

Oracle® Agile

Engineering Data Management

Web Services Guide for Agile e6.1.3.0

Part No. E50970-02

March 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

Introduction to Web Services 1

About Web Services 1
Core Technologies 2
Web Services Architecture 3
About Agile e6 Web Services 4
The Core Web Services 5
About Agile e6 Web Services Framework 5
Components of Agile e6 Web Services Framework 5

Getting Started with Web Services 7

Prerequisites 7
Operating Environment 7
Web Services Engines 7
Standards Compliance 8
Understanding Web Services Authentication and Performance 9
The Agile e6 Session Handling 9
Understanding the Agile e6 Web Services Requests 11
Obtaining the Agile e6 Metadata 11
Understanding the Agile e6 Web Services Responses 12
Response Status Code 12
Whitelist Mechanism for Masks 12
Exceptions and Warnings 14
countOnly Query Support 14

Setting Up the Web Services Infrastructure 15

Installing the Agile e6 Web Services Framework 15
Creating the WebLogic Agile e6 Domain 15
Testing Inbound Web Services with JDeveloper HTTP Analyzer 15

Configuring Agile e6 Web Services Security 17

Setting Up the Web Services Security Policies 17
Setting Up the Web Services Security 17
Authenticating a Web Service Client 19

Working with Agile e6 Web Services 21

Developing the Outbound Web Services Wrapper 21

The Web Services Wrapper Interface 21

Developing a Custom Wrapper 22

Calling a Custom Wrapper from Agile e6 23

Deploying a Custom Wrapper 24

Web Service Wrapper Log Messages 25

Agile e6 Core Web Services Operations 27

Bulk Operations 27

Bulk Processing of Requests 28

Handling Bulk Requests 28

BusinessObject Web Service 28

Binary Data Transfer 29

Bulk Operations 29

DocumentManagement Web Service 41

DFM Support 41

StreamingFileServices 41

DocumentManagement CoreService 45

Bulk Operations 57

Metadata Web Service 61

Bulk Operations 62

Configuration Web Service 71

Bulk Operations 75

EngineeringCollaborationService Web Service 76

Appendix 87

SAMPLES 87

EchoServiceWrapper.java 87

SampleWrapper.java 89

Web Services Security 95

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 e6 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 e6 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 e6 can be found in the Release Notes file on the Oracle Technology Network (OTN) Web site (http://www.oracle.com/technology/documentation/agile_eseries.html)

Agile Training Aids

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Chapter 1

Introduction to Web Services

About Web Services

Web Services are technologies for building distributed applications. These services, which can be made available over the internet, use a standardized XML messaging system and are not tied to specific operating systems or programming languages. Through Web Services, companies can encapsulate existing business processes, publish them as services, search for and subscribe to other services, and exchange information throughout and beyond the enterprise. Web Services are based on universally agreed upon specifications for structured data exchange, messaging, discovery of services, interface description, and business process design.

A Web Service makes remote procedure calls across the internet using:

- HTTP/HTTPS, or other protocols to transport requests and responses.
- Simple Object Access Protocol (SOAP) to communicate request and response information.

The key benefits provided by Web Services are:

- Service-oriented Architecture

  Unlike packaged products, Web Services can be delivered as streams of services that allow access from any platform. Components can be isolated; only the business-level services need be exposed.

- Interoperability

  Web Services ensure complete interoperability between systems.

- Integration

  Web Services facilitate flexible integration solutions, particularly if you are connecting applications on different platforms or written in different languages.

- Modularity

  Web Services offer a modular approach to programming. Each business function in an application can be exposed as a separate Web Service. Smaller modules reduce errors and result in more reusable components.

- Accessibility

  Web Services can be completely decentralized. They can be distributed over the internet and accessed by a wide variety of communications devices.

- Efficiency

  Web Services constructed from applications meant for internal use can be used externally without changing code. Incremental development using Web Services is relatively simple because Web Services are declared and implemented in a human readable format.

Core Technologies

Agile e6 Web Services use industry standard core technologies. These are:

1. Web Services Description Language (WSDL)
2. XML and XML Schema
3. Simple Object Access Protocol (SOAP)

Web Service Description Language (WSDL)

WSDL is an XML-based format for describing the interface of a Web Service. WSDL describes the endpoints, location, protocol binding operations, parameters, and data types of all aspects of a Web Service:

- The WSDL that describes a Web Service has the following characteristics:
  - It is published by the service provider.
  - It is used by the client to format requests and interpret responses.
  - It can be optionally submitted to a registry or service broker to advertise a service.
- Additionally, WSDL describes the following:
  - The operations that are provided by a Web Service.
  - The input and output message structures for each Web Service operation.
  - The mechanism to contact the Web Service.

XML and XML Schema

A WSDL file is published via an XML file. Document or Literal are required as part of the WS-I interoperability standard. This standard sets the basis for modern Web Service usage.

- Document

  The payload for an operation, however complex, must be defined in a single XML element.

- Literal

  The definition of single XML elements must be described by an XML Schema embedded in the WSDL file.

When using Document or Literal formatting, the WSDL file will contain an XML Schema definition that defines all messages and data types that are used for a particular service. The payload itself consists entirely of XML data structures.

Simple Object Access Protocol (SOAP)

SOAP is a lightweight protocol intended for exchanging structured information in a decentralized distributed environment. SOAP uses XML to define an extensible messaging framework.

SOAP messages consist of the following:

- An envelope for wrapping messages, including addressing and security information.
- A set of serialized rules for encoding data types in XML.
- Conventions for a procedure call and, or response.

Web Services Architecture

You can view Web Services architecture in terms of roles and the protocol stack:

- Roles:
  - Service provider

    This provides the service by implementing it and making it available on the internet.

  - Service requester

    This is the user of the service who accesses the service by opening a network connection and sending an XML request.

  - Service registry

    This is a centralized directory of services where developers can publish new services

or find existing ones.

- Protocol Stack:
  - Service transport layer

    This layer uses the HTTP protocol to transport messages between applications.
  - XML messaging layer

    This layer encodes messages in XML format using SOAP to exchange information between computers. It defines an envelope specification for encapsulated data that is transferred, the data encoding rules, and remote procedure call (RPC) conventions.
  - Service description layer

    This layer describes the public interface to a specific Web Service using the Web Service Description Language (WSDL) protocol. With WSDL, it defines an XML grammar to describe network services. The operations and messages are described abstractly, and then bound to a network protocol and message format. WSDL allows description of endpoints and their messages regardless of what message formats or network protocols are used to communicate.
  - Service discovery layer

    This layer centralizes services into a common registry using the Universal Description Discovery and Integration (UDDI) protocol. UDDI is a platform-independent XML-based registry for businesses worldwide to list themselves on the internet.

About Agile e6 Web Services

Agile e6 Web Services expose a subset of the Engineering Data Management (EDM) functionalities of the Agile e6 application. These services support functionalities provided by EDM modules in 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, standard based discoverable interface
- Java based Web Services framework using Oracle WebLogic
- Modularized Agile e6 Schema (XSD) and WSDL for easy maintenance
- Standard-based WSDL to ensure compatibility across various clients (.NET, Java, and BPEL)
- Bulk APIs wherever applicable for better performance

The Core Web Services

Agile e6 Core Web Services is a set of services for the following functionalities:

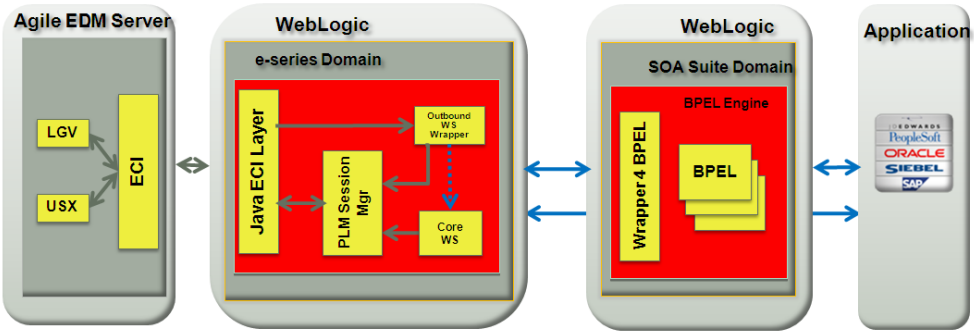
- BusinessObject Web Service
- DocumentManagement Web Service
- Metadata Web Service
- Configuration Web Service
- EngineeringCollaborationService Web Service

About Agile e6 Web Services Framework

The Web Service Framework is an additional layer on top of Agile e6, which supports inbound and outbound communication based on standard Web Services technology. It provides the means to call external Web Services from inside Agile e6 LogiView procedures (outbound direction). In addition, it allows external applications (Web Service Clients) to call the Agile e6 APIs through 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



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.

Chapter 2

Getting Started with Web Services

Prerequisites

Agile e6 Web Services are deployed on an Agile e6 WebLogic application domain.

To use the Agile e6 Web Services framework for inbound and outbound Web Services based on business data transaction, you are required to ensure the following:

- Operational environment is set:
  - Agile e6.1.3.0 is installed.
  - WebLogic Server is installed
- Web Services framework is configured for the following:
  - Authentication provider in WebLogic
  - Web Service Security
- Test the inbound Web Services

You can now call available core services or implement your own outbound Web Service wrappers.

Operating Environment

Agile e6 Application	Release e6.1.3.0 or higher
Default Web Services engine	Oracle WebLogic server 12c or higher Note The version of this server is the one that is released with the Agile e6 application.
Java 2 Platform Standard Edition Development Kit	7.0

Web Services Engines

All application server vendors, such as Oracle, have built-in Web Services infrastructure solutions that are integrated with their application servers. For non-Web Services integrated applications, there are stand-alone products, such as AXIS from Apache, which provide Web Services infrastructure that can be integrated with different application servers.

The Web Service framework works with WebLogic 12c platforms. Additionally, in a Distributed

File Management (DFM) environment, a subset of the Web Service will be available on the DFM client side. In addition to the WebLogic 12c platform, Tomcat or Metro runtime systems can be used as Web Service engines.

Both platforms provide support for MTOM, and for Fileservice operations, the need for MTOM with Streaming SOAP attachments.

The Agile e6 Web Services framework works on the following Web Service engines:

- Oracle Apps Server Web Service Infrastructure
- Oracle SOA Suite
- WebLogic Web Service Infrastructure
- Axis version 2.0 to support JAX-WS features, especially the MTOM

Standards Compliance

The Agile e6 Web Services are implemented in compliance with the following standards:

Standard	Location
Simple Object Access Protocol (SOAP) 1.1/1.2	http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
Web Service Description Language (WSDL) 1.2	http://www.w3.org/TR/2001/NOTE-wsdl-20010315
WS-I Basic Profile 1.1	http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html
XML Schema 1.1	http://www.w3.org/XML/Schema
SOAP Message Transmission Optimization Mechanism (MTOM)	http://www.w3.org/TR/soap12-mtom/
JAX-WS 2.0/2.1/2.2 (JSR 224)	http://java.sun.com/developer/technicalArticles/J2SE/jax_ws_2/
JAXB 2.0/2.1/2.2 (JSR-222)	http://jaxb.java.net/

Understanding 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, this sessionID is 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 will be 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 Web Service security, while fully supported as part of Agile e6 WS-I Basic Profile compliance, does not yield the same benefit as using the higher performance session optimization facility provided by the Agile e6 implementation.

Note For information about Web Service single sign-on please refer to the Security Guide for Agile e6.1.3.0.

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

  First, the Web Service 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.

- 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 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.

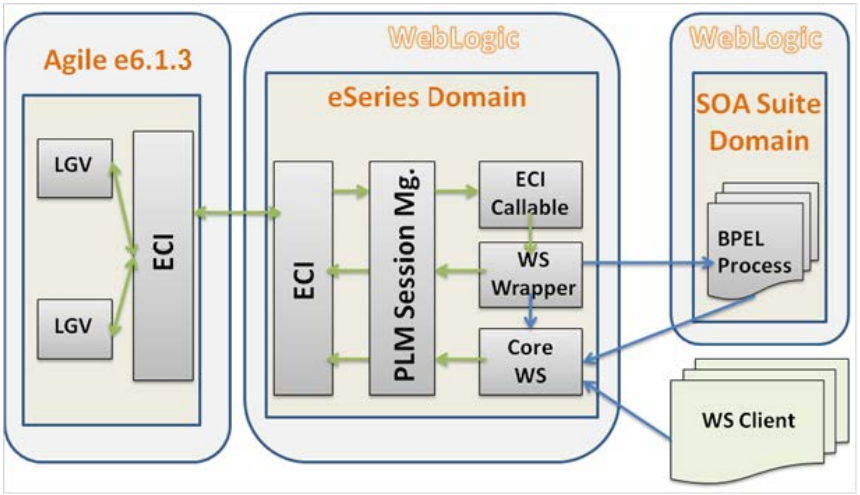
The PLM ticket mechanism is the only way to let the calls to different core Web Services, such as metadata, or Business Object Web Services use the same EDM Server instance.

A Web Service client must 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 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 PLM ticket as the password. This shuts down the EDM 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 EDM 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 EDM Server.

To retrieve a PLM session, a PLM Ticket is provided. When a new PLM Session is created, the PLM Ticket is set to the EDM Server instance, which is then set into the SOAP message to the client side.

The life cycle of a PLM session is the same as the given HTTP session. The timeout of a HTTP session is specified in the file 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 EDM Server instance that was used in the last request. The ticket remains valid only as long as the EDM Server instance is running. After obtaining a ticket, the client code needs to configure the port by setting the ticket string as password. See BindingProvider in the example given under Authenticating a Web Service Client.

The PLM Ticket improves the Web Services performance and simplifies the session management. If different Web Services are used in a usecase flow, which is very likely, the ticket returned by the response(s) 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.

Understanding the Agile e6 Web Services Requests

In the Agile e6 Web Services framework, each operation has its own request data type, which is inherited from RequestHeaderType. The RequestHeaderType for all the requests has only the following elements:

- messageID (String, optional):

  Default value for the ID is the current system time in milliseconds.

- messageName (String, optional):

  Default value for the message name is the simple class name.

Obtaining the Agile e6 Metadata

You can obtain the basic Agile e6 metadata through Agile e6 Java Client. Look for the data model of the Agile e6 application.

To obtain Agile e6 metadata through a Web Services operation, use the metadata service. This service requires an entity name and a mask name.

For further information see section Configuration Parameters.

Understanding the Agile e6 Web Services Responses

The ResponseHeaderType has the following members:

messageId (String, required)	Default value for the ID is the current system time in milliseconds.
messageName (String, required)	Default value for the message name is the simple class name.
statusCode (ResponseStatusCode, required)	Default value for the status code is SUCCESS.
exceptions (List<PlmExceptionType>, optional): warnings (List<PlmWarningType>, optional)	
ticket (String, optional)	

Response Status Code

The response obtained from every Web Service call contains a response statusCode, which indicates the success or failure of a Web Service operation.

These response status codes are of four types:

SUCCESS	Indicates that all Web Services in the batch were executed successfully and that all operations worked as intended.
FAILURE	Indicates that all Web Services in the batch failed during execution, indicating the intended operations were not performed.
WARNING	Indicates that while Web Services in the batch were successfully executed, however certain warnings were also encountered during the execution. These warnings need to be analyzed by the client to verify that all operations worked as intended.
PARTIAL_SUCCESS	Indicates a partial success in the execution of batch Web Services when one or more but not all batch requests have failed. Even if a single Web Service fails among a batch of Web Services, the response status code indicates PARTIAL_SUCCESS.

Whitelist Mechanism for Masks

To ensure that only the masks designed for the access of Web Services are used, all the mask names are checked against a Whitelist that is maintained by the administrator of the Agile e6 installation.

List of Mask Names

A configuration rubric named EDB-WSI-MASKS is used. It contains sub-parameters such as EDB-ARTICLE-WSI and each of these sub-parameters contain a mask name rule pattern. All rules for an entity are checked for an operation that involves an item as an entity (or as parent entity in case of a relation).

Important	It is recommended to implement special masks designed for Web Services, instead of only adding the standard masks like EDB-ART-SLI (or “*”) to the Whitelist .

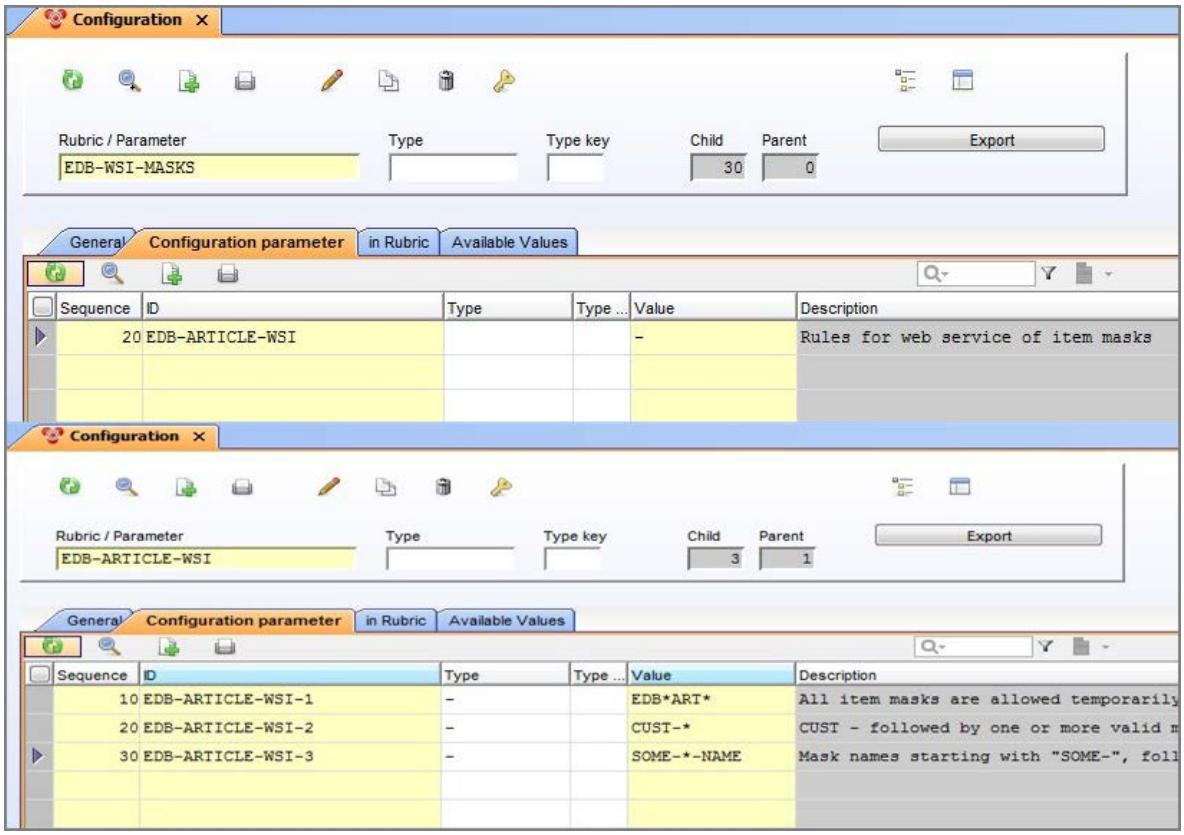
Adding standard masks can have certain implications due to the following:

- Standard masks may expose sensitive data.
- To be able to access invisible fields, you are required to make them visible. This must be done as they are not opened to the customer.
- Performance suffers as the standard masks contain too many fields.

Note	For all fields located on sub-forms that are part of a combined form, the pre-field userexit will not be executed. Thus, we recommend using lists instead of combined forms.

Configuration Parameters

The Configuration Parameters are entered as shown in the image below.



The rule pattern consists of valid mask name characters and may contain one or more asterisks (*) to indicate one or more mask name characters.

The Web Service session reads these configuration rules and checks each combination of entity and mask name passed by a client against the rules for the respective entity. If one of the rules accepts the mask name, access is granted, If not, the access to the mask is denied and an IllegalAccessException is thrown and marshaled back to the client.

The rules are cached by the session so that the subsequent operations do not have the overhead of reading the rules again. The rules can also be cached in the Business Service, this provides a domain wide cache, instead of a session based cache.

Exceptions and Warnings

The Agile e6 framework throws an exception (WebFault) only if a severe technical problem occurs, for instance, in an event of a connection loss to the EDM Server. When an operation is not successful, the system throws an exception or a warning.

- In case of FAILURE, an exception is issued, while a warning may be issued.
- In case of WARNING, only a warning is issued.

When the status is WARNING, the outcome of the operation is unknown. You are manually required to check if the operation was successful.

countOnly Query Support

A count request is indicated by the flag countOnly in the query request object. The service then executes a count, which ignores the mask limit, and returns a pseudo PlmObject with a COUNT attribute and a RECORD_LIMIT attribute.

The value of the COUNT attribute is an Integer with the number of objects matching the query, and the value of the RECORD_LIMIT attribute is the record limit of the mask used for the count operation.

Chapter 3

Setting Up the Web Services Infrastructure

Installing the Agile e6 Web Services Framework

The Agile e6 Web Services framework is installed during the basic installation of the Agile e6 application. The WebLogic domains are created with the Agile e6 application installer and/or the Agile e6 Administration Client. By default, the installation has two WebLogic domains – one for installation, and an additional one for every application. Web Services are deployed on the application specific WebLogic domain.

For complete details, please refer to the Administration Guide for Agile e6.1.3.0.

Creating the WebLogic Agile e6 Domain

You are required to obtain a WebLogic Agile e6 domain name from the Agile e6 administrator, or create a new domain. This is required as the Web Services are deployed into this WebLogic Agile e6 domain.

During the installation of the WebLogic Server, two domains are created. Each domain consists of an AdminServer and an eSeries-01 server. The AdminServer is only for the administration of the domain, while the eSeries-01 contains the Agile e6 deployments.

For example, the domains directory C:\Oracle\Middleware\user_projects\domains contains the domain names eSeries_domain, and eSeries_domain_plmref.

For complete details on how to install the WebLogic Server and create a domain, please refer to the Server Installation Guide on Windows and UNIX for Agile e6.1.3.0.

Testing Inbound Web Services with JDeveloper HTTP Analyzer

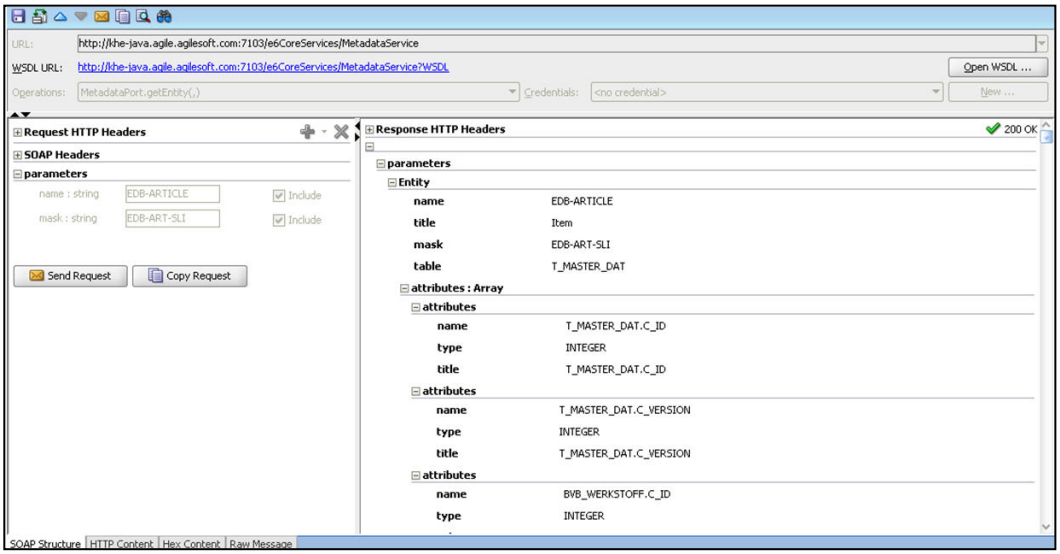
The Web Service operations can also be called and tested using the JDeveloper HTTP Analyzer.

You are required to use HTTP/S for the basic authentication with the HTTP analyzer; otherwise you cannot call an operation.

Note For more details, see chapter Configuring the Agile e6 Web Services Security.

To test the Web Services with the JDeveloper HTTP Analyzer:

1. Launch the JDeveloper IDE.
2. Open the HTTP Analyzer screen.
3. Enter the URL of the WSDL of the Web Service you want to call.
4. In the Request area on the left, enter the input parameters for name:string and mask:string, such as EDB-ARTICLE and EDB-ART-SLI.
5. Click on Send Request.
6. Upon successful call of the Web Service, the result data is displayed in the Response area of the console.



Chapter 4

Configuring Agile e6 Web Services Security

The Web Services configuration can be used to configure the Web Services security. Since Web Services are not secured by default, it is required to configure/establish a Web Service policy that meets your security strategy requirements.

Note The Agile e6 Web Service needs user credentials to connect to the Agile e6 application server. These credentials are provided by the Agile e6 authentication provider which passes this information to the Web Service. The Web Service has to be configured to be secure. An unauthorized Web Service cannot work.

Note For information about Web Service single sign-on please refer to the Security Guide for Agile e6.1.3.0.

Note The SSL port needs to be activated for the domain where Web Services are deployed. The standard listen port (non-SSL) must be disabled for the domain where Web Services are deployed.

In the examples used in this chapter, a Web Service security policy is used to secure the entire Web Service. The client has to provide the WSS: SOAP message security user name token encrypted with an X.509 certificate.

Note The certificate will be authenticated, too. The certificate must be valid and the certificate name must be available in the system.

Setting Up the Web Services Security Policies

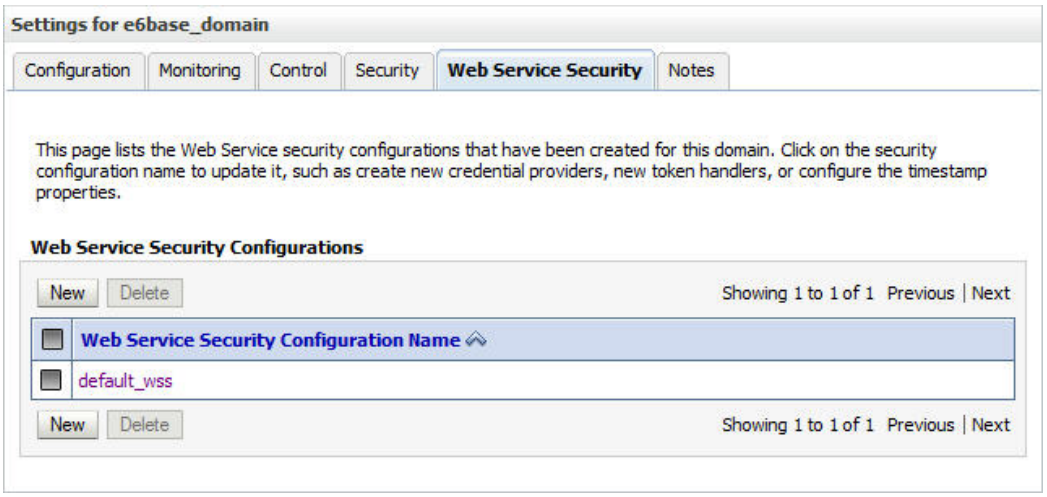
Note In the current release of Agile e6 Core Web Services security policies are supported, except for the StreamingFileServices uploadFile and downloadFile. The file streaming only works with SSL (by using HTTPS). It does not work if HTTPS is enforced by adding a policy. All other Web Services can be controlled by the Web Services security policies.

For more details on setting up Web Service Security Policies, see http://docs.oracle.com/cd/E24329_01/web.1211/e24488/message.htm#i244059

Setting Up the Web Services Security

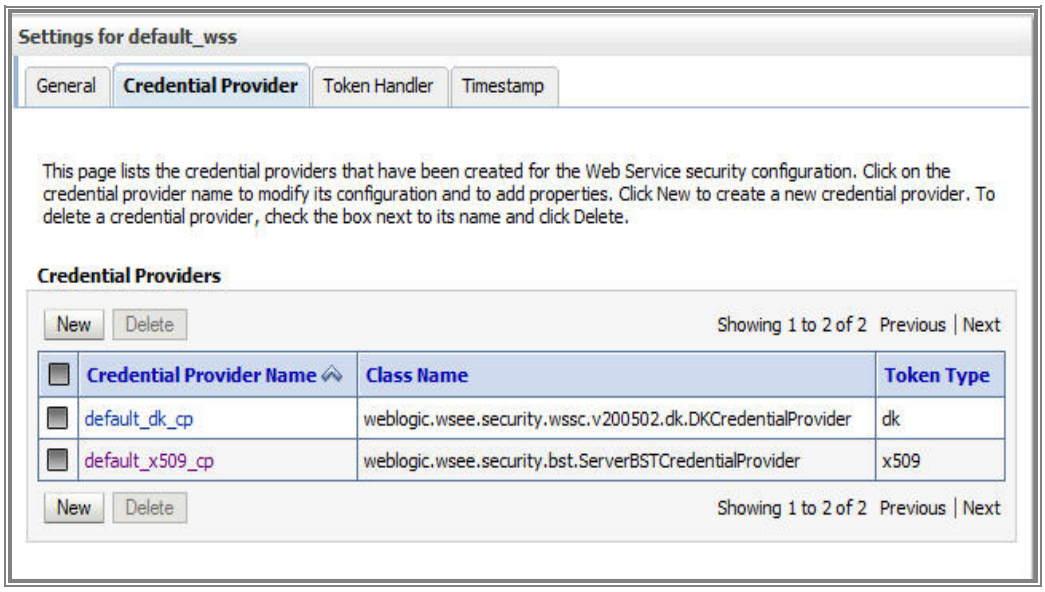
1. X.509 Authentication

   For X.509 authentication, you need to configure the Web Service Security Configurations.



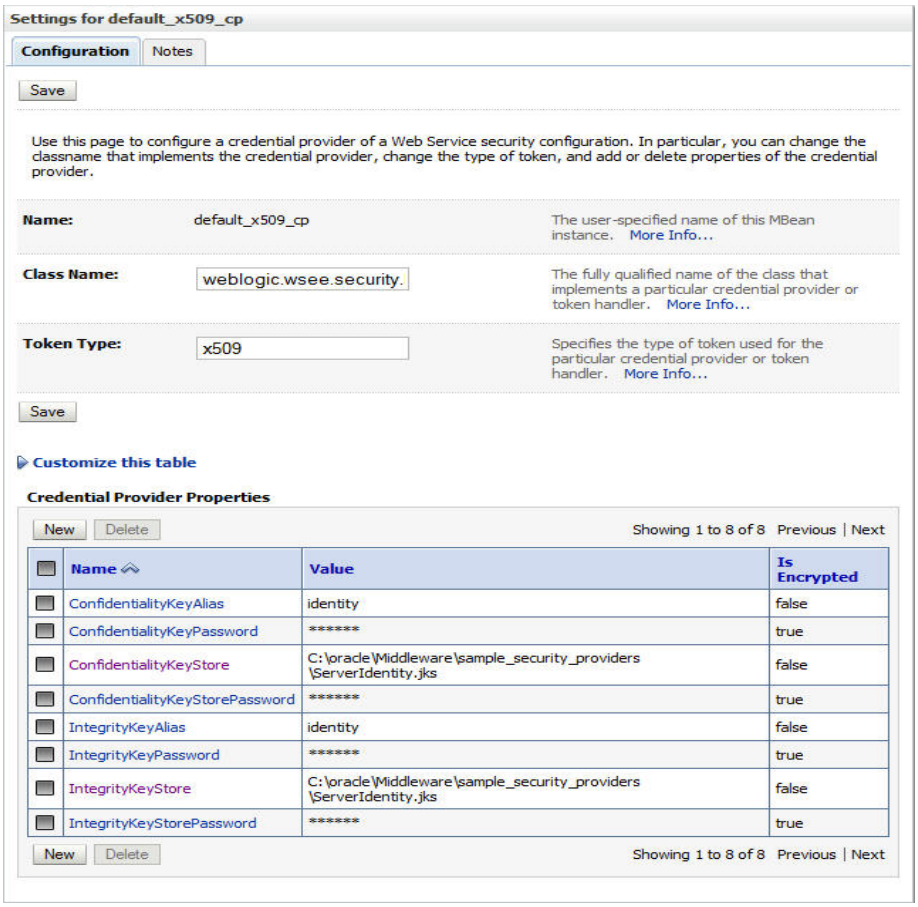
2. Use the default providers given by the WebLogic server.

   In this example, when you click on the default Web Service default_wss, the Settings for default_wss mask opens. The Credential Provider column lists a number of available default providers, created for the Web Service security configuration.



3. Store the x509 server certificate in a Java keystore.

The following screenshot shows how it can be configured.



Authenticating a Web Service Client

The caller of an Agile e6 Web Service has to provide user credentials to gain access to the Agile e6 application. The attributes of these credentials depend on the used Web Service policy. The following is an example of an unauthenticated Web Service call:

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns4:getVersion
xmlns:ns2="http://xmlns.oracle.com/Agile/e6/Metadata/v0"

xmlns:ns3="http://xmlns.oracle.com/Agile/e6/plm"
xmlns:ns4="http://xmlns.oracle.com/Agile/e6/HelloWorld/v0" />
    </S:Body>
</S:Envelope>
```

A Sample of HTTP/S Authentication

You can use the basic authentication of HTTP/S to secure a Web Service. With this basic authentication of HTTP/S, the user credentials are stored in the HTTP/S header. The SOAP message does not carry any security information.

Note Basic authentication without SSL must not be used in a production environment as the passwords and data are transferred in plain text. Ideally, HTTP/S must be configured.

WebLogic production servers should always use SSL (HTTP/S) and should not provide unencrypted access to any of the services.

For complete details on WebLogic Security Fundamentals and Transport Level Security, refer to the WebLogic documentation on OTN.

To add the HTTP/S basic authentication to a SOAP request, the code may look like the following example:

```
MetadataService service = new MetadataService(wsdlURL, serviceQName);

BindingProvider bindingProvider = (BindingProvider) service.getPort();

bindingProvider.getRequestContext().put(BindingProvider.USERNAME_PROPER
TY, username);
bindingProvider.getRequestContext().put(BindingProvider.PASSWORD_PROPER
TY, password);
bindingProvider.getRequestContext().put(BindingProvider.SESSION_MAINTAI
N_PROPERTY, Boolean.TRUE);
```

Note The last line of this example configures HTTP/S session handling. If you do not add this code, each Web Service call creates a new HTTP/S session which leads to a new EDM Server instance starting up.

A detailed sample for Web Services Security can be found in chapter Appendix, section Web Services Security.

Chapter 5

Working with Agile e6 Web Services

Developing the Outbound Web Services Wrapper

The Web Services 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.

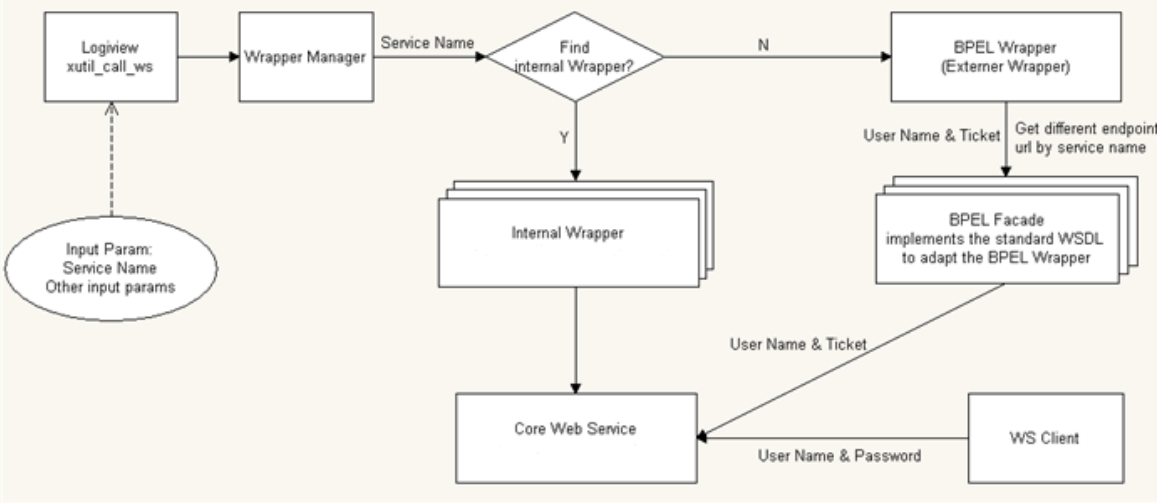
If it is an asynchronous Web Service, the wrapper returns a string list with the correlation ID. Otherwise, it transforms the XML payload returned by the Web Service into a string list that is expected by the calling EDM Server process.

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 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 BPEL facade.

Normally, this facade must 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 must implement a standard WSDL. The External wrapper uses this standard WSDL to generate the proxy. It can then 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 directory APP-INF/classes together with the file application.properties for the Web Services.

Example:

```
bpel.wrapper.SampleExternalService2.wsdl=http://<server><port>/soa-
infra/services/default/SOAComposite2/BPELFacadeService?WSDL
```

In the example above, when we pass the service name as SampleExternalService2 in xutil_call_ws, the wrapper manager first looks for an internal wrapper named as SampleExternalService2. If this internal wrapper does not exist, then the wrapper manager calls the external wrapper and gets the endpoint with the key value bpel.wrapper.SampleExternalService2.wsdl in the ExternalWrapper.properties file.

Note You are required to use the custom staging mechanism of the Agile e6 installer to add your own mappings to the file ExternalWrapper.properties.

Session Management Integration

In the external wrapper, the user name and ticket must be passed to the BPEL facade. The BPEL facade uses this information to invoke the Agile e6 core Web Services and reconnect to the same Agile e6 PLM session to get more data.

Developing a Custom Wrapper

In order to compile a custom wrapper, you need the libraries contained in the Web Services application. These libraries can be found at ${ep_root}/staging/product/WebServices/WebServices.ear/APP-INF/lib.

If your wrapper calls an external or internal Web Service, you also need to add the generated

client classes or any other infrastructure classes needed. This depends on the Web Service client framework that you use.

Important	We highly recommend using the WebLogic Web Service framework when implementing a wrapper for a Web Service because the wrapper is run inside WebLogic.

The name of the wrapper class must be <ID>Wrapper, as the wrapper manager looks for this string when the EDM Server tries to call it.

The imports used by the wrapper are in the libraries contained in the lib directory inside the file WebServices.ear (at APP-INF/lib). By default, the wrapper class must be in the package com.agile.ws.e6.wrappers so that the wrapper manager can find it at the runtime. However, it is also possible to add other package names to the search path by adding them to the application.properties file of the Web Services application:

```
WrapperPackage.Custom.1 = some.custom.wrapperpackage
WrapperPackage.Custom.2 = another.custom.wrapperpackage
```

More sample details at EchoServiceWrapper.Java and SampleWrapper.java.

Calling a Custom Wrapper from Agile e6

A new C/C++ userexit is provided that can be called from LogiView or C/C++ to make an outbound Web Service call. The request is sent as an ECI call to the ECI server embedded in the application server, which then calls the respective wrapper for XML processing.

To limit any XML parsing in LogiView and C/C++, the userexit sends a string containing a userexit parameter as an input to the wrapper and it expects a list of string as a result from the wrapper.

The userexit accepts an input argument that uses the syntax prescribed by CallableParam and zag_cnv_arg. This argument is passed to the ECI callable Eci_call Web Service in the application server.

This userexit is called xutil_call_ws and can be used in LogiView as follows:

```
EP_APP_CMD = "EchoService"
EP_APP_CID_STRING = ""
EP_APP_CONTENTS = "Please echo: something"
RES = #xutil_call_ws(EP_APP_CMD, EP_APP_CID_STRING, EP_APP_CONTENTS,
EP_APP_RESULT, EP_APP_ERROR)
if (RES == 0)
put(strprint("Web service %s returned: %s", EP_APP_CMD, EP_APP_RESULT))
     else
put(strprint("Error %d when calling web service %s, error message
is:\n%s", RES, EP_APP_CMD, EP_APP_ERROR))
     endif
```

This LogiView code uses four existing string variables and two new ones to call the wrapper for the EchoService. The EchoService returns the input arguments as a result, which is provided as a standard wrapper to test the infrastructure.

The userexit xutil_call_ws consists of the following parameters:

EP_APP_CMD	Wrapper Name
EP_APP_CID_STRING	Correlation ID. This is used to correlate a later response, which may come in asynchronously, to the initiating request.
EPP_APP_CONTENTS	Any string value that the wrapper must interpret.
RES	Result string from the Wrapper
EP_APP_RESULT	A new string variable that needs to be created to call the wrapper for the EchoService.
EP_APP_ERROR	A new string variable that needs to be created to call the wrapper for the EchoService.
The userexit provides a return code - "0" for success and any other number for errors.	

Deploying a Custom Wrapper

1. Identify the classes you need for your wrapper and create a JAR file containing them.

   Give your JAR files proper names to avoid naming conflicts with existing wrapper implementations.

   For example, use a unique prefix that identifies your company.

2. If you need third party JAR files.

   Ensure that they are not already provided by WebLogic server or the Agile e6 Web Services application. Do not use JAR files that duplicate features or come in conflict with the WebLogic server, like using different Web Service client frameworks.

   Note Your wrapper implementation will run inside and as part of a Web Service application deployed into WebLogic server.

3. Copy all JAR files needed by your wrapper implementation to the custom staging directory for the Web Services application. This must be:

   `${ep_root}/staging/custom/WebServices/${app_name}.ear/APP-INF/lib.`

   Note Do not copy any library that are already part of the Web Services application into the custom staging area. These libraries are only replaced by updates from Oracle, and are always put in the product staging area.

4. Redeploy the Web Services into the WebLogic Server.

For complete details on how to deploy an application, see the Administration Guide for Agile e6.1.3.0 or the Hotfix Readme.

Web Service Wrapper Log Messages

A log file looks similar to the one shown below, depending on what has been implemented in the Java based wrapper for message logging.



Chapter 6

Agile e6 Core Web Services Operations

This chapter describes the Agile e6 Core Web Services operations. The use of these operations to process the bulk requests is described in section Bulk Processing of Requests.

Some of the operations, such as getRelations, require counting the number of records. For more information on these, see section countOnly Query Support in this document.

For additional information on these Web Services and operations, the following documents will also be available from the Oracle Software Delivery Cloud website: https://edelivery.oracle.com:

- Agile e6 Web Services Schema Docs, including the PLM Data Types document
- Agile e6 Web Services SOAP Samples

Bulk Operations

Certain situations require several calls to be made to a Web Service operation. In a WAN environment, calling the operation several times would result in an unacceptable performance penalty. So it is critical to limit the number of Web Service calls for each use case. Therefore, most of the Agile e6 Web Service operations support bulk mechanism.

Example:

Instead of calling the Web Service operation BusinessObjectService.createObject one hundred times, you can call the operation BusinessObjectService.AN and assign a list of the one hundred single requests to create an object. Each nested request in the bulk call has the full flexibility of the single request, so it is possible to create ten projects, fifty items and forty documents in one bulk call.

To link the objects together, you can call another bulk operation called BusinessObjectService.createRelationBulk.

Note You are allowed to create a complex graph of objects in only two Web Service calls.

Note The bulk mechanism can be configured to either stop on the first failure of a nested request, or to continue and report all errors and warnings that occurred during the processing of the nested requests.

The result is a bulk response with nested responses, matching the list of nested requests in the bulk request.

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 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, as a result of which the results are in random sequence. 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 is 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.

The content of the response object depends on the status code, as listed below:

SUCCESS	A list with all response objects matching the list of requests. All requests succeeded.
FAILURE	A list with all response objects matching the list of requests. One or more requests failed, check the respective responses. The last executed request failed with an exception, which is returned as the fault in the bulk response. If one request fails due to lack of mask in Whitelist, status code would be FAILURE instead of PARTIAL_SUCCESS even if other services are successful.
WARNING	The bulk request does not contain any request.
PARTIAL_SUCCESS	A list with all response objects matching the list of requests. One or more requests failed, check the respective responses. If stopOnFailure was requested, the list ends with the first response that failed.

BusinessObject Web Service

The BusinessObject Web Service enables you to create retrieve, update, and delete PLM objects belonging to an entity, an entity type, and a relation. All operations require one request

object as input, and return one response object.

- The request contains attribute values used to search or create a PLM object.
- The response contains the data of the respective objects.
- Bulk operations allow you to execute a whole list of requests with a single Web Service call. The response of a bulk operation contains a list of responses matching the list of requests.

The PLM objects are read from the Agile e6 application using the mask specified in the request. It is only possible to access Agile e6 attributes that are visible in this mask, with the exception of the ID fields EDB_ID and C_ID. Only the masks listed in the Web Service Whitelist of the Agile e6 application can be accessed.

- Use the EBD_ID if you need to keep the reference to a PLM object, especially if it is stored in another system.
- The C_ID needs to be used only to build internal object graphs, for instance when filling a UI element.

Note Not all PLM objects have an EDB_ID or a C_ID. Check the customization of the respective Agile e6 system before deciding how to make external references to PLM objects of that system.

All operations allow you to specify the attributes that you require to return from the PLM object. If no return attributes are specified, all the accessible attributes are returned, which correspond to all visible fields of the mask. Optionally, all languages of multi-lingual attributes can be returned, but may result in a considerably larger response object.

Binary Data Transfer

It is possible to request the transfer of binary data (BLOBs). However, by default, binary data is not transmitted; binary attributes that are to be transferred need to be visible in the underlying Agile e6 mask. BLOBs are transferred using the MTOM feature of JAX-WS.

Bulk Operations

The following are the bulk operation names of the single request operations for Business Object Web Services describing the concurrent sections.

- createObjectBulk
- getObjectsBulk
- updateObjectBulk
- deleteObjectBulk

- createRelationBulk
- updateRelationBulk
- getRelationsBulk
- deleteRelationBulk
- setReservationBulk

createObject

<table>
<tr><td>Service</td><td>To create an object in the Agile e6 system.</td></tr>
<tr><td>Usage</td><td>The request includes a PlmObject which describe the object to create and a PlmResult to define the return values of the response. By default, the response returns values of all visible fields (plus some important ID fields like EDB_ID and C_ID) that are contained in the mask.

Response information can be restricted by defining the list of fields that must be returned in the PlmResult of the request, but note that you can only return values for fields which are available in the mask. The settings of the mask determines the sorting of the data. Return values are provided in a standardized format as marshaled by the JAXB framework. Date values are in UTC, based on the assumption that the data returned by the EDM server is in server local time.

By default, only the current language value is retrieved for multi-lingual fields. However, the Web Service client can request to retrieve all language values in the same call.

If the object already exists in Agile e6, or an error occurs during creation of the object, an error code is returned.</td></tr>
<tr><td>Request Type</td><td>CreateObjectRequestType
▫ messageId (String): ID to be returned in the response (optional)
▫ messageName (String): Name to be returned in the response (optional)
▫ plmObject: The PLM class reference and a list of object attributes (name/value pairs) used to create the new object.
▫The metadata of the plmObject is currently ignored by this operation, thus you can pass empty values for these elements.
▫Only the list of attribute values is used to fill the new record in Agile e6.
▫ plmResult: Describes how the result of the operation is returned to the client.
<table>
<tr><td>attributeNames (Boolean)</td><td>List of field names to be returned</td><td>▫ Optional
▫ Default
▫All visible fields and ID fields</td></tr>
<tr><td>includeBinaryValues (Boolean)</td><td>Include binary values?</td><td>▫ Optional
▫ Default – False</td></tr>
<tr><td>includeAllLanguages (Boolean)</td><td>Include all languages?</td><td>▫ Optional
▫ Default – False</td></tr>
</table>
</td></tr>
<tr><td>Response Type</td><td>CreateObjectResponseType</td></tr>
</table>

<table>
<tr>
<td></td>
<td>

statusCode (ResponseStatusCode):
<table>
<tr><td>SUCCESS</td><td>The newly created PLM object including all attributes defined by the PlmResult of the request.</td></tr>
<tr><td>PARTIAL_SUCCESS</td><td>The newly created PLM object including some of the attributes defined by the PlmResult of the request.</td></tr>
<tr><td>FAILURE</td><td>A faulty description including an error message and the type of error.</td></tr>
</table>

ticket (String): A PLM ticket.

The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
The ticket is only valid for the EDM server instance that generated it.

messageId (String): Copied from the request, or generated.
messageName (String): Copied from the request, or the operation name.
warnings: List of warnings (PlmWarningType) that occurred during the operation.
exceptions: List of exceptions (PlmExceptionType) that occurred during the operation.
object (PlmObject): The object created by this operation.
It contains the attributes requested in the PlmResult, or all visible attributes if no attributes have been requested.

</td>
</tr>
</table>

getObjects

Service	To retrieve the requested objects from the Agile e6 system.
Usage	It executes a query in the Agile e6 system and returns the matching objects. The operation uses the mask that is specified in the request to retrieve the data (search in mask). Response information can be restricted by defining the list of fields that must be returned. Note that you can get the values only for fields that are available in the mask. The sorting of data determines the settings of the mask. Date values are returned in UTC, based on the assumption that the data returned by the EDM server is in Server Local Time. By default, only the current language value is retrieved for multi-lingual fields. However, the Web Service client can request to retrieve all language values in the same call. Objects for which the current user does not have access to, does not show up in the response (this is contrary to client behavior where one can see such objects as dotted (...) objects. At the server, this is controlled by a DataView default. The countOnly flag of the request is very important here. For more information on this, refer to countOnly Query Support in this document.
Request Type	GetObjectsRequestType ▫ messageId (String): ID to be returned in the response (optional).

<table>
<tr><td></td><td>

messageName (String): Name to be returned in the response (optional).
plmQuery (PlmQuery):

plmClass: The object type as a PlmClassRef (PLM class reference, as provided by the metadata service).
selection: A list of PlmCondition objects representing the search criteria for object attributes.

<table>
<tr><td>ignoreRecordLimit (Boolean)</td><td>Ignore the record limit?</td><td>Optional
Default - False</td></tr>
<tr><td>countOnly (Boolean)</td><td>Count only?</td><td>Optional
Default - False</td></tr>
<tr><td>attributeNames (Boolean)</td><td>List of field names to be returned?</td><td>Optional
Default
All visible fields and ID fields</td></tr>
<tr><td>includeBinaryValues (Boolean)</td><td>Include binary values?</td><td>Optional
Default - False</td></tr>
<tr><td>includeAllLanguages (Boolean)</td><td>Include all languages?</td><td>Optional
Default - False</td></tr>
</table>
</td></tr>
<tr><td>Response Type</td><td>
GetObjectsResponseType

statusCode (ResponseStatusCode):

<table>
<tr><td>SUCCESS</td><td>The query was executed without any problem. The list of objects might be empty.</td></tr>
<tr><td>PARTIAL_SUCCESS</td><td>One or more attributes, defined by the PlmResult of the request, were not found in the queried objects. Details can be found in the list of warnings and exceptions.</td></tr>
<tr><td>FAILURE</td><td>A faulty description including error message and the type of error.</td></tr>
</table>

ticket (String): A PLM ticket.

The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
The ticket is only valid for the EDM server instance that generated it.

messageId (String): Copied from the request, or generated.
messageName (String): Copied from the request, or the operation name.
warnings: List of warnings (PlmWarningType) that occurred during the operation.
exceptions: List of exceptions (PlmExceptionType) that occurred during the operation.
recordLimitHit (Boolean): Indicates whether the query result has hit the mask limit.
objects: List of objects (PlmObject) found including all the object attribute values listed in the PlmResult of the request.

If a count request was made, the response will have one PlmObject with an Integer attribute named COUNT containing the number of objects matching the query, and an Integer attribute called RECORD_LIMIT containing the current record limit of the mask used for the count operation.

</td></tr>
</table>

updateObject

<table>
<tr><td>Service</td><td>To update an object in the Agile e6 system.</td></tr>
<tr><td>Usage</td><td>The operation supports an automatic creation of an object if it was not found. The flag autoCreate in the request controls this behavior.

By default, the response returns values of all visible fields (plus some important ID fields like EDB_ID and C_ID) that are contained in the mask. The response information can be restricted by defining the list of fields that must be returned in the PlmResult of the request, but note, that you can only return values for fields which are available in the mask. The settings of the mask determines the sorting of the data. Return values are provided in a standardized format as marshaled by the JAXB framework. Date values are in UTC, based on the assumption that the data returned by the EDM server is in server local time.

By default, only the current language value is retrieved for multi-lingual fields. However, the Web Service client can request to retrieve all language values in the same call.

If an error occurs while creating the object, an error message is returned.</td></tr>
<tr><td>Request Type</td><td>UpdateObjectRequestType
▫ messageId (String): ID to be returned in the response (optional).
▫ messageName (String): Name to be returned in the response (optional).
▫ plmObject: The PLM class reference, the ID of the object to update, and a list of object attributes (name/value pairs) used to update or create the object.
<table>
<tr><td>autoCreate (Boolean)</td><td>Automatically create the object if it does not exist.</td><td>▫ Optional
▫ Default - False
▫False = Returns a FAILURE code in case the object ID does not match the existing Agile e6 record.
▫True = Object will be created with the attributes passed in the request, in case the object ID does not match an existing Agile e6 record.</td></tr>
</table>
▫ plmResult: Describes how the result of the operation is returned to the client.
<table>
<tr><td>attributeNames (Boolean)</td><td>List of field names to be returned</td><td>▫ Optional
▫ Default
▫All visible fields and ID fields</td></tr>
<tr><td>includeBinaryValues (Boolean)</td><td>Include binary values</td><td>▫ Optional
▫ Default – False</td></tr>
<tr><td>includeAllLanguages (Boolean)</td><td>Include all languages</td><td>▫ Optional
▫ Default - False</td></tr>
</table></td></tr>
<tr><td>Response Type</td><td>UpdateObjectResponseType
▫ statusCode (ResponseStatusCode):</td></tr>
</table>

	<table><tr><td>SUCCESS</td><td>The newly created PLM object, including all attributes defined by the PlmResult of the request.</td></tr><tr><td>PARTIAL_SUCCESS</td><td>The newly created PLM object including some of the attributes defined by the PlmResult of the request.</td></tr><tr><td>FAILURE</td><td>A fault description including error message and the type of error.</td></tr></table> ▫ ticket (String): A PLM ticket. ▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the PLM session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session. ▫The ticket is only valid for the EDM server instance that generated it. ▫ messageId (String): Copied from the request, or generated. ▫ messageName (String): Copied from the request, or the operation name. ▫ warnings: List of warnings (PlmWarningType) that occurred during the operation. ▫ exceptions: List of exceptions (PlmExceptionType) that occurred during the operation. ▫ object (PlmObject): The object updated or created by this operation. ▫It contains the attributes requested in the PlmResult, or all visible attributes if no attributes have been requested. ▫ autoCreated (Boolean): This flag is true, if the object has been created by this operation, and false if the object has been updated.

deleteObject

Service	To delete an object in the Agile e6 system.
Usage	Note The object is deleted permanently without using the trash basket. A mask can be specified, which is used to process the delete operation. This allows to fire customer specific triggers on the EDM server during record deletion.
Request Type	DeleteObjectRequestType ▫ messageId (String): ID to be returned in the response (optional). ▫ messageName (String): Name to be returned in the response (optional). ▫ plmObject (PlmObject or PlmObjectRef): The PLM class reference and the ID query of the object to be deleted.
Response Type	DeleteObjectResponseType ▫ statusCode (SUCCESS or FAILURE) ▫ ticket (String): A PLM ticket. ▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the PLM session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session. ▫The ticket is only valid for the EDM server instance that generated it. ▫ messageId (String): Copied from the request, or generated.

	▫ messageName (String): Copied from the request, or the operation name. ▫ warnings: List of warnings (PlmWarningType) that occurred during the operation. ▫ exceptions: List of errors (PlmExceptionType) that occurred during the operation. ▫ object (PlmObject): PLM object with ID information like EDB_ID and C_ID, if available (optional). ▫This is only returned if the object is found, but the operation fails.

createRelation

Service	To create a relation between one parent object and one child object of an Agile e6 entity.
Usage	The request includes a PlmObject which describes the relation object to create a PlmObjectReference for the parent and the child object, a PlmMetaRelation object defining the relation between them and a PlmResult to define the return values of the response. By default, the response returns values of all visible fields (plus some important ID fields like EDB_ID and C_ID) that are contained in the mask. Response information can be restricted by defining the list of fields that must be returned in the PlmResult of the request, but note that you can only return values for fields which are available in the mask. The settings of the mask determines the sorting of the data. Return values are provided in a standardized format as marshaled by the JAXB framework. Date values are in UTC, based on the assumption that the data returned by the EDM server is in server local time. By default, only the current language value is retrieved for multi-lingual fields. However, the web service client can request to retrieve all language values in the same call. If the object already exists in Agile e6, or an error occurs while creating the relation object, an error code is returned.
Request Type	CreateRelationRequestType ▫ messageId (String): ID to be returned in the response (optional). ▫ messageName (String): Name to be returned in the response (optional). ▫ relationObject (PlmObject): The relation Object with the PLM class reference and a list of relationship attributes (name/value pairs) used to create the new object. ▫ The metadata of the relationObject is currently ignored by this operation, so you can pass empty values for these elements. ▫ Only the list of attribute values is used to fill the new relation record in Agile e6. ▫ parent (PlmObjectRef) Object reference: This is either a PlmObject as returned by the BusinessService.getObjects operation, or a PlmObjectReference containing the query for the record. ▫ child (PlmObjectRef) as object reference: This is either a PlmObject as returned by the BusinessService.getObjects operation, or a PlmObjectReference containing the query for the record.

<table>
<tr><td></td><td>

relation (PlmMetaRelation) a relationship metadata: This is a PlmMetaRelation as returned by the MetadataService.getRelation operation. It defines the relation of the parent and the child object.
plmResult (PlmResult):This describes how the result of the operation is returned to the client.

<table>
<tr><td>attributeNames (Boolean)</td><td>List of field names to be returned?</td><td>▫ Optional
▫ Default
▫All visible fields and ID fields</td></tr>
<tr><td>includeBinaryValues (Boolean)</td><td>Include binary values</td><td>▫ Optional
▫ Default - False</td></tr>
<tr><td>includeAllLanguages (Boolean)</td><td>Include all languages?</td><td>▫ Optional
▫ Default - False</td></tr>
</table>
</td></tr>
<tr><td>Response Type</td><td>
CreateRelationResponseType

statusCode (ResponseStatusCode)

<table>
<tr><td>SUCCESS</td><td>The newly created PLM object including all attributes defined by the PlmResult of the request.</td></tr>
<tr><td>PARTIAL_SUCCESS</td><td>The newly created PLM object including some of the attributes defined by the PlmResult of the request.</td></tr>
<tr><td>FAILURE</td><td>A faulty description including error message and the type of error.</td></tr>
</table>

ticket (String): A PLM ticket.
▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the PLM session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
▫The ticket is only valid for the EDM server instance that generated it.
messageId (String): Copied from the request, or generated.
messageName (String): Copied from the request, or the operation name.
warnings: List of warnings (PlmWarningType) that occurred during the operation.
exceptions: List of exceptions (PlmExceptionType) that occurred during the operation.
relationObject (PlmObject): The created PLM relation object contains the attributes listed in the PlmResult of the request.
▫Additionally, the ID attributes of the parent and the child record are added.
▫The parent IDs are named PARENT.EDB_ID and PARENT.C_ID, the attributes for child IDs are named CHILD.EDB_ID and CHILD.C_ID.
▫This is necessary to prevent name clashes if the parent and child entity are the same (for instance in an Item BOM).

</td></tr>
</table>

updateRelation

Service	Updates an existing relationship entry in the Agile e6 system.

<table>
<tr><td>Usage</td><td>By default, the response returns values of all visible fields (plus some important ID fields like EDB_ID and C_ID) that are contained in the mask. The response information can be restricted by defining the list of fields that must be returned in the PlmResult of the request, but note, that you can only return values for fields which are available in the mask. The settings of the mask determines the sorting of the data. Return values are provided in a standardized format as marshaled by the JAXB framework. Date values are in UTC, based on the assumption that the data returned by the EDM server is in server local time.

By default, only the current language value is retrieved for multi-lingual fields. However, the Web Service client can request to retrieve all language values in the same call.

If the object already exists in Agile e6, or an error occurs while creating the relation object, an error code is returned.</td></tr>
<tr><td>Request Type</td><td>UpdateRelationRequestType
▫ messageId (String): ID to be returned in the response (optional).
▫ messageName (String): Name to be returned in the response (optional).
▫ plmRelationQuery (PlmRelationQuery):
▫PlmObject or PlmObjectReference: Defines the parent record of the relation. This is either a PlmObject as returned by the BusinessObject Web Service, or (to improve the performance) only a reference to it.
▫PlmMetaRelationRef: The metadata of the relation to be read (parent, child, type, view). This allows specifying the aggregate, refined, constraint, or type relation.
▫Selection: A list of PlmCondition objects representing the search criteria for object attributes. The rest of the PlmRelationQuery members are ignored.
▫ relationObject (PlmObject): The relation object with the PLM class reference, and a list of relationship attributes (name/value pairs) is used to create the new object.
▫ plmResult (PlmResult):This describes how the result of the operation is returned to the client.
<table>
<tr><td>attributeNames (Boolean)</td><td>List of field names to be returned</td><td>▫ Optional
▫ Default
▫All visible fields and ID fields</td></tr>
<tr><td>includeBinaryValues (Boolean)</td><td>Include binary values</td><td>▫ Optional
▫ Default - False</td></tr>
<tr><td>includeAllLanguages (Boolean)</td><td>Include all languages</td><td>▫ Optional
▫ Default - False</td></tr>
</table>
</td></tr>
<tr><td>Response Type</td><td>UpdateRelationResponseType
▫ statusCode (ResponseStatusCode)
<table>
<tr><td>SUCCESS</td><td>The newly created PLM object including all attributes defined by the PlmResult of the request.</td></tr>
<tr><td>PARTIAL_SUCCESS</td><td>The updated PLM object including only some of the attributes defined by the PlmResult of the request.</td></tr>
<tr><td>FAILURE</td><td>A faulty description including error message and the type of error.</td></tr>
</table>
▫ ticket (String): A PLM ticket.</td></tr>
</table>

	▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the PLM session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session. ▫The ticket is only valid for the EDM server instance that generated it. ▫ messageId (String): Copied from the request, or generated. ▫ messageName (String): Copied from the request, or the operation name. ▫ warnings: List of warnings (PlmWarningType) that occurred during the operation. ▫ exceptions: List of exceptions (PlmExceptionType) that occurred during the operation. ▫ relationObject (PlmObject): The updated PLM relation object contains the attributes listed in the PlmResult of the request. ▫Additionally, the ID attributes of the parent and the child record are added. ▫The parent IDs are named PARENT.EDB_ID and PARENT.C_ID, the attributes for child IDs are named CHILD.EDB_ID and CHILD.C_ID. ▫This is necessary to prevent name clashes if the parent and child entity are the same (for instance in an Item BOM).

getRelations

Service	To get the relationship records of a PLM object for a specified PLM relation.
Usage	The operation executes a query in the Agile e6 system and returns the list of matching relationship records of a PLM object for a specified PLM relation. The operation allows reading aggregate, refined, constraint, and type relation. The countOnly flag of the request is very important here. For more information on this, refer to countOnly Query Support in this document.
Request Type	CreateRelationRequestType ▫ messageId (String): ID to be returned in the response (optional). ▫ messageName (String): Name to be returned in the response (optional). ▫ plmRelationQuery (PlmRelationQuery): ▫PlmObject or PlmObjectReference: To get the parent record of the relation. This is either a PlmObject as returned by the BusinessObject Web Service or - to improve performance - only a reference to it. ▫PlmMetaRelationRef: The metadata of the relation to be read (parent, child, type, view). ▫This allows you specifying the aggregate, refined, constraint, or type relation. ▫selection: A list of PlmCondition objects representing the search criteria for object attributes.

ignoreRecordLimit (Boolean)	Ignore the record limit	▫ Optional ▫ Default - False
countOnly (Boolean)	Count only	▫ Optional ▫ Default - False
attributeNames	List of field names to	▫ Optional

	(Boolean)	be returned	▫ Default ▫All visible fields and ID fields
	includeBinaryValues (Boolean)	Include binary values	▫ Optional ▫ Default - False
	includeAllLanguages (Boolean)	Include all languages	▫ Optional ▫ Default - False
Response Type	GetRelationsResponseType ▫ Status code (SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE). ▫ A PLM ticket (String) ▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed or - in case of a backward flow - if there is no HTTP session. ▫The ticket is only valid for the EDM server instance that generated it. ▫ messageId (String): Copied from the request, or generated. ▫ messageName (String): Copied from the request, or the operation name. ▫ warning: List of warnings (PlmWarningType) that occurred during the operation. ▫ List of errors (PlmExceptionType) that occurred during the operation. ▫ Record limit hit? [true/false]: Indicates if the query result has hit the mask limit. ▫ List of objects (PlmObject) found including all the object attribute values listed in the PlmResult of the request. ▫Additionally, the ID attributes of the parent and the child record are added. The parent IDs are named PARENT.EDB_ID and PARENT.C_ID, the attributes for child IDs are named CHILD.EDB_ID and CHILD.C_ID. ▫This is necessary to prevent name clashes if the parent and child entity are the same (for instance in an Item BOM). ▫If a count request was made, the response will have one PlmObject with an Integer attribute named COUNT containing the number of objects matching the query, and an Integer attribute called RECORD_LIMIT containing the current record limit of the mask used for the count operation.		

deleteRelation

Service	Deletes a relationship entry in the Agile e6 system.
Usage	Note The object is deleted permanently without using the trash basket. A mask can be specified, which is used to process the delete operation. This allows to fire customer specific triggers on the EDM server during record deletion.
Request Type	DeleteRelationRequestType ▫ plmRelationQuery (PlmRelationQuery): ▫PlmObject or PlmObjectReference: Defines the parent record of the relation. This is either a PlmObject as returned by the BusinessObject Web Service, or – to improve performance – only a reference to it.

	▫PlmMetaRelationRef: The metadata of the relation to be read (parent, child, type, view). This allows specifying the aggregate, refined, constraint, or type relation. ▫Selection (List<PlmCondition>: A list of PlmCondition objects representing the search criteria for object attributes. This query has to match exactly one relation record, otherwise the operation fails. The rest of the PlmRelationQuery members are ignored. ▫ messageId (String): ID to be returned in the response (optional). ▫ messageName (String): Name to be returned in the response (optional).
Response Type	DeleteRelationResponseType ▫ status code (ResponseStatusCode): SUCCESS: Relation object is deleted. FAILURE: A faulty description including error message and the type of error. ▫ A PLM ticket (String) ▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the PLM session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session. ▫The ticket is only valid for the EDM server instance that generated it. ▫ messageId (String): Copied from the request, or generated. ▫ messageName (String): Copied from the request, or the operation name. ▫ warnings: List of warnings (PlmWarningType) that occurred during the operation. ▫ exceptions: List of exceptions (PlmExceptionType) that occurred during the operation. ▫ plmObject (PlmOjbect): PLM object with ID information like EDB_ID and C_ID, if available (optional). This is only returned if the object is found, but the operation fails.

setReservation

Service	The Web Service operates similar to the corresponding ECI functions: ▫ eci_res_ent ▫ eci_fre_ent
Usage	Reserves or un-reserves an object (usually a document object) in the Agile e6 system.
Request Type	SetReservationRequestType ▫ plmObject (PlmObject or PlmObjectRef): The PLM entity reference and the ID query of the object to be reserved. ▫ Boolean value true or false: ▫True = reserved ▫False = un-reserved ▫ messageId (String): ID to be returned in the response (optional). ▫ messageName (String): Name to be returned in the response (optional).
Response Type	SetReservationResponseType ▫ status code (ResponseStatusCode):

	SUCCESS: Object is reserved/un-reserved. FAILURE: A faulty description including error message and the type of error. ▫ A PLM ticket (String) ▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow)if there is no HTTP session. ▫The ticket is only valid for the EDM server instance that generated it. ▫ messageId (String): Copied from the request, or generated. ▫ messageName (String): Copied from the request, or the operation name. ▫ warnings: List of warnings (PlmWarningType) that occurred during the operation. ▫ exceptions: List of exceptions (PlmExceptionType) that occurred during the operation. ▫ plmObject (PlmOjbect): PLM object with ID information like EDB_ID and C_ID, if available (optional). This is only returned if the object is found, but the operation fails.

DocumentManagement Web Service

A document is a business object that specifies one or more files that are stored in the Agile e6 File Vault. You can load a document and add one or more files to its files table. You can also search for a document and its files, like searching for an item. The CAD fastload feature is available via the getCADAssembly Web Service.

DFM Support

The current implementation does not support Distributed File Management (DFM). The upLoadFile and downLoadFile web services, which are not a part of the core Services and are deployed as File Services, are used to access the local File Server on the remote DFM location. The file services are deployed on WebLogic and for remote DFM locations on TOMCAT.

The use cases describe how different Web Services interact to implement a DFM operation.

Note For information about the DFM Web Service Configuration, please refer to the Administration Guide for Agile e6.1.3.0.

StreamingFileServices

In the installed environment, these are deployed as the StreamingFileService web application, containing the DocumentFileService Web Service. It is deployed on WebLogic and for remote DFM locations on TOMCAT.

- downloadFile
- uploadFile

downloadFile

<table>
<tr><td>Service</td><td>Downloads a file directly from the local File Server. The request needs the response data provided by the getFiles or getCADAssembly Web Service call.</td></tr>
<tr><td>Usage</td><td>The Web Service is deployed on a separate installation on the local DFM location. The service can be deployed on a TOMCAT or WebLogic server.
▫ If the Web Service is deployed on WebLogic, we recommend updating metadata (file replication) directly. Do not use the updatedFileObject Web Service for this.
▫ The deployment only needs tracing.
▫ The Web Service should use the HTTPs protocol to secure the file transfer.</td></tr>
<tr><td>Request Type</td><td>downloadFileRequestType
▫ messageId (String, required)
▫ messageName (String, required)
▫ statusCode (ResponseStatusCode, required)
▫SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE.
▫ url (String, optional)
▫The URL to the createUpdateFileObject Web Service.
▫ fmsVaultTokens (List<PlmFMSVaultToken>, required)
▫ PlmFMSVaultToken
▫Containing all attributes to contact the File Server via FMS Java Daemon.
<table>
<tr><td>vaultName</td><td>String, required</td></tr>
<tr><td>vaultType</td><td>String, required</td></tr>
<tr><td>vaultKind</td><td>String, required</td></tr>
<tr><td>vaultNode</td><td>String, required</td></tr>
<tr><td>vaultNetRef</td><td>String, required</td></tr>
<tr><td>vaultPath</td><td>String, required</td></tr>
<tr><td>fmsJadeNode</td><td>String, required</td></tr>
<tr><td>fmsJadePort</td><td>String, required</td></tr>
<tr><td>tokenTimestamp</td><td>String, required</td></tr>
<tr><td>token</td><td>String, required</td></tr>
</table>
▫ fmsToken (PlmFmsToken, required)
▫ PlmFmsToken
▫Containing all attributes to contact the File Server via FMS Java Daemon.
<table>
<tr><td>id</td><td>String, required</td><td>Document to file relation record CID</td></tr>
<tr><td>parentId</td><td>String, required</td><td>Document to file relation record parent CID (Document CID)</td></tr>
<tr><td>childId</td><td>String, required</td><td>Document to file relation record child CID (File CID)</td></tr>
</table>
</td></tr>
</table>

	sourceFileToken	PlmFMSFileToken, optional	
	targetFileToken	PlmFMSFileToken, required	

Response Type	downloadFileResponseType ▫ messageId (String, required) ▫ messageName (String, required) ▫ statusCode(ResponseStatusCode, required) ▫SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE ▫ exceptions (List<PlmExceptionType>, optional) ▫List of exceptions that occurred during the operation. ▫ warnings (List<PlmWarningType>, optional) ▫List of warnings that occurred during the operation. ▫ fileData (a javax.activation.DataHandler, optional) ▫The file data as a binary MTOM stream with content type “application/octet-stream”. ▫Use the streaming API of your JAX-WS implementation to transfer the file. Note If streaming is not used, the file is transferred to XML data. This can lead to performance issues, creates big SOAP messages, and can cause an OutOfMemoryException on the server or client.

uploadFile

Service	Uploads a file directly to the local File Server. The request needs the response data provided by the getFMSVault Web Service call.
Usage	The Web Service is deployed on a separate installation on the local DFM location. The service can be deployed on a TOMCAT or WebLogic server. ▫ If the Web Service is deployed on WebLogic, we recommend updating metadata (file replication) directly. Do not use the createUpdatedFileObject Web Service for this. ▫ The deployment only needs tracing. ▫ The Web Service should use the HTTP/S protocol to secure the file transfer. ▫ The request contains a token which allows the Web Service client to upload a file into a specific vault. ▫ The URL, document and file reference, allows the Web Service to contact the createUpdateFileObject Web Service to update the metadata in Agile e6. ▫ For authentication, the PLM ticket can be used, or a system account.
Request Type	uploadFileRequestType ▫ messageId (String, optional) ▫ messageName (String, optional) ▫ ticket (String, required) The PLM ticket is used to call createUpdateFileObject () Web Service. ▫ DocumentFileQuery (PlmDocumentFileQuery, required)

Document	PlmObjectRef, required	The Agile e6 document object as PlmObjectRef
documentFileMaskName	String, required	The document file relation list to use for the query
Selection	List<PlmCondition>, required	List of PlmConditions which contain the field name/value pairs to search for document records
ignoreRecordLimit	Boolean, optional	A flag indicating if the record limit should be ignored (default = false)
attributeNames	List<String>, optional	Inherited from PlmResult
includeBinaryValues	Boolean, optional	Inherited from PlmResult
includeAllLanguages	Boolean, optional	Inherited from PlmResult

- documentFileObject (PlmObject, required)
  - The relation object with the Agile e6 class reference, and a list of relationship attributes (name/value pairs) used to create the new object.
  - The metadata of the relationObject is currently ignored by this operation, thus empty values for these elements can be passed.
  - Only the list of attribute values is used to fill the new relation record in Agile e6.
- url (String, required)
  The URL to the createUpdateFileObject Web Service.
- fmsVaultToken (PlmFMSVaultToken, required)
- PlmFMSVaultToken
  Contains all attributes to contact the File Server via FMS Java Daemon.

vaultName	String, required
vaultType	String, required
vaultKind	String, required
vaultNode	String, required
vaultNetRef	String, required
vaultPath	String, required
fmsJadeNode	String, required
fmsJadePort	String, required
tokenTimestamp	String, required
Token	String, required

- fileDate (javax.activation.DataHandler, required)

<table>
<tr><td>Response Type</td><td>uploadFileResponseType
▫ messageId (String, required)
▫ messageName (String, required)
▫ statusCode(ResponseStatusCode, required)
▫SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE
▫ exceptions (List<PlmExceptionType>, optional)
▫List of exceptions that occurred during the operation.
▫ warnings (List<PlmWarningType>, optional)
▫List of warnings that occurred during the operation.
▫ dfmResult (DFMResultData, optional)
<table>
<tr><td>Id</td><td>String, required</td><td>Document to Document relation Record CID</td></tr>
<tr><td>parented</td><td>String, required</td><td>Document to Document relation Record Parent CID (Document CID)</td></tr>
<tr><td>childId</td><td>String, required</td><td>Document to Document relation Record Child CID (Document CID)</td></tr>
<tr><td>Attributes</td><td>List<PlmAttributeChoice>, required</td><td></td></tr>
</table>
</td></tr>
</table>

Note A PlmFault will be thrown if an unexpected technical problem occurs, e.g. connection loss.

DocumentManagement CoreService

All operations in DocumentManagement Web Service support DFM.

Note To create a document, use the Business Object Web Services.

The following are the Web Service operation names of the single request operations for DocumentManagement Web Services describing the concurrent sections:

- getFiles
- getCADAssembly
- getCADAssemblyNextDataBlock
- getFMSVault
- createUpdateFileObject

getFiles

<table>
<tr><td>Service</td><td>Retrieves the list of files assigned to a specific document.

The operation will use the document file relation mask that is specified in the request to retrieve the data – search in a mask.</td></tr>
<tr><td>Usage</td><td>The response provides the necessary data to call the downloadFile Web Service and is divided into the following parts:

1. List of Vault definitions including:
▫ FMS Server definition
▫ FMS Java Daemon definition
▫ FMS vault tokens
2. List of FMS file token including:
▫ Source file with file tokens
▫ Target file with file tokens

Note In case of replication, the source file and target file will be filled. Otherwise, only the target file will be filled.

3. List of files

By default, the response will return values of all visible document file relation fields, plus some important ID fields (e.g. EDB_ID and C_ID) that are contained in the mask. Response information can be restricted by defining the list of fields that should be returned.

Note Only values for fields, which are available in the mask, can be returned. The settings of the mask will determine the sorting of the data.

The response will return file information and vault information. Return values will be provided in a standardized format as marshaled by the JAXB framework. Date values will be in UTC, based on the assumption that the data returned by the EDM server is in server local time.

By default, only the current language value will be retrieved for multi-lingual fields. However, the Web Service client can request to retrieve all language values in the same call.

The method fills a list of vault in the form of PlmFMSVaultToken in the response, which are referenced in one or many from the PlmFMSTokens.

The file tokens in the form PlmFMSTokens in response contains file details and its generated access tokens. Each file token represents one file from the file list and can be referenced by doc_fil ID.

The bulk operation is called getFilesBulk.</td></tr>
<tr><td>Request Type</td><td>GetFilesRequestType

▫ messageId (String, optional)
▫ messageName (String, optional)
▫ plmDocumentFileQuery (PlmDocumentFileQuery):

<table>
<tr><td>document (PlmObjectRef, required)</td><td>The Agile e6 document object as PlmObjectRef reference.</td></tr>
<tr><td>documentFileMaskName</td><td>The document file relation list to use for the query.</td></tr>
</table>
</td></tr>
</table>

	(String, required)	
	Selection (list<PlmCondition>, required)	List of PlmConditions which contain the field name/value pairs to search for document records.
	ignoreRecordLimit (Boolean, optional)	A flag indication if the record limit should be ignored (default is false).
	excludeVaultValues (Boolean, optional)	A flag indication if the vault values should be returned, too (default is true).
	attributeNames (List<String>, optional)	Inherited from PlmResult.
	includeBinaryValues (Boolean, optional)	Inherited from PlmResult.
	includeAllLanguages (Boolean, optional)	Inherited from PlmResult

Response Type

GetFileResponeType

- messageId (String, required)
- messageName (String, required)
- statusCode(ResponseStatusCode, required)
  - SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE
- exceptions (List<PlmExceptionType>, optional)
  - List of exceptions that occurred during the operation.
- warnings (List<PlmWarningType>, optional)
  - List of warnings that occurred during the operation.
- ticket (String, optional)
  - The PLM ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
  - The ticket is only valid for the EDM server instance that generated it.
- recordLimitHit (Boolean, required)
  - [true|false] Indicates whether the query result has hit the mask limit.
- url (String, optional)
  - The URL to the createUpdateFileObject Web Service.
- fmsVaultTokens (List<PlmFMSVaultToken>, optional)
  - List of PlmFMSVaultToken containing all attributes to contact the File Server via FMS Java Daemon.

PlmFMSVaultToken	vaultName (String, required) vaultType (String, required) vaultKind (String, required) vaultNode (String, required) vaultNetRef (String, required) vaultPath (String, required) fmsJadeNode (String, required)

		▫ fmsJadePort (String, required) ▫ url (String, required) ▫ tokenTimestamp (String, required) ▫ token (String, required)

▫ fmsTokens (List,PlmFMSToken>, optional)

▫List of PlmFMSToken containing all attributes to contact the File Server via FMS Java Daemon.

PlmFMSToken	▫ Id (String, required) Document to File relation Record CID ▫ parentId (String, required) Document to File relation Record Parent CID (Document CID) ▫ childId (String, required) Document to File relation Record Child CID (File CID) ▫ sourceFileToken (PlmFMSFileToken, optional) ▫ targetFileToken (PlmFMSFileToken, required)

▫ objects (List<DFMResultData>, required)

▫List of Document File relation records found.

DFMResultData	▫ Id (String, required) Document to File relation Record CID ▫ parentId (String, required) Document to File relation Record Parent CID (Document CID – T_DOC_DAT.C_ID) ▫ childId (String, required) Document to File relation Record Child CID (File CID) ▫ attributes (List<PlmAttributeChoice>, required)

▫ The created objects contain the attributes listed in the request.

▫ In case one or more of the requested attributes do not exist, PARTIAL_SUCCESS is returned.

▫ In case one or more of the requested attributes are not accessible, a WARNING is returned.

▫ If no attributes have been requested (attribute list is missing), the whole object including all visible attributes and the ID attributes (EDB_ID and C_ID) are returned. If none of the requested attributes exists, or the list is empty, only the ID attributes (EDB_ID and C_ID) are returned.

getCADAssembly

Service	Retrieves the list of document structure records and files needed to load a 3D assembly into a CAD system. The operation is similar to the fastload ECI function which allows collecting all files required to represent a CAD assembly. The CAD fastload feature is also available for this Web Service.
Usage	The Web Service uses the stored procedure to traverse the CAD structure and to create the document hierarchy list and the file list. This stored procedure specifics of the CAD structures like external references, drawings, family tables, etc. takes into

<table>
<tr><td></td><td>account.
Note See also ECI function eci_loa_cax for more details on the required parameters.
The Web Service is used to load the document structure according to:
▫ the version view which is currently active
▫ the snapshot view (baseline).
The Web Service returns the hierarchical document structure information and the list of files.
Note In case of the snapshot, the Web Service returns a flat list of document instead of the document hierarchy.
The entry point is a document.
Snapshot (Baselines) handling requires providing additional information when calling this Web Service. The Snapshot-ID identifies the snapshot to be loaded.
The response provides the necessary data to call the downloadFile Web Service and is divided into the following parts:
▫ List of vault definitions including:
▫FMS Server definition
▫FMS Java Daemon definition
▫ List of files
▫ List of document hierarchy
By default, the response returns values provided by the temporary table used by the stored procedure.
The response information can be restricted by defining the list of fields that must be returned, but note that you can only return values for fields which are available in the temporary table.
The stored procedure checks if all files are available of the DFM location. If one or more files are out-dated the procedure marks the records with the DFM status.
The response returns file information and vault information (see Response Type). Return values are provided in a standardized format as marshaled by the JAXB framework. Date values are in UTC, based on the assumption that the data returned by the EDM server is in server local time.
Note Since the operation handles large structures, there will be no bulk support.</td></tr>
<tr><td>Request Type</td><td>GetCADAssemblyRequestType
▫ messageId (String, optional)
▫ messageName (String, optional)
▫ plmDocumentCADQuery (PlmDocumentCADQuery):
<table>
<tr><td>document (PlmObjectRef, required)</td><td>An object reference to the document.</td></tr>
<tr><td>creationSystem</td><td>(String, required)</td></tr>
<tr><td>logicType</td><td>(String, required)</td></tr>
<tr><td>structureFlag</td><td>(boolean, required)</td></tr>
</table></td></tr>
</table>

	<table><tr><td>bomFunction</td><td>(String, required)</td></tr><tr><td>snapshotID (String, optional)</td><td>The ID of the snapshot.</td></tr><tr><td>externalReferences (boolean, optional)</td><td>Obey external references for snapshots? ▫ True ▫ False</td></tr></table> ▫ plmFileResult (PlmResult, optional): This describes how the result of the operation is returned to the client. <table><tr><td>attributeNamesForFileList(List<(String>)</td><td>List of field names of the file list to be returned</td><td>▫ Optional ▫ Default ▫All visible fields and ID fields</td></tr><tr><td>includeBinaryValues (Boolean)</td><td>Include binary values</td><td>▫ Optional ▫ Default – False</td></tr><tr><td>includeAllLanguages (Boolean)</td><td>Include all languages</td><td>▫ Optional ▫ Default – False</td></tr></table> ▫ plmDocStrResult (PlmResult, optional): This describes how the result of the operation is returned to the client. <table><tr><td>attributeNamesForDocStructureList (List<String>)</td><td>List of field names of the document structure list to be returned</td><td>▫ Optional ▫ Default ▫All visible fields and ID fields</td></tr><tr><td>includeBinaryValues (Boolean)</td><td>Include binary values</td><td>▫ Optional ▫ Default – False</td></tr><tr><td>includeAllLanguages (Boolean)</td><td>Include all languages</td><td>▫ Optional ▫ Default – False</td></tr></table>
Response Type	GetCAD AssemblyResponeType ▫ messageId (String, required) ▫ messageName (String, required) ▫ statusCode(ResponseStatusCode, required) ▫SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE ▫ exceptions (List<PlmExceptionType>, optional) ▫List of exceptions that occurred during the operation. ▫ warnings (List<PlmWarningType>, optional) ▫List of warnings that occurred during the operation. ▫ ticket (String, optional) ▫The PLM ticket can be used instead of the password in subsequent calls. It allows the client to reference the PLM session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session. ▫The ticket is only valid for the EDM server instance that generated it. ▫ fileWidgetID (String, optional)

- structureWidgetID (String, optional)
- url (String, optional)

  The URL to the createUpdateFileObject Web Service. This URL is provided if DFM is active.
- fmsVaultTokens (List<PlmFMSVaultToken>, optional)

  List of PLMFMSVaultToken containing all attributes to contact the File Server via FMS Java daemon.

PlmFMSVaultToken	vaultName (String, required) vaultType (String, required) vaultKind (String, required) vaultNode (String, required) vaultNetRef (String, required) vaultPath (String, required) fmsJadeNode (String, required) fmsJadePort (String, required) url (String, required) token Timestamp (String, required) token (String, required)

- fmsTokens (List,PlmFMSToken>, optional)

  List of PlmFMSToken containing all attributes to contact the file server via FMS Java daemon.

PlmFMSToken	id (String, required) Document to File relation Record CID parentId (String, required) Document to File relation Record Parent CID (Document CID) childId (String, required) Document to File relation Record Child CID (File CID) sourceFileToken (PlmFMSFileToken, optional) targetFileToken (PlmFMSFileToken, required)

- files (List<DFMResultData>, optional)

  List of Document to File relation records found.

DFMResultData	Id (String, required) Document to File relation Record CID parentId (String, required) Document to File relation Record Parent CID (Document CID) childId (String, required) Document to File relation Record Child CID (File CID) attributes (List<PlmAttributeChoice>, required)

- structure (List<DFMResultData>, optional)

  List of Document to Document relation records found.

DFMResultData	Id (String, required) Document to Document relation Record CID

			▫ parentId (String, required) Document to Document relation Record Parent CID (Document CID) ▫ childId (String, required) Document to Document relation Record Child CID (File CID) ▫ attributes (List<PlmAttributeChoice>, required)

▫The created objects contain the attributes listed in the request.

▫In case one or more of the requested attributes do not exist, PARTIAL_SUCCESS is returned.

▫In case one or more of the requested attributes are not accessible, a WARNING is returned.

▫If no attributes have been requested (attribute list is missing), the whole object including all visible attributes and the ID attributes (EDB_ID and C_ID) are returned. If none of the requested attributes exists, or the list is empty, only the ID attributes (EDB_ID and C_ID) are returned.

getCADAssemblyNextDataBlock

Service	The service retrieves the next set of data for the widget IDs and closes if the next set of data is empty. It does not open the new widgets.
Usage	This operation can only be used in conjunction with "getCADAssembly" webservice. This service takes the widgetIDs of file and structure for the mask EDB-CAX-FIL-TMP-SLI and EDB-CAX-STR-TMP-SLI, which is returned by the getCADAssembly service.
Request Type	GetCADAssemblyNextDataBlockRequestType ▫ messageId (String, optional) ▫ messageName (String, optional) ▫ fileWidgetID (String, optional) ▫ structureWidgetID (String, optional) ▫ plmFileresult (PlmResult, optional) This describes how the result of the operation is returned to the client.

attributeNamesForFileList(List<String>)	List of field names of the file list to be returned	▫ Optional ▫ Default ▫All visible fields and ID fields
includeBinaryValues (Boolean)	Include binary values	▫ Optional ▫ Default – False
includeAllLanguages (Boolean)	Include all languages	▫ Optional ▫ Default – False

▫ PlmDocStrResult (PlmResult, optional): This describes how the result of the operation is returned to the client.

attributeNamesForDocStructureList (List<String>)	List of field names of the	▫ Optional ▫ Default

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>document structure list to be returned</td><td>▫All visible fields and ID fields</td></tr>
<tr><td>includeBinaryValues (Boolean)</td><td>Include binary values</td><td>▫ Optional
▫ Default – False</td></tr>
<tr><td>includeAllLanguages (Boolean)</td><td>Include all languages</td><td>▫ Optional
▫ Default – False</td></tr>
</table>
</td></tr>
<tr><td>Response Type</td><td>
GetCAD AssemblyResponeType

messageId (String, required)
messageName (String, required)
statusCode(ResponseStatusCode, required)
▫SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE
exceptions (List<PlmExceptionType>, optional)
▫List of exceptions that occurred during the operation.
warnings (List<PlmWarningType>, optional)
▫List of warnings that occurred during the operation.
ticket (String, optional)
▫The PLM ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
▫The ticket is only valid for the EDM server instance that generated it.
fileWidgetID (String, optional)
structureWidgetID (String, optional)
url (String, optional)
▫The URL to the createUpdateFileObject Web Service. This URL is provided if DFM is active.
fmsVaultTokens (List<PlmFMSVaultToken>, optional)
▫List of PLMFMSVaultToken containing all attributes to contact the File Server via FMS Java daemon.
<table>
<tr><td>▫PlmFMSVaultToken</td><td>

vaultName (String, required)
vaultType (String, required)
vaultKind (String, required)
vaultNode (String, required)
vaultNetRef (String, required)
vaultPath (String, required)
fmsJadeNode (String, required)
fmsJadePort (String, required)
url (String, required)
token Timestamp (String, required)
token (String, required)

</td></tr>
</table>

fmsTokens (List,PlmFMSToken>, optional)
▫List of PlmFMSToken containing all attributes to contact the File Server via FMS Java

</td></tr>
</table>

daemon.

PlmFMSToken	▫ Id (String, required) Document to File relation Record CID ▫ parentId (String, required) Document to File relation Record Parent CID (Document CID) ▫ childId (String, required) Document to File relation Record Child CID (File CID) ▫ sourceFileToken (PlmFMSFileToken, optional) ▫ targetFileToken (PlmFMSFileToken, required)

▫ files (List<DFMResultData>, optional)

▫List of Document to File relation records found.

DFMResultData	▫ Id (String, required) Document to File relation Record CID ▫ parentId (String, required) Document to File relation Record Parent CID (Document CID) ▫ childId (String, required) Document to File relation Record Child CID (File CID) ▫ attributes (List<PlmAttributeChoice>, required)

▫ structure (List<DFMResultData>, optional)

▫List of Document To Document relation records found.

DFMResultData	▫ Id (String, required) Document to Document relation Record CID ▫ parentId (String, required) Document to Document relation Record Parent CID (Document CID) ▫ childId (String, required) Document to Document relation Record Child CID (File CID) ▫ attributes (List<PlmAttributeChoice>, required)

▫The created objects contain the attributes listed in the request.

▫In case one or more of the requested attributes do not exist, PARTIAL_SUCCESS is returned.

▫In case one or more of the requested attributes are not accessible, a WARNING is returned.

▫If no attributes have been requested (attribute list is missing), the whole object including all visible attributes and the ID attributes (EDB_ID and C_ID) are returned. If none of the requested attributes exists, or the list is empty, only the ID attributes (EDB_ID and C_ID) are returned.

getFMSVault

Service	This operation gets the upload information for different types of documents, Cax, file and creation systems, or for the given vault name that is used if DFM is not active.
Usage	The Web Service client provides the different needed combinations as request and gets the information for the upload. The provided metadata will be cached for later use (createUpdateFileObject)
Request Type	getFMSVaultRequestType

	▫ messageId (String, optional) ▫ messageName (String, optional) ▫ document (PlmObjectRef, optional) ▫An object reference to the document (the parent). ▫ fileFormat (String, optional) ▫ stepCreationSystem (String, optional) ▫The Step Creation System. ▫ vaultName (String, optional) ▫The vault name for which details will be retrieved.
Response Type	getFMSVaultResponseType ▫ messageId (String, required) ▫ messageName (String, required) ▫ statusCode(ResponseStatusCode, required) ▫SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE ▫ exceptions (List<PlmExceptionType>, optional) ▫List of exceptions that occurred during the operation. ▫ warnings (List<PlmWarningType>, optional) ▫List of warnings that occurred during the operation. ▫ fileData (javax.activation.DataHandler, optional) ▫The file data as a binary MTOM stream with content type "application/octet-stream". ▫Use the Streaming API of your JAX-WS implementation to transfer the file. If streaming is not used, the file is transferred to XML data, which takes considerably longer leading to huge SOAP messages, and might even result in an OutOfMemoryException on the server or the client.

createUpdateFileObject

Service	This operation allows uploading a file directly to the local File Server and distributed File Server. The request needs the response data provided by the getFMSVault web service call.
Usage	The Web Service is normally deployed on a separate installation on the local DFM location. Note The service can be deployed on a TOMCAT or WebLogic server. The deployment itself does not need any special configuration settings except for tracing. If the Web Service is deployed on WebLogic, we recommend updating the metadata directly and not to use the updateFileObject Web Service for that task. Note The Web Service is not protected by any Web Service security but must use the HTTP/S protocol to secure the file transfer. The request contains a token which allows the Web Service client to upload a file into that specific vault. The URL, document reference, and file reference allows the Web Service to contact the createupdateFileObject Web Service to update the metadata in Agile e6.

<table>
<tr><td></td><td>The PLM ticket can be used for authentication, or a system account is used for this task.</td></tr>
<tr><td>Request Type</td><td>uploadFileRequestType

messageId (String, optional)
messageName (String, optional)
ticket (String, required). The plm ticket is used to call createUpdateFileObject Web Service.
documentFileQuery (PlmDocumentFileQuery, required):

<table>
<tr><td>document (PlmObjectRef, required)</td><td>The PLM document object as PlmObjectReference.</td></tr>
<tr><td>documentFileMaskName (String, required)</td><td>The document file relation list to use for the query.</td></tr>
<tr><td>selection (List<PlmCondition>, required)</td><td>List of PlmConditions which contain the field name/value pairs to search for document records.</td></tr>
<tr><td>ignoreRecordLimit (boolean, optional, default=false)</td><td>A flag indicates if the record limit should be ignored. The default is false.</td></tr>
<tr><td>attributeNames (List<String>, optional)</td><td>inherited from PlmResult</td></tr>
<tr><td>includeBinaryValues (boolean, optional)</td><td>inherited from PlmResult</td></tr>
<tr><td>includeAllLanguages (boolean, optional)</td><td>inherited from PlmResult</td></tr>
</table>

documentFileobject (PlmObject, required) the relation Object with the PLM class reference and a list of relationship attributes (name/value pairs) used to create the new object.
url (String, required)
The URL to the createUpdateFileObject web service.
fmsToken (PlmFMSToken, required)
PlmFMSVaultToken, required, containing all attributes to contact the File Server via File Server Java daemon.
<table>
<tr><td>PlmFMSVaultToken</td><td>

vaultName (String, required)
vaultType (String, required)
vaultKind (String, required)
vaultNode (String, required)
vaultNetRef (String, required)
vaultPath (String, required)
fmsJadeNode (String, required)
fmsJadePort (String, required)
tokenTimestamp (String, required)
token (String, required)

</td></tr>
</table>
fileData (javax.activation.DataHandler, required)
</td></tr>
</table>

Response Type	uploadFileResponseType ▫ messageId (String, required) ▫ messageName (String, required) ▫ statusCode(ResponseStatusCode, required) ▫SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE ▫ exceptions (List<PlmExceptionType>, optional) ▫List of exceptions that occurred during the operation. ▫ warnings (List<PlmWarningType>, optional) ▫List of warnings that occurred during the operation. ▫ dfmResult (DFMResultData, optional) ▫Id (String, required) Document to Document relation Record CID ▫parentId (String, required) Document to Document relation Record Parent CID (Document CID) ▫childId (String, required) Document to Document relation Record Child CID (Document CID) ▫attributes (List<PlmAttributeChoice>, required)

Bulk Operations

The following are the bulk operation names of the single request operations for DocumentManagement Web Services describing the concurrent sections:

- getFiles(Bulk)
- createUpdateFileObject(Bulk)
- getFMSVault(Bulk)

getFiles

Service	To get a list of files assigned to a specific document.
Usage	The operation uses the document file relation mask that is specified in the request to retrieve the data (search in mask). The response provides the necessary data to call the downloadFile Web Service and is divided into the following parts: ▫ List of Vault definitions including: ▫FMS Server definition ▫FMS Java Daemon definition ▫FMS Vault Tokens ▫ List of FMS File Tokens including (in-case of replication, the source file and target file are filled if not only the target file): ▫Source File with file tokens ▫Target file with file tokens ▫ List of Files

<table>
<tr><td></td><td>By default the response returns values of all visible document file relation fields (plus some important ID fields like EDB_ID and C_ID) that are contained in the mask. Response information can be restricted by defining the list of fields that must be returned, but note that you can only return values for fields which are available in the mask.

The settings of the mask determines the sorting of the data.

Return values are provided in a standardized format as marshaled by the JAXB framework. Date values are in UTC, based on the assumption that the data returned by the EDM server is in server local time.

By default, only the current language value is retrieved for multi-lingual fields. However, the Web Service client can request to retrieve all language values in the same call.

The method fills a list of Vault in the form of PlmFMSVaultToken in the response, which is referenced in one or many from the PlmFMSTokens.

The file tokens in the form PlmFMSTokens in the response contains file details and its generated access tokens. Each file token represents one file from the file list and can be referenced by doc_fil ID.</td></tr>
<tr><td>Request Type</td><td>GetFilesRequestType
▫ messageId (String, optional)
▫ messageName (String, optional)
▫plmDocumentFileQuery (PlmDocumentFileQuery):
<table>
<tr><td>document (PlmObjectRef, required)</td><td>The PLM document object as PlmObjectReference.</td></tr>
<tr><td>documentFileMaskName (String, required)</td><td>The document file relation list to use for the query.</td></tr>
<tr><td>selection (List<PlmCondition>, required)</td><td>List of PlmConditions which contain the field name/value pairs to search for document records.</td></tr>
<tr><td>ignoreRecordLimit (boolean, optional, default=false)</td><td>A flag indicates if the record limit must be ignored.</td></tr>
<tr><td>excludeVaultValues (boolean, optional, default=true)</td><td>A flag indicates if the vault values must be returned, too.</td></tr>
<tr><td>attributeNames (List<String>, optional)</td><td>inherited from PlmResult</td></tr>
<tr><td>includeBinaryValues (boolean, optional)</td><td>inherited from PlmResult</td></tr>
<tr><td>includeAllLanguages (boolean, optional)</td><td>inherited from PlmResult</td></tr>
</table></td></tr>
<tr><td>Response Type</td><td>GetFilesResponeType
▫ messageId (String, required)
▫ messageName (String, required)</td></tr>
</table>

- statusCode(ResponseStatusCode, required)
  - SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE
- exceptions (List<PlmExceptionType>, optional)
  - List of exceptions that occurred during the operation.
- warnings (List<PlmWarningType>, optional)
  - List of warnings that occurred during the operation.
- ticket (String, optional)
  - The PLM ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
  - The ticket is only valid for the EDM server instance that generated it.
- recordLimitHit (boolean, required)
  - [true | false] Indicates if the query result has hit the mask limit.
- url (String, optional)
  - The URL to the createUpdateFileObject web service.
- fmsVaultTokens (List<PlmFMSVaultToken>, optional)
  - List of PlmFMSVaultToken containing all attributes to contact the File Server via FMS Java daemon.

PlmFMSVaultToken	vaultName (String, required) vaultType (String, required) vaultKind (String, required) vaultNode (String, required) vaultNetRef (String, required) vaultPath (String, required) vaultfmsJadeNode (String, required) fmsJadePort (String, required) url (String, required) tokenTimestamp (String, required) token (String, required)

- fmsTokens (List<PlmFMSToken>, optional)
  - List of PlmFMSToken containing all attributes to contact the File Server via FMS Java daemon.

PlmFMSToken	Id (String, required) Document to File relation Record CID parentId (String required) Document to File relation Record Parent CID (Document CID) childId (String, required) Document to File relation Record Child CID (File CID) sourceFileToken (PlmFMSFileToken), optional) targetFileToken (PlmFMSFileToken, required)

- objects (List<DFMResultData>, required)
  - List of Document File relation record found.

		▫DFMResultData	▫ Id (String, required) Document to File relation Record CID ▫ parentId (String required) Document to File relation Record Parent CID (Document CID-T_DOC_DAT.C_ID)) ▫ childId (String, required) Document to File relation Record Child CID (File CID) ▫ attributes (List<PlmAttributeChoice., required)
	▫The created objects contain the attributes listed in the request. ▫In case one or more of the requested attributes do not exist, PARTIAL_SUCCESS is returned. ▫In case one or more of the requested attributes are not accessible, a WARNING is returned. ▫If no attributes have been requested (attribute list is missing), the whole object including all visible attributes and the ID attributes (EDB_ID and C_ID) are returned. If none of the requested attributes exists or the list is empty, only the ID attributes (EDB_ID and C_ID) are returned.		

createUpdateFileObject

Note This operation is an internal operation. It creates/updates metadata for the upload file object.

getFMSVault

Service	This operation retrieves upload information for different types of documents, Cax, file and creation systems, or for the given Vault name used. (applies whether DFM is active or not)
Usage	The Web Service client provides the different needed combinations as request and gets the information for the upload of the different combinations.
Request Type	getFMSVaultRequestType ▫ messageId (String, optional) ▫ messageName (String, optional) ▫ document (PlmObjectRef, optional): An object reference to the document (the parent). ▫ fileFormat (String, optional) ▫ stepCreationSystem (String, optional): The Step Creation System ▫ vaultName (String, optional): The Vault name for which the details are to be retrieved.
Response Type	getFMSVaultResponseType ▫ messageId (String, required)

	▫ messageName (String, required) ▫ statusCode(ResponseStatusCode, required) ▫SUCCESS, PARTIAL_SUCCESS, WARNING, or FAILURE ▫ exceptions (List<PlmExceptionType>, optional) ▫List of exceptions that occurred during the operation. ▫ warnings (List<PlmWarningType>, optional) ▫List of warnings that occurred during the operation. ▫ ticket (String, optional) ▫The PLM ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session. ▫The ticket is only valid for the EDM server instance that generated it. ▫ url (String, required) ▫The URL to the createUpdateFileObject Web Service. ▫ fmsVaultToken (PlmFMSVaultToken,required) ▫PlmFMSVaultToken containing all attributes to contact the File Server via FMS Java daemon. PlmFMSVaultTokent: ▫ vaultName (String, required); ▫ vaultType (String, required); ▫ vaultKind (String, required); ▫ vaultNode (String, required); ▫ vaultNetRef (String, required); ▫ vaultPath (String, required); ▫ fmsJadeNode (String, required); ▫ fmsJadePort (String, required); ▫ url (String, required); ▫ token Timestamp (String, required); ▫ token (String, required) ▫

Metadata Web Service

The Metadata Service enables you to read the definition of Agile e6 classes like entities, entity types, and relations.

▫ All operations need one request object as input and return one response object.

- The request contains at least the name of Agile e6 class and a mask name.
- The response contains the metadata of the respective Agile e6 class.

- Bulk operations allow you to execute a whole list of requests with one Web Service call. The response of a bulk operation contains a list of responses matching the list of requests.

- The Agile e6 metadata is read from Agile e6 using the mask specified in the request.
- It is only possible to access Agile e6 attributes that are visible in this mask, with the exception of the ID fields EDB_ID and C_ID.
- Only masks listed in the so called Web Service Whitelist of Agile e6 can be accessed.

Bulk Operations

The following are the bulk operation names of the single request operations for Metadata Web Services describing the concurrent sections.

- getEntity(Bulk)
- getEntityType(Bulk)
- getEntityRelation(Bulk)
- getNumberCycles(Bulk)
- getNumbers(Bulk)

getEntity

Service	To get the metadata of an Agile e6 entity.
Usage	The data is based on the mask used to access the data. If the request does not pass a specific mask name, the default mask of the entity is used. The response contains the definition of all visible attributes in the mask. If an attribute has mode specific access, it is returned regardless of the mode specific access value.
Request Type	GetEntityRequestType ▫ name (String): The entity name. ▫ mask (String): The mask name to use to read this entity (optional). ▫If empty, the default list of the entity is used. ▫ messageId (String): ID to be returned in the response (optional). ▫ messageName (String): Name to be returned in the response (optional).
Response Type	GetEntityResponseType ▫ statusCode (ResponseStatusCode): SUCCESS or FAILURE. ▫ ticket (String): A PLM ticket. ▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session. ▫The ticket is only valid for the EDM server instance that generated it.

- messageId (String): copied from the request, or generated.
- messageName (String): copied from the request, or the operation name.
- warnings: List of warnings (PlmWarningType) that occurred during the operation.
- exceptions: List of exceptions (PlmExceptionType) that occurred during the operation.
- entity (PlmEntity): contains the entity definition (all information is returned with type String, unless noted otherwise):

▫ Entity ID ▫ Entity name ▫ Mask name ▫ Default form ▫ Default list ▫ Table name ▫ Join table name ▫ Entity title (in the current session language) ▫ Mask title (in the current session language) ▫ Record limit ▫ Number of significant fields		
▫ List of attribute definitions (PlmMetaAttribute) containing:	▫ Attribute name ▫ Attribute type ▫ Attribute title (in the current session language) ▫ Attribute description (in the current session language) ▫ Attribute format ▫ Attribute default value ▫ Attribute Checkstring	
	▫ Attribute access (visible/invisible/mandatory/read-only/mode specific):	▫ Query mode specific attribute access (if applicable) ▫ Update mode specific attribute

				access (if applicable) ▫ Insert mode specific attribute access (if applicable)
			▫ Data size of the attribute in this mask. ▫ Visible field width in this mask. (useful if a UI is generated from this metadata) ▫ Visible field height in this mask. (useful if a UI is generated from this metadata) ▫ Attribute visible? Tells you if the attribute is visible in this mask.	
		▫ List of relation metadata (PlmMetaRelation) defined for this entity:	▫ The name of the parent entity ▫ The name of the child entity ▫ The relation type ▫ The name of the view ▫ The name of the relation ▫ The internal ID of the relation ▫ The table name ▫ The title of the relation (in the current session language) ▫ The default list ▫ The default form (empty for most relations) ▫ The mask name used to read the metadata ▫Later, this data can be used to call MetadataService.getEntityRelation to get all attribute metadata for one specific relation	
		▫ List of type metadata (PlmMetaType) defined for this entity:	▫ The name of the master entity ▫ The type name ▫ The table name ▫ The title of the type ▫ The default list ▫ The default form ▫ The mask name used to read the metadata ▫Later, this data can be used to	

			call MetadataService.getEntityType to get all attribute metadata for one specific type.	

getEntity Type

Service	To get the metadata of an Agile e6 entity type.
Usage	The data is based on the mask used to access the data. If the request does not pass a specific mask name, the default mask of the entity type is used. The response contains the definition of all visible attributes in the mask. If an attribute has mode specific access, it is returned regardless of the mode specific access value. If the mask contains visible multi-lingual attributes, all generated invisible multi-lingual attribute siblings are returned.
Request Type	GetEntityTypeRequestType ▫ name (String): the name of the master entity for this type. ▫ type (String): the name of the type. ▫ mask (String): The mask name to be used to read this entity (optional). ▫If empty, the default list of the entity is used. ▫ messageId (String): ID to be returned in the response (optional). ▫ messageName (String): Name to be returned in the response (optional).
Response Type	GetEntityTypeResponseType ▫ statusCode (ResponseStatusCode): SUCCESS or FAILURE. ▫ ticket (String): A PLM ticket. ▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session. ▫The ticket is only valid for the EDM server instance that generated it. ▫ messageId (String): copied from the request, or generated. ▫ messageName (String): copied from the request, or the operation name. ▫ warnings: List of warnings (PlmWarningType) that occurred during the operation. ▫ exceptions: List of exceptions (PlmExceptionType) that occurred during the operation. ▫ An entity type definition of type PlmEntityType (all information is returned with type String, unless noted otherwise):

▫ Entity name ▫ Type name ▫ Mask name ▫ Table name ▫ Join table name ▫ Entity type title (in the current session		

		language). ▫ Mask title (in the current session language). ▫ Mask limit. ▫ Number of significant fields ▫ Master: the reference to the master entity as a PlmClassRef		
		▫ List of attribute definitions containing:	▫ Attribute type ▫ Attribute title (in the current session language) ▫ Attribute description (in the current session language) ▫ Attribute format ▫ Attribute default value	
			▫ Attribute access (visible/invisible/mandatory read-only/mode specific):	▫ Query mode specific attribute access (if applicable) ▫ Update mode specific attribute access (if applicable) ▫ Insert mode specific attribute access (if applicable)
			▫ Data size of the attribute in this mask ▫ Visible field width in this mask. (useful if a UI is generated from this metadata) ▫ Visible field height in this mask. (useful if a UI is generated from this metadata) ▫ Attribute visible? Tells you if the attribute is visible in this mask.	
		▫ List of relation metadata (PlmMetaRelation) defined for this	▫ The name of the parent entity ▫ The name of the child entity ▫ The relation type	

		entity:	▫ The name of the view ▫ The name of the relation ▫ The internal ID of the relation ▫ The table name ▫ The title of the relation (in the current session language) ▫ The default list ▫ The default form ▫This data can be used later to call MetadataService.getEntityRelation to get all attribute metadata for one specific relation.	
	▫			

getEntityRelation

Service	To get the metadata of an Agile e6 constraint, refine, or aggregate relation.
Usage	The data is based on the mask used to access the data. If the request does not pass a specific mask name, the default mask of the entity is used. The response contains the definition of all visible attributes in the mask. If an attribute has mode specific access, it is returned regardless of the mode specific access value. In addition, the response contains the default ID fields of the relation (EDB_ID and C_ID, if available), even if these are invisible. If the mask contains visible multi-lingual attributes, all generated invisible multi-lingual attribute siblings are returned.
Request Type	GetEntityRelationRequestType ▫ metaRelation (PlmMetaRelation): Describes the relation to be read, and must contain at least the following information: parent: The name of the parent entity (e.g. "EDB-ARTICLE") child: The name of the child entity (e.g. "EDB-DOCUMENT") type: The relation type (e.g. PlmRelationTypeEnum.REFINE) view: The name of the view. (e.g. "STR") ▫The meta relation object can either be taken from the response of a call to getEntity/getEntity type, or it can be created using hard coded default values. The four attributes listed above are needed to identify the relation data, the rest of the attributes in PlmMetaRelation are not relevant. ▫ messageId (String): ID to be returned in the response (optional) ▫ messageName (String): Name to be returned in the response (optional)
Response Type	GetEntityRelationResponseType ▫ statusCode (ResponseStatusCode): SUCCESS or FAILURE.

- ticket (String): A PLM ticket.
  - The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
- The ticket is only valid for the EDM server instance that generated it.
- messageId (String): copied from the request, or generated.
- messageName (String): copied from the request, or the operation name.
- warnings: List of warnings (PlmWarningType) that occurred during the operation.
- exceptions: List of exceptions (PlmExceptionType) that occurred during the operation.
- plmRelation (PlmRelation): The relation definition containing:

• Relation name. • PlmMetaRelation: Detailed meta information of the relation. • Mask name. • Table name. • Entity type title (in the current session language). • Mask title (in the current session language). • Mask limit. • Number of significant fields.		
• List of attribute definitions containing:	• Attribute type • Attribute title (in the current session language) • Attribute description (in the current session language) • Attribute format • Attribute default value	
	• Attribute access (visible/invisible/mandatory/read-only/mode specific):	• Query mode specific attribute access (if applicable) • Update mode specific attribute access (if applicable) • Insert mode specific attribute access (if

				applicable)
			Data size of the attribute in this mask. Visible field width in this mask. (useful if a UI is generated from this metadata) Visible field height in this mask. (useful if a UI is generated from this metadata) Attribute visible? Tells you if the attribute is visible in this mask.	

getNumberCycles

Service	This operation is used to retrieve several number cycles which are used in a mask of an entity or a relation.
Usage	The operation scans the field default definition in the mask and returns the number cycle names. The operation considers field defaults as well as mask specific field defaults. The Web Service returns number cycles from visible fields only, because invisible fields cannot be populated with a number cycle number returned by getNumber Web Service.
Request Type	GetNumberCyclesRequestType ▫ plmClass (PlmClassRef) ▫The class definition containing the entity and a mask name (optional). Note Leave it empty to read a relation mask, metaRelation is used in this case. ▫ metaRelation (PlmMetaRelationRef) ▫The relation definition including parent and child entity, relation type, view, and mask name (optional). Note Leave it empty to read an entity mask, plmClass is used in this case. ▫ messageId (String) ▫ID to be returned in the response (optional). ▫ messageName (String) ▫Name to be returned in the response (optional).
Response Type	GetNumberCyclesResponseType ▫ statusCode (ResponseStatusCode): ▫SUCCESS or FAILURE. ▫ messageId (String): Copied from the request, or generated. ▫ messageName (String): Copied from the request, or the operation name. ▫ numberCycles (List<PlmFieldNumberCycle>) ▫The list of all number cycles used in this mask, consisting of field name and number cycle name.

getNumbers

<table>
<tr><td>Service</td><td>This operation is used to get one or more numbers generated by a number generator. Number generators are used to create instance object IDs. The format of the number can be defined in a flexible way by defining the numbering template.</td></tr>
<tr><td>Usage</td><td>Note It is not possible to hand back unused numbers, once the numbers have been retrieved. Thus, it is important to use this Web Service carefully; otherwise the number cycle might run out of range.

In order to control which number cycles are available for the Web Service call "getNumbers", field "Max # of Numbers per Web Service" is available. If the value is "0" or NULL, the number cycle is not accessible for the Web service, else the value defines how many numbers (<=) can be generated by one getNumbers Web Service call.</td></tr>
<tr><td>Request Type</td><td>GetNumbersRequestType
▫ messageId (String)
▫▫ID to be returned in the response (optional).
▫ messageName (String)
▫▫Name to be returned in the response (optional).
▫ numberCycleName (String)
▫▫The name of the number cycle to be used.
▫ Range (Integer)
▫▫The amount of numbers to be requested from the number server.</td></tr>
<tr><td>Response Type</td><td>GetNumberCyclesResponseType
▫ statusCode (ResponseStatusCode): SUCCESS, PARTIAL_SUCCESS, WARNING or FAILURE.
<table>
<tr><td>SUCCESS</td><td>Object deleted successfully.</td></tr>
<tr><td>PARTIAL_SUCCESS</td><td>Is returned when all numbers are generated, but threshold warning is issued (number variant will be exhausted soon).</td></tr>
<tr><td>WARNING</td><td>Is returned with a list of only some of the requested numbers that are generated. Reasons are that number variant is exhausted after generating a few numbers or that more numbers than allowed were requested for one Web Service call.</td></tr>
<tr><td>FAILURE</td><td>Is returned when the number variant is not found, has the wrong format (missing #), when the Web Service is not enabled, exhausted, or the definition for the number variant is incorrect.</td></tr>
</table>
▫ ticket (String): A PLM ticket.
▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
▫ The ticket is only valid for the EDM server instance that generated it.
▫ messageId (String): Copied from the request, or generated.</td></tr>
</table>

	▫ messageName (String): Copied from the request, or the operation name. ▫ warnings (List< PlmWarningType >): List of warnings that occurred during the operation. ▫ exceptions (List< PlmExceptionType >): List of exceptions that occurred during the operation. ▫ numberCycleName (String): Name of the number cycle, copied from the request. ▫ numbers (List<String>): List of numbers returned from this number cycle.

Configuration Web Service

The Configuration Web Service enables you to retrieve PLM objects from Agile e6. It retrieves configuration data of a PLM object, such as Default, User Context, etc., specified by its name.

The requests include the value(s) for specifying the requested object. Responses include the requested objects.

Bulk operations require a list of requests to execute. These return with a list of responses.

The following are the Web Service operation names of the single request operations for Configuration Web Services describing the concurrent sections.

- getUserContext
- setUserContext

getUserContext

<table>
<tr><td>Service</td><td>To get all the information of the current user context of an Agile e6 session.</td></tr>
<tr><td>Usage</td><td>The request object carries current Agile e6 user's details and only an optional message ID and name. The response object delivers the user attributes, all group assignments attributes, current view (Released, Global etc.), current context (DSG, ENG), current project assignment (Project ID) and current Org assignment (Org ID).</td></tr>
<tr><td>Request Type</td><td>GetUserContextRequestType
▫ messageId (String): ID to be returned in the response (optional).
▫ messageName (String): Name to be returned in the response (optional).</td></tr>
<tr><td>Response Type</td><td>GetUserContextResponseType
▫ statusCode (ResponseStatusCode): SUCCESS or FAILURE.
<table>
<tr><td>SUCCESS</td><td>All objects from user context are returned, if they are available.
There could be some warnings in the message block, if some objects (e.g. currentJob, currentRole) are not available. It could make sense for setUserContext, but it is not necessary to check for getUserContext.</td></tr>
</table></td></tr>
</table>

FAILURE	There was a connection problem.

- ticket (String) : A PLM ticket.
  - The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
  - The ticket is only valid for the PLM server instance that generated it.
- messageId (String): copied from the request, or generated.
- messageName (String): copied from the request, or the operation name.
- warnings: List of warnings (PlmWarningType) that occurred during the operation.
- exceptions: List of exceptions (PlmExceptionType) that occurred during the operation.
- plmUserContext (PlmUserContext):

plmUserInfo (PlmUserContextUserInfo):	▫ userName ▫ userID ▫ group ▫ groupID ▫ userLanguage ▫ userLocale
plmViewInfo (PlmUserContextView):	▫ preliminaryFlag (Boolean.TRUE, Boolean.False, null) ▫ referenceDate (yyyy-MM-dd HH:mm:ss, @NOW, empty_string) ▫ versionViewTitle (Current, Production, Global, Development_extended) ▫ versionView (PlmVersionViewEnum)
plmChangeManagementInfo (PlmUserContextChg):	▫ currentChgFlag (Boolean.TRUE, Boolean.False,null) ▫ currentWorkOrder (EDB_ID, empty_string) ▫ currentWorkSet (EDB_ID, empty_string)
plmMoaMpaInfo (PlmUserContextMoaMpa):	▫ moaMpaConfFlag(PlmUserContextMOAEnum, null) ▫ currentJob (empty_string) ▫ currentRole (empty_string) ▫ currentProjectOrOrganisation (empty_string) ▫ currentProjectOrOrganisationCid (empty_string)
plmDFMInfo (PlmUserContextDFM):	▫ dfmConfigFlag (Boolean.TRUE, Boolean.False,null) ▫ site (Reserved for future use)

setUserContext

<table>
<tr><td>Service</td><td>To modify one or more settings of the current user context in the current Agile e6 session.</td></tr>
<tr><td>Usage</td><td>This operation is used to set current job in MOA/MPA environment. The request object carries the user context details, such as the new view, new assignment, new organization assignment, and new project assignment. The response object delivers the user attributes, group assignments, new current view, new current context, new current project assignment, new current org assignment.</td></tr>
<tr><td>Request Type</td><td>SetUserContextRequestType
▫ messageId (String): ID to be returned in the response
▫ messageName (String): Name to be returned in the response
▫ plmUserContext (PlmUserContext):
<table>
<tr><td>plmUserInfo (PlmUserContextUserInfo):</td><td>▫ userName (not changeable by current user)
▫ userID (not changeable by current user)
▫ group (Reserved for future use: currently there is no API to implement it.)
▫ groupID (Reserved for future use : currently there is no API to implement it.)
▫ userLanguage (e.g. ENG, GER, FRA)
▫ userLocale (always changed with user language)</td></tr>
<tr><td>plmViewInfo (PlmUserContextView):</td><td>▫ preliminaryFlag (Boolean.TRUE, Boolean.FALSE)
▫ referenceDate (in the form of yyyy-MM-dd HH:mm:ss)
▫ versionViewTitle (ignored)
▫ versionView (PlmVersionViewEnum)</td></tr>
<tr><td>plmChangeManagementInfo (PlmUserContextChg):</td><td>▫ currentChgFlag (not changeable by current user)
▫ currentWorkOrder (EDB-ID of work order, no check of currentChgFlag)
▫ currentWorkSet (EDB-ID of work set, no check of currentChgFlag)</td></tr>
<tr><td>plmMoaMpaInfo (PlmUserContextMoaMpa):</td><td>▫ moaMpaConfFlag (not changeable by current user)
▫ currentJob EDB_ID of the job, access is checked by EDM server
▫ currentProjectOrOrganisation - ignored in the request, filled by the response
▫ currentProjectOrOrganisationCid - ignored in the request, filled by the response
▫ currentRole - ignored in the request, filled by the response</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>plmDFMInfo (PlmUserContextDFM):</td><td>▫ dFMConfigFlag (not changeable by current user)
▫ site (reserved for future use)</td></tr>
<tr><td>Response Type</td><td colspan="3">SetUserContextRequestType
▫ statusCode (ResponseStatusCode): SUCCESS. PARTIAL_SUCCESS,or FAILURE.
<table>
<tr><td>SUCCESS</td><td>All objects could be set successfully; some objects could not be set with warnings.</td></tr>
<tr><td>PARTIAL_SUCCESS</td><td>Some objects could be set successfully, while some could not be set with warnings.</td></tr>
<tr><td>FAILURE</td><td>There was a connection problem.</td></tr>
</table>
▫ ticket (String): A PLM ticket.
▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
▫The ticket is only valid for the EDM server instance that generated it.
▫ messageId (String): copied from the request, or generated.
▫ messageName (String): copied from the request, or the operation name.
▫ warnings: List of warnings (PlmWarningType) that occurred during the operation.
▫ exceptions: List of exceptions (PlmExceptionType) that occurred during the operation.
▫ plmUserContext (PlmUserContext):
<table>
<tr><td>plmUserInfo (PlmUserContextUserInfo):</td><td>▫ userName
▫ userID
▫ group
▫ groupID
▫ userLanguage
▫ userLocale</td></tr>
<tr><td>plmViewInfo (PlmUserContextView):</td><td>▫ preliminaryFlag (Boolean.TRUE, Boolean.False, null)
▫ referenceDate (yyyy-MM-dd HH:mm:ss, @NOW, empty_string)
▫ versionViewTitle (Current, Production, Global, Development_extended)
▫ versionView (PlmVersionViewEnum)</td></tr>
<tr><td>plmChangeManagementInfo (PlmUserContextChg):</td><td>▫ currentChgFlag (Boolean.TRUE, Boolean.False,null)
▫ currentWorkOrder (EDB_ID, empty_string)
▫ currentWorkSet (EDB_ID, empty_string)</td></tr>
<tr><td>plmMoaMpaInfo (PlmUserContextMoaMpa):</td><td>▫ moaMpaConfFlag(PlmUserContextMOAEnum, null)
▫ currentJob (empty_string)
▫ currentRole (empty_string)</td></tr>
</table>
</td></tr>
</table>

			▫ currentProjectOrOrganisation (empty_string) ▫ currentProjectOrOrganisationCid (empty_string)
		plmDFMInfo (PlmUserContextDFM):	▫ dfmConfigFlag (Boolean.TRUE, Boolean.False,null) ▫ site (Reserved for future use)

Bulk Operations

The following are the bulk operation names of the single request operations for Configuration Web Services describing the concurrent sections.

- getDefault(Bulk)
- createDefault(Bulk)

getDefault

Service	To get the contents of a PLM Default value.
Usage	It is possible to read all available defaults from system configuration.
Request Type	GetDefaultRequestType ▫ defaultName (String): ▫Name of the DataView default. ▫ messageId (String): ▫ID to be returned in the response (optional). ▫ messageName (String): ▫Name to be returned in the response (optional).
Response Type	GetDefaultResponseType ▫ Status Code PARTIAL_SUCCESS: Typed default value is NULL FAILURE: Default "Non-existent" - PLM Object is NULL~ ▫ PLM Default as complex data type (String, Integer, Float, Boolean)

createDefault

<table>
<tr><td>Service</td><td>This operation creates/updates the content of a PLM Default value.</td></tr>
<tr><td>Usage</td><td>Similar to the corresponding ECI function “eci_add_dfv”, the Web Service does not persist the default value. Thus, a default value set via createDefault will exist only at runtime and for the current Web Service session.
Note The Web Service allows creating defaults for the current user only.
It is possible to create/update any defaults in system configuration.</td></tr>
<tr><td>Request Type</td><td>CreateDefaultRequestType
▫ messageId (String):
▫ID to be returned in the response (optional).
▫ messageName (String):
▫Name to be returned in the response (optional).
▫ PLM Default as complex data type (String, Integer, Float):
▫Compared to getDefault and createDefault, it only supports String, Integer, and Float as input values.
▫The format for other types is not defined by DataView, and the value is always stored as String.</td></tr>
<tr><td>Response Type</td><td>GetDefaultResponseType
▫ Status Code
<table>
<tr><td>SUCCESS</td><td>The default is created.</td></tr>
<tr><td>PARTIAL_SUCCESS</td><td>Typed default value is NULL</td></tr>
<tr><td>FAILURE</td><td>Default "Non-existent" - PLM Object is NULL~</td></tr>
</table>
▫ticket (String,): A PLM ticket. PLM Default as complex data type (String, Integer, Float, Boolean)
▫The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
▫ The ticket is only valid for the EDM server instance that generated it.
▫ messageId (String): Copied from the request, or generated.
▫ messageName (String): Copied from the request, or the operation name.
▫ warnings (List< PlmWarningType >): List of warnings that occurred during the operation.
▫ exceptions (List< PlmExceptionType >): List of exceptions that occurred during the operation.</td></tr>
</table>

EngineeringCollaborationService Web Service

Agile e6 ECService Web Services are a set of Business Services that supplement EDM's Core Web Services.

Example:

Below are examples given of how CAD scenarios are used to describe the generic character of the **createUpdateStructure** Web Service operation.

This operation allows you to process relations like Create/Read/Update/Delete between the following objects:

- Document - Document
  - Document structure (STR):

    The document structure is used in the context of CAD integration to represent the assembly structure (top-level-assembly -> sub-assembly -> single part).

  - Snapshot structure (SNP):

    The Snapshot structure allows saving specific document structure configurations, independent from the version view setting. Furthermore, it is used to store design variants in Agile e6. The CAD system has full control of the Snapshot structure. The CAD system is the only authoring system creating Snapshot structures.

  - Intern-Extern Relationship (IER):

    CAD assemblies sometimes make use of external references. For instance, if an engineer is designing a tool to manufacture the work piece, he is using the work piece geometry to derive the tool geometry. In order to assign the work piece geometry to the tool assembly structure, the Intern-Extern Relationship is used.
- Document - Item (STR Structure)
  - Optional use case: Is used if items are derived from CAD structure.
- Item-Item (STR Bill of Material)
  - Optional use case: Is used if BOM is derived from CAD structure.

1. Filtering

   Document-Document STR relations, which are created/owned by the CAD integration, can be identified by the field T_DOC_STR.UG2_IDENT. “UG2_IDENT” contains a name which indicates the source CAD integration. If the relation record is owned by the CAD integration, and the access rights of the relation record allow deletion, the CAD integration is allowed to delete the record. In other words, if the relation record was created manually by a user, the CAD integration shall never delete such a record. The UG2_IDENT mechanism is also used for BOM and Item-Document relation.

   In general, the Web Service shall allow defining a set of field value pairs which are used to identify objects that are managed by the CAD integration. It is possible to define separate sets of field value pairs for the different relation (BOM, Item-Document, Document-Document).

   Note You can use wildcards to specify filter criteria.

In order to find out if data records need to be created, updated, or deleted, an additional comparison pattern can be defined. For the CAD document structure the comparison pattern typically consists of the following attributes:

- Parent-ID(C_ID_1)
- Child-ID (C_ID_2)
- Transf.-Matrix (ECC_XMAT)
- Old Filename (CAX_COM)
- Structure Ref (CAX_REF)
- CAX-internal value (CAX_04)

This list of attributes is configurable. It is possible to define separate list of fields for the different relation types (BOM, Item-Document, Document-Document, Document-File).

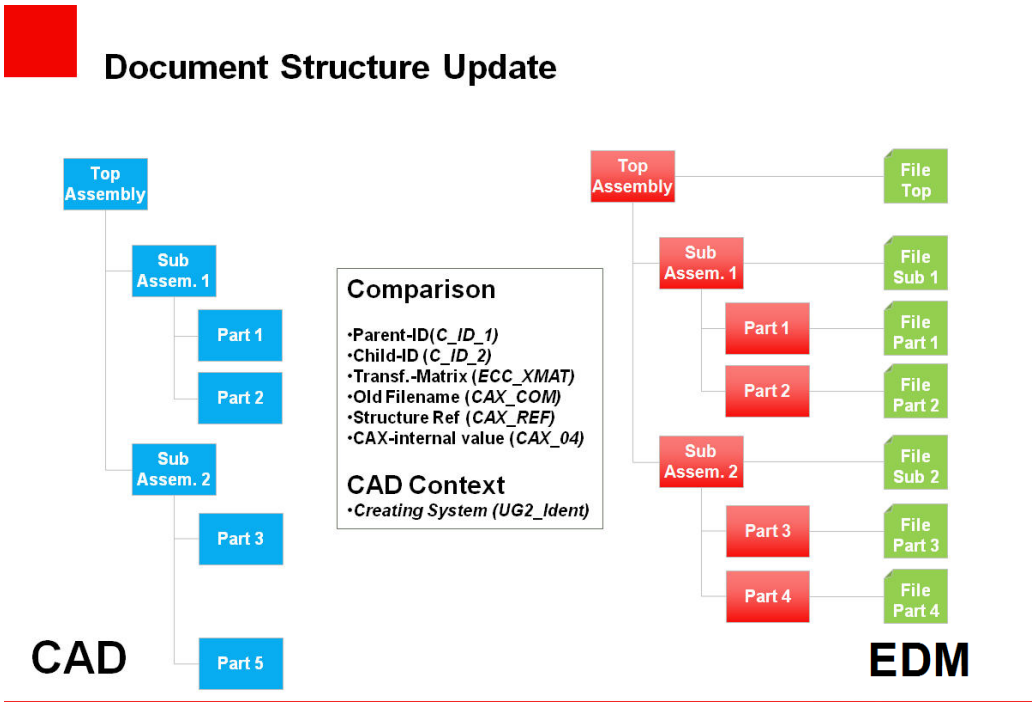
If the record, defined by the values of the comparison pattern, already exists in Agile e6, the record is updated. Otherwise, it is created. Remaining records, which are not listed in the Web Service, but are owned by the CAD integration (see UG2_Ident above), are deleted in Agile e6. Records which are not owned by the CAD integration remain untouched. If updating or deletion fails due to insufficient access rights, the Web Service reports an error.

2. Scenarios

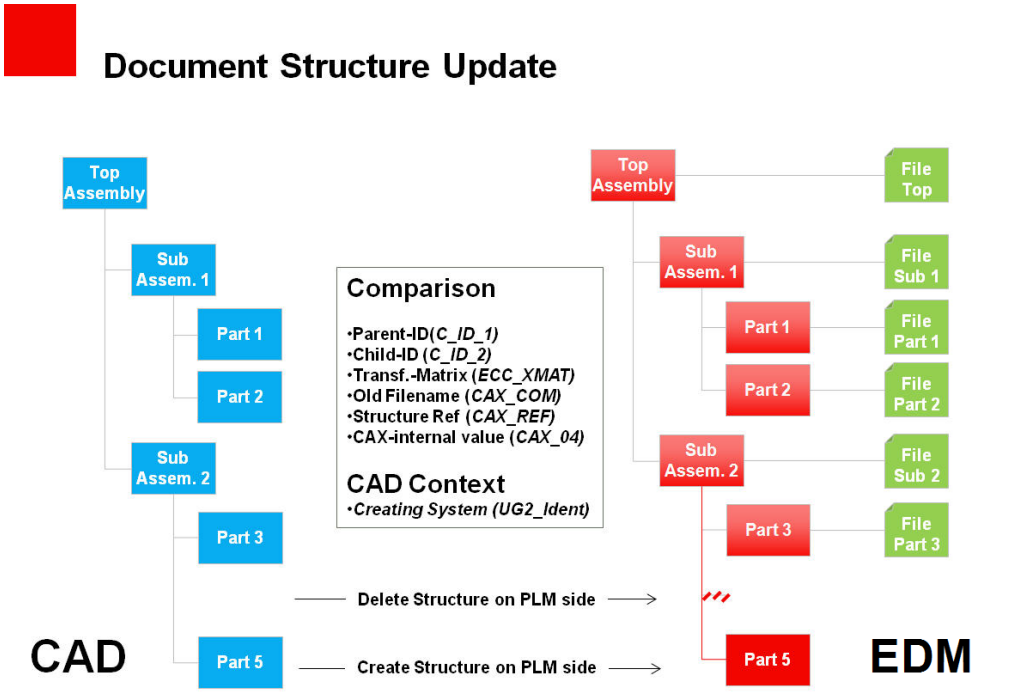
The scenario below illustrates the basics of structure update.

Note It focuses on the document structure only.

- Sample Data: CAD Document Structure (Pre-Update)



- Sample Data: CAD Document Structure (Post-Update)



3. Create

   The create use case is executed if a relation record cannot be found in Agile e6 while comparing the comparison pattern attributes.

4. Read

Reading of data is part of the create/update use case, as create/update returns the attribute values of created/updated records.

5. Update

   In the CAD document structure context, the update is always executed with a delete and a create operation. Thus, all data is first deleted and then newly created.

   However, in the BOM, and maybe in the Item-Document relation, too, the PosNo information is significant and the approach of deleting and recreating does not work. Hence, the update use case is required.

6. Delete

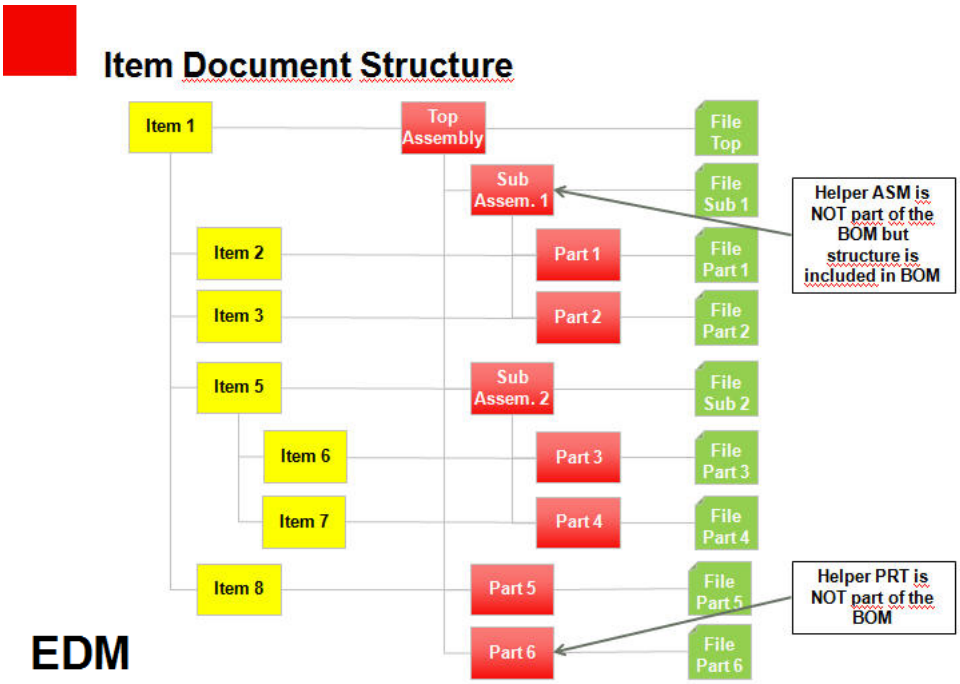
   The delete operation is implicitly used as part of the update of a CAD document structure. Additionally, the delete use case applies if a structure contains spare records in Agile e6. However, this behavior is optional for an object type level.

   For instance, there are use cases where the CAD integration loads only part of the CAD assembly structure and hence saves only part of the CAD structure back to Agile e6. By setting the delete option for every object type appropriately, the deletion of alleged spare records is omitted.

Note In case a complete structure needs to be deleted, it is initiated by sending an empty structure.

- Document-Item-BOM Scenario

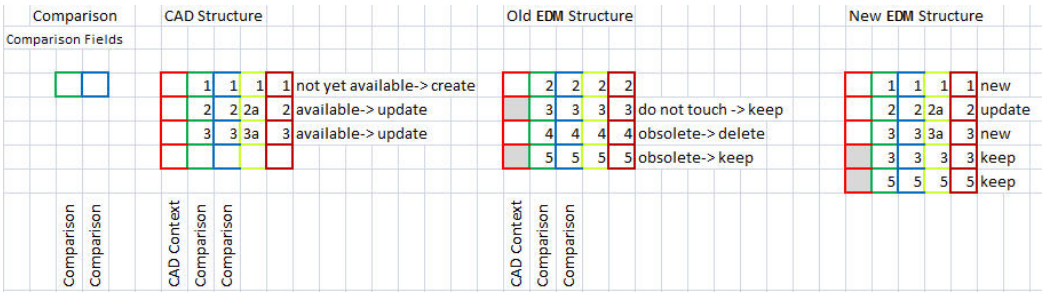
Sometimes, not all document structure nodes make it into an item node and the Web Service ignores this. The input parameter structure for the Web Service clearly defines which item structure needs to be created.



7. Generic Systematic

- Comparison fields (see green and blue columns in graphic below)
- Defines the list of fields which are used to identify a specific record in the old Agile e6 data structure (see also graphic above)
- CAD Structure
- Shows the new structure as maintained in the CAD system.
- Old Agile e6 structure
- Represents the structure as it is stored in Agile e6 before the update
- New Agile e6 structure
- Represents the structure as it will be stored after the update
- "CAD Context" (red column):

  Defines the field value pair(s), which is/are used to identify all records that are managed by the CAD integration. Cells with grey fill color are not managed by the CAD integration.



Note Record 4|4|4|4 will only be deleted if the delete flag for the specific relation type is set to true.

8. Access Control

   The access control can be applied on different levels.

   - For "update"/"delete" use case the Agile e6 access rights define if the write and/or delete access is available for the object and for the current user.
   - Especially in the "create" use case, the parent access might be considered. If current user has write or delete access to the parent record, he is allowed to create/update/delete the relation record.
   - Specific for documents: The modification of references to files and sub-documents might require the parent document to be "reserved" by the current user.

   The first two cases will be handled by the widget, which is used in the Web Service to create/update/delete the relation objects. Specific triggers at the form/widget assure you that the checks will be performed. Customer specific checks can be included by adding specific userexits.

   It might be sufficient to add a trigger at the forms to check if the parent document is reserved by the current user. However, it might be better to control the behavior with Web Service parameters for specific relation types (e.g. reservation of parent document required in general to create/update/delete the file assignment), or even on relation record level.

9. Error Handling

   While creating/updating/deleting relation records, the following issues can appear:

   - A unique index violation during create/update, because a duplicate record will be created.
   - The parent and/or child record is not available (either does not exist at all, or is not accessible).
   - The user has no access to write and/or delete.
   - Any other relation object trigger throws an error.

In case of an error, the processing of the remaining relation objects continues. At the end, the Web Service reports which objects failed to be created/updated/deleted, together with a corresponding error message.

To ensure that only masks designed for Web Service access are used, all mask names are checked against a Whitelist that is maintained by the administrator of the Agile e6 installation.

createUpdateStructure

<table>
<tr><td>Service</td><td>This Web Service operation allows you to process relations like Create/Read/Update/Delete and many more between the following objects:
▫ Document – Document
▫ Document - Item (STR Structure)
▫ Item-Item (STR Bill of Material)</td></tr>
<tr><td>Usage</td><td>This single request operation for ECService Web Services describes the generic character of the Web Service.</td></tr>
<tr><td>Request Type</td><td>CreateUpdateStructureRequestType
▫ messageId (String): ID to be returned in the response (optional).
▫ messageName (String): Name to be returned in the response (optional).
▫ relationType (List<ECRelationDescriptor<): List of relations to process.
<table>
<tr><td>contextFilter (List<PlmCondition)</td><td>CAD Context filter: attribute names (For example: UG2_IDENT) and values. This is used to query for the relation records of the parent in the Agile e6 relation list.</td></tr>
<tr><td>comparisonPattern (List<String>)</td><td>List of attribute names used as comparison pattern for relation record identification.
Comparison pattern (e.g. Parent-ID, Child-ID, Transf.-Matrix, Old Filename, Structure Ref, CAX-internal value)</td></tr>
<tr><td>cleanupFlag (Boolean)</td><td>Defines if records not matching any relation object are deleted.</td></tr>
<tr><td>Relation (PlmMetaRelationRef)</td><td>The metadata of the relation to be read (parent, child, type, view). This allows specifying the aggregate, refined, constraint, or type relation.</td></tr>
<tr><td>Structures (List<ECStructure>)</td><td>List of parents and their relation objects.
parentId (PlmAttributeChoice): Unique ID of the parent (EDB_ID or C_ID).
data (List<ECData>): The relation data for this parent.
If the list is empty and cleanup is requested, all relation records found in the CAD context are deleted.
Otherwise, the comparison algorithm checks which records need to be created, updated, or deleted from the EDM server.</td></tr>
</table></td></tr>
</table>

			attributes (List<PlmAttributeChoice>: List of attribute values to use for updating or creating the relation record. This must contain the attribute CHILD.C_ID if a create operation is required.

plmResult(plmResult): This describes how the result of the operation is returned to the client.

attributeNames (Boolean)	List of field names to be returned	▫ Optional ▫ Default ▫All visible fields and ID fields
includeBinaryValues (Boolean)	Include binary values	▫ Optional ▫ Default - False
includeAllLanguages (Boolean)	Include all languages	▫ Optional ▫ Default - False

Response Type

CreateUpdateStructureResponseType

- statusCode (ResponseStatusCode):

SUCCESS	The newly created PLM object including all attributes defined by the PLM result of the request.
PARTIAL_SUCCESS	The newly created PLM object including some of the attributes defined by the PlmResult of the request.
FAILURE	A faulty description including an error message and the type of error.

- ticket (String): A PLM ticket.
  - The ticket can be used instead of the password in subsequent calls. It allows the client to reference the Agile e6 session, even if the HTTP session is closed, or (in case of a backward flow) if there is no HTTP session.
  - The ticket is only valid for the EDM server instance that generated it.
- messageId (String): copied from the request, or generated.
- messageName (String): copied from the request, or the operation name.
- warnings (List< PlmWarningType >): List of warnings that occurred during the operation.
- exceptions (List< PlmExceptionType >): List of exceptions that occurred during the operation.
- relations (List<ECRelationData>): Data with error/success flag, plus error message (if available). The content and order in the list correspond to the list of ECRelationDescriptors in the request:

List<ECResultStructure>	parentId (PlmAttributeChoice): Unique ID of the parent (EDB_ID or C_ID) warnings (List< PlmWarningType >): List of warnings that occurred when processing this structure (optional). exceptions (List< PlmExceptionType >): List of exceptions that occurred when processing this

			structure (optional). This contains the information about failed delete attempts if structure cleanup was requested
			resultData (List<ECResultData>): List of structure elements for this parent. It contains all updated or created relation objects. ▫ statusCode (ResponseStatusCode): SUCCESS or FAILURE. ▫ operationType (OperationType): CREATE or UPDATE. DELETED records are not reported, only the new or updated structure records are returned. ▫ attributes (List<PlmAttributeChoice>: List of attribute values as requested. These contain the attributes requested in the PlmResult for the respective relation, or all visible attributes if no attributes have been requested. The list contains the injected attributes CHILD.EDB_ID, CHILD.C_ID (EDB_ID only if available). Parent information is not injected to reduce the payload; the parent is identical for all rows. ▫ exception (PlmExceptionType): Exception that occurred during the creat or update operation (optional).

Chapter 7

Appendix

SAMPLES

EchoServiceWrapper.java

This wrapper does not call an outbound Web Service. It returns the arguments that have been passed. This wrapper is already contained inside the Web Services application and can be used to test the infrastructure.

```
/*
 * $Id: EchoServiceWrapper.java,v 1.3 2010/10/15 15:11:56 brg Exp $
 *
 * Copyright (c) 1992, 2010, Oracle. All rights reserved.
 */
package com.agile.ws.e6.wrappers;

import com.agile.eci.EciConnection;
import com.agile.eci.EciPar;
import com.agile.eci.EciParBuffer;
import com.agile.share.callable.CallableParam;
import com.agile.share.trace.Logger;
import com.agile.share.trace.Trace;
import com.agile.share.util.StringArray;
import com.agile.share.util.StringList;
import com.agile.ws.e6.PlmSession;
import com.agile.ws.e6.wrappersupport.WrapperContext;

import java.net.InetAddress;

/**
 * A simple echo wrapper to test the wrapper mechanism.
 */
public class EchoServiceWrapper extends AbstractWrapper {

   private static Logger log =
Trace.getLogger(EchoServiceWrapper.class);

   /**
    * The name of this service wrapper.
    */
   public static final String NAME =
EchoServiceWrapper.class.getSimpleName();
```

```
   /**
    * Creates a new echo service.
    */
   public EchoServiceWrapper() {
      super(NAME);
   }

   /** {@inheritDoc} */
   @Override
   public StringList callWebService(WrapperContext context,
CallableParam args) {

      log.enter("callWebService", "context="+context+", args="+args);
      PlmSession session = null;
      String userName = "<unknown>";
      String processId = "<unknown>";

      try {
         session = createPlmSession(context);
         EciConnection con = session.getConnection();

         EciPar par = con.call("eci_rea_edb_usr");

         userName = par.get(1);
         log.info("My name is " + userName + "," +
               "\nI belong to the group " + par.get(2) +
               ".\nMy user ID is " + par.get(3) + "," +
               "\nmy group ID is " + par.get(4) +
               "\nand I am " + ("y".equals(par.get(5)) ? "" : "not ") +
"a manager" +
               ".\n" +
               "\nOur session is " + session + ".\n");

         par = con.call("eci_get_pid");
         processId = par.get(1);

         EciParBuffer buf = new EciParBuffer();
         buf.add("EDB-BAS-WARNING");
         buf.addNew("This is the e6 EchoService running inside WebLogic
on host " +
               InetAddress.getLocalHost().getHostName() + "\n\n" +
               "You are " + userName + " and your process ID is " +
processId);
         buf.end();
         con.call("eci_mes_wri", buf);
      }
      catch (Exception e) {
```

```
            log.error("Unable to reconnect to e6: ", e);
        }
        finally {
            if (session != null) {
                session.close();
            }
        }
        StringList result = new StringArray(args.getParam());
        log.leave("callWebService", "result=" + result);
        return result;
    }
}
```

SampleWrapper.java

This wrapper calls an Agile e6 core service as an example. The generated client code needed to call the Agile e6 Core Web Service is not included. The generated classes belong to the wrapper and need to be deployed along with it.

The SampleWrapper class needs a property file, SampleWrapper.properties, which contains the URL of the Web Service.

When Agile e6 application wants to call a wrapper called Sample, it passes Sample as the first argument to xutil_call_ws. The wrapper manager then looks for a class called SampleWrapper in all the packages in its search path. To call the EchoServiceWrapper, pass EchoService to xutil_call_ws.

```
/*
 * $Id: SampleWrapper.java,v 1.1.2.2 2011/06/30 15:53:27 brg Exp $
 *
 * Copyright (c) 1992, 2011, Oracle. All rights reserved.
 */

package com.agile.ws.e6.wrappers;

import com.agile.eci.EciConnection;
import com.agile.eci.EciPar;
import com.agile.eci.EciParBuffer;
import com.agile.security.tickets.plm.PlmTicket;
import com.agile.share.callable.CallableException;
import com.agile.share.callable.CallableParam;
import com.agile.share.trace.Logger;
import com.agile.share.trace.Trace;
import com.agile.share.util.StringArray;
import com.agile.share.util.StringList;
import com.agile.ws.e6.PlmSession;
import com.agile.ws.e6.WebServiceEnum;
```

```
import com.agile.ws.e6.client.core.common.PlmUserContext;
import com.agile.ws.e6.client.core.common.PlmUserContextUserInfo;
import com.agile.ws.e6.client.core.configuration.Configuration;
import com.agile.ws.e6.client.core.configuration.ConfigurationService;
import
com.agile.ws.e6.client.core.configuration.GetUserContextRequestType;
import
com.agile.ws.e6.client.core.configuration.GetUserContextResponseType;
import com.agile.ws.e6.wrappersupport.WrapperContext;

import java.net.InetAddress;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.UnknownHostException;
import java.util.Properties;

import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;

/**
 * A simple example of a wrapper for an out-bound web service call.
 *
 * <p>It extends the abstract wrapper implementation, which is shipped
 * in the library agile-ws-e6-callables as part of the WebServices.ear
file.
 *
* <p>Alternatively, a JAR file containing the wrapper can be added to
the
 * WebServices application into the APP_INF/lib directory,
 * so that it is deployed as part of the WebServices application.
 */
public class SampleWrapper extends AbstractWrapper {
   /** The Logger used to print trace messages */
   private static Logger log = Trace.getLogger(SampleWrapper.class);

   /** The name of our properties file */
   private static final String SAMPLE_PROPERTIES =
"SampleWrapper.properties";
   /** Property containing the URL of the external web service */
   private static final String PROP_ENDPOINT = "Sample.endPoint";

   /**
    * The service we want to contact.
    *
    * <p>This class - and all the others in the package
com.agile.ws.e6.client.core - is
    * generated by the WebLogic clientgen Ant task.
    */
   private ConfigurationService configurationService;
```

```
    /** The URL to the Configuration service */
    private String configurationEndPoint;

    /**
     * The name of this service wrapper.
     */
    public static final String NAME = "Sample";

    /**
     * Creates a new HelloWorld service.
     */
    public SampleWrapper() {
       super(NAME);
    }

    /**
     * @return the RCS information of this object's class (polymorphic)
     */
    @Override
    public final String getRCSId() {
       return getClassRCSId();
    }

    /**
     * @return the RCS information of this class (static)
     */
    public static String getClassRCSId() {
       return "$Id: SampleWrapper.java,v 1.1.2.2 2011/06/30 15:53:27 brg
Exp $";
    }

    /**
     * Creates the port to access the Configuration service.
     *
     * @param endPoint the WSDL URL of the service.
     * @param ticket the PLM ticket provided by the e6 server
     *
     * @return the port to call the Configuration service
     */
    private Configuration getConfigurationPort(PlmTicket ticket) throws
MalformedURLException {
       if (configurationService == null) {
          configurationService = new ConfigurationService(
                new URL(configurationEndPoint),
                new QName(WebServiceEnum.CONFIGURATION.getNamespace(),

WebServiceEnum.CONFIGURATION.getServiceName()));
       }
```

```
      Configuration port = configurationService.getConfigurationPort();

      BindingProvider binding = (BindingProvider) port;

      // Add authentication, we have a ticket so we do not need the
password

binding.getRequestContext().put(BindingProvider.USERNAME_PROPERTY,
ticket.getUserName());

binding.getRequestContext().put(BindingProvider.PASSWORD_PROPERTY,
String.valueOf(ticket.getRawTicket()));

      // Maintain the HTTP session, in case we do several calls to the
server

binding.getRequestContext().put(BindingProvider.SESSION_MAINTAIN_PROPER
TY, Boolean.TRUE);

      return port;
   }

   /**
    * Calls the web service.
    *
    * @param context the context that contains the reference to the e6
server
    * @param args arguments passed by the e6 server
    * @return list of return values, or null
    *
    * @throws CallableException Web service call failed
    */
   @Override
   public final StringList callWebService(final WrapperContext context,
final CallableParam args) throws CallableException {
      log.enter("callWebService", "context="+context+", args="+args);

      // Try to load our properties
      Properties sampleProps = new Properties();

      try {
         sampleProps.load(getClass().getResourceAsStream("/" +
SAMPLE_PROPERTIES));
         configurationEndPoint =
sampleProps.getProperty(PROP_ENDPOINT);
      }
      catch (Exception e) {
         String msg = "Unable to load SampleWrapper.properties";
         log.error(msg, e);
```

```
            throw new CallableException(this, msg);
         }
         if (configurationEndPoint == null) {
            log.error("No WSDL URL found in " + SAMPLE_PROPERTIES);
            throw new CallableException(this, "No web service URL
configured for the Sample wrapper");
         }

         log.info("Using WSDL at " + configurationEndPoint);

         Configuration port = null;
         StringList result = new StringArray();

         try {
            // Create a session object:
            //
            // It gives us access to an ECI connection to the same e6
server instance that called us,
            // and - if needed - it will create an AxalantRepository
instance for us,
            // if we want to make use of the high level Java ECI (JET
layer).
            PlmSession session = createPlmSession(context);

            // First some ECI calls
            callEciDemo(session);

            // Now the "external" web service call.
            //
            // Here, we would normally call an external service of another
system,
            // but for demo purposes, we just call an e6 Core service.
            port = getConfigurationPort(context.getPlmTicket());
            log.info("Port created with PLM ticket");

            GetUserContextRequestType request = new
GetUserContextRequestType();
            GetUserContextResponseType response =
port.getUserContext(request);
            log.info("Web method getUserContext() returned with status
code " + response.getStatusCode());

            PlmUserContext e6Context = response.getPlmUserContext();
            PlmUserContextUserInfo userInfo = e6Context.getPlmUserInfo();

            result.add("User name  = " + userInfo.getUserName());
            result.add("Group name = " + userInfo.getGroup());
            result.add("Language   = " + userInfo.getUserLanguage());
            result.add("Locale     = " + userInfo.getUserLocale());
```

```
      }
      catch (Exception e) {
         log.error("Unable to call web service", e);
         throw new CallableException(this, "Web service call failed",
e);
      }
      finally {
         if (port != null) {
            try {
               // Tell the server that we no longer need the e6
session.
               port.closeSession();
            }
            catch (Exception e) {
               log.error("Unable to close port", e);
            }
         }
      }
      return result;
   }

   /**
    * Calls some ECI functions to demonstrate how to get additional
data.
    *
    * @param session the e6 PLM session
    */
   private void callEciDemo(PlmSession session) {
      String userName = null;
      String processId = null;
      String host;

      try {
         host = InetAddress.getLocalHost().getHostName();
      }
      catch (UnknownHostException e) {
         host = "unknown host";
      }

      EciConnection con = session.getConnection();

      EciPar par = con.call("eci_rea_edb_usr");

      userName = par.get(1);
      log.info("My name is " + userName + "," +
            "\nI belong to the group " + par.get(2) +
            ".\nMy user ID is " + par.get(3) + "," +
```

```
            "\nmy group ID is " + par.get(4) +
            "\nand I am " + ("y".equals(par.get(5)) ? "" : "not ") + "a
manager" +
            ".\n" +
            "\nOur session is " + session + ".\n");

      par = con.call("eci_get_pid");
      processId = par.get(1);

      EciParBuffer buf = new EciParBuffer();
      buf.add("EDB-BAS-WARNING");
      buf.addNew("This is the e6 SampleWrapper running inside WebLogic
on host " +
            host + "\n\n" +
            "You are " + userName + " and your process ID is " +
processId);
      buf.end();
      // Prints a message on the client that contacted us
      con.call("eci_mes_wri", buf);
   }
}
```

Web Services Security

In the following example, a Web Services policy is used for the WSS: SOAP Message Security.

You are required to provide the username token and a client certificate. The complete security information is embedded into the SOAP message.

Note By configuring WSS: SOAP Message Security, the WSDL gets modified.

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header>
        <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
                       S:mustUnderstand="1">
            <ns1:EncryptedKey
xmlns:ns1="http://www.w3.org/2001/04/xmlenc#" Id="JOLZ6aDu8pt9PRPe">
                <ns1:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                    <ns2:DigestMethod
xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"

Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                </ns1:EncryptionMethod>
                <ns3:KeyInfo
xmlns:ns3="http://www.w3.org/2000/09/xmldsig#">
```

```
                    <wsse:SecurityTokenReference
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"

xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"

wsu:Id="str_kNu7Vfo6pZLqdYcv">
                        <X509Data
xmlns="http://www.w3.org/2000/09/xmldsig#">
                            <X509IssuerSerial>
                                <X509IssuerName>CN=CertGenCAB,OU=FOR
TESTING ONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US</X509IssuerName>
                                <X509SerialNumber>-
135694037818432800534509206009756866711</X509SerialNumber>
                            </X509IssuerSerial>
                        </X509Data>
                    </wsse:SecurityTokenReference>
                </ns3:KeyInfo>
                <ns1:CipherData>

<ns1:CipherValue>JHUAfXjSBYxXKAGrpQ.....NUWGQ9IPL9M1uODqmnQ8Nlk=
</ns1:CipherValue>
                </ns1:CipherData>
                <ns1:ReferenceList>
                    <ns1:DataReference URI="#PJr5jO7puKh5OL5b" />
                </ns1:ReferenceList>
            </ns1:EncryptedKey>
            <wsse:BinarySecurityToken xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
                                      xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                                      EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary"
                                      ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"

wsu:Id="bst_GpDR1niRFucsZsbm">MIICKzCc.......KMuSA1XAQ==</wsse:BinarySe
curityToken>
            <dsig:Signature
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
                <dsig:SignedInfo>
                    <dsig:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                    <dsig:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
                    <dsig:Reference URI="#Timestamp_KKvWCLdlrCRB2SNF">
                        <dsig:Transforms>
                            <dsig:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                        </dsig:Transforms>
```

```
                        <dsig:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />

<dsig:DigestValue>xndjH7PWB/yinv/uFzmElQzAezI=</dsig:DigestValue>
                    </dsig:Reference>
                    <dsig:Reference URI="#Body_0lUdcO0zqWY2bBvU">
                        <dsig:Transforms>
                            <dsig:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                        </dsig:Transforms>
                        <dsig:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />

<dsig:DigestValue>gt0av56Xh/gca30jxtDChJkFZck=</dsig:DigestValue>
                    </dsig:Reference>
                    <dsig:Reference URI="#unt_UROJpKRFSAIZLKFf">
                        <dsig:Transforms>
                            <dsig:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                        </dsig:Transforms>
                        <dsig:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />

<dsig:DigestValue>QBSH0z6BxmZgEM56+g3ZS2w00lg=</dsig:DigestValue>
                    </dsig:Reference>
                    <dsig:Reference URI="#bst_GpDR1niRFucsZsbm">
                        <dsig:Transforms>
                            <dsig:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                        </dsig:Transforms>
                        <dsig:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />

<dsig:DigestValue>62PwFQZD1Nj1R77qudrvzJCIUNE=</dsig:DigestValue>
                    </dsig:Reference>
                </dsig:SignedInfo>

<dsig:SignatureValue>TfFLyCR9MF4ZepqwmnCned7mj5TavfwjDg69.......MIFR3kB
U=</dsig:SignatureValue>
                <dsig:KeyInfo>
                    <wsse:SecurityTokenReference
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"

xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-
secext-1.1.xsd"

xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"

wsse11:TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
```

```
wsu:Id="str_KVJy1AtKNAxVYsKq">
                        <wsse:Reference URI="#bst_GpDR1niRFucsZsbm"
                                        ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" />
                    </wsse:SecurityTokenReference>
                </dsig:KeyInfo>
            </dsig:Signature>
            <ns1:EncryptedData
xmlns:ns1="http://www.w3.org/2001/04/xmlenc#" Encoding="UTF-8"
Id="PJr5jO7puKh5OL5b" MimeType="text/xml"

Type="http://www.w3.org/2001/04/xmlenc#Element">
                <ns1:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
                <ns1:CipherData>

<ns1:CipherValue>yG4ULSKvJFL8......OgqkcPmY6yhdpoE=</ns1:CipherValue>
                </ns1:CipherData>
            </ns1:EncryptedData>
            <wsu:Timestamp xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                           wsu:Id="Timestamp_KKvWCLdlrCRB2SNF">
                <wsu:Created>2010-02-03T14:44:31Z</wsu:Created>
                <wsu:Expires>2010-02-03T14:45:31Z</wsu:Expires>
            </wsu:Timestamp>
        </wsse:Security>
    </S:Header>
    <S:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd"
            wsu:Id="Body_0lUdcO0zqWY2bBvU">
        <ns4:hello
xmlns:ns2="http://xmlns.oracle.com/Agile/e6/Metadata/v0"
                   xmlns:ns3="http://xmlns.oracle.com/Agile/e6/plm"

xmlns:ns4="http://xmlns.oracle.com/Agile/e6/HelloWorld/v0" />
    </S:Body>
</S:Envelope>
```

In the following example, WSS: SOAP Message Security information is provided for the SOAP request:

```
MetadataService service = new MetadataService(wsdlURL, serviceQName);
BindingProvider bindingProvider = (BindingProvider) service.getPort();
List<CredentialProvider> credProviders = new
ArrayList<CredentialProvider>();
try {
  // Load server certificate
  X509Certificate serverCert =
(X509Certificate)CertUtils.getCertificate(serverCertFile);
```

```
  // Check server certificate
  serverCert.checkValidity();

  // Create a new certificate credential provider
  CredentialProvider cp = new
ClientBSTCredentialProvider(clientKeyStore,
               clientKeyStorePass, clientKeyAlias, clientKeyPass,
"JKS", serverCert);

  // Add certificate credential provider to the array list
 credProviders.add(cp);

  // Create a new username token credential provider
  CredentialProvider up = new
ClientUNTCredentialProvider(username.getBytes(), password.getBytes());

  // Add certificate username token credential provider to the array
list
  credProviders.add(up);

  Map<String, Object> rc =
((BindingProvider)portName).getRequestContext();

  // Add the credential providers to the request context
  rc.put(WSSecurityContext.CREDENTIAL_PROVIDER_LIST, credProviders);

  // Add a trusr manager to the request context, you can do here some
validation tests on the return certificate of the SOAP message
  rc.put(WSSecurityContext.TRUST_MANAGER,
                   new TrustManager(){
                     public boolean
certificateCallback(X509Certificate[] chain, int validateErr){
                       return true;
                     }
                   } );
} catch (Exception e) {
  log.printStackTrace(e);}
```