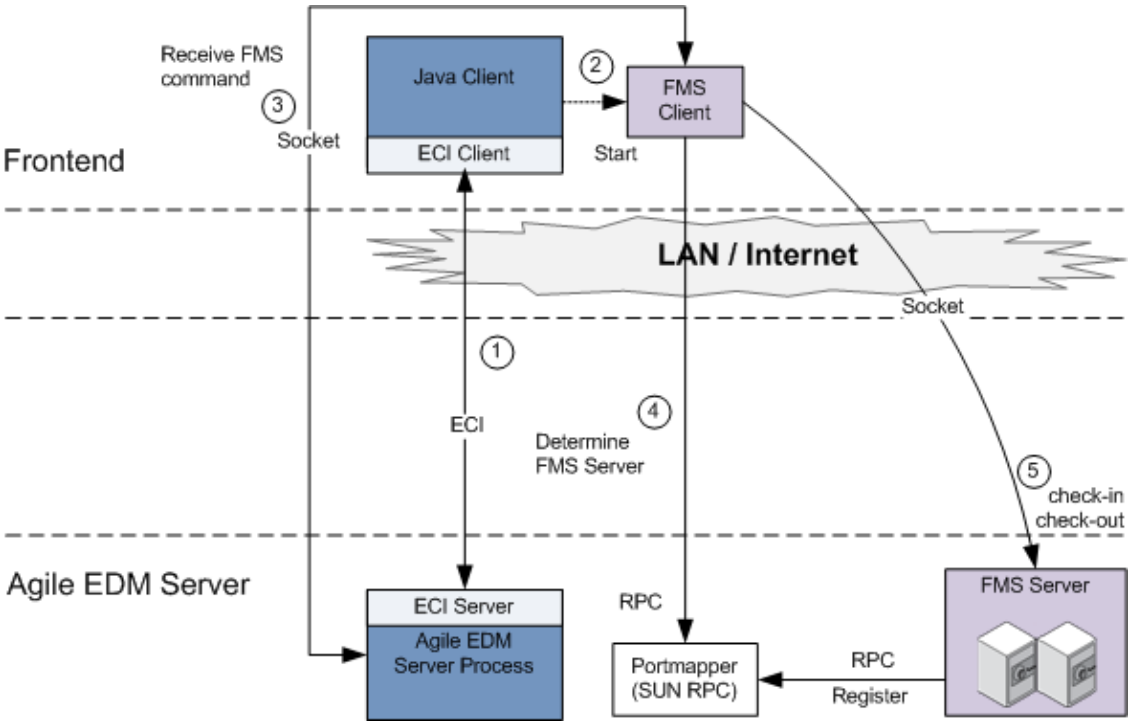
7. The Agile EDM server delegates the authentication to the Business Services which validates the service ticket which is used for the authentication

Line of Communication - File Access (using FMS)

The Java Client supported different options to access files depending on how the Java Client is deployed on the client machine.

Local Installed Java Client

The local installed (MSI Installation) Java Client uses the external FMS Client to communicate with the FMS Server directly.



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

1. The user initiates the FMS operation and the Agile EDM Server calls the Java Client to start the external FMS Client.

2. The Java Client starts the external FMS Client and passes the connection information of the Agile EDM Server.

3. The FMS Client opens the command channel between the Agile EDM Server and the FMS Client. Via this channel, the FMS Client receives the command and the information to access the FMS Server.

4. The FMS Client contacts the RPC portmapper to gain the RPC connection information of the FMS Server.

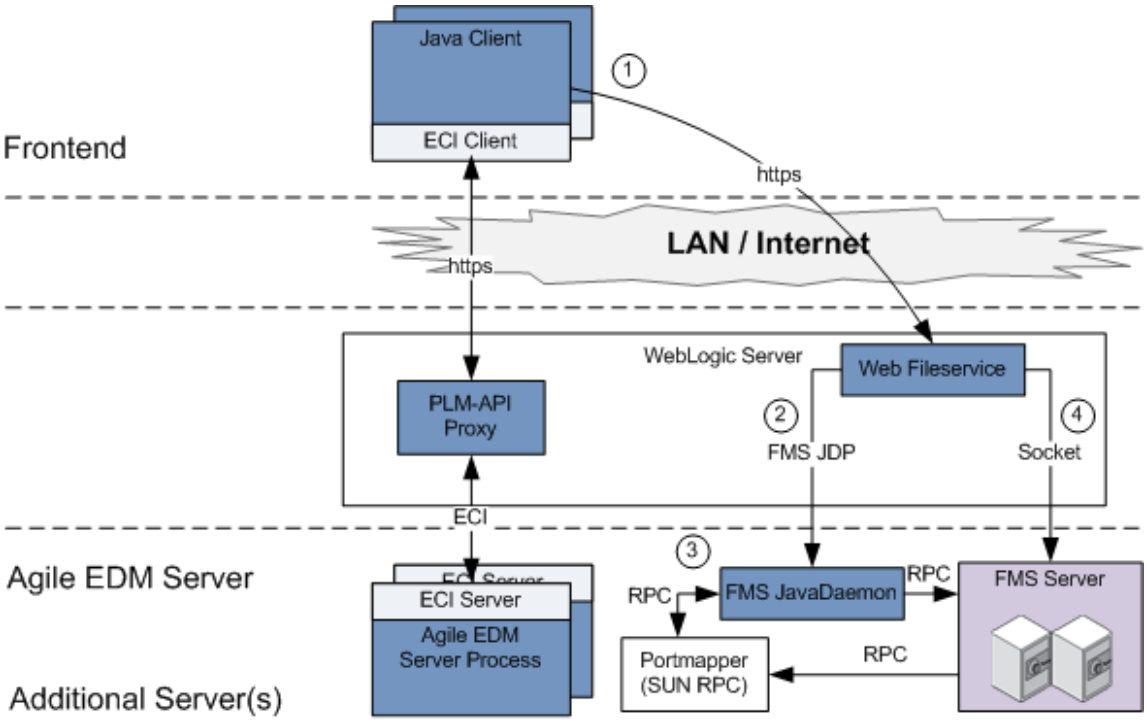
5. The FMS Client connects to the FMS Server to check-in or a check-out the file.

WebStart Deployed Java Client

The WebStart Java Client supports a build-in native FMS Client for direct connections to the FMS Server. This is normally used in DFM (Distributed File Management) environments where RPC communication is possible.

In a pure web environment, the Java Client communicates via HTTPS with the Agile system.

The communication between the Java Client, the EDM Server process, the Web Fileservice, and the FMS Server is depicted in the following figure:



Note If the Java Client uses the HTTP protocol to communicate with the Agile EDM Server, the client must use HTTP for the communication with the FMS server, too.

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 Fileservice, which is hosted by a servlet engine. The address of the Web Fileservice 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.

3. The Web Fileservice can now request the FMS Server to create an individual thread and return the connection parameter.

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

Files are not stored temporarily, but transferred directly between the frontend and the FMS Server.

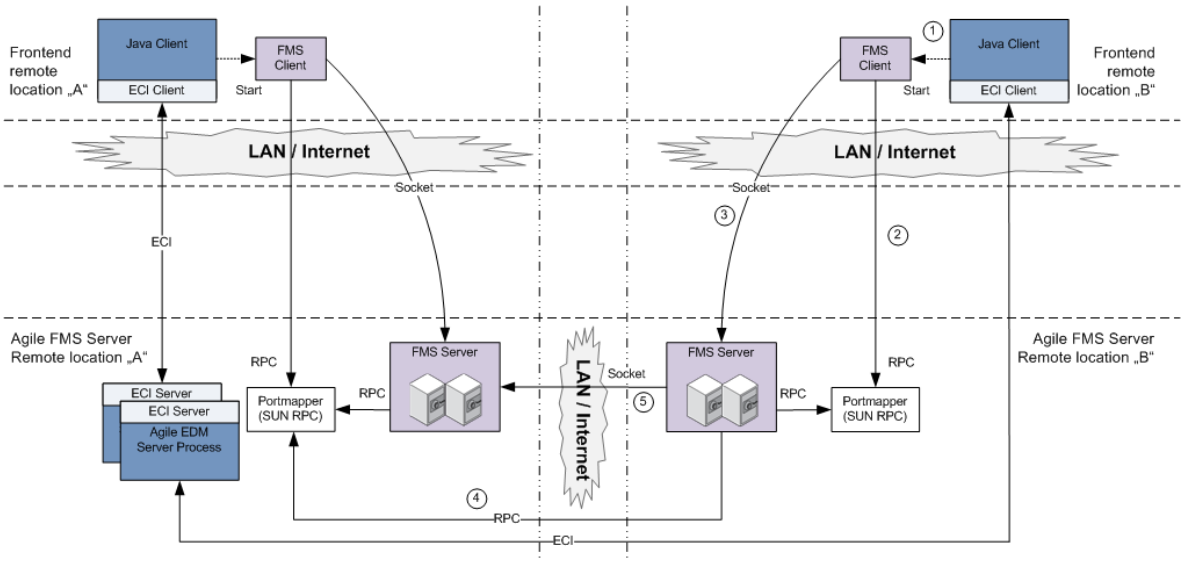
The following table contains the details about the relevant lines of communication:

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

Line of Communication - Distributed File Management (DFM)

The Agile e6 system supports a distributed file management which allows having a File Server installed on the remote location. The different File Servers are able to replicate files for performance reasons.

The file replication can happen online, on demand, or asynchronous via a batch replication. With the batch replication you can execute the file transfer during a time slot where the bandwidth is not used by other users.



In this example the remote location “A” is the main DFM location where the Agile EDM server is installed. The remote location “B” is a pure DFM location, here is no Agile EDM server available. Client on the remote location “B” are connected to the Agile EDM server process which is running on the remote location “A”.

The following steps are executed when the user performs a replication of a file (On-Line):

1. The Java Client starts the external FMS Client if necessary.

2. The FMS Client requests the connection information for the FMS Server of the local location (B).

3. The FMS Client connects to the FMS Server and triggers a replication operation.

4. The FMS Server on the local location requests the connection information for the remote FMS Server (A).

5. The FMS Server requests the file to replicate from location “A” to the location “B”.

In a DFM environment the FMS Client always connects to the FMS Server on the same location. If a file replication is necessary the local FMS Server requests the file from the remote FMS Server, the FMS Client never contacts the remote FMS Server.

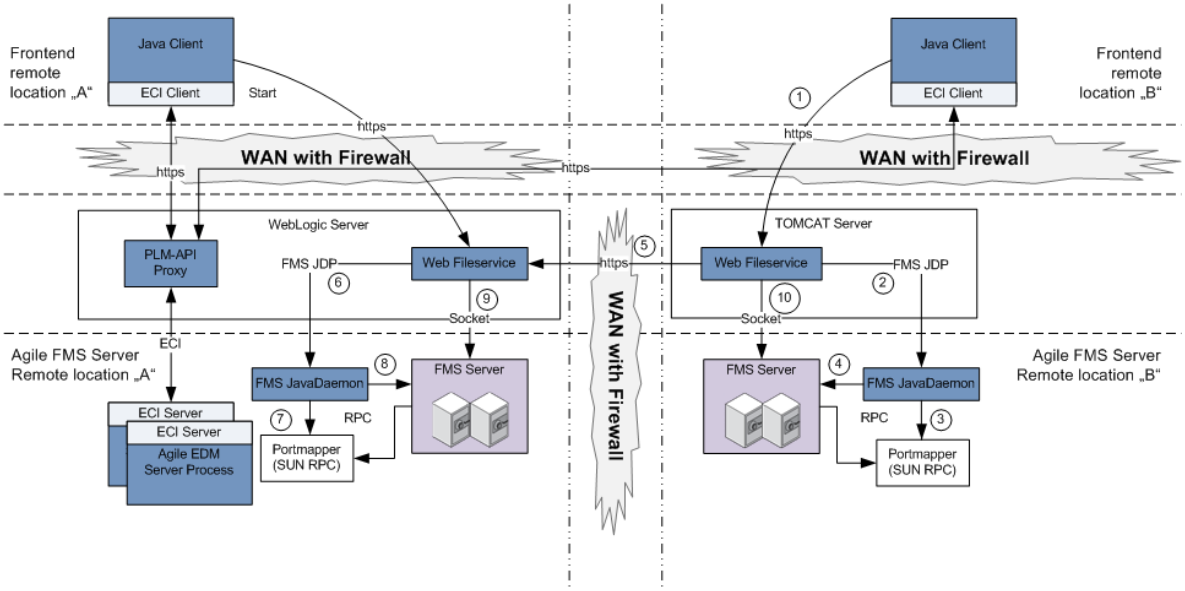
All FMS Servers are master File Servers. The Agile EDM Server administers on which FMS Server the instances of a file are present and if an instance is out-dated.

For example; if a file instance is only available on location “A”, and a user wants to change it on location “B”, Agile e6 initiates a file replication and the FMS Server on location “B” requests the file from the FMS Server on location “A”. Now both FMS servers have the newest file instances. Now, when the user changes and re-checks-in the file on location “B”, the Agile e6

system marks the file instance on location “A” as outdated in the database.

For batch replication, a Batch Client can be configured to execute batch replication jobs.

Line of Communication - Firewall Friendly DFM



This example shows the same scenario as above, but with firewall friendly communication between the remote locations.

The following steps are executed when the user performs a replication of a file (On-Line):

1. The Java Client initiates the file replication by calling the Web Fileservice.
2. The Web Fileservice calls the FMS Java Daemon to get the access to the FMS Server.
3. The FMS Java Daemon gets the RPC connect information from the Portmapper.
4. The FMS Java Daemon calls the FMS Server to get the port for the file transfer.
5. After the Web Fileservice received the port to access the FMS Server, it calls the remote Web Fileservice to request the file which should be replicated.
6. The remote Web Fileservice uses its FMS Java Daemon to get the access port for the FMS Server.
7. The FMS Java Daemon gets the RPC connect information from the Portmapper.

8. The FMS Java Daemon calls the remote FMS Server to get the port for the file transfer.

9. After the Web File Service received the port to access the FMS Server, it requests the file from the FMS Server and sends the file back to the calling Web Fileservice.

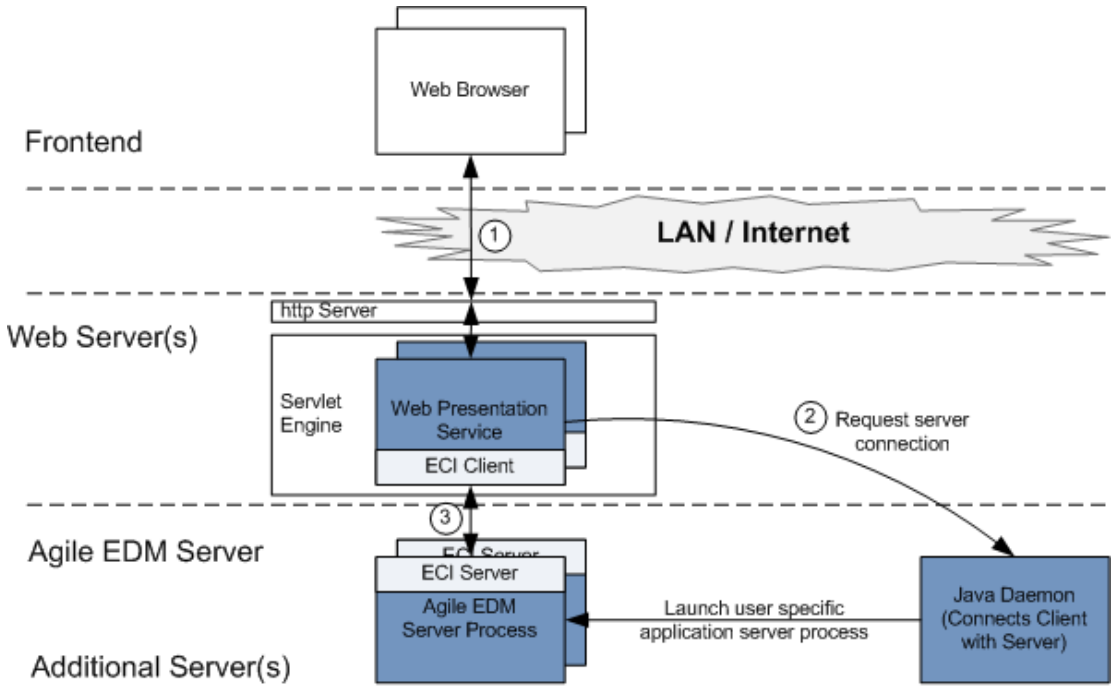
10. The Web Fileservice sends the received file to its local FMS Server.

Chapter 6

Web Client Communication

Line of Communication - 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 graphic:



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

1. The web browser sends a request to the Web Presentation Service 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 communicates 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.

3. The Web Presentation Service connects to the given Application Server Process via ECI.

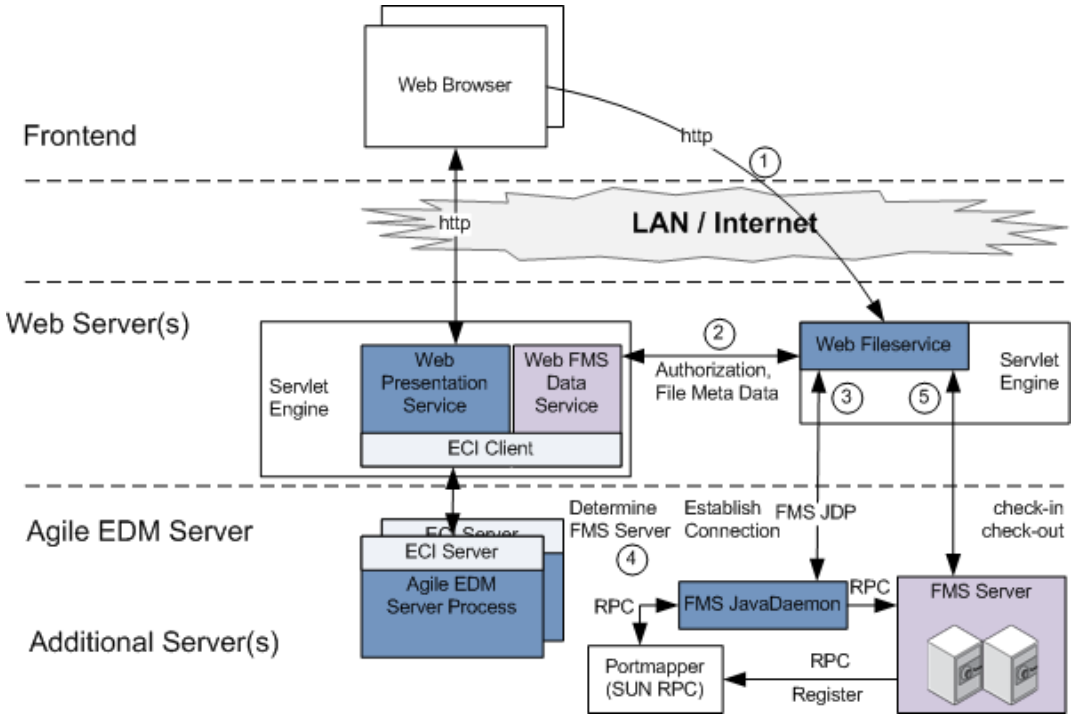
The Web Presentation Service will process requests from the web browser by translating these into ECI calls to the Application Server Process. The Web Presentation Service is also responsible for the rendering of lists and forms into DHTML (dynamic HTML - HTML enriched with JavaScript). The JavaScript 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 - 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 Fileservice. The address of the Web Fileservice 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 Fileservice 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 to bypass the HTTP server and communicates 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 Fileservice, 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 Fileservice provides the authorization ticket, and in return, receives required file metadata.

3. With the RPC-number of the FMS Server, the Web Fileservice connects to the Portmapper to determine the port of the corresponding FMS Server.

4. Now the Web Fileservice can request the FMS Server to initiate an individual thread and return the connection parameter.

5. The file data are transmitted between the Web Fileservice and the FMS Server thread. The Web Fileservice 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 by all users - and increases performance especially in the case of large files, because only a minimum of file write/read operations are performed.

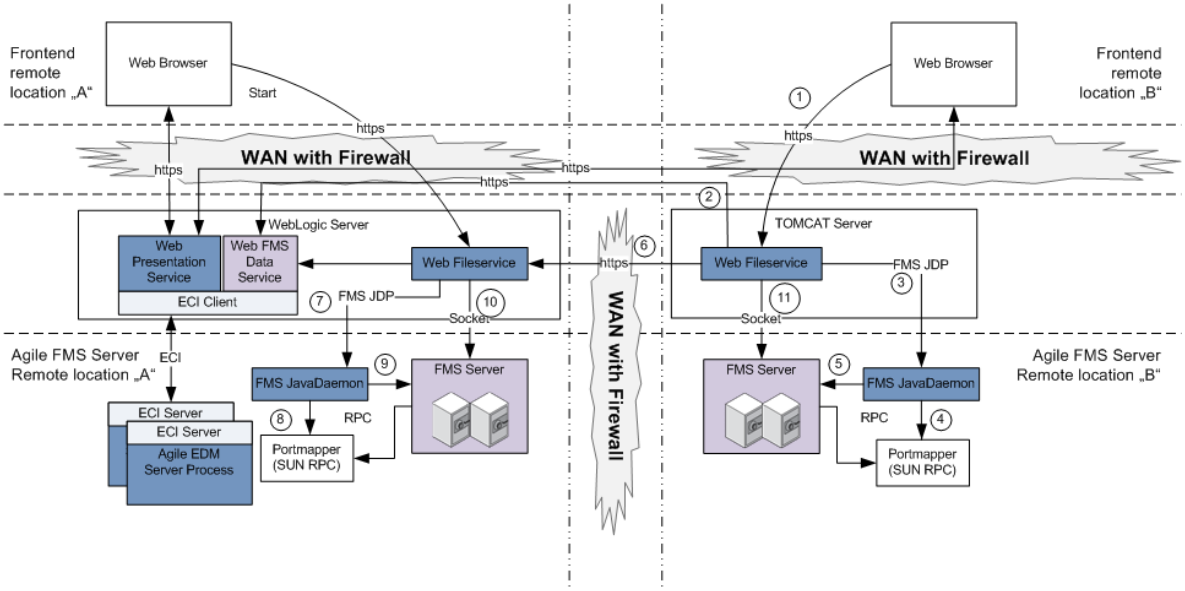
The following table contains the details about the relevant lines of communication:

Line of Communication	Type	Port or Range of Ports
(1)	HTTP (or HTTPS)	Port 8088 by default (can be configured during installation)

Line of Communication	Type	Port or Range of Ports
(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.

Line of Communication - DFM

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 file replication operation in the Web Client:

1. The Web Client calls the Web Fileservices to execute a file replication operation.

2. The Web Fileservice uses the URL information passed with the call to request the access information from the Web FMS Data Service.

3. With the gained access information, the Web Fileservice calls the FMS Java Daemon to get the access port of the local FMS Server.

4. The FMS Java Daemon requests the RPC connection information of the FMS Server from the Portmapper.

5. The FMS Java Daemon retrieves the FMS Server access port and sends it back to the Web Fileservice.

6. The Web Fileservice calls the remote Web Fileservice to request the file which should be replicated. This call is a POST request which contains the access information for the remote FMS Server.

7. The Web Fileservice uses the FMS Java Daemon to get the FMS Server access port.

8. The FMS Java Daemon requests the RPC connection information of the FMS Server from the Portmapper.

9. The FMS Java Daemon retrieves the FMS Server access port and sends it back to the Web Fileservice.

10. The Web Fileservice requests the file to replicate from the FMS Server and sends it back to the remote Web Fileservice.

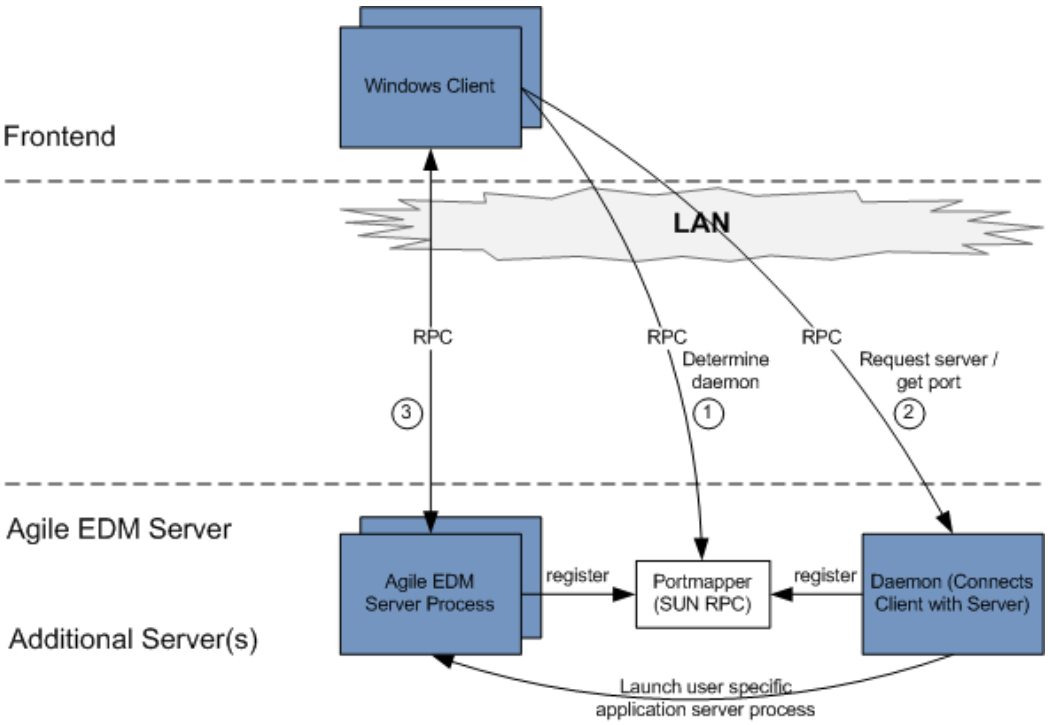
11. The Web Fileservice sends the file to the local FMS Server.

Chapter 7

Windows Client Communication

Line of Communication - 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. The Daemon had registered at the Portmapper initially and is therefore known to the Portmapper.

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

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

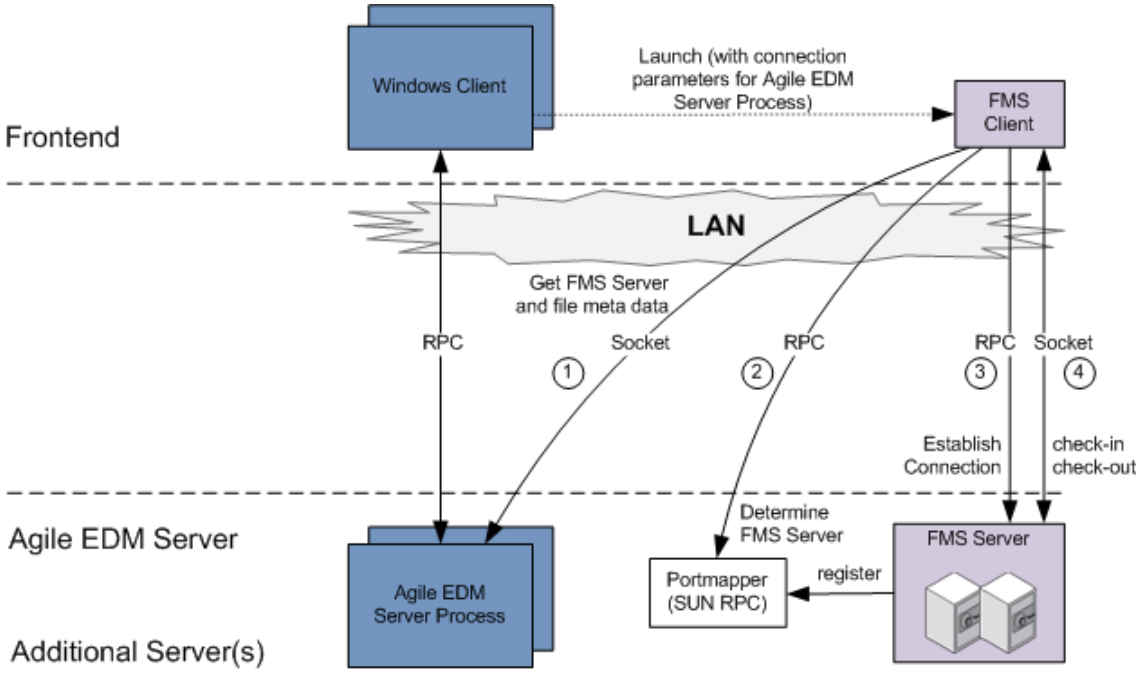
The communication line between the Windows Client and the Application Server Process uses a proprietary protocol that transfers only a minimum amount of data.

The following table describes 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 - 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 (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 (2).

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

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

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

The following table describes 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 8

Web Services Interface

Agile e6 Web Services expose a subset of the Product Lifecycle Management (EDM) functionalities of Agile e6. These services support functionalities provided by the EDM modules in Agile e6, 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

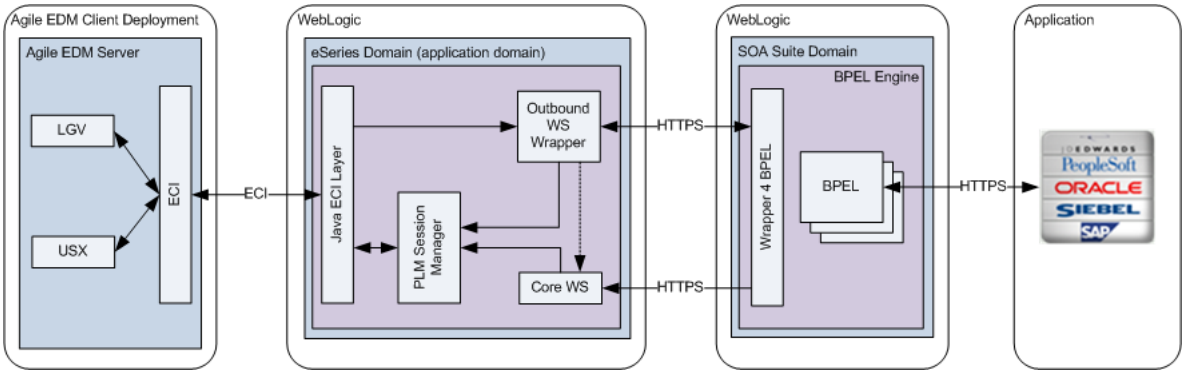
The Web Services Framework is an additional layer on top of Agile e6, which supports inbound and outbound communication based on standard Web Services technology.

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 EDM object or get EDM 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 Service Wrapper Interface

Each wrapper has to implement the Web Service Wrapper 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 EDM Server process.

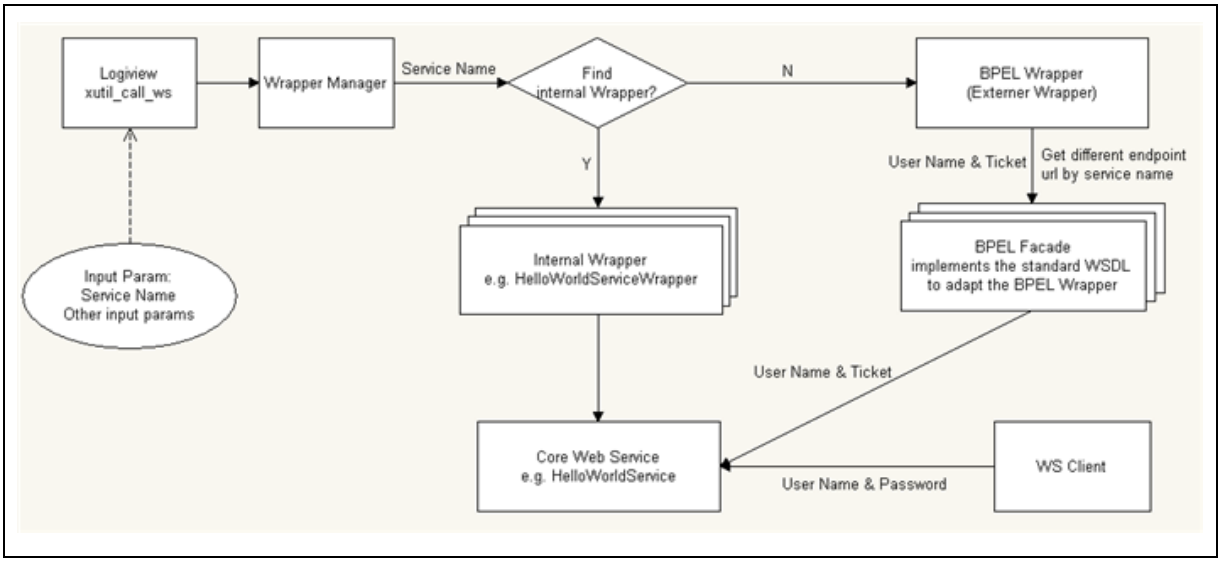
In case the wrapper is implemented in BPEL (or in any other external Web Service language), you require a special wrapper called the ExternalWrapper. This wrapper delegates the call to the respective external Web Service Wrapper.

The BPEL Facade

For the outbound calls, you are required to use the External Wrapper 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 can be found in the APP-INF/classes directory, along with the file application.properties for the Web Services.

The Agile e6 Session Handling

Every call of an Agile e6 Core Web Service needs an EDM server instance. It is very important to limit the number of EDM 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 EDM Server instance assigned to the HTTP session of the current Web Service call. If it is found, the Web Service call uses the existing EDM Server instance.

- EDM Ticket

  An EDM ticket is returned in response to a core Web Service operation. This ticket can be used to access the same EDM 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 EDM Server instance, even if no EDM Server is assigned to the HTTP session.

The EDM ticket mechanism is the only way to let the calls to different core Web Service, such as Metadata or BusinessObject Web Service use the same EDM Server instance.

A Web Service client should use the EDM 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 EDM 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 EDM ticket as the password. This shuts down the EDM Server instance and frees up the server resources.

The EDM Session Manager

The EDM Session Manager lets you manage the EDM Session objects which are used to keep the existing connections and user contexts to the EDM Server.

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

The EDM Session provides connection to EDM Server.

To retrieve an EDM Session, EDMTicket is provided. When a new EDM Session is created, the EDMTicket is set to the EDM Session, which is then set into the SOAP message to the client side.

The lifecycle of EDM Session is the 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 EDM Ticket

A Response contains a string that can be used in subsequent calls. This string is called the EDM Ticket. The ticket gives the caller access to the same Agile e6 instance (EDM 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 EDM Ticket improves the Web Service 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 (e.g. Configuration.setUserContext) is used as a password when the client makes a call for another service operation (e.g. BusinessObject.getObjects).

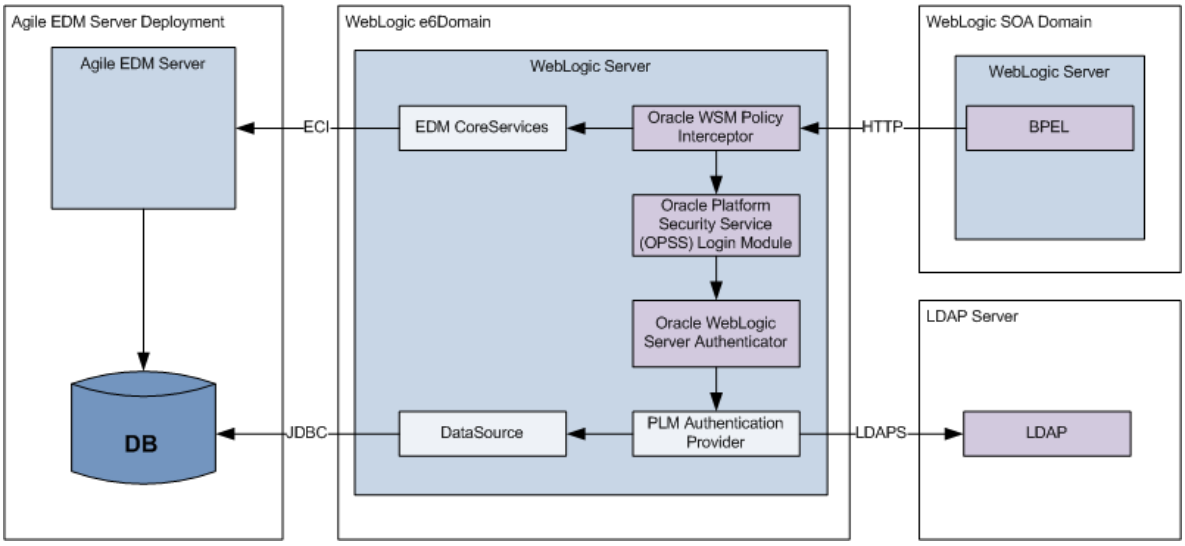
The ticket sharing among different client ports eliminates the need for the server to start new Agile e6 sessions, which would result in new EDM Servers being started.

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 EDM user name and EDM 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 EDM 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 PLM Authentication Provider is installed at the time of Agile e6 application installation. A JDBC connection is made from within WebLogic, thus the user and password are the one for the JDBC data source.

For complete details on Agile e6 application installation, including the preliminary settings of the authentication provider, refer to the installation guides for Agile e6.1.3.0.

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 by 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, this 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 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 Web Service 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.

The core Web Services supports the feature FastInfoSet; by default it is part of WebLogic Service 12.1.2.

The StreamingFileSevice uses “Streaming MTOM” for file transfers from the client to the service host, which then does the upload/download to the nearest File Server and the callback to the EDM server to update the file metadata.

Single Sign-On with WebLogic and SAML Web Service Authentication

The Web Service single sign-on solution uses WebLogic and the SAML (Security Assertions Markup Language) protocol.

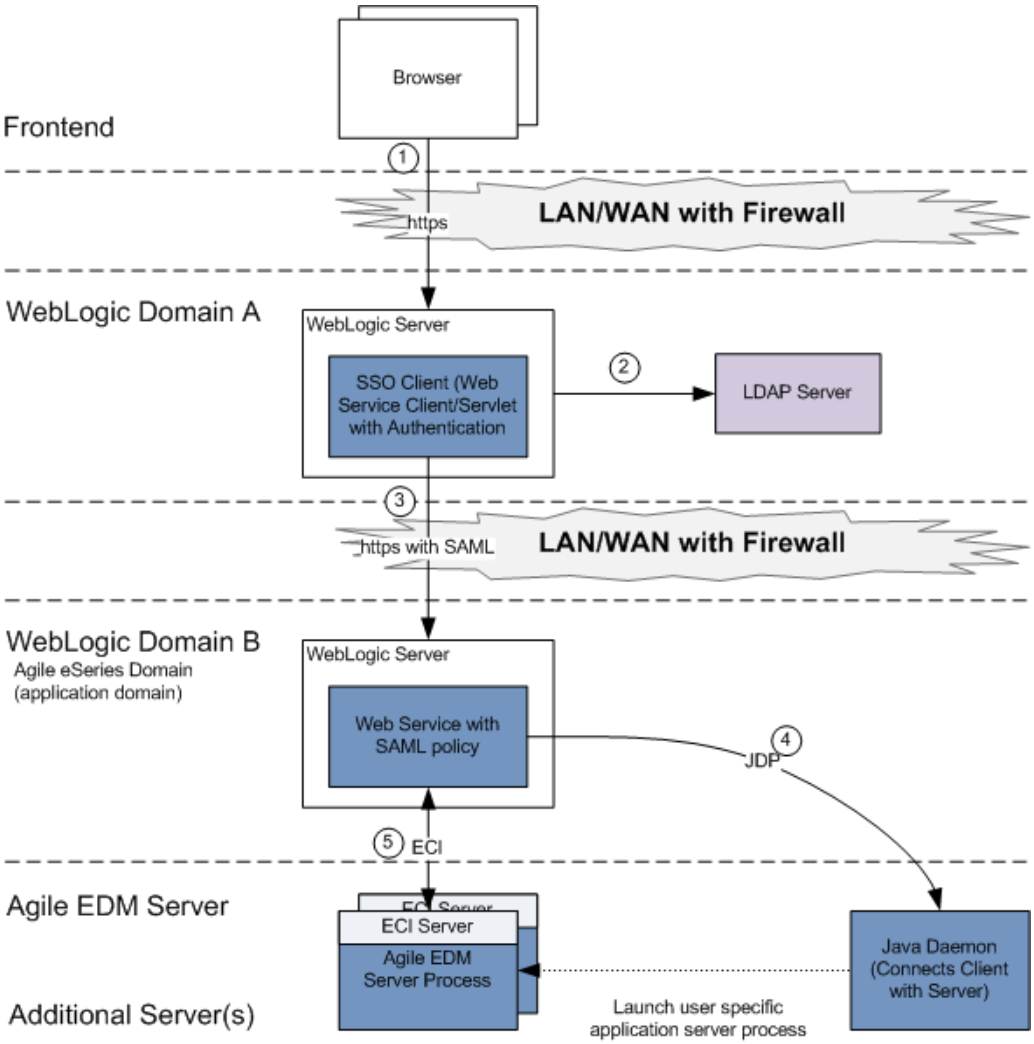
The following graphic depicts the possible single sign-on use case by using a web application in case of the single sign-on client.

- WebLogic Domain A

  Here, the single sign-on client is running and consists of a login JSP page and a Java servlet.

- WebLogic Domain B

  Here, the SAML protected Web Service is running.



The following steps are executed when the user performs a single sign-on with WebLogic and SAML Web Service authentication for Agile e6.

1. The user enters the login credentials in the login JSP page in the browser and sends the request to Domain A where the web application is running.

   This domain is configured with a SAML Identity Provider and an LDAP provider.

2. Authentication occurs by contacting the configured LDAP server.

3. The servlet sends the Web Service request to Domain B where the SAML protected Web Services are running.

   WebLogic attaches the SAML assertion (created by the Identity Provider) to the Web Service request.

4. The Web Presentation Service with SAML policy connects the Java Daemon, requesting to launch a user specific Application Server Process and to return the connection information for this process.

5. If it is a valid user and the SAML assertion is accepted by WebLogic on Domain B, the Web Service invokes the login procedures for Agile e6 and executes the Web Service.

   Finally, the response is sent back via Domain A to the browser.

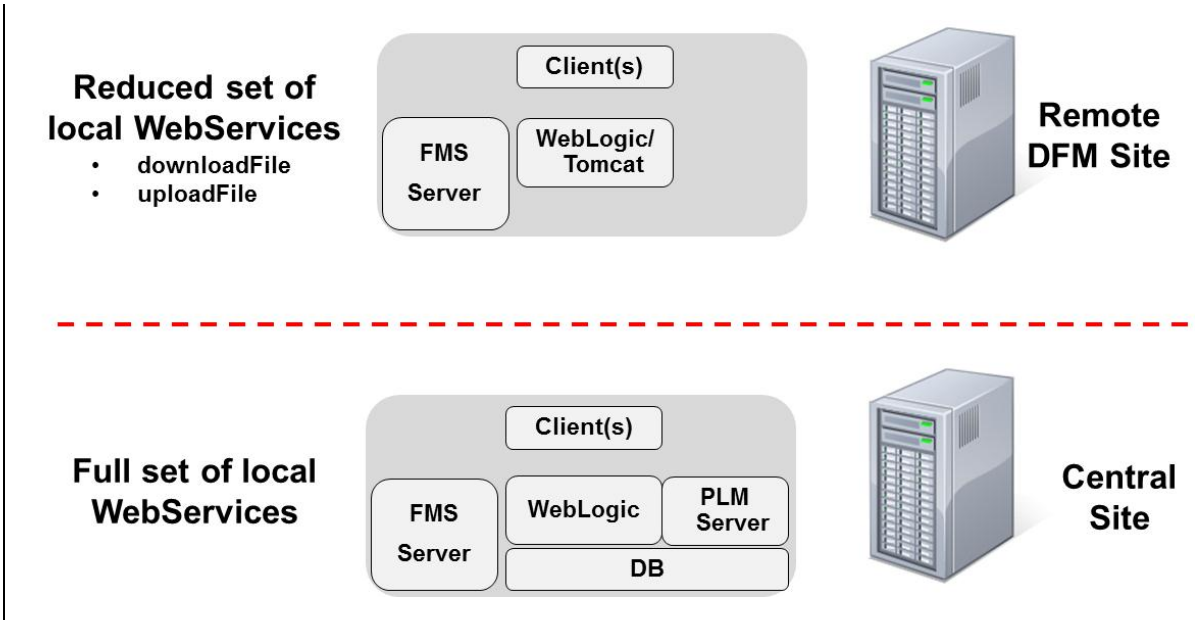
Note For further information about single sign-on with SAML please also refer to the respective Oracle WebLogic Server documentation.

File Handling in a DFM Infrastructure

The Web Service interface leverages all the performance benefits of a DFM infrastructure, by providing key Web Services in a local deployment at the DFM site.

It ensures that file transfer happens between the client and the local File Server, whenever possible.

The following graphic depicts the typical architecture and deployment at a remote DFM site in comparison to the central site of Agile e6.



For further information please refer to the Web Service Guide for Agile e6.1.3.0.