



# **Agile Product Lifecycle Management**

Agile PLM Core Web Services User Manual

v9.3.0.2

Part No. E17317-01

June 2010

# Oracle Copyright

*Copyright © 1995, 2010, 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 software or related documentation 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 RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software 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 which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software 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.

# CONTENTS

---

Oracle Copyright.....	ii
<b>Chapter 1 Introduction to Agile PLM Web Services .....</b>	<b>1</b>
About Service Oriented Architecture (SOA) .....	1
About Web Services .....	1
Core Technologies .....	2
Web Services Description Language (WSDL) .....	2
XML and XML Schema.....	3
Simple Object Access Protocol (SOAP) .....	3
Web Services Architecture .....	3
About Agile PLM Web Services .....	4
Agile PLM Core Web Services .....	4
Agile PLM EC Services .....	5
Agile PLM Web Service Authentication and Performance .....	5
Impact on Existing Agile PLM Extensions and Services .....	5
Casual User Interface Integration Examples.....	6
User Interface Integration - MS Word.....	6
User Interface Integration - MS Excel.....	7
User Interface Integration - Portals and Agile Web Client.....	7
User Interface Integration - Mobile ADF .....	8
CAD Integration through EC Services .....	8
Building Casual User Interfaces .....	9
Developing User Interfaces for MS Office .....	9
Developing User Interfaces for Oracle WebCenter and ADF .....	10
<b>Chapter 2 Getting Started with Agile Web Services .....</b>	<b>13</b>
Before Building a Web Services Client.....	13
Operational Environment.....	14
Standards Compliance .....	14
Web Services Engines .....	14
Generating and Initializing the Stubs.....	15
Generating Agile Stubs.....	15
Initializing the Client Stubs .....	15
Understanding the MessageElement .....	15
Obtaining the API Names and Attribute IDs .....	16
Special Handling of MessageElements .....	16

Unit of Measure .....	16
Multilist and List .....	16
Money .....	17
Date .....	18
User/Supplier/Customer/Analyst .....	18
Agile Attributes without API Names.....	19
Item Constants.....	19
User Constants .....	19
Understanding the Web Services Responses.....	19
Response Status Code.....	19
Exceptions and Warnings.....	20
Working with Warnings .....	21
<b>Chapter 3 Working with Business Objects.....</b>	<b>23</b>
Getting an Object.....	23
Special Handling in the getObject Operation.....	24
Creating an Object.....	24
Saving As a New Object.....	25
Special Handling in the saveAsObject Operation.....	26
Deleting and Undeleting an Object.....	26
Checking the Delete Status .....	27
Updating an Object.....	28
Getting the Status of an Object .....	28
Getting the AutoNumbers .....	28
Getting the Classes .....	29
Getting the Subclasses .....	29
<b>Chapter 4 Working with Tables.....</b>	<b>31</b>
About Tables .....	31
Operations Supported on Tables .....	31
Loading a Table.....	33
Special Handling in the loadTable Operation .....	34
Working with the Readonly Tables.....	35
Retrieving the Metadata of a Table .....	35
Adding Rows to a Table .....	35
Special Handling in the addRows Operation .....	36

Adding a Site to the Sites Tab of an Item .....	36
Adding Suppliers to the Suppliers Tab of an Item .....	36
Adding Suppliers to a Manufacturer Part .....	37
Adding Manufacturer Part to AML of an Item .....	38
Adding Manufacturer Part to the Relationships Tab .....	38
Adding Affected Item to a Change .....	38
Adding Site Specific Item to the BOM Tab .....	39
Adding Site Specific AML to the Manufacturers Tab .....	39
Updating Rows in a Table .....	40
Removing Rows from a Table .....	41
Clearing a Table .....	41
Copying Tables.....	42
Redlining a Table.....	42
<b>Chapter 5 Working with Searches .....</b>	<b>45</b>
Agile PLM Searches .....	45
Specifying Search Criteria .....	45
Search Conditions .....	46
Search Operation Keywords.....	46
Specifying Search Attributes .....	47
Getting the Searchable Attributes.....	48
Using Relational Operators .....	48
Using Unicode Escape Sequences .....	49
Using Between, Not Between, In, and Not In Operators .....	49
Using the Nested Criteria to Search for Values in Object Lists .....	50
Searching for Words or Phrases Contained in Attachments .....	51
Using Logical Operators .....	51
Using Wildcard Characters with the Like Operator.....	52
Using Parentheses in Search Criteria .....	52
Using SQL Syntax to specify Search Criteria .....	53
Using SQL Wildcards .....	54
Setting Result Attributes for a Search .....	54
Specifying Result Attributes.....	59
Examples of Searches .....	60
Quick Search .....	60
Advanced Search .....	60
Getting the Searchable Attributes.....	61
<b>Chapter 6 Working with File Folders and Attachments .....</b>	<b>63</b>
Agile File Folders .....	63
Managing File Folders .....	63
Creating a File Folder .....	63
Checking Out a File Folder .....	64

Setting the Version of File Folder Files.....	65
Cancelling a File Folder Checkout.....	66
Checking In a File Folder.....	66
Deleting the File Folders.....	66
Getting a File from a File Folder.....	67
Getting a File from a File Folder using a Download URL.....	68
Getting a File from a particular Version of File Folder.....	68
Adding Files to a File Folder Object.....	68
Adding Files in a File Folder.....	69
Managing Attachments.....	70
Getting Attachments of an Object.....	70
Getting a Specific Attachment and a File Folder.....	70
Getting a Specific Attachment using a URL.....	71
Adding Attachments to an Object.....	72
Adding attachments by File Reference.....	72
Adding Multiple Attachments into Single Folder.....	73
Adding Files using SOAP Attachment.....	73
Checking Out the Attachments.....	74
Checking Out All the Attachments.....	74
Checking Out Multiple Attachments from a Folder.....	74
Checking In the Attachments.....	75
Checking In an Attachment with FileId Identification.....	75
Deleting the Attachments.....	76
<b>Chapter 7 Managing Workflows.....</b>	<b>77</b>
About Agile PLM Workflows.....	77
How the Status of a Change Affects Workflow Functionality.....	77
Getting the Status of a Workflow.....	78
Getting the Workflow of a Routable Object.....	78
Setting a Workflow.....	78
Checking User Privileges.....	79
Adding and Removing Approvers.....	79
Getting Approvers.....	80
Approving a Routable Object.....	80
Rejecting a Routable Object.....	81
Commenting a Change.....	81
Auditing a Change.....	82
Changing the Workflow Status of an Object.....	82

<b>Reference: Core Operations .....</b>	<b>83</b>
Chapter 8 Admin and Metadata Web Services .....	85
getAllClasses .....	85
getSubClasses .....	87
getNode .....	89
getLists .....	91
getAttributes .....	93
getTableMetadata .....	95
getAutoNumbers .....	97
getUsers .....	99
getUserGroups .....	101
convertCurrency .....	102
Chapter 9 Attachment Web Services .....	105
getFileFF .....	105
getFileAttachment .....	107
addFileAttachment .....	109
checkoutFF .....	111
checkinFF .....	113
cancelCheckoutFF .....	115
checkoutAttachment .....	117
checkinAttachment .....	119
addFileFF .....	121
Chapter 10 Business Web Services .....	125
createObject .....	125
getObject .....	129
updateObject .....	132
deleteObject .....	134
undeleteObject .....	136
isDeletedObject .....	138
sendObject .....	140
saveAsObject .....	142
checkPrivilege .....	144
Chapter 11 Collaboration Web Services .....	147
getWorkflows .....	147
getStatus .....	149
auditRObj .....	151
getApprovers .....	153
changeStatus .....	155

approveRObjct.....	157
rejectRObjct.....	159
setWorkFlow.....	161
addApprovers.....	163
removeApprovers.....	165
commentRObjct.....	167
Chapter 12 PC Web Services .....	171
setIncorporate.....	171
getRevisions .....	173
undoRedline .....	175
isRedlineModified .....	177
Chapter 13 Search Web Services .....	181
quickSearch.....	181
advancedSearch.....	184
getSearchableAttributes .....	186
Chapter 14 Tables Web Services.....	189
isReadOnlyTable .....	189
clearTable.....	191
copyTable .....	193
addRows.....	195
updateRows.....	197
removeRows.....	199
loadTable.....	201
<b>Appendix A .....</b>	<b>Error! Bookmark not defined.</b>
Working with Java Samples .....	205
Building Stubs and Compiling the Samples.....	205
Executing the Samples using ant Task .....	205
Executing the Samples using a Java IDE.....	206
Understanding the Code.....	206
AddFileSOAPAttachment Method .....	207
Helper Methods .....	209
getRowId Method .....	209
getFileId Method.....	212
Troubleshooting.....	<b>Error! Bookmark not defined.</b>



# Preface

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

---

The [Oracle Technology Network \(OTN\) Web site](http://www.oracle.com/technology/documentation/agile.html) <http://www.oracle.com/technology/documentation/agile.html> can be accessed through **Help > Manuals** in both Agile Web Client and Agile JavaClient. If you need additional assistance or information, please contact My Oracle Support (<https://support.oracle.com>) for assistance.

---

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

---

## TTY Access to Oracle Support Services

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

## Readme

Any last-minute information about Agile PLM can be found in the Readme file on the [Oracle Technology Network \(OTN\) Web site](http://www.oracle.com/technology/documentation/agile.html) <http://www.oracle.com/technology/documentation/agile.html>

## Agile Training Aids

Go to the [Oracle University Web page](http://www.oracle.com/education/chooser/selectcountry_new.html) [http://www.oracle.com/education/chooser/selectcountry\\_new.html](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.

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.



# Introduction to Agile PLM Web Services

**This chapter includes the following:**

---

▪ About Service Oriented Architecture (SOA) .....	1
▪ About Web Services .....	1
▪ About Agile PLM Web Services .....	4
▪ Casual User Interface Integration Examples .....	6
▪ CAD Integration through EC Services .....	8
▪ Building Casual User Interfaces .....	9

## About Service Oriented Architecture (SOA)

Service Oriented Architecture (SOA) is a business-centric IT architecture for building enterprise applications through adaptable and re-usable business processes and services. Each service implements one action such as creating a product record, viewing a BOM table, or updating the Price and Compliance data.

Leading companies are gaining operational efficiencies and business agility through adaptable, re-usable business processes and services built on truly flexible Service-Oriented Architecture (SOA) platforms.

The guiding principles of SOA are:

- Self contained and loosely coupled
- Well defined standards-based interfaces
- Right-sized interfaces
- Location independent and interoperable in a standards-based manner
- Implementation agnostic

One SOA implementation is the Web services approach where the basic unit of communication is a message, rather than an operation. This is often referred to as "message-oriented" services. Web services make functional building-blocks that are accessible over standard Internet protocols and independent of platforms and programming languages. SOA is gaining wide customer adoption because of its reliance on standards-based protocols and enabling rapid development of applications using Web Services. SOA and Web services are supported by most major software vendors.

## 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** – Business 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 for 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

*Oracle's Agile Web services use industry standard core technologies.* The bulleted list that follows lists these technologies. Each core technology is explained in detail in the topics that follow.

- Web Services Description Language (WSDL)
- XML and XML Schema
- Simple Object Access Protocol (SOAP)

### Web Services 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 as an XML file. Document/Literal is 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 element must be described by an XML Schema embedded in the WSDL file.

When using Document/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 will consist 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:

- Web services 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.
- Web services 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 PLM Web Services

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

- Well defined standards based discoverable Interface
- XML based Web Service Framework - Apache Axis 1.4
- Modularized PLM Schema (XSD) and WSDL for easy maintenance
- Standards-based WSDL to ensure compatibility across various clients (.NET, Java, and BPEL)
- Batch APIs wherever applicable for better performance
- Web Service versioning for backward compatibility

Agile PLM Web Services expose all key PLM functionalities in the following services.

- **Agile PLM Core Web Services** – These services support functionalities provided by PLM solutions such as PC, PQM, PCM, PPM, PG&C. See [Agile PLM Core Web Services Operations](#) on page 83.
- **Agile PLM EC Web Services** – These services support functionalities provided by Agile PLM's Engineering Services (EC) solution. See Agile PLM EC Services Guide at [Oracle Technology Network \(OTN\) Web site](#) <http://www.oracle.com/technology/documentation/agile.html>.

## Agile PLM Core Web Services

Agile PLM Core Web Services are a set of services for the following PLM functionalities:

- Business Object CRUD (Create, Read, Update, Delete) data services
- Collaboration services
- Meta data Services
- Search Services
- Attachment Services
- Table Services

## Agile PLM EC Services

Agile PLM Engineering Collaboration (EC) Services are a set of Business Services that supplement PLM's Core Web Services for CAD use cases. They also offer a set of higher level BPEL orchestration services. Customers and partners can build next generation MCAD and ECAD connectors utilizing Agile PLM Web Services and Engineering Collaboration Services.

Some of the benefits are:

- Significantly Improves WAN performance for CAD connectors because the bulk of the logic is deployed to the server
- Makes it easier for development partners and customers to implement CAD connectors
- Provides the unique and interface friendly API Name field to access PLM metadata

## Agile PLM Web Service Authentication and Performance

In 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:

- 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 are placed in the SOAP header of the Web service reply.
- Whenever the `sessionID` is included by the client in subsequent Web service requests, that `sessionID` will be used to restore cached session information, thus bypassing the substantially more expensive process of re-executing the authentication. Note that, 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 WS-Security. Using the `UserName` token as provided in WS-Security, while fully supported as part of Agile PLM's WSI Basic Profile compliance, will not yield the same benefit as using the higher-performance session optimization facility provided by the Agile PLM implementation.

## Impact on Existing Agile PLM Extensions and Services

Agile PLM provides tools and process extensions to customize the Agile PLM to meet unique user requirements, provide access to external databases, extend automation capabilities, and develop UI extensions. These tools and services are listed below. Agile PLM Web Services implementation has no impact on these capabilities; they are in addition to the existing services.

- **Agile SDK** – The SDK is a set of Java APIs that enable building custom applications to access or extend the Agile PLM server functionalities. **For information, refer to *Agile PLM SDK Developer Guide*.**
- **Agile Integration Services (AIS)** – AIS is a collection of predefined Web Services in the Agile Integration framework that enable communication between PLM server and disparate

database. **For information, refer to *Agile PLM AIS Developer Guide*.**

- **Agile Content Services (ACS)** – ACS is a process for transferring data to other Agile PLM solutions or to any other external system. **For information, refer to *Agile PLM ACS User Guide*.**
- **Process Extensions (PX)** – PX is a framework for extending the functionality of the Agile PLM system. The functionality can be server-side extensions such as custom automations, or client-side functionality such as new commands added to the Java/Web Client's Actions or Tools menus. **For information, refer to *Agile PLM SDK Developer Guide*.**
- **Web Service Extensions (WSX)** – WSX is a Web service engine that enables communication between Agile PLM and internal and external systems. **For information, refer to *Agile PLM SDK Developer Guide*.**
- **Dashboard Management Extensions (DX)** – Similar to PX, DX extends the functionalities of the Agile PLM system. **For information, refer to *Agile PLM SDK Developer Guide*.**

## Casual User Interface Integration Examples

Agile Web Client and Agile Java Client are targeted towards those who use the more complex product lifecycle management features of Agile PLM on a daily basis to perform assigned tasks and duties. There is also another set of users who use the auxiliary capabilities of Agile PLM to perform lightweight tasks such as document management, importing compliance and price data, or approving ECO and Sales RFQ.

The tools of choice for these users are the popular desktop products provided by Microsoft Office or Adobe Acrobat, Mobile devices. They prefer simple user-friendly interfaces, for example:

- Microsoft Word and Acrobat for document management
- Microsoft Excel to import price and compliance data
- Oracle WebCenter and Oracle Application Development Framework (ADF) for simple document management tasks
- Oracle WebCenter and ADF for simple item management tasks
- Mobile devices to access sales RFQ
- Mobile devices to access ECO Approval
- Microsoft Sharepoint for simple document management tasks

## User Interface Integration - MS Word

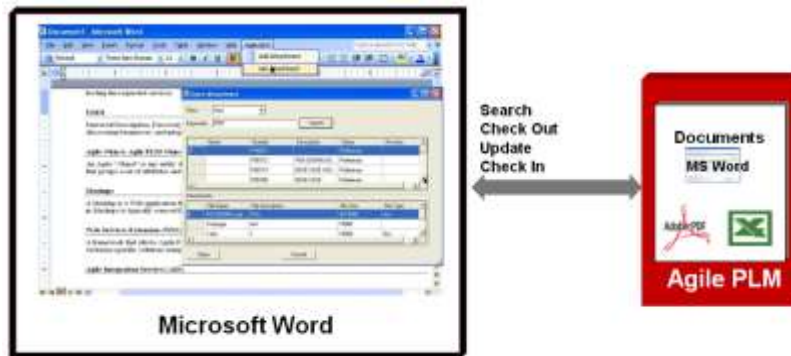
This example demonstrates document management capabilities of PLM's Web Services. Currently, when casual users want to view or update a document in Agile PLM, they do so by logging in to the Web Client to retrieve and view the Word document. The steps are:

1. Log in to PLM Client
2. Search and locate the document
3. Check out the document (in Word)
4. Modify the documents (in Word)



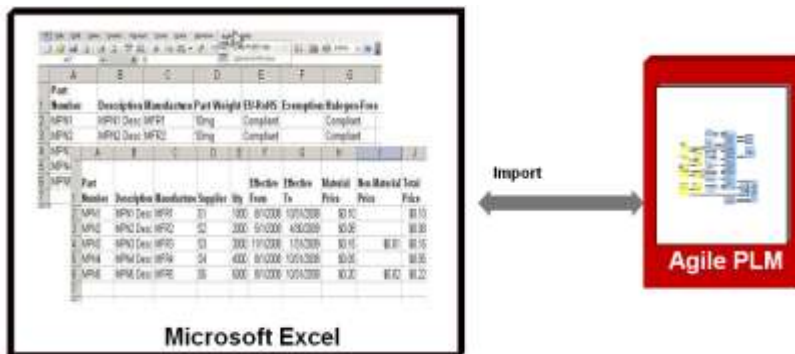
5. Check In the document
6. Log out

Using Agile PLM's Web Services, the casual user directly accesses Agile PLM documents from MS Word. This simple UI will encourage and accelerate greater user participation. Agile PLM is transparent to this class of users which eliminates training and exposure in PLM Web Client.



## User Interface Integration - MS Excel

This is similar to MS - Word integration. In this case, the casual user is one of your partners and suppliers. Using PLM's Web Services, you can provide a simple UI in Excel template for suppliers and partners. Then when necessary, suppliers import information such as compliance and price data directly into PLM system from Excel. Benefits include greater and more convenient supplier participation in the PLM process with no training in Agile PLM Web Client.



## User Interface Integration - Portals and Agile Web Client

Before PLM Web Services, the practice was to create custom Web applications using Agile PLM SDK with various tools and technologies. With Web Services, you can build rich Web applications in Oracle Web Center (and ADF) by taking advantage of Web 2.0 UI and mobile services.

Once you develop the custom UI Web application for casual users, you can also integrate the custom UI with Agile Web Client using Agile PLM's URL Process Extensions (refer to *Agile PLM SDK Developer Guide*) and Smart URL features.



## User Interface Integration - Mobile ADF

One of the key demands in Agile PLM installations is mobile access for management and executive personnel. One such example is ECO Approval by the senior or management staff using mobile devices. PLM's Web Services enable developing simple ECO Approval applications for users of mobile devices.

The following illustrations depict a sales RFQ implementation from a sales manager's perspective:

Using the mobile device's browser, the sales manager launches the Mobile application built using Agile Web Services. The first screen is the Search RFQ screen. The second is the RFQ Details screen and the third, the Send RFQ screen.



## CAD Integration through EC Services

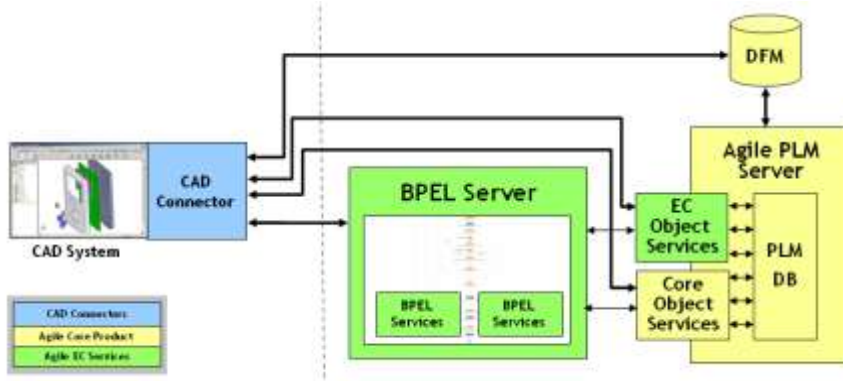
Customers and partners can build next generation MCAD and ECAD connectors with the aid of Agile PLM Core and Engineering Collaboration Web Services. The benefits were summarized in [Agile PLM EC Services](#) on page 5.

Some of the EC services are orchestrated using Business Process Execution Language (BPEL).

BPEL, short for Web Services Business Process Execution Language is an executable language for specifying interactions with Web Services. Processes in Business Process Execution Language export and import information by using Web Service interfaces exclusively.

The CAD integration and the role of BPEL server is shown in the following illustration.

**Figure 1: CAD integration architecture**



## Building Casual User Interfaces

The following paragraphs describe the tools and the steps in developing some the UI integration examples in MS Office and Oracle Web Center (and ADF) environments.

### Developing User Interfaces for MS Office

Microsoft supports building UI integration interfaces by providing the *Microsoft Office Add-in* (a piece of code) for this purpose. MS Office Add-in support integration at Application level and document level.

To develop the MS Word Add-in with PLM, the following tools and applications are necessary:

- Application software
  - Microsoft Visual Studio 2005/2008
  - Dot NET framework 3.5
  - Agile PLM (v9.3 or above) server
  - Microsoft Word 2003/2007
- Programming languages
  - C#
  - Visual Basic for .NET
  - Microsoft Visual C++/ATL
- Plug-in templates
  - Shared Add-in Extensibility template

---

**Note** The *Shared Add-in Extensibility* templates are used to deploy a single add-in onto multiple Microsoft Office applications (common add-ins across Word, Excel, and other office applications). This Add-in is always installed only at the application-level.

---

- Office 2003/2007 Add-in template

### Steps in developing an MS Office Add-in:

1. Evaluate Add-in type: application level versus document level
2. Evaluate programming language: C#, Visual Basic
3. Create a project in Microsoft Visual Studio 2005/2008
4. Generate the C#/Visual Basic Stubs from Agile WSDL

For information on Stubs, see [Generating and Initializing the Stubs](#) on page 15.

5. Create Windows Forms
6. Bind data to UI controls
  - Populate documents with data from Agile Web Services
7. Build & test the Add-in
8. Deploy the Add-in
9. Extend Agile 9.3 Samples to fit your business needs:
  - MS Word Document Management - Application Level Add-In
  - MS Excel – Import BOM/Price/ Compliance - document Level Add-in

---

**Note** The source code for these two MS Office Add-ins is available for download from Oracle OTN Website. A good source of information for developing an MS Office Add-in is the MSDN forum.

---

## Developing User Interfaces for Oracle WebCenter and ADF

This section provides basic information to develop the following UIs in Oracle Web Center and ADF environments.

- Document Management UI in Oracle WebCenter (and ADF 10g)
- Item Management UI in Oracle WebCenter (and ADF 11g)
- Sales RFQ UI in Mobile device
- ECO Approval UI in Mobile device

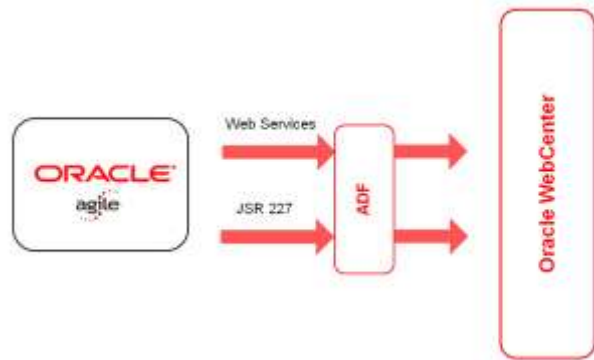
You need the following tools and applications to develop the Oracle WebCenter (and ADF) with PLM:

- Software
  - Oracle jDeveloper 10g/11g
  - Agile PLM (v9.3 or above) server

▫ Programming Languages

- Java

**Figure 2: Oracle WebCenter with Agile PLM**



**Steps in developing ADF applications**

1. Create a Project in jDeveloper
2. Generate the Java Stubs from Agile WSDL
3. Map XML schema to Java Classes
4. Create UI forms
5. Create page flow
6. Bind data to UI controls
7. Build and test the applications
8. Deploy the applications
9. Extend Agile 9.3 samples to meet business needs:
  - Document management
  - Item management
  - Sales RFQ

For information on Oracle Web Center, ADF, and jDeveloper, visit Oracle Web site at:  
<http://www.oracle.com/technology/products/webcenter/index.html> and  
<http://www.oracle.com/technology/products/adf/index.html>



# Getting Started with Agile Web Services

This chapter includes the following:

---

▪ Operational Environment .....	14
▪ Generating and Initializing the Stubs .....	15
▪ Understanding the MessageElement.....	15
▪ Agile Attributes without API Names .....	19
▪ Understanding the Web Services Responses .....	19

## Before Building a Web Services Client

Verify that the following are in place:

- The following jars are present in classpath:
  - axis.jar
  - axis-ant.jar
  - commons-discovery.jar

- Ant libraries are present.

- The Ant build file contains the following:

```
<path id="build.classpath">
  <fileset dir="${axislib.dir}">
    <include name="**/*.jar" />
  </fileset>
</path>
<taskdef resource="axis-tasks.properties">
  <classpath refid="build.classpath"/>
</taskdef>
<target name="wsdl2java-Generate-Client">
  <echo message="Generating all the client side stubs"/>
  <axis-wsdl2java
    all="true"
    output="./src"
    verbose="true"
    url="http://<host>:<port>/core/services/<serviceName>?wsdl">
  </axis-wsdl2java>
</target>
```

Replace the `<serviceName>` with the name of the Web Service. For example, BusinessObject, Collaboration and so on.

## Operational Environment

Development platforms vary in their SOAP implementations. Implementation differences in certain development platforms may prevent access to some or all of the features in the API. If you are using Visual Studio for .NET development, it is recommended that you use Visual Studio 2003 or higher.

Agile PLM Application	Release 9.3
Default Web Services Engine	Apache Axis 1.4
Java 2 Platform Standard Edition Development Kit	5.0

## Standards Compliance

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

Standard	Location
Simple Object Access Protocol (SOAP) 1.1/1.2	<a href="http://www.w3.org/TR/2000/NOTE-SOAP-20000508/">http://www.w3.org/TR/2000/NOTE-SOAP-20000508/</a>
Web Service Description Language (WSDL) 1.2	<a href="http://www.w3.org/TR/2001/NOTE-wsdl-20010315">http://www.w3.org/TR/2001/NOTE-wsdl-20010315</a>
WS-I Basic Profile 1.1	<a href="http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html">http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html</a>
XML Schema 1.1	<a href="http://www.w3.org/XML/Schema">http://www.w3.org/XML/Schema</a>

## Web Services Engines

All Application Server vendors, such as Oracle, BEA, IBM, 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 following Web Services Engines are supported:

- Oracle Apps Server Web Service Infrastructure
- WebLogic Web Service Infrastructure
- Axis - version 1.4
- Axis2 - version 1.4

<b>Important</b> Axis 1.4 is the default Web Services Engine for Agile PLM Release 9.3.
---



## Generating and Initializing the Stubs

The Stub acts as a gateway for client side objects and all outgoing requests to server side objects that are routed through it. The stub wraps client object functionality and by adding the network logic ensures the reliable communication channel between client and server. The stub can be written up manually or generated automatically depending on chosen communication protocol.

For Agile Web Services to function successfully, you first need to create Agile PLM Server Stubs, and initialize the Client side Stubs.

### Generating Agile Stubs

Execute the Ant target **wsdl2java-Generate-Client**.

All the stubs are created in **src** folder.

### Initializing the Client Stubs

In the following sample, the generated stubs are being initialized for Business Object Web Services client. You may adapt it for other Web Services.

```
String SERVER_URL = "http://<host>:<port>/core/services/BusinessObject";
String USERNAME = "admin";
String PASSWORD = "agile";
BusinessObjectServiceLocator locator = new BusinessObjectServiceLocator();
BusinessObject_BindingStub businessObjectStub =
    (BusinessObject_BindingStub) locator.getBusinessObject(new java.net.URL(SERVER_URL));
((org.apache.axis.client.Stub)businessObjectStub).setUsername(USERNAME);
((org.apache.axis.client.Stub)businessObjectStub).setPassword(PASSWORD);
```

## Understanding the MessageElement

- A MessageElement is a part of a Request that specifies attributes of Agile Objects, which can use Agile API Names.
- Most MessageElements are String Type, while some can be Unit of Measure Type or AgileListEntryType.
- A MessageElement can be assigned any Tag Name. When API name is used as Tag Name, you need not pass the 'attributeld'.

```
MessageElement dataCell = new MessageElement(namespaceUri, "Numeric01");
dataCell.setObjectValue(9144.0);
```

- When API name is not used as Tag Name, attributeld has to be explicitly passed as an XML attribute having the name 'attributeld'

```
MessageElement dataCell = new MessageElement(namespaceUri, "key");
dataCell.setObjectValue(9144.0);
dataCell.addAttribute(namespaceUri, SchemaConstants.attributeId.getValue(), "AttributeID");
```

## Obtaining the API Names and Attribute IDs

To obtain the API Names, open the desired Class in Agile Java Client. You will find all the API Names in the General Information page or under the General Information tab.

## Special Handling of MessageElements

The following MessageElement Types require special handling, as described below.

### Unit of Measure

The attributes of an Agile object that require 'Unit of Measure' as an input type are updated with the UOM values. These values are denoted by the Unit of Measure object. The corresponding object is **AgileUnitOfMeasureType**.

To do this, you have to send the data as an instance of AgileUnitOfMeasureType. In addition, you need to pass the corresponding Namespace URI as an attribute.

The format is:

```
messageElement.addAttribute(PREFIX, NAMESPACEURI, "type", CLASSNAME);
```

You can choose any meaningful value for PREFIX. The type should be passed as "type".

You are required to send correct values for NAMESPACEURI and CLASSNAME. For example:

```
COMMONNAMESPACEURI = "http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
CLASSNAME = "AgileUnitOfMeasureType"
```

You can define your own namespaceUri or use COMMONNAMESPACEURI.

#### Example: MessageElement for a UOM

```
AgileUnitOfMeasureType uom = new AgileUnitOfMeasureType();
uom.setUnitName("Kilogram");
uom.setUnitValue(1000.0);
MessageElement dataCell=new MessageElement(namespaceUri, "mass");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, SchemaConstants.type.getValue(),
"AgileUnitOfMeasureType");
dataCell.setObjectValue(uom);
message[0] = dataCell;
```

## Multilist and List

The corresponding object is **AgileListEntryType**. You are required to pass the namespace attribute.

#### Example: MessageElement for a Multilist

```
AgileListEntryType multilist01 = new AgileListEntryType();
SelectionType[] multiSelect = new SelectionType[3];
multiSelect[0] = new SelectionType();
multiSelect[0].setValue("Canceled");
multiSelect[1] = new SelectionType();
multiSelect[1].setValue("Complete");
```

```
multiSelect[2] = new SelectionType();
multiSelect[2].setValue("Accepted");
multilist01.setSelection(multiSelect);
MessageElement dataCell = new MessageElement(namespaceUri, "multilist01");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, SchemaConstants.type.getValue(),
    "AgileListEntryType");
dataCell.setObjectValue(multilist01);
message[1] = dataCell;
```

### Example: MessageElement for a List

```
AgileListEntryType list01 = new AgileListEntryType();
SelectionType[] selection = new SelectionType[1];
selection[0] = new SelectionType();
selection[0].setValue("Alternate");
list01.setSelection(selection);
MessageElement dataCell = new MessageElement(namespaceUri, "list01");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, SchemaConstants.type.getValue(),
    "AgileListEntryType");
dataCell.setObjectValue(list01);
message[1] = dataCell;
```

## List of Objects

In certain cases, Agile SDK expects a list of **IDataObject** to be passed. For such cases, Agile Web Services use **AgileObjectListEntryType**.

```
AgileObjectListEntryType multilist01 = new AgileObjectListEntryType();
ObjectReferentIdType[] obj = new ObjectReferentIdType[1];
obj[0] = new ObjectReferentIdType();
obj[0].setClassIdentifier("8750");
obj[0].setObjectIdentifier("SAP0265");
multilist01.setSelection(obj);
MessageElement dataCell = new MessageElement(namespaceUri, "supplier");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, SchemaConstants.type.getValue(),
    "AgileObjectListEntryType");
dataCell.setObjectValue(multilist01);
message[0] = dataCell;
```

## Money

The corresponding object is **AgileMoneyType**. You are required to pass the *namespace* attribute.

```
AgileMoneyType money = new AgileMoneyType();
money.setAmount(997777.9);
money.setCurrency("USD");
MessageElement dataCell = new MessageElement(namespaceUri, "money01");
dataCell.setObjectValue(money);
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, SchemaConstants.type.getValue(),
    "AgileMoneyType");
message[0] = dataCell;
```

## Date

Java provides an object either as `Date` or as `Calendar` objects. You are required to pass the *namespace* attribute.

`Date` is a special case. Even though it is an XSD type, you must pass URI for `Date`.

### Example: MessageElement for Date

```
xsdnamespace = "http://www.w3.org/2001/XMLSchema"
MessageElement dataCell = new MessageElement(namespaceUri, "date01");
dataCell.addAttribute(XSIPREFIX, xsdnamespace, SchemaConstants.type.getValue(), "dateTime");
dataCell.setObjectValue(new Date());
message[0] = dataCell;
```

## User/Supplier/Customer/Analyst

The corresponding object is `ObjectReferentIdType`. You are required to pass the *namespace* attribute.

### Example: MessageElement for a User

```
ObjectReferentIdType user = new ObjectReferentIdType();
user.setObjectIdentifier("EMS1");
user.setClassIdentifier("supplier");
MessageElement dataCell=new MessageElement(namespaceUri, "supplier");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, SchemaConstants.type.getValue(),
"ObjectReferentIdType");
dataCell.setObjectValue(user);
message[0] = dataCell;
```

### Example: MessageElement for a Customer

```
ObjectReferentIdType customer = new ObjectReferentIdType();
customer.setObjectIdentifier("DEMO CUSTOMER 1");
customer.setClassIdentifier("customer");
MessageElement dataCell=new MessageElement(namespaceUri, "customer");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, SchemaConstants.type.getValue(),
"ObjectReferentIdType");
dataCell.setObjectValue(customer);
message[0] = dataCell;
```

---

**Note** Other values, like `String`, numbers etc can be passed as they are. However, numbers should not be passed as strings.

---

## Agile Attributes without API Names

The following attributes do not have an API name. You are required to use the Attribute IDs, listed below. These values have been picked from Agile SDK Constants. For more information on this, refer to *Agile SDK Developer Guide*.

### Item Constants

TABLE_REDLINEBOM	new Integer(-803);
TABLE_REDLINEMANUFACTURERS	new Integer(-1491);
TABLE_REDLINETITLEBLOCK	new Integer(-801);
TABLE_REDLINEPAGETWO	new Integer(-810);
TABLE_REDLINEPAGETHREE	new Integer(-1501);
FLAG_IS_REDLINE_MODIFIED	new Integer(-101);
FLAG_IS_REDLINE_REMOVED	new Integer(-102);
FLAG_IS_REDLINE_ADDED	new Integer(-103);

### User Constants

ATT_LOGIN_PASSWORD	new Integer(-1);
ATT_APPROVAL_PASSWORD	new Integer(-2);
ATT_SUPPLIER	new Integer(-3);
ATT_LOCALE	new Integer(-4);
ATT_TIMEZONE	new Integer(-5);
ATT_DATEFORMAT	new Integer(-6);
ATT_DATETIMEFORMAT	new Integer(-7);

## Understanding the Web Services Responses

### 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 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 will indicate PARTIAL\_SUCCESS.

## Exceptions and Warnings

When an operation is not successful, the system will throw an Exception or a Warning.

- In case of **FAILURE**, an exception is issued, while a warning may or may not 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 required to check it manually whether the operation was successful or not.

The Exceptions require code correction or system administration, while the Warnings can be resolved as described in [Working with Warnings](#) on page 21.

The response header for Web Services calls consists of a list of exceptions and warnings populated as **AgileExceptionListType** and **AgileWarningListType** objects. The application client must check for exceptions and warnings at all times to ensure that the code has performed all operations as intended.

The exception and warning lists contain a reference element 'id' which may be used to identify the corresponding requested in the batch that was the source of the exception(s) or warning(s).

Refer to the schema for dealing with response objects for a particular Web Service.

---

**Note** The code snippets provided in the first section of this document discuss only about the Request Objects. See [Core Operations Reference](#) on page 83 for complete sample code.

---

### Example: Getting Exceptions and Warnings

```
if( !approveObjectResponseType.getStatusCode().toString().equals(
ResponseStatusCode.SUCCESS.getValue() ) ){
    AgileExceptionListType[] agileExceptionListType =
approveObjectResponseType.getExceptions();
    if(agileExceptionListType!=null)
        for(int i=0; i<agileExceptionListType.length; i++){
            AgileExceptionType exceptions[] = agileExceptionListType[i].getException();
            for(int j=0; j<exceptions.length; j++){
                System.out.println("Exception Id:" + exceptions[j].getExceptionId() + "\nMessage: "
+
                exceptions[j].getMessage() );
            }
        }
    AgileWarningListType agileWarningListType[] = approveObjectResponseType.getWarnings();
    if(agileWarningListType!=null)
        for(int i=0; i<agileWarningListType.length; i++){
            AgileWarningType warnings[] = agileWarningListType[i].getWarning();
```

```
for(int j=0; j<warnings.length; j++)
    System.out.println("Warning Id: " + warnings[j].getWarningId() + "\nMessage: " +
        warnings[j].getMessage() );
}
```

## Working with Warnings

---

**Note** By default, all warnings are enabled.

---

You can work with warnings in following ways:

- Use **setWarningResolution** to selectively Enable or Disable a select set of warnings.

```
AgileWarningResolutionType warningRes[] = new AgileWarningResolutionType[1];
warningRes[0] = new AgileWarningResolutionType();
warningRes[0].setId(182);
warningRes[0].setResolution(AgileWarningResolutionConstantsType.DISABLE);
approveRObjectRequestType.setWarningResolution(warningRes);
```

- Use **diasableAllWarnings** function to disable ALL the warnings

```
approveRObjectRequestType.setDisableAllWarnings();
```

- Enable a select set of warnings and disable the rest with a combination of **diasableAllWarnings** and **setWarningResolution**.

```
approveRObjectRequestType.setDisableAllWarnings();
AgileWarningResolutionType warningRes[] = new AgileWarningResolutionType[1];
warningRes[0] = new AgileWarningResolutionType();
warningRes[0].setId(182);
warningRes[0].setResolution(AgileWarningResolutionConstantsType.ENABLE);
approveRObjectRequestType.setWarningResolution(warningRes);
```





# Working with Business Objects

This chapter includes the following:

▪ Getting an Object.....	23
▪ Creating an Object.....	24
▪ Saving As a New Object.....	25
▪ Deleting and Undeleting an Object.....	26
▪ Updating an Object.....	28
▪ Getting the Status of an Object.....	28
▪ Getting the AutoNumbers.....	28
▪ Getting the Classes.....	29
▪ Getting the Subclasses.....	29

This chapter describes how to work with the Agile PLM Business Objects and provides sample code snippets.

## Getting an Object

To get an Agile PLM object, use the getObject operation. This operation lets you specify the `objectType` and `objectNumber` parameters.

An `objectType` is the API name or ID of a Subclass. For example, a 'Part' is an `objectType` of the Agile Class 'Item'; ECO is an `objectType` of the Agile Class 'Change'. An `objectNumber` is number of the Agile Object being retrieved.

The actual information about the object is obtained through the response in the form of `AgileObjectType` objects.

Use the following syntax to get an object by specifying `objectType` and `objectNumber` parameters, as shown in examples.

```
GetObjectRequestType getObjectRequestType = new GetObjectRequestType();
AgileGetObjectRequest agileGetObjectRequest[] = new AgileGetObjectRequest[1];
agileGetObjectRequest[0] = new AgileGetObjectRequest();
agileGetObjectRequest[0].setClassIdentifier("objectType");
agileGetObjectRequest[0].setObjectNumber("objectNumber");
```

### Example: Getting an Item

```
GetObjectRequestType getObjectRequestType = new GetObjectRequestType();
AgileGetObjectRequest agileGetObjectRequest[] = new AgileGetObjectRequest[1];
agileGetObjectRequest[0] = new AgileGetObjectRequest();
agileGetObjectRequest[0].setClassIdentifier("Part");
agileGetObjectRequest[0].setObjectNumber("P00001");
```

**Example: Getting a Change**

```
GetObjectRequestType getObjectRequestType = new GetObjectRequestType();
AgileGetObjectRequest agileGetObjectRequest[] = new AgileGetObjectRequest[1];
agileGetObjectRequest[0] = new AgileGetObjectRequest();
agileGetObjectRequest[0].setClassIdentifier("ECO");
agileGetObjectRequest[0].setObjectNumber("ECO-0001");
```

## Special Handling in the getObject Operation

To get certain Agile Objects, the getObject operation requires an additional parameter to be set. This parameter, `setOptions(propertyType)`, accepts a name-value pair - **propertyName** and **propertyValue**.

**Example: Getting an Manufacturer Part**

```
GetObjectRequestType getObjectRequestType = new GetObjectRequestType();
AgileGetObjectRequest agileGetObjectRequest[] = new AgileGetObjectRequest[1];
agileGetObjectRequest[0] = new AgileGetObjectRequest();
agileGetObjectRequest[0].setClassIdentifier("ManufacturerPart");
agileGetObjectRequest[0].setObjectNumber("manufPartNumber");
PropertyType[] propertyType = new PropertyType[1];
propertyType[0] = new PropertyType();
propertyType[0].setPropertyName( SchemaConstants.manufacturer_name.getValue() );
propertyType[0].setPropertyValue(manufName);
agileGetObjectRequest[0].setOptions(propertyType);
```

## Creating an Object

To create a new Agile PLM object, use the createObject operation. This operation requires you to specify the **objectType** parameter, for example, a *Part*.

**Example: Creating a Part**

```
CreateObjectRequestType createObjectRequestType = new CreateObjectRequestType();
AgileCreateObjectRequest agileCreateObjectRequest[] = new AgileCreateObjectRequest[1];
agileCreateObjectRequest[0].setClassIdentifier("Document");
AgileRowType row_1 = new AgileRowType();
MessageElement messages_1[] = new MessageElement[2];
String namespaceUri = null;
messages_1[0] = new MessageElement(namespaceUri, "number");
messages_1[0].addTextNode(documentNumber);
messages_1[1] = new MessageElement(namespaceUri, "description");
messages_1[1].addTextNode("Doc Desc");
```

---

<b>Note</b>	Agile Web Services do not support setting the Life Cycle Phase (LCP)/workflow status attribute of an object while you are creating that object. This is because the necessary settings for LCP are not available until the object is created.
-------------	---

---

## Saving As a New Object

You can save an existing Agile Object as a new object by using the `saveAsObject` operation. This operation calls `AgileSaveAsObjectRequestType`, which requires the values of `objectName`, `objectNumber` and `newObjectName`.

You can specify a `newObjectNumber` as a `MessageElement` to generate number for a new object, as shown in the following syntax:

```
SaveAsObjectRequestType saveAsObjectRequestType = new SaveAsObjectRequestType();
AgileSaveAsObjectRequestType agileSaveAsObjectRequestType[] = new
AgileSaveAsObjectRequestType[1];
agileSaveAsObjectRequestType[0].setParentClassIdentifier("objectName");
agileSaveAsObjectRequestType[0].setParentObjectNumber("objectNumber");
agileSaveAsObjectRequestType[0].setNewClassIdentifier("newObjectName");
    AgileRowType row = new AgileRowType();
    MessageElement messages[] = new MessageElement[1];
    String namespaceUri = null;
    messages[0] = new MessageElement(namespaceUri, "TagName");
    messages[0].addAttribute(namespaceUri, "attributeId", "Attribute ID" );
    messages[0].addTextNode(newObjectNumber);
    row.set_any(messages);
    agileSaveAsObjectRequestType[0].setData(row);
```

You can also use `autoNumberSource` to generate number for a new object, using the following syntax:

```
SaveAsObjectRequestType saveAsObjectRequestType = new SaveAsObjectRequestType();
AgileSaveAsObjectRequestType agileSaveAsObjectRequestType[] = new
AgileSaveAsObjectRequestType[1];
agileSaveAsObjectRequestType[0].setParentClassIdentifier("objectName");
agileSaveAsObjectRequestType[0].setParentObjectNumber("objectNumber");
agileSaveAsObjectRequestType[0].setNewClassIdentifier("newObjectName");
agileSaveAsObjectRequestType[0].setAutoNumberSource("autoNumberSource");
```

---

**Note** See [Getting AutoNumbers](#) on page 28 for more details.

---

### Example: Saving a Part As a New Part

```
SaveAsObjectRequestType saveAsObjectRequestType = new SaveAsObjectRequestType();
AgileSaveAsObjectRequestType agileSaveAsObjectRequestType[] = new
AgileSaveAsObjectRequestType[1];
agileSaveAsObjectRequestType[0].setParentClassIdentifier("Part");
agileSaveAsObjectRequestType[0].setParentObjectNumber(partNumber1);
agileSaveAsObjectRequestType[0].setNewClassIdentifier("Part");
    AgileRowType row = new AgileRowType();
    MessageElement messages[] = new MessageElement[1];
    String namespaceUri = null;
    messages[0] = new MessageElement(namespaceUri, "Message_Num");
    messages[0].addTextNode(newPartNumber);
    row.set_any(messages);
    agileSaveAsObjectRequestType[0].setData(row);
```

## Special Handling in the saveAsObject Operation

In case of saving an object as a **Program**, you need to specify the `TemplateType` - *Active*, *Template* or *Proposed*. Optionally, you can also pass additional attributes, such as Scheduled Start Data, Tables to be copied, Apply to children, etc.

---

**Note** The default `TemplateType` of a Program is *Active*.

---

### Example: Saving an Object as a Program of type Template

```
SaveAsObjectRequestType saveAsObjectRequestType = new SaveAsObjectRequestType();
AgileSaveAsObjectRequestType agileSaveAsObjectRequestType[] = new
AgileSaveAsObjectRequestType[1];
agileSaveAsObjectRequestType[0] = new AgileSaveAsObjectRequestType();
agileSaveAsObjectRequestType[0].setParentClassIdentifier("Program");
agileSaveAsObjectRequestType[0].setParentObjectNumber( parentProgramNumber );
agileSaveAsObjectRequestType[0].setNewClassIdentifier("Program");
AgileRowType row = new AgileRowType();
    MessageElement messages[] = new MessageElement[1];
    String namespaceUri = null;
    messages[0] = new MessageElement(namespaceUri, "name");
    messages[0].addTextNode(newProgramNumber);
    row.set_any(messages);
    agileSaveAsObjectRequestType[0].setData(row);
        PropertyType properties[] = new PropertyType[1];
        properties[0] = new PropertyType();
        properties[0].setPropertyName( SchemaConstants.program_template.getValue() );
        properties[0].setPropertyValue("Template");
        agileSaveAsObjectRequestType[0].setOptions(properties);
    String tables[] = {"PageTwo", "Team"};
    agileSaveAsObjectRequestType[0].setTablesToCopy(tables);
    agileSaveAsObjectRequestType[0].setApplyToChildren(true);
```

When you create a program, you can specify that it is a template by setting the value of the Template attribute to "Template". You can do this only when you create a program or when you save it as a new program. Existing programs cannot be changed from the "Active" or "Proposed" state to "Template".

## Deleting and Undeleting an Object

The deletion of an object in Agile is of two types - soft delete and hard delete.

With soft delete, which is carried out using the operation `deleteObject`, the object is marked as 'Deleted'. It is however not removed from the database, so that it can be restored using the operation `undeleteObject`. A soft-deleted object does not appear in search results however with the operation `isDeletedObject` you can find these deleted objects.

With hard delete, the object is removed from the database permanently. These objects do not appear in search queries or pre-defined query results.

---

**Note** To delete and undelete an object, you must have Delete and Undelete privileges, respectively, for the particular object type. However, soft-deleted changes that have items on the Affected Items tab cannot be restored, regardless of the user's privileges.

---

Not all Agile PLM objects can be deleted. If you attempt to delete these objects, the deleteObject operation throws an exception. Also, if you try to delete an Item that is used on the BOM tab of another item, the Agile PLM server throws an exception.

Some of the objects that cannot be deleted are:

- An item with a pending change
- An item with a revision history
- An item with a canceled change
- A released change
- A manufacturer with one or more manufacturer parts
- A manufacturer part currently used on the Manufacturers tab of another object

#### Example: Deleting a Part

```
DeleteObjectRequestType deleteObjectRequestType = new DeleteObjectRequestType();
AgileDeleteObjectRequest agileDeleteObjectRequest[] = new AgileDeleteObjectRequest[1];
agileDeleteObjectRequest[0] = new AgileDeleteObjectRequest();
agileDeleteObjectRequest[0].setClassIdentifier("Part");
agileDeleteObjectRequest[0].setObjectNumber(partNumber);
```

#### Example: Undeleting a Part

```
UndeleteObjectRequestType undeleteObjectRequestType = new UndeleteObjectRequestType();
AgileUndeleteObjectRequest agileUndeleteObjectRequest[] = new AgileUndeleteObjectRequest[1];
agileUndeleteObjectRequest[0] = new AgileUndeleteObjectRequest();
agileUndeleteObjectRequest[0].setClassIdentifier("Part");
agileUndeleteObjectRequest[0].setObjectNumber(partNumber);
```

## Checking the Delete Status

You can verify whether an Object has been deleted or not by using the isDeletedObject operation.

#### Example: Checking if a Part is deleted

```
IsDeletedObjectRequestType isDeletedObjectRequestType = new IsDeletedObjectRequestType();
AgileIsDeletedObjectRequest agileIsDeletedObjectRequest[] = new
AgileIsDeletedObjectRequest[1];
agileIsDeletedObjectRequest[0] = new AgileIsDeletedObjectRequest();
agileIsDeletedObjectRequest[0].setClassIdentifier("Part");
agileIsDeletedObjectRequest[0].setObjectNumber(partNumber);
```

## Updating an Object

You can update any object with the operation `updateObject` by setting the values of the desired attributes.

### Example: Updating a Part

```
UpdateObjectRequestType updateObjectRequestType = new UpdateObjectRequestType();
AgileUpdateObjectRequest agileUpdateObjectRequest[] = new AgileUpdateObjectRequest[1];
agileUpdateObjectRequest[0] = new AgileUpdateObjectRequest();
agileUpdateObjectRequest[0].setClassIdentifier("Part");
agileUpdateObjectRequest[0].setObjectNumber(partNumber);
    AgileRowType rows = new AgileRowType();
    MessageElement messages[] = new MessageElement[1];
    String namespaceUri = null;
    messages[0] = new MessageElement(namespaceUri, "Message_Desc");
    messages[0].addTextNode("Updated value of Doc Description");
    rows.set_any(messages);
    agileUpdateObjectRequest[0].setData(rows);
```

## Getting the Status of an Object

In a workflow or a lifecycle, a routable object passes through various states. Subsequent action on this object requires ascertaining its current state, the states it has already crossed and the states it must go through. This information is obtained using the operation `getStatus`.

The response object will consist of `AgileStatusType` objects for `nextDefaultStatus`, `nextValidStatuses`, `currentStatus`. For complete details, refer Schema Documentation at Oracle eDelivery Site.

### Example: Getting the status of an ECO

```
GetStatusRequestType getStatusRequestType = new GetStatusRequestType();
AgileGetStatusRequestType agileGetStatusRequestType[] = new AgileGetStatusRequestType[1];
agileGetStatusRequestType[0] = new AgileGetStatusRequestType();
agileGetStatusRequestType[0].setClassIdentifier("ECO");
agileGetStatusRequestType[0].setObjectNumber( changeNumber );
```

## Getting the AutoNumbers

An `AutoNumber` source is a predefined sequence of numbers that automatically assign a number to an object. An Agile PLM class can have one or more `AutoNumber` sources. These are defined in the Admin node of the Agile Java Client.

To get a 'next in sequence' `AutoNumber`, specify the `autoNumberSource` and `objectType` attributes in the operation `getAutoNumbers`.

---

**Note** The Manufacturers and Manufacturer Parts classes, and their user-defined subclasses, do not support automatic numbering.

---

### Example: Getting Autonumbers for Part Class

```
GetAutoNumbersRequestType getAutoNumbersRequestType = new GetAutoNumbersRequestType();
AgileGetAutoNumbersRequestType agileGetAutoNumbersRequestType[] = new
AgileGetAutoNumbersRequestType[2];
agileGetAutoNumbersRequestType[0].setClassIdentifier("Part");
agileGetAutoNumbersRequestType[0].setAutoNumberIdentifier( new String[]{"PartNumber"} );
agileGetAutoNumbersRequestType[0].setSize(3);
    agileGetAutoNumbersRequestType[1].setClassIdentifier("ECO");
    agileGetAutoNumbersRequestType[1].setIncludeAllAutoNumberSource(true);
    agileGetAutoNumbersRequestType[1].setSize(2);
getAutoNumbersRequestType.setRequests(agileGetAutoNumbersRequestType);
```

## Getting the Classes

You can retrieve the classes for each object type with the operation `getAllClasses`. Your program can then provide a method to pick the desired class from the list.

You can specify the `ClassFilterType` as follows:

- ClassFilterType.ALL** - To retrieve all the classes and their subclasses.
- ClassFilterType.TOP** - To retrieve all the classes only
- ClassFilterType.CONCRETE** - To retrieve all the subclasses only

The syntax for this operation is:

```
GetAllClassesRequestType getAllClassesRequestType = new GetAllClassesRequestType();
getAllClassesRequestType.setLevel(ClassFilterType.ALL);
```

## Getting the Subclasses

Although you can retrieve all Subclasses by using the operation `getAllClasses` with `ClassFilterType` filter, you may require Subclasses of a particular Class. This can be achieved by using the operation `getSubClasses`, in which, you can specify the Agile API name of the desired Class.

`ClassType` objects are obtained from the response.

### Example: Getting all Subclasses of the Class 'Changes'

```
GetSubClassesRequestType getSubClassesRequestType = new GetSubClassesRequestType();
AgileGetSubClassesRequestType agileGetSubClassesRequestType[] = new
AgileGetSubClassesRequestType[1];
agileGetSubClassesRequestType[0].setClassIdentifier("Changes");
```





## Working with Tables

This chapter includes the following:

▪ Operations Supported on Tables.....	31
▪ Loading a Table .....	33
▪ Working with the Readonly Tables .....	35
▪ Retrieving the Metadata of a Table.....	35
▪ Adding Rows to a Table.....	35
▪ Updating Rows in a Table.....	40
▪ Removing Rows from a Table .....	41
▪ Clearing a Table .....	41
▪ Copying Tables.....	42
▪ Redlining a Table.....	42

This chapter describes how to work with the Agile PLM Table and provides sample code snippets.

### About Tables

Agile data is contained in tables. In Agile Web Client, these tables are equivalent to the separate tabs in a window, such as the Manufacturers and BOM tabs.

The Agile Web Services do not support random access of rows to a table. This means that you cannot retrieve a specific row by index number and then update it.

### Operations Supported on Tables

Web Services supports table operations for Agile PLM's PC and PQM solutions.

#### PC

Table Name	Objects	Web Services API
Item Changes Pending Changes	Item	loadTable, isReadOnlyTable
Item Changes Change History	Item	loadTable, isReadOnlyTable
Item BOM	Item	loadTable, copyTable, clearTable, isReadOnlyTable, addRows, removeRows, updateRows
Item Manufacturers	Item	loadTable, copyTable, clearTable, isReadOnlyTable, addRows, removeRows, updateRows

Table Name	Objects	Web Services API
Item Sites	Item	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Item Prices	Item	loadTable, isReadonlyTable
Item Quality	Item	loadTable, isReadonlyTable
Item Compliance Compositions	Item	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Item Compliance Substances	Item	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Item Compliance Specifications	Item	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Item Relationships	Item	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Item Where Used	Item	loadTable, isReadonlyTable
Changes Affected Items Table	Changes	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Changes AI Redline Title Block	Changes	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Changes AI Redline BOM	Changes	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Changes AI Redline Manufacturers	Changes	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Changes AI Redline Attachments	Changes	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Changes Relationships	Changes	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Mfrs Relationships	Mfrs	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Mfrs Where Used	Mfrs	loadTable, isReadonlyTable
Mfr Parts Prices	Mfrs	loadTable, isReadonlyTable
Mfr Parts Compliance Compositions	Mfr Parts	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Mfr Parts Compliance Substances	Mfr Parts	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Mfr Parts Compliance Specifications	Mfr Parts	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Mfr Parts Suppliers	Mfr Parts	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows

Table Name	Objects	Web Services API
Mfr Parts Relationships	Mfr Parts	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Mfr Parts Where Used	Mfr Parts	loadTable, isReadonlyTable
Sites Relationships	Sites	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
Sites History	Sites	loadTable, isReadonlyTable

**PQM**

Table Name	Objects	Web Services API
PSR Affected Items	PSR	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
PSR Related PSR	PSR	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
PSR Relationships	PSR	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
QCR Affected Items	QCR	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows
QCR Relationships	QCR	loadTable, copyTable, clearTable, isReadonlyTable, addRows, removeRows, updateRows

## Loading a Table

You can use the operation `loadTable` to load a table from Agile PLM system. This operation takes `tablesIdentifier` parameter along with `classIdentifier` and `objectIdentifier`.

Tables vary for each Agile PLM dataobject. Tables for change objects are different from tables for items. Each table for a particular dataobject is identified by a constant in the constants class or by the API name for that dataobject. Item constants are contained in the `ItemConstants` class, change constants are contained in the `ChangeConstants` class, and so on.

### Example: Loading the Table of a Part

```
RequestTableType table[] = new RequestTableType[1];
table[0] = new RequestTableType();
table[0].setClassIdentifier("Part");
table[0].setObjectNumber( partNumber );
table[0].setTableIdentifier("table01");
loadTableRequestType.setTableRequest(table);
```

## Special Handling in the loadTable Operation

### Example: Loading a Table for an Object Version

```
table[0] = new RequestTableType();
table[0].setClassIdentifier("FileFolder");
table[0].setObjectNumber( folderNumber );
table[0].setTableIdentifier("Files");
    PropertyType properties[] = new PropertyType[1];
    properties[0] = new PropertyType();
    properties[0].setPropertyName( SchemaConstants.folderVersion.getValue() );
    properties[0].setPropertyValue( folderVersion );
    table[0].setOptions(properties);
    loadTableRequestType.setTableRequest(table);
```

### Example: Loading a Table for an Object Revision

```
table[0] = new RequestTableType();
table[0].setClassIdentifier( "Part" );
table[0].setObjectNumber( partNumber );
table[0].setTableIdentifier("TitleBlock");
    PropertyType properties[] = new PropertyType[1];
    properties[0] = new PropertyType();
    properties[0].setPropertyName( SchemaConstants.revision.getValue() );
    properties[0].setPropertyValue( partVersion );
    table[0].setOptions(properties);
    loadTableRequestType.setTableRequest(table);
```

### Example: Loading a Table for a Redline Change

```
table[0] = new RequestTableType();
table[0].setClassIdentifier( "Part" );
table[0].setObjectNumber( partNumber );
table[0].setTableIdentifier("TitleBlock");
    PropertyType properties[] = new PropertyType[1];
    properties[0] = new PropertyType();
    properties[0].setPropertyName(SchemaConstants.redline_change.getValue() );
    properties[0].setPropertyValue( changeNumber );
    table[0].setOptions(properties);
    loadTableRequestType.setTableRequest(table);
```

### Example: Loading a Table for a Site Object

```
table[0] = new RequestTableType();
table[0].setClassIdentifier( "Part" );
table[0].setObjectNumber( parentPartNumber );
table[0].setTableIdentifier("BOM" );
    PropertyType properties[] = new PropertyType[1];
    properties[0] = new PropertyType();
    properties[0].setPropertyName( SchemaConstants.site.getValue() );
    properties[0].setPropertyValue( sitel );
    table[0].setOptions(properties);
```

```
loadTableRequestType.setTableRequest(table);
```

## Working with the Readonly Tables

Several Agile PLM tables store history information or data about related objects. These tables are read-only and as such, you cannot modify these tables. When you write code to access a table, use the operation `isReadOnlyTable` to check if the table is read-only.

## Retrieving the Metadata of a Table

You may require the metadata information of a table, which is the underlying data that describes a table's properties. This is useful when you need to identify the attributes of a particular table, its ID, or its table name without having to load a dataobject. The metadata is obtained in the form of `AttributeType` objects from the response.

For this, use the operation `getTableMetadata` specifying the `tableIdentifier` and `classIdentifier`.

### Example: Retrieving Metadata of a Table

```
agileGetTableMetadataRequestType[0].setClassIdentifier("Part");
agileGetTableMetadataRequestType[0].setTableIdentifier("table01");
getTableMetadataRequestType.setRequests(agileGetTableMetadataRequestType);
```

## Adding Rows to a Table

To create a table row, use the operation `addRows`, which creates a new row and initializes it with the data specified in the `rows` parameter. The `rows` parameter of `addRows` is available to pass the following data:

- a set of attributes and values for the row's cells
- an Agile PLM object (such as an `Item`) to add to the table

When you add a row to a table, it is not necessarily added at the end of the table.

---

**Note** You cannot add an empty row to a table.

---

### Example: Adding Rows in a BOM Table

With the `addRows` operation, you can add a child element to a `Part` by adding rows to the BOM table of the parent object.

```
RequestTableType table = new RequestTableType();
table.setClassIdentifier("Part");
table.setObjectNumber(parentPartNumber);
table.setTableIdentifier("BOMtable_API_Name");
AgileRowType[] rows = new AgileRowType[1];
rows[0] = new AgileRowType();
    String namespaceUri = null;
```

```
MessageElement messages[] = new MessageElement[1];
rows[0].set_any(messages);
messages[0] = new MessageElement(namespaceUri, "BOM_Child_Number");
messages[0].addAttribute(namespaceUri, "itemNumber");
messages[0].addTextNode( BOMchildPartNumber );
agileAddRowsRequest[0].setRow(rows);
agileAddRowsRequest[0].setObjectInfo(table);
addRowsRequestType.setData(agileAddRowsRequest);
```

## Special Handling in the addRows Operation

---

**Note** All additional attributes like revision, site etc should be passed as options. Site and Revision should be passed along the individual row.

---

### Adding a Site to the Sites Tab of an Item

```
final String COMMONNAMESPACEURI = "http://xmlns.oracle.com/AgileObjects/Core/Common/V1";
final String XSIPREFIX = org.apache.axis.Constants.NS_PREFIX_SCHEMA_XSI;
final String TYPE = SchemaConstants.type.getValue();
AgileListEntryType lst03 = new AgileListEntryType();
SelectionType[] multiSelect = new SelectionType[1];
multiSelect[0] = new SelectionType();
multiSelect[0].setValue("Bangalore");
lst03.setSelection(multiSelect);
MessageElement dataCell = new MessageElement(namespaceUri, "siteName");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, TYPE, "AgileListEntryType");
dataCell.setObjectValue(lst03);
message[0] = dataCell;
row[0].set_any(message);
```

### Adding Suppliers to the Suppliers Tab of an Item

You can add suppliers to the suppliers tab of an item using the following two methods:

#### Method 1

```
final String COMMONNAMESPACEURI = "http://xmlns.oracle.com/AgileObjects/Core/Common/V1";
final String XSIPREFIX = org.apache.axis.Constants.NS_PREFIX_SCHEMA_XSI;
final String TYPE = SchemaConstants.type.getValue();
ObjectReferentIdType multiSelect = new ObjectReferentIdType();
multiSelect.setClassIdentifier("Broker");
multiSelect.setObjectIdentifier("SAP0265");
MessageElement dataCell = new MessageElement(namespaceUri, "supplier01");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, TYPE, "ObjectReferentIdType");
dataCell.setObjectValue(multiSelect);
message[0] = dataCell;
row[0].set_any(message);
```

#### Method 2

```

final String COMMONNAMESPACEURI = "http://xmlns.oracle.com/AgileObjects/Core/Common/V1";
final String XSIPREFIX = org.apache.axis.Constants.NS_PREFIX_SCHEMA_XSI;
final String TYPE = SchemaConstants.type.getValue();
    AgileObjectListEntryType multilist01 = new AgileObjectListEntryType();
    ObjectReferentIdType[] obj = new ObjectReferentIdType[1];
    obj[0] = new ObjectReferentIdType();
    obj[0].setClassIdentifier("Broker");
    obj[0].setObjectIdentifier("SAP0265");
    multilist01.setSelection(obj);
    MessageElement dataCell = new MessageElement(namespaceUri, "supplier");
    dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, TYPE, "AgileObjectListEntryType");
    dataCell.setObjectValue(multilist01);
    message[0] = dataCell;
    row[0].set_any(message);

```

## Adding Suppliers to a Manufacturer Part

```

final String COMMONNAMESPACEURI = "http://xmlns.oracle.com/AgileObjects/Core/Common/V1";
final String XSIPREFIX = org.apache.axis.Constants.NS_PREFIX_SCHEMA_XSI;
final String TYPE = SchemaConstants.type.getValue();
    RequestTableType objectInfo = new RequestTableType();
    objectInfo.setClassIdentifier(subclassId);
    objectInfo.setObjectNumber(objectNumber);
    objectInfo.setTableIdentifier("tableId");
    agileAddRowsRequests[0].setObjectInfo(objectInfo);
    PropertyType[] options = new PropertyType[1];
    options[0] = new PropertyType();
    options[0].setPropertyName(SchemaConstants.manufacturer_name.getValue());
    options[0].setPropertyValue("Cisco");
        agileAddRowsRequests[0].setOptions(options);
        AgileAddRowsRequest[] agileAddRowsRequests = new AgileAddRowsRequest[1];
        agileAddRowsRequests[0] = new AgileAddRowsRequest();
        AgileRowType[] row = new AgileRowType[1];
        row[0] = new AgileRowType();
        agileAddRowsRequest.setRow(row);
        String namespaceUri = null;
    MessageElement[] message = new MessageElement[1];
    AgileObjectListEntryType multilist01 = new AgileObjectListEntryType();
    ObjectReferentIdType[] obj = new ObjectReferentIdType[1];
    obj[0] = new ObjectReferentIdType();
    obj[0].setClassIdentifier("Broker");
    obj[0].setObjectIdentifier("SAP0265");
    multilist01.setSelection(obj);
    MessageElement dataCell = new MessageElement(namespaceUri, "supplier");
    dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, TYPE, "AgileObjectListEntryType");
    dataCell.setObjectValue(multilist01);
    message[0] = dataCell;
    row[0].set_any(message);

```

## Adding Manufacturer Part to AML of an Item

You can add a Manufacturer Part to the AML of an Item using the following two methods:

### Method 1

```
MessageElement[] message = new MessageElement[2];
MessageElement dataCell = new MessageElement(namespaceUri, "mfrPartNumber");
dataCell.setObjectValue("bosco");
message[0] = dataCell;
MessageElement dataCell = new MessageElement(namespaceUri, "mfrName");
dataCell.setObjectValue("cisco");
message[1] = dataCell;
row[0].set_any(message);
```

### Method 2

```
MessageElement[] message = new MessageElement[1];
ObjectReferentIdType obj = new ObjectReferentIdType();
obj.setClassIdentifier("ManufacturerPart");
obj.setObjectIdentifier("MfrP_01");
PropertyType[] options = new PropertyType[1];
options[0] = new PropertyType();
options[0].setPropertyName(SchemaConstants.manufacturer_name.getValue());
options[0].setPropertyValue("Manu_4570");
obj.setOptions(options);
MessageElement dataCell = new MessageElement(namespaceUri, "mfrPartNumber");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, TYPE, "ObjectReferentIdType");
dataCell.setObjectValue(obj);
message[0] = dataCell;
row[0].set_any(message);
```

## Adding Manufacturer Part to the Relationships Tab

```
ObjectReferentIdType obj = new ObjectReferentIdType();
obj.setClassIdentifier("ManufacturerPart");
obj.setObjectIdentifier("m12444");
PropertyType[] options = new PropertyType[1];
options[0] = new PropertyType();
options[0].setPropertyName(SchemaConstants.manufacturer_name.getValue());
options[0].setPropertyValue("Cisco");
obj.setOptions(options);
MessageElement dataCell = new MessageElement(namespaceUri, "name");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, TYPE, "ObjectReferentIdType");
dataCell.setObjectValue(obj);
message[0] = dataCell;
row[0].set_any(message);
```

## Adding Affected Item to a Change

```
MessageElement dataCell1 = new MessageElement(namespaceUri, "itemNumber");
```



```

dataCell11.setObjectValue("P00400");
message[0] = dataCell11;

MessageElement dataCell2 = new MessageElement(namespaceUri, "effectiveDate");
dataCell2.setObjectValue(new Date());
dataCell2.addAttribute(XSIPREFIX, Constants.URI_DEFAULT_SCHEMA_XSD, "type", "dateTime");
message[1] = dataCell2;
MessageElement dataCell3 = new MessageElement(namespaceUri, "newRev");
dataCell3.setObjectValue("Item_01"); dataCell3.addAttribute(XSIPREFIX,
Constants.URI_DEFAULT_SCHEMA_XSD, TYPE, "string"); message[2] = dataCell3;
row[0].set_any(message);

```

## Adding Site Specific Item to the BOM Tab

To add a child object to a specific site, we can utilize either the `setOptions` feature on the table object, or use the `setAdditionalRowInfo` method on the row object. Using `setOptions` on the table object will add all new rows to a particular site. On the other hand, `setAdditionalRowInfo` may be used to specify a site for each individual row, meaning that if several rows are to be added with a web service call, each row may be added to a different site.

### Example: Adding Site Specific Item to the BOM Tab

In this example, `setAdditionalRowInfo` is used to add a given row to a specific site of a Part using the operation `addRows`.

```

RequestTableType table = new RequestTableType();
table.setClassIdentifier("Part");
table.setObjectNumber( parentPartNumber );
table.setTableIdentifier( "BOM" );
AgileRowType[] rows = new AgileRowType[1];
rows[0] = new AgileRowType();
String namespaceUri = null;
MessageElement messages[] = new MessageElement[1];
    rows[0].set_any(messages);
    messages[0] = new MessageElement(namespaceUri, "itemNumber");
    messages[0].addTextNode( childPartNumber );
    AdditionalInfoType additionalInfoType = new AdditionalInfoType();
    additionalInfoType.setSite(site1);
    rows[0].setAdditionalRowInfo(additionalInfoType);
    agileAddRowsRequest[0].setRow(rows);
    agileAddRowsRequest[0].setObjectInfo(table);
    addRowsRequestType.setData(agileAddRowsRequest);

```

## Adding Site Specific AML to the Manufacturers Tab

For adding a Manufacturer to a Part at a specific site, use the `setOptions` feature by providing a name-value pair using which a particular site is identified. Subsequently, the web service adds the manufacturer to the site as specified in the options.

A manufacturer part is specified through a `MessageElement` in the operation `addRows`, the message element cannot be specified in the usual manner. In this case, the message element for the manufacturer part must be of type `ObjectReferentIdType`. Consequently, an object identifier type object is created and appropriate class and object identifier values are set, using the

manufacturer part class and its number, respectively.

```
RequestTableType table = new RequestTableType();
table.setClassIdentifier("Part");
table.setObjectNumber( partNumber );
table.setTableIdentifier( "Manufacturers" );
PropertyType[] properties = new PropertyType[1];
properties[0] = new PropertyType();
properties[0].setPropertyName( SchemaConstants.site.getValue() );
properties[0].setPropertyValue( sitel );
table.setOptions(properties);
AgileRowType[] rows = new AgileRowType[1];
rows[0] = new AgileRowType();

String namespaceUri = null;
MessageElement messages[] = new MessageElement[1];
messages[0] = new MessageElement(namespaceUri, "mfrPartNumber");
ObjectReferentIdType objRefId = new ObjectReferentIdType();
objRefId.setClassIdentifier( "ManufacturerPart" );
objRefId.setObjectIdentifier( manufPartNumber );
messages[0].addAttribute(Constants.NS_PREFIX_SCHEMA_XSI, COMMONNAMESPACEURI, "type",
"ObjectReferentIdType");
PropertyType[] properties_manufName = new PropertyType[1];
properties_manufName[0] = new PropertyType();
properties_manufName[0].setPropertyName( SchemaConstants.manufacturer_name.getValue() );
properties_manufName[0].setPropertyValue( manufName );
objRefId.setOptions(properties_manufName);
messages[0].setObjectValue(objRefId);

rows[0].set_any(messages);
agileAddRowsRequest[0].setRow(rows);
agileAddRowsRequest[0].setObjectInfo(table);
addRowsRequestType.setData(agileAddRowsRequest);
```

## Updating Rows in a Table

Rows in a table are updated using the operation `updateRows`.

In the following example, the `rowID` is set after performing the operation `loadTable` and getting the `rowID` from the response.

```
UpdateRowsRequestType updateRowsRequestType = new UpdateRowsRequestType();
AgileUpdateRowsRequest agileUpdateRowsRequest[] = new AgileUpdateRowsRequest[1];
agileUpdateRowsRequest[0] = new AgileUpdateRowsRequest();
RequestTableType table = new RequestTableType();
table.setClassIdentifier( "ECO" );
table.setObjectNumber( changeNumber );
table.setTableIdentifier("AffectedItems" );
    AgileUpdateRow updateRow[] = new AgileUpdateRow[1];
    updateRow[0] = new AgileUpdateRow();
    updateRow[0].setRowId(getRowID("ECO", changeNumber, "AffectedItems", partNumber ) );
    AgileRowType row = new AgileRowType();
    String namespaceUri = null;
```

```

MessageElement messages[] = new MessageElement[1];
    Date date = new Date();
    date.setTime( date.getTime() );
    messages[0] = new MessageElement(namespaceUri, "effectiveDate");
    messages[0].addAttribute("date_px", Constants.URI_DEFAULT_SCHEMA_XSD, "type",
        "dateTime");
    messages[0].setObjectValue(date);
    row.set_any(messages);
    updateRow[0].setRow(row);
    agileUpdateRowsRequest[0].setRow(updateRow);
    agileUpdateRowsRequest[0].setObjectInfo(table);
    updateRowsRequestType.setData(agileUpdateRowsRequest);

```

---

**Note** See [Appendix](#) on page 205 for [getRowId](#) on page 209 helper method.

---

## Removing Rows from a Table

To remove a row from a table, use the operation `removeRows` operation, which requires `tableIdentifier`, `rowID`, besides `objectIdentifier`, `objectNumber`, and `objectInfo`.

If a table is read-only, you can't remove rows from it. To check the read/write status of a tables, see [Working with ReadOnly Tables](#) on page 35.

If you are working with a released revision of an item, you cannot remove a row from the item's tables until you create a change order for a new revision.

### Example: Removing a Table Row

```

RemoveRowsRequestType removeRowsRequestType = new RemoveRowsRequestType();
AgileRemoveRowsRequest agileRemoveRowsRequest[] = new AgileRemoveRowsRequest[1];
agileRemoveRowsRequest[0] = new AgileRemoveRowsRequest();
RequestTableType table = new RequestTableType();
table.setClassIdentifier("Part");
table.setObjectNumber( parentPartNumber );
table.setTableIdentifier( ItemConstants.TABLE_BOM.toString() );
agileRemoveRowsRequest[0].setObjectInfo(table);
agileRemoveRowsRequest[0].setRowId( new Integer[] {getRowID("Part", parentPartNumber, "BOM",
childPartNumber)} );
removeRowsRequestType.setRows(agileRemoveRowsRequest);

```

---

**Note** See Appendix for the [getRowId helper method](#). on page 209

---

## Clearing a Table

You can clear the entire table by removing all the rows. This can be done by setting the `tableIdentifier` in the operation `clearTable`.

### Example: Clearing a Table

```

RequestTableType table1 = new RequestTableType();
table1.setClassIdentifier("Part");
table1.setObjectNumber( partNumber );
table1.setTableIdentifier( "tableAPIName" );

```

```
agileClearTableRequestType[0].setAgileTable(table1);
clearTableRequestType.setClearTable(agileClearTableRequestType);
```

## Copying Tables

You can copy all the rows of a table in an Agile object to another table by using the operation `copyTable`. This operation requires `classIdentifier`, `objectNumber` and `tableIdentifier`, and setting of the `SourceTable` and `TargetTable` values.

```
agileCopyTableRequestType[0] = new AgileCopyTableRequestType();
RequestTableType table1 = new RequestTableType();
RequestTableType table2 = new RequestTableType();
    table1.setClassIdentifier("Part");
    table1.setObjectNumber( partNumber1 );
    table1.setTableIdentifier("Compositions");
    table2.setClassIdentifier("Part");
    table2.setObjectNumber( partNumber2 );
    table2.setTableIdentifier( "Compositions" );
agileCopyTableRequestType[0].setSourceTable(table1);
agileCopyTableRequestType[0].setTargetTable(table2);
copyTableRequestType.setCopyTable(agileCopyTableRequestType);
```

## Redlining a Table

When you issue a change for a released item or a price agreement, the Agile Web Services lets you redline certain tables affected by the change. In the Agile PLM clients, redline tables visually identify values that have been modified from the previous revision. Red underlined text - thus the term “redline”, indicates values that have been added, and red strikethrough text indicates values that have been deleted. Those responsible for approving the change can review the redline data.

The Agile PLM system provides the following Redline tables:

- Redline BOM
- Redline Manufacturers (AML)
- Redline Price Lines
- Redline Title Block

### Example: Adding a Redlined BOM

```
RequestTableType table = new RequestTableType();
table.setClassIdentifier("Part");
table.setObjectNumber( parentPartNumber );
table.setTableIdentifier( "-803" );
ObjectReferentIdType multiSelect = new ObjectReferentIdType();
multiSelect.setClassIdentifier("Part");
multiSelect.setObjectIdentifier("P00407");
MessageElement dataCell = new MessageElement(namespaceUri, "itemNumber");
dataCell.addAttribute(XSIPREFIX, COMMONNAMESPACEURI, TYPE, "ObjectReferentIdType");
dataCell.setObjectValue(multiSelect);
message[0] = dataCell;
```

```
row[0].set_any(message);
PropertyType[] options = new PropertyType[1];
options[0] = new PropertyType();
options[0].setPropertyName(SchemaConstants.redline_change.getValue());
options[0].setPropertyValue("C00644");
row[0].setOptions(options);
agileAddRowsRequest[0].setObjectInfo(table);
```



# Working with Searches

This chapter includes the following:

---

▪ Specifying Search Criteria .....	45
▪ Using SQL Syntax to specify Search Criteria .....	53
▪ Setting Result Attributes for a Search.....	54
▪ Examples of Searches.....	60

This chapter describes how to work with the Agile PLM Searches and provides sample code snippets.

## Agile PLM Searches

Agile PLM Searches can have multiple search criteria (like an Advanced Search in the Agile Web Client), or it can be a simple search that specifies only one criteria.

## Specifying Search Criteria

You can narrow the number of objects returned from a search by specifying search criteria. If you specify \* as the search criteria, the query returns references to all objects in the specified query class. It's a good practice to limit the search criteria as much as possible, as the amount of data returned may be excessively large, resulting in decreased performance.

You can use the `setCriteria(criteria)` method to specify query criteria, which sets the search criteria from a specified String. This String references one or more parameters.

```
advancedSearchRequestType.setClassIdentifier("Part");
advancedSearchRequestType.setCaseSensitive(false);
String criteria = "[Title Block.Number] contains 'P0' && " +
"[Title Block.Description] is not null";
advancedSearchRequestType.setCriteria(criteria);
String attribute1 = "Title Block.Number";
String attribute2 = "Title Block.Description";
String attribute3 = "Title Block.Lifecycle Phase";
advancedSearchRequestType.setResultAttributes(new String[]{attribute1, attribute2,
attribute3} );
advancedSearchRequestType.setDisplayName("Search123");
AdvancedSearchResponseType advancedSearchResponseType =
agileStub.advancedSearch(advancedSearchRequestType);
```

## Search Conditions

The Agile Web Services provides a simple yet powerful query language for specifying search criteria. The query language defines the proper syntax for filters, conditions, attribute references, relational operators, logical operators, and other elements.

Search criteria consist of one or more search conditions. Each search condition contains the following elements:

- **Left operand** – The left operand is always an attribute enclosed in brackets, such as `[Title Block.Number]`. You can specify the attribute as an attribute name (fully qualified name or short name) or attribute ID number. The attribute specifies which characteristic of the object to use in the search.
- **Relational operator** – The relational operator defines the relationship that the attribute has to the specified value, for example, “equal to” or “not equal to.”
- **Right operand** – The matching value for the specified attribute in the left operand. The right operand can be a constant expression or a set of constant expressions. A set of constant expressions is needed if the relational operator is “between,” “not between,” “in,” or “not in.”

Following is an example of a search condition:

```
[Title Block.Description] == 'Computer'
```

This is another example where the right operand is a set of constant expressions:

```
[Page Two.Numeric01] between ('1000', '2000')
```

## Search Operation Keywords

When you specify a search condition, you must use proper keywords to construct the statement. The following keywords are available:

and	does	less	or	to
asc	equal	like	order	union
between	from	minus	phrase	where
by	greater	none	select	with
contain	in	not	start	word
contains	intersect	null	starts	words
desc	is	of	than	

These keywords are not localized. You must use English keywords, regardless of locale. You can use the keywords in lower case or upper case. In addition to keywords, you can use Agile PLM variables such as `$USER` (for current user) and `$TODAY` (for today's date) in Agile Searches.

The "in" operator does not support MultiList in (set) query criteria.



## Specifying Search Attributes

Every Agile PLM object that you can search for also has an associated set of attributes, which are inherent characteristics of the object. You can use these attributes as the left operand of a search condition. The right operand of the search condition specifies the attribute's value(s).

A search attribute must be enclosed within brackets, for example, `[Title Block.Number]`. The brackets are needed because many attribute names have spaces. If a search attribute is not enclosed within brackets, your query will fail.

You can specify a search attribute in the following ways:

Attribute reference	Example
attribute ID number	<code>[1001]</code>
fully-qualified attribute name	<code>[Title Block.Number]</code>
short attribute name	<code>[Number]</code>

**Note** Because attribute names can be modified, Agile recommends referencing attributes by ID number or constant. However, many of the examples in this chapter reference attributes by name simply to make them more readable. If you choose to reference attributes by name, use the fully-qualified attribute name instead of the short name. Short attribute names are not guaranteed to be unique and could therefore cause your query to fail or produce unexpected results.

**Note** Specifying the search attributes using Attribute APIName is not supported.

Attribute names, whether you use the long or short form, are case-insensitive. For example, `[Title Block.Number]` and `[TITLE BLOCK.NUMBER]` are both allowed. Attribute names are also localized. The names of Agile PLM attributes vary based on the locale of your Agile Application Server. If you are creating a query that is going to be used on servers in different locales, you should reference attributes by ID number (or the equivalent constant) instead of by name.

If the attribute name contains special characters, such as quotes or backslashes, you can type these characters using the backslash (\) as an escape character. For example, to include a quote character in your string, type \'. If you want to write a backslash, type two of them together (\\). If the attribute name contains square brackets, enclose the entire name in quotes:

```
['Page Two.Unit of Measure [g or oz]']
```

There are other, perhaps less intuitive, ways to specify attributes. For example, you could pass in an `IAttribute` reference using a parameter of the `setCriteria()` method. In the following example, '%0' references the attribute in the `params` parameter.

```
advancedSearchRequestType.setCriteria("[Title Block.Number] starts with %0 and [Title
Block.Part Category] in %1 and [Title Block.Description] contains %2");
ParamListType[] params = new ParamListType[3];
params[0] = new ParamListType();
params[0].setParameter(new String[]{"P00"});
params[1] = new ParamListType();
params[1].setParameter(new String[]{"Electrical", "Mechanical"});
params[2] = new ParamListType();
params[2].setParameter(new String[]{"Resistor"});
```

```
advancedSearchRequestType.setParams(params);
```

You can also use `String` concatenation to reference an attribute constant:

```
advancedSearchRequestType.setCriteria("[\" + ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION + \"]  
== 'Computer'");
```

## Getting the Searchable Attributes

The searchable attributes for a query depend on the specified query class or subclass. However, the searchable attributes for a subclass can differ greatly from searchable attributes for its parent class.

Due to database considerations, not all attributes are searchable. Generally, a few select Page One attribute (namely: Title Page, Cover Page, and General Info attributes) are searchable for each class.

If a tab is not configured in Java Client to be visible, you can still search for an attribute on that tab in the Agile Web Services. However, you must search for the Table name that corresponds to the Tab name.

To find the searchable attributes for a query, use the `getSearchableAttributes` operation.

Even though an attribute may not be searchable, it can still be included as a column in the query results. For more information, see [Setting Result Attributes for a Query](#).

## Using Relational Operators

Table below lists relational operators that are supported by the Agile Web Services search operations.

English operator	Notation	Description
equal to	==	Finds only an exact match with the specified value.
not equal to	!=	Finds any value other than an exact match with the specified value.
greater than	>	Finds any value greater than the specified value.
greater than or equal to	>=	Finds any value greater than or equal to the specified value.
less than	<	Finds any value less than the specified value.
less than or equal to	<=	Finds any value less than or equal to the specified value.
contains, contains all		Finds any value that includes the specified value.
does not contain, does not contain all		Finds any value that does not include the specified value.
contains any		Finds any value that includes the specified value.
does not contain any		Finds any value that does not include the specified value.

English operator	Notation	Description
contains none of		Finds any value that includes none of the specified values.
does not contain none of		Behaves the same as <code>does not contain any</code> .
starts with		Finds values that begin with characters in the specified value.
does not start with		Finds values that do not begin with characters in the specified value.
is null		Finds objects where the selected attribute contains no value.
is not null		Finds objects where the selected attribute contains a value.
like		Performs a wildcard search, finding objects that match a single character or any string.
not like		Performs a wildcard search, finding objects that do not match a single character or any string.
between		Finds objects that fall between the specified values.
not between		Finds objects that do not fall between the specified values.
in		Finds objects that match any of the specified values.
not in		Finds objects that do not match any of the specified values.
contains phrase		Finds objects with files that contain the specified phrase.
contains all words		Finds objects with files that contain all of the specified words.
contains any word		Finds objects with files that contain any of the specified words.
contains none of		Finds objects with files that contain none of the specified words.

Relational operators are not localized. You must use English keywords, regardless of locale. As with other query language keywords, you can use them in lower case or upper case.

## Using Unicode Escape Sequences

Agile Web Services Search operations support Unicode escape sequences. The primary usage of Unicode escape sequences in a query string is to search for nonburnable or foreign local character sets. A Unicode character is represented with the Unicode escape sequence `\uxxxx`, where `xxxx` is a sequence of four hexadecimal digits.

For example, to search for an item with Unicode 3458, use the following query:

```
Select * from [Items] where [Description] contains '\u3458'
```

There is another query operation for “contains” usage in the case of MultiList.

## Using Between, Not Between, In, and Not In Operators

The ‘between’, ‘not between’, ‘in’, and ‘not in’ relational operators are not supported directly by Agile PLM Java and Web clients. These relational operators provide a convenient shorthand method for specifying ‘equal to’, ‘not equal to’, ‘greater than or equal to’, or ‘less than or equal to’ operations with a set of values.

Short form	Equivalent long form
[Number] between ('1','6')	[Number] >= '1' and [Number] <= '6'
[Number] not between ('1','6')	[Number] < '1' and [Number] > '6'
[Number] in ('1','2','3','4','5','6')	[Number] == '1' or [Number] == '2' or [Number] == '3' or [Number] == '4' or [Number] == '5' or [Number] == '6'
[Number] not in ('1','2','3','4','5','6')	[Number] != '1' and [Number] != '2' and [Number] != '3' and [Number] != '4' and [Number] != '5' and [Number] != '6'

As shown in the table, when you use the ‘between’, ‘not between’, ‘in’, and ‘not in’ relational operators, each value in the set of values must be enclosed in quotes and delimited by commas. Here are more criteria examples that use ‘between’ and ‘in’ relational operators:

```
[Title Block.Number] in ('1000-02', '1234-01', '4567-89')
[Title Block.Effectivity Date] between ('01/01/2001', '01/01/2002')
[Page Two.Numeric01] between ('1000', '2000')
```

---

**Note** The relational operators any, all, none of, and not all are not supported in the Web Services.

---

## Using the Nested Criteria to Search for Values in Object Lists

Several lists in Agile PLM contain business objects, such as Agile PLM users. To search for an object in such a list, you can specify nested query criteria. Nested criteria are enclosed in parentheses and separated from each other by a logical AND (&&) or OR (||) operator. A comma can also be used to separate nested criteria; it's equivalent to a logical OR.

The following criteria finds a user with the first name Christopher OR the last name Nolan.

```
[Page Two.Create User] in ([General Info.First Name] == 'Christopher',
[General Info.Last Name] == 'Nolan')
```

The following criteria finds a user with the first name Christopher AND the last name Nolan.

```
[Page Two.Create User] in ([General Info.First Name] == 'Christopher' &&
[General Info.Last Name] == 'Nolan')
```

The parameter query is not supported in nested queries and multiple values for one placeholder in query parameters must be specified in two dimensional arrays as shown in the example below.

### Example: Correct and incorrect parameter query in nested query criteria

- The parameter query specified in the following nested query criteria will fail to execute:  

```
[Page Two.User1] in ([General Info.First Name] == %0)
```
- However, when it is explicitly specified as a string value, instead of the placeholder, it will succeed:

```
[Page Two.User1] in ([General Info.First Name] == 'Christopher')
```

## Searching for Words or Phrases Contained in Attachments

Two special attributes, `[Attachments.File Document Text]` and `[Files.Document Text]`, are used to index the content of files stored on the Agile file management server. If you are hosting your database on Oracle, you can take advantage of a feature that lets you search for words or phrases contained in attachments. When you create search criteria that uses either of these attributes, there are four additional relational operators you can use:

- `contains phrase`
- `contains all words`
- `contains any word`
- `contains none of`

The following table shows several search conditions that search for words or phrases in attachments.

Search Condition	Finds
<code>[Attachments.File Document Text]</code> <code>contains phrase 'adding new materials'</code>	Objects in which any of their attachments contain the phrase "adding new materials."
<code>all [Attachments.File Document Text]</code> <code>contains all words 'adding new materials'</code>	Objects in which all their attachments contain the words "adding," "new," and "materials."
<code>none of [Attachments.File Document Text]</code> <code>contains any word 'containers BOM return output'</code>	Objects in which none of their attachments contain any of the words "containers," "BOM," "return," or "output."
<code>[Attachments.File Document Text]</code> <code>contains none of 'containers BOM output'</code>	Objects in which any of their attachments do not contain the words "containers," "BOM," or "output."

## Using Logical Operators

You can use logical operators to combine multiple search conditions into a complex filter. When you have two or more conditions defined in a set of query criteria, the relationship between them is defined as either 'and' or 'or'.

- **and** narrows the search by requiring that both conditions are met. Each item in the results must match both conditions. The 'and' logical operator can also be specified using two ampersands, '&&'.
- **or** broadens the search by including any object that meets either condition. Each item in the results table needs to match only one of the conditions, but may match both. The 'or' logical operator can also be specified using two vertical bars, '||'.

Logical operators are case-insensitive. For example, 'and' or 'AND' are both allowed.

The following query criteria finds parts that have both a part category equal to Electrical and a lifecycle phase equal to Inactive.

```
[Title Block.Part Category] == 'Electrical' and  
[Title Block.Lifecycle Phase] == 'Inactive'
```

If you replace the ‘and’ operator with ‘or’, the query locates all parts with either a part category of Electrical or a lifecycle phase of Inactive, which could be a large number of parts.

```
[Title Block.Part Category] == 'Electrical' or  
[Title Block.Lifecycle Phase] == 'Inactive'
```

The Agile Web Services provides three where-used set operators. For more information, see [Creating a Where-Used Query](#).

Logical operators, including the where-used set operators, are not localized. You must use English keywords, regardless of locale.

## Using Wildcard Characters with the Like Operator

If you define a search condition using the ‘like’ operator, you can use two wildcard characters: the asterisk (\*) and question mark (?). The asterisk matches any string of any length, so **\*at** finds cat, splat, and big hat.

For example, `[Title Block.Description] like '*book*' returns all objects that contain the word “book,” such as textbook, bookstore, books, and so on.`

The question mark matches any single character, so **?at** finds hat, cat, and fat, but not splat.

For example, `[Title Block.Description] like '?al*' matches any word containing “al” that is preceded by a single letter, such as tall, wall, mall, calendar, and so on.`

## Using Parentheses in Search Criteria

Where-used, set operators have higher priority than **and** and **or** logical operators, as shown by the following table.

Priority	Operator(s)
1	<ul style="list-style-type: none"><li>▫ union</li><li>▫ intersection</li><li>▫ minus</li></ul>
2	<ul style="list-style-type: none"><li>▫ and</li><li>▫ or</li></ul>

Therefore, search conditions joined by **union**, **intersection**, and **minus** operators are evaluated before conditions joined by **and** or **or**.

If you use where-used set operators (‘union’, ‘intersect’, or ‘minus’) in search criteria, you can use parentheses to change the order that criteria are evaluated. If only ‘and’ or ‘or’ logical operators are used in a search criteria, additional parentheses aren’t needed because they don’t change the result of criteria evaluation.

The following two criteria, although they contain the same search conditions, provide different results because parentheses are placed differently:

```
([Title Block.Part Category] == 'Electrical' and  
[Title Block.Description] contains 'Resistor') union
```

```

([Title Block.Description] contains '400' and
 [Title Block.Product Line(s)] contains 'Taurus')

[Title Block.Part Category] == 'Electrical' and
([Title Block.Description] contains 'Resistor' union
 [Title Block.Description] contains '400') and
 [Title Block.Product Line(s)] contains 'Taurus'

```

## Using SQL Syntax to specify Search Criteria

In addition to its standard query language, the Agile Web Services also supports SQL-like syntax to specify search criteria. If you are familiar with SQL statements, you may find this extended query language more flexible, more powerful and easier to work with. It combines in one operation the specification of the query result attributes, the query class, the search condition, and the sort column(s).

This is a simple example that demonstrates the syntax:

- Query result attributes: `SELECT [Title Block.Number], [Title Block.Description]`
- Query class: `FROM [Items]`
- Search condition: `WHERE [Title Block.Number] starts with 'P'`
- Sort column(s): `ORDER BY 1 asc`

To improve readability, it's recommended that SQL key words such as `SELECT` and `FROM` are all typed using capital letters and each part of the statement appears on a separate line. This is merely a convention, not a requirement. SQL key words are not case-sensitive, and you can write the entire query string on one line if you prefer.

The best way to demonstrate the advantages of SQL syntax is to compare the code for a query that uses standard Agile API query syntax for search criteria with one that uses SQL syntax. The following example shows a query created using the standard Agile API query syntax:

### Example: Query using standard Agile API query syntax

```

advancedSearchRequestType.setCriteria("[Page Two.Numeric01] between (1000, 2000)");
//Set result attributes
String[] attrs = { "Title Block.Number", "Title Block.Description",
    "Title Block.Lifecycle Phase" };
advancedSearchRequestType.setResultAttributes(attrs);

```

This example shows the same query rewritten in SQL syntax. Although the example doesn't have fewer lines of code, you may find that it's more readable than Agile API query syntax, particularly if you're familiar with SQL.

### Example: Search criteria using SQL syntax

```

String criteria = "SELECT " +
    "[Title Block.Number],[Title Block.Description], " +
    "[Title Block.Lifecycle Phase] " +
    "FROM " +
    "[Items] " +
    "WHERE " +

```

```
    "[Title Block.Number] between (1000, 2000)";  
advancedSearchRequestType.setCriteria(criteria);
```

The following example shows a query written with SQL syntax that specifies the search criteria.

**Example: Using SQL syntax to specify query attributes**

```
try {  
    String statement =  
        "SELECT " +  
        "[Title Block.Number], [Title Block.Description] " +  
        "FROM " +  
        "[Items] " +  
        "WHERE " +  
        "[Title Block.Description] like %0";  
    advancedSearchRequestType.setCriteria(statement);  
}
```

---

**Note** Remember, the FROM part of the search condition specifies the query class. If you use the classIdentifier attribute to also specify a query class, the query class specified in the SQL search condition takes precedence.

---

## Using SQL Wildcards

You can use both the asterisk (\*) and question mark (?) as wildcards in a query that uses SQL syntax. In Agile Web Services search operation. The asterisk matches any string and the question mark matches any single character. You can use wildcards in the SELECT statement (the specified query result attributes) and the WHERE statement (the search condition).

For example, "SELECT \*" specifies all available query result attributes.

## Setting Result Attributes for a Search

When you use the operation `advancedSearch`, it returns several output fields, which are also called result attributes. By default, there are only a few result attributes for each query class. You can add or remove result attributes using the `setResultAttributes()`.

The following table shows the default query result attributes for each predefined Agile PLM class.



Query class	Default result attributes
<b>Changes</b> Change Orders ECO Change Requests ECR Deviations Deviation Manufacturer Orders MCO Price Change Orders PCO Sites Change Orders SCO Stop Ships Stop Ship	Cover Page.Change Type Cover Page.Number Cover Page.Description Cover Page.Status Cover Page.Workflow
<b>Customers</b> Customers Customer	General Info.Customer Type General Info.Customer Number General Info.Customer Name General Info.Description General Info.Lifecycle Phase
<b>Declarations</b> Homogeneous Material Declarations Homogeneous Material Declaration IPC 1752-1 Declarations IPC 1752-1 Declaration IPC 1752-2 Declarations IPC 1752-2 Declaration JGPSSI Declarations JGPSSI Declaration Part Declarations Part Declaration Substance Declarations Substance Declaration Supplier Declarations of Conformance Supplier Declaration of Conformance	Cover Page.Name Cover Page.Description Cover Page.Supplier Cover Page.Status Cover Page.Workflow Cover Page.Compliance Manager Cover Page.Due Date Cover Page.Declaration Type

Query class	Default result attributes
<b>Discussions</b> Discussions Discussion	Cover Page.Subject Cover Page.Status Cover Page.Priority Cover Page.Type
<b>File Folders</b> File Folders File Folder	Title Block.Type Title Block.Number Title Block.Description Title Block.Lifecycle Phase
<b>Items</b> Parts Part Documentation Document	Title Block.Item Type Title Block.Number Title Block.Description Title Block.Lifecycle Phase Title Block.Rev
<b>Manufacturers</b> Manufacturers Manufacturer	General Info.Name General Info.City General Info.State General Info.Lifecycle Phase General Info.URL
<b>Manufacturer Parts</b> Manufacturer Parts Manufacturer Part	General Info.Manufacturer Part Number General Info.Manufacturer Name General Info.Description General Info.Lifecycle Phase
<b>Packages</b> Packages Package	Cover Page.Package Number Cover Page.Description Cover Page.Assembly Number Cover Page.Status Cover Page.Workflow
<b>Part Groups</b> Part Groups Commodity Part Family	General Info.Name General Info.Description General Info.Lifecycle Phase General Info.Commodity Type General Info.Overall Compliance

Query class	Default result attributes
<b>Prices</b> Published Prices Contracts Published Price Quote History Quote History	General Info.Price Number General Info.Description General Info.Rev General Info.Price Type General Info.Lifecycle Phase General Info.Program General Info.Customer General Info.Supplier
<b>Product Service Requests</b> Non-Conformance Reports NCR Problem Reports Problem Report	Cover Page.PSR Type Cover Page.Number Cover Page.Description Cover Page.Status Cover Page.Workflow
<b>Programs</b> Activities Program Phase Task Gates Gate	General Info.Name General Info.Description General Info.Status General Info.Health General Info.Owner General Info.Root Parent General Info.Workflow General Info.Type
<b>Projects</b> Sourcing Projects Sourcing Project	General Info.Project Type General Info.Number General Info.Description General Info.Manufacturing Site General Info.Ship To Location General Info.Program General Info.Customer General Info.Lifecycle Phase
<b>Quality Change Requests</b> Corrective Action/Preventive Action CAPA Audits Audit	Cover Page.QCR Type Cover Page.QCR Number Cover Page.Description Cover Page.Status

Query class	Default result attributes
	Cover Page.Workflow
<b>RFQ Responses</b> RFQ Responses RFQ Response	Cover Page.RFQ Number Cover Page.RFQ Description Cover Page.Lifecycle Phase Cover Page.Requested Cover Page.Completed Cover Page.Due Date
<b>RFQs</b> RFQs RFQ	Cover Page.RFQ Number Cover Page.RFQ Description Cover Page.MFG Site Cover Page.Ship-To Location Cover Page.Program Cover Page.Customer Cover Page.Lifecycle Phase Cover Page.RFQ Type
<b>Sites</b> Sites Site	General Info.Name General Info.Contact General Info.Phone
<b>Specifications</b> Specifications Specification	General Info.Name General Info.Description General Info.Lifecycle Phase General Info.Jurisdictions General Info.Validation Type General Info.Specification Type

Query class	Default result attributes
<b>Substances</b> Materials Material Subparts Subpart Substance Groups Substance Group Substances Substance	General Info.Name General Info.Description General Info.CAS Number General Info.Lifecycle Phase General Info.Substance Type
<b>Suppliers</b> Suppliers Component Manufacturer Contract Manufacturer Distributor Manufacturer Rep	General Info.Supplier Type General Info.Number General Info.Name General Info.Description General Info.Status
<b>Transfer Orders</b> Content Transfer Orders CTO Automated Transfer Orders ATO	Cover Page.Transfer Order Type Cover Page.Transfer Order Number Cover Page.Description Cover Page.Status Cover Page.Workflow

## Specifying Result Attributes

If you run a query and find that the resulting `table` object does not contain the attributes you expected, it's because you didn't specify result attributes. The following example shows how to specify the result attributes for a query.

### Example: Setting query result attributes

```
String attribute1 = "Title Block.Number";
String attribute2 = "Title Block.Description";
String attribute3 = "Title Block.Lifecycle Phase";
advancedSearchRequestType.setResultAttributes(new String[]{attribute1, attribute2,
attribute3} );
```

The `ResultAttributes` element takes an array of `String` where you can array of attribute names (such as `{"Title Block.Description", "Title Block.Number"}`) or attribute ID constants or attribute API names. The following example shows how to specify result attributes using ID constants.

```
Setting query result attributes by specifying ID constants private void
setQueryResultColumns(IQuery query) throws APIException {
    // Put the attribute IDs into an array
    String[] attrs = { ItemConstants.ATT_TITLE_BLOCK_NUMBER+"" ,
                      ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION+"" ,
                      ItemConstants.ATT_TITLE_BLOCK_LIFECYCLE_PHASE+"" ,
```

```
ItemConstants.ATT_PAGE_TWO_TEXT01+""",
ItemConstants.ATT_PAGE_TWO_NUMERIC01+""",
ItemConstants.ATT_PAGE_THREE_TEXT01+"""};
// Set the result attributes for the query
advancedSearchRequestType.setResultAttributes(attrs);
}
```

When you use the `setResultAttributes()` method, make sure you specify valid result attributes. Otherwise, the `setResultAttributes()` method will fail.

## Examples of Searches

The examples below show how to create quick search, advanced search and how to get the searchable attributes.

### Quick Search

#### *Operation - quickSearch*

```
QuickSearchRequestType quickSearchRequestType = new QuickSearchRequestType();
quickSearchRequestType.setClassIdentifier("Part");
quickSearchRequestType.setKeywords("P0*");
quickSearchRequestType.setSearchFiles(false);
QuickSearchResponseType quickSearchResponseType =
agileStub.quickSearch(quickSearchRequestType);
```

### Advanced Search

#### *Operation - advancedSearch*

```
AdvancedSearchRequestType advancedSearchRequestType = new AdvancedSearchRequestType();
advancedSearchRequestType.setClassIdentifier("Part");
advancedSearchRequestType.setCaseSensitive(false);
String criteria = "[Title Block.Number] contains 'P0' && " +
    "[Title Block.Description] is not null";
advancedSearchRequestType.setCriteria(criteria);
String attribute1 = "Title Block.Number";
String attribute2 = "Title Block.Description";
String attribute3 = "Title Block.Lifecycle Phase";
advancedSearchRequestType.setResultAttributes(new String[]{attribute1, attribute2,
attribute3});
advancedSearchRequestType.setDisplayName("Search123");
AdvancedSearchResponseType advancedSearchResponseType =
agileStub.advancedSearch(advancedSearchRequestType);
```

## Getting the Searchable Attributes

### ***Operation - getSearchableAttributes***

```
QueryGetSearchableAttributesRequestType queryGetSearchableAttributesRequestType = new
QueryGetSearchableAttributesRequestType();
queryGetSearchableAttributesRequestType.setClassIdentifier("Part");
QueryGetSearchableAttributesResponseType queryGetSearchableAttributesResponseType =
agileStub.getSearchableAttributes(queryGetSearchableAttributesRequestType);
```





# Working with File Folders and Attachments

**This chapter includes the following:**

---

▪ Managing File Folders .....	63
▪ Getting a File from a File Folder .....	67
▪ Adding Files to a File Folder Object.....	68
▪ Managing Attachments .....	70

This chapter describes how to work with the Agile PLM File Folders and Attachments, and provides sample code snippets.

## Agile File Folders

A File Folder is a business object that specifies one or more files or URLs that are stored in the file PLM server vault. In addition, a file folder has its own set of tables. This means that you can create and load an independent file folder and add one or more files to its Files table. You can also search for a file folder, just as you would search for an Item or Change.

The File Folder Base Class has two Classes and each of these classes have their own respective Subclasses. This section describes File Folder features and components, and provides procedures to add, modify, or remove them.

## Managing File Folders

The Agile Web Services let you perform File Folder related tasks, such as:

- checking-in and checking-out files associated with the objects in the rows of an Attachments table
- adding files and URLs to an Attachments table
- deleting attachments

This section lists and describes these features, and provides necessary procedures to use the Agile Web Services to perform these tasks.

## Creating a File Folder

With the operation createObject, you can create a File Folder, as a file folder is an Agile Object.

### Example: Creating a File Folder

```
CreateObjectType createObjectType = new CreateObjectType();
```

```
AgileCreateObjectRequest agileCreateObjectRequest[] = new AgileCreateObjectRequest[1];
agileCreateObjectRequest[0] = new AgileCreateObjectRequest();
agileCreateObjectRequest[0].setClassIdentifier("FileFolder");
AgileRowType row_1 = new AgileRowType();
MessageElement[] messages = new MessageElement[2];
String namespaceUri = null;
messages[0] = new MessageElement(namespaceUri, "number");
messages[0].addTextNode(folderNumber);
messages[1] = new MessageElement(namespaceUri, "description");
messages[1].addTextNode("File Folder Description");
row_1.set_any(messages);
agileCreateObjectRequest[0].setData(row_1);
createObjectRequestType.setRequests(agileCreateObjectRequest);
```

---

**Note** When you add a file or a URL to the row of the Attachments table of a business object, you will automatically create automatically a new file folder object that contains the associated file or URL.

---

### Example: Creating a Design Object

```
CreateObjectRequestType createObjectRequestType = new CreateObjectRequestType();
AgileCreateObjectRequest agileCreateObjectRequest[] = new AgileCreateObjectRequest[1];
agileCreateObjectRequest[0] = new AgileCreateObjectRequest();
agileCreateObjectRequest[0].setClassIdentifier("Design");
AgileRowType row_1 = new AgileRowType();
MessageElement[] messages = new MessageElement[2];
String namespaceUri = null;
messages[0] = new MessageElement(namespaceUri, "number");
messages[0].addTextNode(designNumber);
messages[1] = new MessageElement(namespaceUri, "description");
messages[1].addTextNode("Design Desc");
row_1.set_any(messages);
agileCreateObjectRequest[0].setData(row_1);
createObjectRequestType.setRequests(agileCreateObjectRequest);
```

## Checking Out a File Folder

Before you can add, delete, or modify the files contained in a file folder, you must check out the file folder. With the appropriate privileges, you can check out a file folder as long as it is not already checked out by another user. Once a file folder is checked out, no one else can check it out or modify it.

The user who checked out a file folder, as well as other users who are change analysts or component engineers, can check it in. If the file folder was checked out to a location on the network, or to a shared drive or directory, anyone who has access to that network location or to that shared directory can check in the file folder.

Use the operation checkOutFF for checking out a file folder.

```
CheckOutFFRequestType checkOutFFRequestType = new CheckOutFFRequestType();
AgileCheckOutFFRequest agileCheckOutFFRequest[] = new AgileCheckOutFFRequest[1];
agileCheckOutFFRequest[0] = new AgileCheckOutFFRequest();
agileCheckOutFFRequest[0].setFolderNumber(folderNumber);
checkOutFFRequestType.setRequests(agileCheckOutFFRequest);
```

## Setting the Version of File Folder Files

A file folder can have several versions. When you add a file folder to the Attachments table of another business object, you can specify the file version to use. If you don't specify a file version, the Agile Web Services use the default or latest version. If you specify a file version, the row on the Attachments table is linked to that version.

If the parent object containing the Attachments table is an item, you can incorporate the item to lock the specified versions of its attachments.

### Example: Setting the version when adding a row to the Attachments table

This is carried out in two stages. First, add a row using the operation `addRows` to add a file folder and then update the table using the operation `updateRows`.

#### Stage 1 - Adding the file folder to the attachment table of a Part

```
AddRowsRequestType addRowsRequestType = new AddRowsRequestType();
AgileAddRowsRequest agileAddRowsRequest[] = new AgileAddRowsRequest[1];
agileAddRowsRequest[0] = new AgileAddRowsRequest();
RequestTableType table = new RequestTableType();
table.setClassIdentifier("Part");
table.setObjectNumber( parentPartNumber );
table.setTableIdentifier( "Attachments" );
folderNumber = "FOLDER00441";
    AgileRowType[] rows = new AgileRowType[1];
    rows[0] = new AgileRowType();
    String namespaceUri = null;
    String COMMONNAMESPACEURI = "http://xmlns.oracle.com/AgileObjects/Core/Common/V1";
    MessageElement messages[] = new MessageElement[1];
    messages[0] = new MessageElement(namespaceUri, "folderNumber");
    messages[0].addAttribute(Constants.NS_PREFIX_SCHEMA_XSI, COMMONNAMESPACEURI, "type",
        "ObjectReferentIdType");
    ObjectReferentIdType objRefId = new ObjectReferentIdType();
    objRefId.setClassIdentifier("FileFolder");
    objRefId.setObjectIdentifier( folderNumber );
    messages[0].setObjectValue(objRefId);
    rows[0].set_any(messages);
    agileAddRowsRequest[0].setRow(rows);
    agileAddRowsRequest[0].setObjectInfo(table);
addRowsRequestType.setData(agileAddRowsRequest);
```

#### Stage 2 - Updating the version of this newly created file folder on the attachment table

```
RequestTableType updateTable = new RequestTableType();
updateTable.setClassIdentifier( "Part" );
updateTable.setObjectNumber( parentPartNumber );
updateTable.setTableIdentifier( "Attachments" );
AgileUpdateRow updateRow[] = new AgileUpdateRow[1];
updateRow[0] = new AgileUpdateRow();
updateRow[0].setRowId(RowId);
AgileRowType row = new AgileRowType();
MessageElement messages_2[] = new MessageElement[1];
messages_2[0] = new MessageElement(namespaceUri, "folderVersion");
messages_2[0].addAttribute(Constants.NS_PREFIX_SCHEMA_XSI, COMMONNAMESPACEURI, "type",
    "AgileListEntryType");
```

```
AgileListEntryType list = new AgileListEntryType();
    SelectionType[] selection = new SelectionType[1];
    selection[0] = new SelectionType();
    selection[0].setValue("2");
    list.setSelection(selection);
    messages_2[0].setObjectValue(list);
    row.set_any(messages_2);
    updateRow[0].setRow(row);
agileUpdateRowsRequest[0].setRow(updateRow);
agileUpdateRowsRequest[0].setObjectInfo(updateTable);
updateRowsRequestType.setData(agileUpdateRowsRequest);
```

## Cancelling a File Folder Checkout

If you check out a file folder and then decide not to modify it, or discard your changes and revert to the original file folder, you can cancel the checkout with the operation `cancelCheckOutFF`. When you cancel a checkout, the file folder is available for other users to check out.

---

**Note** Only the user who checked out a file folder can cancel the checkout.

---

### Example: Cancelling a file folder checkout

```
CancelCheckOutFFRequestType cancelCheckOutFFRequestType = new CancelCheckOutFFRequestType();
AgileCancelCheckOutFFRequest agileCancelCheckOutFFRequest[] = new
AgileCancelCheckOutFFRequest[1];
agileCancelCheckOutFFRequest[0] = new AgileCancelCheckOutFFRequest();
agileCancelCheckOutFFRequest[0].setFolderNumber( folderNumber );
cancelCheckOutFFRequestType.setRequests(agileCancelCheckOutFFRequest);
```

## Checking In a File Folder

After you finish working on the file folder that you checked out, you can check it in using the operation `checkInFF`.

File folders can contain multiple files. When you check in a file folder, you automatically check in all the files that are contained in it. You do not need to specifically list the files contained in the file folder.

### Example: Checking In a File Folder

```
CheckInFFRequestType checkInFFRequestType = new CheckInFFRequestType();
AgileCheckInFFRequest agileCheckInFFRequest[] = new AgileCheckInFFRequest[1];
agileCheckInFFRequest[0] = new AgileCheckInFFRequest();
agileCheckInFFRequest[0].setFolderNumber( folderNumber );
checkInFFRequestType.setRequests(agileCheckInFFRequest);
```

## Deleting the File Folders

To delete a File Folder, use the operation `deleteObject`. You must have the Delete privilege for file folders to be able to delete them.

---

**Note** Deleting a file folder does not automatically remove its associated files from the file server. The Agile PLM administrator is responsible for purging deleted files.

---

To delete a row from the Attachments table of a business object, use the operation `removeRows`.

Removing a row from the Attachments table does not delete the associated file folder. You cannot delete a row from the Attachments table in the following situations:

- The parent object is an Item whose revision is incorporated.
- The selected attachment is currently checked out.

---

**Note** Only the user who checked out a file folder can cancel the checkout.

---

**Note** Deleting a file folder does not automatically remove its associated files from the file server. The Agile PLM administrator is responsible for purging the deleted files.

---

### Example: Deleting a File Folder

```
DeleteObjectRequestType deleteObjectRequestType = new DeleteObjectRequestType();
AgileDeleteObjectRequest agileDeleteObjectRequest[] = new AgileDeleteObjectRequest[1];
agileDeleteObjectRequest[0] = new AgileDeleteObjectRequest();
agileDeleteObjectRequest[0].setClassIdentifier("FileFolder");
agileDeleteObjectRequest[0].setObjectNumber(folderNumber);
deleteObjectRequestType.setRequests(agileDeleteObjectRequest);
```

## Getting a File from a File Folder

If a file folder is checked out by another user, you can still get a copy of the file folder file(s) and save it to your local machine. You can use the operation `getFileFF`, which returns the file stream associated with a row of the Attachments table. The file stream can be for one file or it can be a zipped file stream for multiple files, depending on how many files the associated file folder has.

If you call this operation from the file folder object, you return the zipped file stream for all files listed on the Files table. Whereas, if you call this operation from a row of the Files table of a file folder, you return a file stream for the specific file associated with that row.

```
GetFileFFRequestType getFileFFRequestType = new GetFileFFRequestType();
AgileGetFileFFRequest agileGetFileFFRequest[] = new AgileGetFileFFRequest[1];
agileGetFileFFRequest[0] = new AgileGetFileFFRequest();
agileGetFileFFRequest[0].setFolderNumber(folderNumber);
AgileFileAttachmentRequestType files[] = new AgileFileAttachmentRequestType[1];
files[0] = new AgileFileAttachmentRequestType();
files[0].setRowId(RowId);
agileGetFileFFRequest[0].setFiles(files);
getFileFFRequestType.setRequests(agileGetFileFFRequest);
```

### Example: Getting all the Files from a File Folder

To get all the file of an Agile File Folder object, set `allFiles` variable to true.

```
GetFileFFRequestType getFileFFRequestType = new GetFileFFRequestType();
AgileGetFileFFRequest agileGetFileFFRequest[] = new AgileGetFileFFRequest[1];
```

```
agileGetFileFFRequest[0] = new AgileGetFileFFRequest();
agileGetFileFFRequest[0].setFolderNumber(folderNumber);
agileGetFileFFRequest[0].setAllFiles(true);
getFileFFRequestType.setRequests(agileGetFileFFRequest);
```

## Getting a File from a File Folder using a Download URL

When you have to download extremely large files, you can use the operation `getFileFF` to generate a **downloadURL** for a file attachment present in an Agile File Folder. The request object contains a boolean to indicate that the download URL needs to be fetched and also the specifications that identify the attachment to be downloaded.

### Example: Getting a File using a URL

```
GetFileFFRequestType getFileFFRequestType = new GetFileFFRequestType();
AgileGetFileFFRequest agileGetFileFFRequest[] = new AgileGetFileFFRequest[1];
agileGetFileFFRequest[0] = new AgileGetFileFFRequest();
agileGetFileFFRequest[0].setFolderNumber(folderNumber);
agileGetFileFFRequest[0].setDownloadUrl(true);
AgileFileAttachmentRequestType files[] = new AgileFileAttachmentRequestType[1];
files[0] = new AgileFileAttachmentRequestType();
files[0].setRowId(rowId);
agileGetFileFFRequest[0].setFiles(files);
getFileFFRequestType.setRequests(agileGetFileFFRequest);
```

## Getting a File from a particular Version of File Folder

A file attachment can be downloaded from a particular version of an Agile File Folder Object. The request object contains the specifications that identify the attachment to be downloaded and the version of the folder, an array of bytes is obtained in the response object.

### Example: Getting a File from a Version of a File Folder

```
GetFileFFRequestType getFileFFRequestType = new GetFileFFRequestType();
AgileGetFileFFRequest agileGetFileFFRequest[] = new AgileGetFileFFRequest[1];
agileGetFileFFRequest[0] = new AgileGetFileFFRequest();
agileGetFileFFRequest[0].setFolderNumber(folderNumber);
agileGetFileFFRequest[0].setFolderVersion(folderVersion);
AgileFileAttachmentRequestType files[] = new AgileFileAttachmentRequestType[1];
files[0] = new AgileFileAttachmentRequestType();
files[0].setRowId(rowId);
agileGetFileFFRequest[0].setFiles(files);
getFileFFRequestType.setRequests(agileGetFileFFRequest);
```

## Adding Files to a File Folder Object

With the operation `addFileFF`, you can add files to the 'Files' tab of a File Folder. Before you add the files, the File Folder must be checked out using the operation `checkOutFF`.

```
checkOutFolder(folderNumber);
agileAddFileFFRequestType[0].setFolderNumber(folderNumber);
AddFileFFType files[] = new AddFileFFType[1];
```

```

files[0] = new AddFileFFType();
files[0].setFileName("Filename.txt");
files[0].setDescription("Description for file");
files[0].setFileContent( "File Content...file".getBytes() );
agileAddFileFFRequestType[0].setFiles(files);
addFileFFRequestType.setRequest(agileAddFileFFRequestType);

```

### Example: Adding a File using its DFM Reference

For the files that were already added to DFM, you can add a file to the 'Files' tab of a file folder using a reference obtained from the DFM file server.

```

checkoutFolder(folderNumber);
agileAddFileFFRequestType[0].setFolderNumber(folderNumber);
AddFileReferenceFFType reference[] = new AddFileReferenceFFType[1];
reference[0] = new AddFileReferenceFFType();
reference[0].setFileId(fileId);
reference[0].setFileName("FileThrowReference.txt");
reference[0].setDescription("file added using a reference");
reference[0].setFileSize( new Long(1) );
agileAddFileFFRequestType[0].setFileRefs( reference );
addFileFFRequestType.setRequest(agileAddFileFFRequestType);

```

### Example: Adding URLs to a File Folder

```

agileAddFileFFRequestType[0].setFolderNumber(folderNumber);
AddUrlFFType urls[] = new AddUrlFFType[1];
urls[0] = new AddUrlFFType();
urls[0].setUrl("http://www.testurl_filefolder.com");
urls[0].setDescription("Test url description");
agileAddFileFFRequestType[0].setUrls(urls);
addFileFFRequestType.setRequest(agileAddFileFFRequestType);

```

## Adding Files in a File Folder

The Files table of a file folder lists the files and URLs associated with the object. To edit the table, you must first check out the file folder. You cannot add files or URLs to the Files table or delete them unless the file folder is checked out.

### Example: Adding a URL in a File Folder

Addition of URL attachments to the 'Files' tab of a file folder object can be carried out by specifying the URL and folder number.

```

checkoutFolder(folderNumber);
agileAddFileFFRequestType[0].setFolderNumber(folderNumber);
AddUrlFFType urls[] = new AddUrlFFType[1];
urls[0] = new AddUrlFFType();
urls[0].setUrl("http://www.testurl_filefolder.com");
urls[0].setDescription("Test url description");
agileAddFileFFRequestType[0].setUrls(urls);
addFileFFRequestType.setRequest(agileAddFileFFRequestType);

```

## Managing Attachments

Attachments to objects contain information about the object or a manufacturing process. You can attach files and URLs by referencing them in a File Folder object. The File Folder object holds pertinent content, or Attachments. Most primary Agile API objects, such as Item, Change, Manufacturer, ManufacturerPart, Package, TransferOrder, User, and UserGroup, have an attachments table (or tab in the Java Client) that lists indirect references to the files or URLs that are in separate file folders. Each row in an Attachments table can refer to one file or to all files from a referenced file folder.

### Getting Attachments of an Object

A file attachment is retrieved using the operation `getFileAttachment`.

When the required file is present in a single, separate row, it is downloaded from the attachment tab using its `rowId`. In other cases, when there are several files in the same row and the desired file is one of them, the `fileId` must also be specified.

#### Example: Getting a single Attachment

```
agileGetFileAttachmentRequest[0].setClassIdentifier("Part");
agileGetFileAttachmentRequest[0].setObjectNumber(partNumber);
agileGetFileAttachmentRequest[0].setAllFiles(false);
AgileFileAttachmentRequestType attachments[] = new AgileFileAttachmentRequestType[1];
attachments[0] = new AgileFileAttachmentRequestType();
attachments[0].setRowId(rowId);
agileGetFileAttachmentRequest[0].setAttachments(attachments);
getFileAttachmentRequestType.setRequests(agileGetFileAttachmentRequest);
```

#### Example: Getting all the Attachments of an Object

To get all the attachments of an object, set `allFiles` to `True`.

```
agileGetFileAttachmentRequest[0].setClassIdentifier("Part");
agileGetFileAttachmentRequest[0].setObjectNumber(partNumber);
agileGetFileAttachmentRequest[0].setAllFiles(true);
AgileFileAttachmentRequestType attachments[] = new AgileFileAttachmentRequestType[1];
attachments[0] = new AgileFileAttachmentRequestType();
agileGetFileAttachmentRequest[0].setAttachments(attachments);
getFileAttachmentRequestType.setRequests(agileGetFileAttachmentRequest);
```

### Getting a Specific Attachment and a File Folder

When there are several files in the same row and the desired file is one of them, then a file can be downloaded from the attachment tab using its `fileId` along with the `rowId`. Using the `rowId` alone in this case is ineffective since all the files of that row will be obtained. To download only a specific file from such a set of files in a single row, the `fileId` is also needed.

For such cases, first set the `rowId` information, then obtain the `fileId` value and set the same into the 'files' element of the request object.



```
agileGetFileAttachmentRequest[0].setClassIdentifier("Part");
agileGetFileAttachmentRequest[0].setObjectNumber(partNumber);
agileGetFileAttachmentRequest[0].setAllFiles(false);
AgileFileAttachmentRequestType attachments[] = new AgileFileAttachmentRequestType[1];
attachments[0] = new AgileFileAttachmentRequestType();
attachments[0].setRowId(rowId);
int fileIds[] = new int[] { fileId1, fileId2 };
attachments[0].setFiles(fileIds);
agileGetFileAttachmentRequest[0].setAttachments(attachments);
GetFileAttachmentRequestType.setRequests(agileGetFileAttachmentRequest);
```

Instead of using the `fileId` for obtaining a file from a row with multiple files, you can also use the `rowId` of the files tab in the `fileFolder` object vis-a-vis the desired file.

To download only a specific file from a set of files in a single row in the Attachment tab, first set the `rowId` information, then obtain the `rowId` value of the corresponding `FileFolder` object of the file and set the same into the 'files' element of request object.

```
agileGetFileAttachmentRequest[0].setClassIdentifier("Part");
agileGetFileAttachmentRequest[0].setObjectNumber(partNumber);
agileGetFileAttachmentRequest[0].setAllFiles(false);
AgileFileAttachmentRequestType attachments[] = new AgileFileAttachmentRequestType[1];
attachments[0] = new AgileFileAttachmentRequestType();
attachments[0].setRowId(rowId);
int fileIds[] = new int[] { fileFolderRowId1, fileFolderRowId2 };
attachments[0].setFiles(fileIds);
agileGetFileAttachmentRequest[0].setAttachments(attachments);
GetFileAttachmentRequestType.setRequests(agileGetFileAttachmentRequest);
```

---

**Note** See [Appendix](#) on page 205 for sample helper methods.

---

## Getting a Specific Attachment using a URL

```
GetFileAttachmentRequestType GetFileAttachmentRequestType = new
GetFileAttachmentRequestType();
AgileGetFileAttachmentRequest agileGetFileAttachmentRequest[] = new
AgileGetFileAttachmentRequest[1];
agileGetFileAttachmentRequest[0] = new AgileGetFileAttachmentRequest();
agileGetFileAttachmentRequest[0].setClassIdentifier("Part");
agileGetFileAttachmentRequest[0].setObjectNumber(partNumber);
agileGetFileAttachmentRequest[0].setAllFiles(false);
agileGetFileAttachmentRequest[0].setDownloadUrl(true);
AgileFileAttachmentRequestType attachments[] = new AgileFileAttachmentRequestType[1];
attachments[0] = new AgileFileAttachmentRequestType();
attachments[0].setRowId(rowId);
agileGetFileAttachmentRequest[0].setAttachments(attachments);
GetFileAttachmentRequestType.setRequests(agileGetFileAttachmentRequest);
```

---

**Note** See [Appendix](#) on page 205 for sample helper methods.

---

## Adding Attachments to an Object

When you add a file or a URL to the Attachments table of a business object, the server automatically creates a new file folder containing the associated file or URL. The new row on the Attachments table references the new file folder. Use the operation `addFileAttachment` to add a File or a URL.

When you add a URL attachment, the server stores a reference to the Internet location but does not upload a file. Therefore, you cannot download a URL attachment. Agile Web Services validate URL strings that you attempt to check in as an attachment. If a URL is invalid, the Agile Web Services consider the string a filename instead of a URL.

You cannot add a file or URL to the Attachments table of an item if:

- The current revision has a pending or released MCO.
- The current revision is incorporated.

To add attachments, you have to specify the unique object to whose attachment tab the files will be added. Also, specify its class identifier and object number information.

The exact specification of the attachment to be added is defined as an object of type `AgileAddFileAttachmentRequestType`. This object includes information about the name of the file and its description and content.

### Example: Adding an Attachment to a Part

You can specify if multiple attachments should add into a single folder or multiple folders by supplying boolean value to `setSingleFolder`.

```
agileAddFileAttachmentRequest[0].setClassIdentifier("Part");
agileAddFileAttachmentRequest[0].setObjectNumber( partNumber );
AgileAddFileAttachmentRequestType attachments[] = new AgileAddFileAttachmentRequestType[1];
attachments[0] = new AgileAddFileAttachmentRequestType();
attachments[0].setName("Filename.txt");
attachments[0].setDescription("Description for file ");
attachments[0].setContent( "File Content...file".getBytes() );
agileAddFileAttachmentRequest[0].setAttachments(attachments);
agileAddFileAttachmentRequest[0].setSingleFolder(false);
addFileAttachmentRequestType.setRequests(agileAddFileAttachmentRequest);
```

## Adding attachments by File Reference

You can add an attachment to an Agile object by using a file reference for a file that has already been added to DFM. This reference is obtained from the DFM file server.

### Example: Adding an Attachment to a Part using its File Reference

```
agileAddFileAttachmentRequest[0].setClassIdentifier("Part");
agileAddFileAttachmentRequest[0].setObjectNumber(partNumber);
AgileAddFileAttachmentReferenceRequestType reference[] = new
AgileAddFileAttachmentReferenceRequestType[1];
reference[0] = new AgileAddFileAttachmentReferenceRequestType();
reference[0].setFileId( DFMfileId );
reference[0].setFileName("FileThrowReference.txt");
reference[0].setDescription("File added using a reference");
```

```
reference[0].setFileSize( new Long(1) );
agileAddFileAttachmentRequest[0].setAttachmentRefs (reference);
addFileAttachmentRequestType.setRequests (agileAddFileAttachmentRequest);
```

## Adding Multiple Attachments into Single Folder

Addition of several file attachments to an Agile object requires explicit specification that all the files added to the Agile object must be added to a single folder. The element 'singleFolder' is used to specify that all the attachments added to the object must be added under a single folder.

### Example: Adding multiple Attachments into a single folder

```
AddFileAttachmentRequestType addFileAttachmentRequestType = new
AddFileAttachmentRequestType();
AgileAddFileAttachmentRequest agileAddFileAttachmentRequest[] = new
AgileAddFileAttachmentRequest[1];
agileAddFileAttachmentRequest[0] = new AgileAddFileAttachmentRequest();
agileAddFileAttachmentRequest[0].setClassIdentifier("Part");
agileAddFileAttachmentRequest[0].setObjectNumber(partNumber);
AgileAddFileAttachmentRequestType attachments[] = new AgileAddFileAttachmentRequestType[2];
    for(int i=0; i<attachments.length; i++){
        attachments[i] = new AgileAddFileAttachmentRequestType();
        attachments[i].setName("Filename" + (i+1) + ".txt");
        attachments[i].setDescription("Description for file " + (i+1) );
        attachments[i].setContent( "File Content...file" + (i+1) ).getBytes() );
    }
agileAddFileAttachmentRequest[0].setAttachments(attachments);
agileAddFileAttachmentRequest[0].setSingleFolder(true);
addFileAttachmentRequestType.setRequests (agileAddFileAttachmentRequest);
```

## Adding Files using SOAP Attachment

For adding very large attachments to an Object, use SOAP attachments.

While using SOAP attachments we create a datahandler to specify the file source and add the add the content as a soap attachment to the soap request as shown in the following example. Finally the contentId is set onto AgileAddFileAttachmentRequestType.

```
agileAddFileAttachmentRequest[0].setClassIdentifier("Part");
agileAddFileAttachmentRequest[0].setObjectNumber( partNumber );
AgileAddFileAttachmentRequestType attachments[] = new AgileAddFileAttachmentRequestType[1];
attachments[0] = new AgileAddFileAttachmentRequestType();
attachments[0].setName("Filename.txt");
attachments[0].setDescription("Description for file ");

DataHandler dh = new DataHandler(new FileDataSource("c:\sample.txt") );
AttachmentPart ap = new AttachmentPart(dh);
agileStub.addAttachment(ap);
attachments[0].setContentId(ap.getContentId());
agileAddFileAttachmentRequest[0].setAttachments(attachments);
agileAddFileAttachmentRequest[0].setSingleFolder(false);
addFileAttachmentRequestType.setRequests (agileAddFileAttachmentRequest);
```

**Note** This feature is supported by four operations - addFileFF, addFileAttachment, checkInFF, checkInAttachment.

**Note** See also – sample method [AddFileSOAPAttachment](#) on page 207.

---

## Checking Out the Attachments

Checking out an Attachment process entails obtaining the file information and making changes to the same. The file information is also obtained in the response as an array of bytes. Checking out an attachment for any modifications is carried out with the checkOutAttachment.

### Example: Checking Out an Attachment of a Part

```
agileCheckOutAttachmentRequestType[0].setClassIdentifier("Part");
agileCheckOutAttachmentRequestType[0].setObjectNumber(partNumber);
CheckOutAttachmentType attachments[] = new CheckOutAttachmentType[1];
attachments[0] = new CheckOutAttachmentType();
attachments[0].setRowId(rowId);
agileCheckOutAttachmentRequestType[0].setAttachments(attachments);
checkOutAttachmentRequestType.setRequests(agileCheckOutAttachmentRequestType);
```

## Checking Out All the Attachments

You can check out all the attachments of an object by setting the boolean value of **allFiles** variable.

### Example: Checking Out all the Attachments of a Part

```
agileCheckOutAttachmentRequestType[0].setClassIdentifier("Part");
agileCheckOutAttachmentRequestType[0].setObjectNumber(partNumber);
agileCheckOutAttachmentRequestType[0].setAllFiles(true);
CheckOutAttachmentType attachments[] = new CheckOutAttachmentType[1];
attachments[0] = new CheckOutAttachmentType();
agileCheckOutAttachmentRequestType[0].setAttachments(attachments);
checkOutAttachmentRequestType.setRequests(agileCheckOutAttachmentRequestType);
```

## Checking Out Multiple Attachments from a Folder

When multiple files are associated with a single row, the attachment is identified by using its rowId in conjunction with the rowId of the attachment in the files tab of the file folder object. It requires usage of the rowId of the attachment along with its fileId to distinguish the attachment from all the other attachments in that row.

### Example: Checking Out Multiple Attachments of a Part

```
agileCheckOutAttachmentRequestType[0].setClassIdentifier("Part");
agileCheckOutAttachmentRequestType[0].setObjectNumber(partNumber);
CheckOutAttachmentType attachments[] = new CheckOutAttachmentType[1];
attachments[0] = new CheckOutAttachmentType();
attachments[0].setRowId(rowId);
int fileIds[] = new int[] {fileId};
attachments[0].setFiles(fileIds);
agileCheckOutAttachmentRequestType[0].setAttachments(attachments);
checkOutAttachmentRequestType.setRequests(agileCheckOutAttachmentRequestType);
```

---

**Note** See [Appendix](#) on page 205 for sample helper methods.

---

## Checking In the Attachments

You can Check-In a file attachment of an Agile object after it has undergone any modifications using the operation `checkInAttachment`. The attachment must be checked out prior to the 'check in' operation.

### Example: Checking In an Attachment to a Part

```
CheckInAttachmentRequestType checkInAttachmentRequestType = new
CheckInAttachmentRequestType();
AgileCheckInAttachmentRequestType agileCheckInAttachmentRequestType[] = new
AgileCheckInAttachmentRequestType[1];
agileCheckInAttachmentRequestType[0] = new AgileCheckInAttachmentRequestType();
agileCheckInAttachmentRequestType[0].setClassIdentifier("Part");
agileCheckInAttachmentRequestType[0].setObjectNumber(partNumber);
CheckInAttachmentType attachments[] = new CheckInAttachmentType[1];
attachments[0] = new CheckInAttachmentType();
attachments[0].setFileContent("Modified file information added after the checkin".getBytes());
attachments[0].setFileName("Modified_" + fileName);
agileCheckInAttachmentRequestType[0].setAttachments(attachments);
agileCheckInAttachmentRequestType[0].setRowId(rowId);
checkInAttachmentRequestType.setRequest(agileCheckInAttachmentRequestType);
```

---

**Note** See sample [helper methods](#).

---

## Checking In an Attachment with FileId Identification

In the normal course of usage, `rowId` will prove to be sufficient in identifying the file. However, if the file that has been checked out is part of a row that contains multiple files, then `fileId` is essential to identify that particular file. For such cases, you have to use its `fileId` in conjunction with its `rowId`.

### Example: Checking In an Attachment using Field ID

```
CheckInAttachmentRequestType checkInAttachmentRequestType = new
CheckInAttachmentRequestType();
AgileCheckInAttachmentRequestType agileCheckInAttachmentRequestType[] = new
AgileCheckInAttachmentRequestType[1];
agileCheckInAttachmentRequestType[0] = new AgileCheckInAttachmentRequestType();
agileCheckInAttachmentRequestType[0].setClassIdentifier("Part");
agileCheckInAttachmentRequestType[0].setObjectNumber(partNumber);
CheckInAttachmentType attachments[] = new CheckInAttachmentType[1];
attachments[0] = new CheckInAttachmentType();
attachments[0].setFileName("Modified_" + fileName[0]);
attachments[0].setFileContent("Modified file information added after the checkin".getBytes());
attachments[0].setFileId(fileId);
agileCheckInAttachmentRequestType[0].setAttachments(attachments);
agileCheckInAttachmentRequestType[0].setRowId(rowId);
checkInAttachmentRequestType.setRequest(agileCheckInAttachmentRequestType);
```

---

**Note** See sample [helper methods](#).

---

## Deleting the Attachments

Deleting an attachment is carried out using the operation `removeRows`.

See [Removing Rows from a Table](#) on page 41 for examples.

# Managing Workflows

**This chapter includes the following:**

---

▪ Getting the Status of a Workflow .....	78
▪ Getting the Workflow of a Routable Object.....	78
▪ Setting a Workflow .....	78
▪ Checking User Privileges.....	79
▪ Adding and Removing Approvers .....	79
▪ Getting Approvers .....	80
▪ Approving a Routable Object.....	80
▪ Rejecting a Routable Object .....	81
▪ Commenting a Change .....	81
▪ Auditing a Change .....	82
▪ Changing the Workflow Status of an Object .....	82

This chapter describes how to manage the Agile PLM workflows and provides sample code snippets.

## About Agile PLM Workflows

Agile has electronic routing, notification, and signoff capabilities, thus automating the change control process and providing a simplified but powerful workflow mechanism. With these workflow features, you can:

- Route changes automatically to the users who need to approve or observe the change.
- Send email alerts automatically to approvers and observers to notify them that a change has been routed to them.
- Approve or reject changes online.
- Attach comments to changes.

The workflow functionality available to each user for a particular routable object depends on the status of the routable object and the user's privileges. Your Agile API program should take these workflow dynamics into account and, where possible, adjust your program accordingly.

## How the Status of a Change Affects Workflow Functionality

The workflow actions available for a pending change are different from those for a released change. To check the status of a change to determine whether it's pending or released, use the operation `getStatus`. This operation returns an object for the workflow status.

## Getting the Status of a Workflow

The workflow actions available for a pending change are different from those for a released change. To check the status of a change to determine whether it's pending or released, use the operation `getStatus`. This operation returns **statusName** value for the workflow status, which are Pending, Submitted, Released, etc.

### Example: Getting the status of a change object

```
GetStatusRequestType getStatusRequestType = new GetStatusRequestType();
AgileGetStatusRequestType agileGetStatusRequestType[] = new AgileGetStatusRequestType[1];
agileGetStatusRequestType[0] = new AgileGetStatusRequestType();
agileGetStatusRequestType[0].setClassIdentifier("ECO");
agileGetStatusRequestType[0].setObjectNumber( changeNumber );
getStatusRequestType.setStatusRequest( agileGetStatusRequestType );
```

## Getting the Workflow of a Routable Object

When you create a new change, package, product service request, or quality change order, you must select a workflow. Otherwise, the object is in an unassigned state and cannot progress through a workflow process. Your Agile system can have multiple workflows defined for each type of routable object.

To get the valid workflows for a routable object, which has not yet been assigned a workflow, use the operation `getWorkFlows`.

### Example: Getting a Workflow

```
GetWorkflowsRequestType getWorkflowsRequestType = new GetWorkflowsRequestType();
AgileGetWorkflowsRequestType agileGetWorkflowsRequestType[] = new
AgileGetWorkflowsRequestType[1];
agileGetWorkflowsRequestType[0] = new AgileGetWorkflowsRequestType();
agileGetWorkflowsRequestType[0].setClassIdentifier("ECO");
agileGetWorkflowsRequestType[0].setObjectNumber( changeNumber );
getWorkflowsRequestType.setWorkflowRequest( agileGetWorkflowsRequestType );
```

## Setting a Workflow

If a change is still in the Pending state, you can deselect a workflow to make the change “unassigned” using the operation `setWorkFlow` and specifying the **setWorkFlowIdentifier** parameter.

As long as a change is in the Pending status, you can select a different workflow. Once a change moves beyond the Pending status, you can't change the workflow.

### Example: Setting a WorkFlow

```
SetWorkFlowRequestType setWorkFlowRequestType = new SetWorkFlowRequestType();
AgileSetWorkFlowRequestType agileSetWorkFlowRequestType[] = new
AgileSetWorkFlowRequestType[1];
agileSetWorkFlowRequestType[0] = new AgileSetWorkFlowRequestType();
agileSetWorkFlowRequestType[0].setClassIdentifier("ECO");
agileSetWorkFlowRequestType[0].setObjectNumber( changeNumber );
```



```
agileSetWorkFlowRequestType[0].setWorkFlowIdentifier( workflow );
setWorkFlowRequestType.setSetWorkFlowRequest( agileSetWorkFlowRequestType );
```

## Checking User Privileges

Agile privileges determine the types of workflow actions a user can perform on a Change Object. The Agile system administrator assigns roles and privileges to each user. Table below lists privileges needed to perform workflow actions.

Privilege	Related operation
Change Status	changeStatus
Comment	commentRObjct
Send	sendObject

To determine at run time whether a user has the appropriate privileges to perform a particular action, use the operation `checkPrivilege`. It yields a Boolean value indicating if the user has the specified privilege.

Refer Schema documentation on Oracle eDelivery Site for `AgilePrivilegeType` enumerations.

### Example: Checking the privileges of a user

```
AgileUserIdentifierType user = new AgileUserIdentifierType();
user.setUserIdentifier("admin");
agileCheckPrivilegeRequestType[0].setUserIdentification(user);
AgilePrivilegeType privilege = AgilePrivilegeType.value1;
agileCheckPrivilegeRequestType[0].setPrivilege(privilege);
agileCheckPrivilegeRequestType[0].setClassIdentifier("Part");
agileCheckPrivilegeRequestType[0].setObjectNumber( partNumber );
checkPrivilegeRequestType.setRequests( agileCheckPrivilegeRequestType );
```

## Adding and Removing Approvers

After a change has been routed and the online approval process has begun, it may be necessary to add or remove people from the list of approvers or observers. To add or remove approvers or observers, a user must have both the Agile Product Change Server license and the Route privilege.

You don't need to load the Workflow table to modify the list of approvers. Once you have a routable object, such as an ECO, you can modify its list of approvers using the operation `addApprovers` and the operation `removeApprovers`. With these operations, you can specify the lists of approvers and observers, whether the notification is urgent, and an optional comment. The Agile Web Services provides overloaded operations for adding or removing a user or a user group from the list of approvers.

If any users you select as approvers or observers do not have appropriate privileges to view a change, your program throws an [Exception](#) on page 20. To avoid the possible exception, check the privileges (`checkPrivilege`) of each user before adding him to the approvers or observers list.

### Example: Adding an Approver or Observer

```
agileAddApproversRequestType[0].setClassIdentifier("ECO");
agileAddApproversRequestType[0].setObjectNumber( changeNumber );
```

```
agileAddApproversRequestType[0].setStatusIdentifier("CCB");
    AgileUserUserGroupIdentifierType users[] = new AgileUserUserGroupIdentifierType[2];
    users[0].setClassIdentifier("User");
    users[0].setObjectIdentifier(user1);
    users[1].setClassIdentifier(User);
    users[1].setObjectIdentifier(user2);
    agileAddApproversRequestType[0].setApprovers(users);
    agileAddApproversRequestType[0].setObservers(null);
    agileAddApproversRequestType[0].setUrgent(false);
    agileAddApproversRequestType[0].setComment("Comments");
    addApproversRequestType.setAddApproversRequest(agileAddApproversRequestType);
```

### Example: Removing an Approver or Observer

```
agileRemoveApproversRequestType[0].setClassIdentifier("ECO");
agileRemoveApproversRequestType[0].setObjectNumber( changeNumber );
agileRemoveApproversRequestType[0].setStatusIdentifier("CCB");
    AgileUserUserGroupIdentifierType usergroups[] = new AgileUserUserGroupIdentifierType[1];
    usergroups[0].setClassIdentifier("User");
    usergroups[0].setObjectIdentifier( USERNAME );
    agileRemoveApproversRequestType[0].setApprovers(usergroups);
    agileRemoveApproversRequestType[0].setObservers(null);
    agileRemoveApproversRequestType[0].setComment("Comments");
    removeApproversRequestType.setRemoveApproversRequest(agileRemoveApproversRequestType);
```

## Getting Approvers

Set the `statusIdentifier` in the operation `getApprovers` to obtain the list of approvers.

### Example: Getting Approvers for an Object

```
agileGetApproversRequestType[0].setClassIdentifier("ECO");
agileGetApproversRequestType[0].setObjectNumber( changeNumber );
agileGetApproversRequestType[0].setStatusIdentifier( status );
getApproversRequestType.setApproversRequest(agileGetApproversRequestType);
```

## Approving a Routable Object

This method informs users the object is approved by the approver, or when the approver is approving the object on behalf of one or more user groups. You can also use this method to specify the `secondSignature`, `escalations`, `transfers`, or `signoffForSelf` parameters as they are set in server's Preferences settings. Use the operation `approveRObj`.

### Example: Approving a Routable Object and notifying the users

```
AgileUserUserGroupIdentifierType notifiers[] = new AgileUserUserGroupIdentifierType[1];
notifiers[0] = new AgileUserUserGroupIdentifierType();
notifiers[0].setClassIdentifier("User");
notifiers[0].setObjectIdentifier( notifier1 );
AgileApproveRObjRequestType agileApproveRObjRequestType[] = new
AgileApproveRObjRequestType[1];
agileApproveRObjRequestType[0] = new AgileApproveRObjRequestType();
    agileApproveRObjRequestType[0].setClassIdentifier("ECO");
```

```

agileApproveObjectRequestType[0].setObjectNumber( changeNumber );
agileApproveObjectRequestType[0].setPassword( PASSWORD );
agileApproveObjectRequestType[0].setComment("Comment");
agileApproveObjectRequestType[0].setSecondSignature(null);
agileApproveObjectRequestType[0].setNotifiers(notifiers);
agileApproveObjectRequestType[0].setEscalations(null);
agileApproveObjectRequestType[0].setTransfers(null);
agileApproveObjectRequestType[0].setApproveForGroup(null);
agileApproveObjectRequestType[0].setSignoffForSelf(true);
approveObjectRequestType.setApproveObject( agileApproveObjectRequestType );

```

## Rejecting a Routable Object

This method informs users that the routable object is rejected by the approver, or when the approver is rejecting the object on behalf of one or more user groups. You can also use this method to specify the secondSignature, escalations, transfers, or SignoffForSelf parameters as they are set in server's Preferences settings. Use the operation rejectRObject.

```

Rejecting a Routable Object and notifying the users
agileRejectObjectRequestType[0].setClassIdentifier("ECO");
agileRejectObjectRequestType[0].setObjectNumber( changeNumber );
agileRejectObjectRequestType[0].setPassword( PASSWORD );
agileRejectObjectRequestType[0].setComment("Comment");
agileRejectObjectRequestType[0].setSecondSignature(null);
    AgileUserUserGroupIdentifierType notifiers[] = new AgileUserUserGroupIdentifierType[1];
    notifiers[0] = new AgileUserUserGroupIdentifierType();
    notifiers[0].setClassIdentifier("User");
    notifiers[0].setObjectIdentifier( notifier1 );
    agileRejectObjectRequestType[0].setNotifiers(notifiers);
    agileRejectObjectRequestType[0].setEscalations(null);
    agileRejectObjectRequestType[0].setTransfers(null);
    agileRejectObjectRequestType[0].setRejectForGroups(null);
    agileRejectObjectRequestType[0].setSignoffForSelf(true);
    rejectObjectRequestType.setRejectRObject( agileRejectObjectRequestType );

```

## Commenting a Change

Use the operation CommentRObject operation to comment a change. Use boolean variables to denote whether the originators, change analysts and CCB need to be notified.

```

agileCommentObjectRequestType[0].setClassIdentifier("ECO");
agileCommentObjectRequestType[0].setObjectNumber( changeNumber );
agileCommentObjectRequestType[0].setComment("Comment");
agileCommentObjectRequestType[0].setNotifyOriginator(true);
agileCommentObjectRequestType[0].setNotifyChangeAnalyst(true);
agileCommentObjectRequestType[0].setNotifyCCB(false);
AgileUserUserGroupIdentifierType notifyList[] = new AgileUserUserGroupIdentifierType[1];
notifyList[0] = new AgileUserUserGroupIdentifierType();
notifyList[0].setClassIdentifier("User");
notifyList[0].setObjectIdentifier( USERNAME );
agileCommentObjectRequestType[0].setNotifyList(notifyList);
commentObjectRequestType.setCommentRObjectRequest( agileCommentObjectRequestType );

```

## Auditing a Change

Auditing a routable object, like an ECO, requires specifying the type of routable object in the operation AuditRObj.

```
agileAuditRObjRequestType[0].setClassIdentifier("ECO");
agileAuditRObjRequestType[0].setObjectNumber( changeNumber );
agileAuditRObjRequestType[0].setAuditRelease(true);
auditRObjRequestType.setRequest(agileAuditRObjRequestType);
```

## Changing the Workflow Status of an Object

Use the operation changeStatus to change the workflow status of an object.

```
agileChangeStatusRequestType[0].setClassIdentifier("ECO");
agileChangeStatusRequestType[0].setObjectNumber( changeNumber );
agileChangeStatusRequestType[0].setNewStatusIdentifier( newStatus );
AgileUserUserGroupIdentifierType users[] = new AgileUserUserGroupIdentifierType[1];
    users[0] = new AgileUserUserGroupIdentifierType();
    users[0].setClassIdentifier("User");
    users[0].setObjectIdentifier( user1 );
agileChangeStatusRequestType[0].setApprovers(users);
agileChangeStatusRequestType[0].setObservers(null);
agileChangeStatusRequestType[0].setNotifiers(null);
agileChangeStatusRequestType[0].setComment("Comments");
agileChangeStatusRequestType[0].setPassword("password");
agileChangeStatusRequestType[0].setAuditRelease(false);
agileChangeStatusRequestType[0].setUrgent(false);
agileChangeStatusRequestType[0].setNotifyOriginator(true);
agileChangeStatusRequestType[0].setNotifyChangeAnalyst(true);
agileChangeStatusRequestType[0].setNotifyCCB(true);
changeStatusRequestType.setChangeStatusRequest(agileChangeStatusRequestType);
```

## Reference

# Core Operations

This section describes the Core Web Services Operations. Download Agile Web Services Schema Docs from [Oracle eDelivery Web Site](http://edelivery.oracle.com) edelivery.oracle.com for the details of all the operations.



## Admin and Metadata Web Services

This chapter includes the following:

▪ getAllClasses .....	85
▪ getSubClasses .....	87
▪ getNode .....	89
▪ getLists .....	91
▪ getAttributes .....	93
▪ getTableMetadata .....	95
▪ getAutoNumbers .....	97
▪ getUsers .....	99
▪ getUserGroups .....	101
▪ convertCurrency .....	102

### getAllClasses

**Service** To retrieve all Agile classes from Agile PLM system.

**Usage** The class filtering details are specified in the request object. A list of Agile classes retrieved as per the filter is obtained in the response.

**Basic Steps** To get all classes:

1. Create the request object for the getAllClasses operation.
2. Set the 'level' for the getAllClasses request object. This filter is specified by using a ClassFilterType object which can hold values of 'TOP', 'CONCRETE' or 'ALL'. The ClassFilterType.ALL, ClassFilterType.TOP or ClassFilterType.CONCRETE may be used.
3. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
4. If the Web Service call was successful, then display the list of classes retrieved.

#### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getAllClasses
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <request xmlns="">
          <level>ALL</level>
        </request>
      </getAllClasses>
    </soapenv:Body>
  </soapenv:Envelope>
</pre>

```

```
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getAllClassesResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <class>
            <nodeId>931</nodeId>
            <apiName>ChangesBaseClass</apiName>
            <typeCLASS</type>
            <displayName>Changes</displayName>
            <abstractClass>true</abstractClass>
            <subClass>
              <nodeId>1450</nodeId>
              <apiName>ManufacturerOrdersClass</apiName>
              <typeSUBCLASS</type>
              <displayName>Manufacturer Orders</displayName>
            </subClass>
            <subClass>
              <nodeId>6000</nodeId>
              <apiName>ChangeOrdersClass</apiName>
              <typeSUBCLASS</type>
              <displayName>Change Orders</displayName>
            </subClass>
          </class>
          ... <!-- additional classes -->
        </response>
      </getAllClassesResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      [getSubClasses](#)



## getSubClasses

<b>Service</b>	To retrieve all Agile subclasses for a given base class from Agile PLM system.
<b>Usage</b>	The request object contains relevant details for the same. A list of Agile subclasses are obtained in the response.
<b>Basic Steps</b>	<p>To retrieve all subclasses:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the createObject operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to retrieve several types of subclasses.</li> <li>3. For each batched request, specify the type of object whose subclasses will be retrieved.</li> <li>4. The request objects are set and the Agile Stub (in Java) is used to make the getSubClasses Web Service call. The status code obtained from the response object is printed to verify the success of the getSubClasses operation.</li> <li>5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>6. If the Web Service call was successful, then display the list of subclasses retrieved.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getSubClasses
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <request xmlns="">
          <subClassesRequest>
            <classIdentifier>Changes</classIdentifier>
          </subClassesRequest>
          <subClassesRequest>
            <classIdentifier>Items</classIdentifier>
          </subClassesRequest>
        </request>
      </getSubClasses>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getSubClassesResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </subClassesResponse>
      </getSubClassesResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

```
</classIdentifier>Changes</classIdentifier>
<classes>
  <nodeId>1450</nodeId>
  <apiName>ManufacturerOrdersClass</apiName>
  <typeSUBCLASS</type>
  <displayName>Manufacturer Orders</displayName>
  <abstractClass>true</abstractClass>
  <superClass>
    <nodeId>931</nodeId>
    <apiName>ChangesBaseClass</apiName>
    <typeCLASS</type>
    <displayName>Changes</displayName>
  </superClass>
</classes>
<classes>
  <nodeId>6000</nodeId>
  <apiName>ChangeOrdersClass</apiName>
  <typeSUBCLASS</type>
  <displayName>Change Orders</displayName>
  <abstractClass>true</abstractClass>
  <superClass>
    <nodeId>931</nodeId>
    <apiName>ChangesBaseClass</apiName>
    <typeCLASS</type>
    <displayName>Changes</displayName>
  </superClass>
</classes>
</subClassesResponse>
</response>
</getSubClassesResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      [getAllClasses](#)

## getNode

<b>Service</b>	To retrieve Agile nodes for a given node identifier.
<b>Usage</b>	The request object contains relevant details for the same. The queried node is obtained in the response.
<b>Basic Steps</b>	<p>To retrieve nodes for a given node identifier:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the createObject operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to retrieve several nodes.</li> <li>3. For each batched request, specify the string that identifies the node and whether children of a given node will also be retrieved recursively.</li> <li>4. The request objects are set and the Agile Stub is used to make the getNode Web Service call. The status code obtained from the response object is printed to verify the success of the getNode operation.</li> <li>5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>6. If the Web Service call was successful, then display the list of nodes retrieved.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getNode
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <request xmlns="">
          <nodeRequest>
            <nodeIdentifier>5009</nodeIdentifier>
            <recursive>true</recursive>
          </nodeRequest>
        </request>
      </getNode>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getNodeResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <nodeResponse>
            <node>
              <nodeId>5009</nodeId>
              <apiName>AutoNumbers</apiName>
              <typeNODE</type>
            </node>
          </nodeResponse>
        </response>
      </getNodeResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

```
</displayName>AutoNumbers</displayName>
<childNodes>
  <nodeId>990</nodeId>
  <apiName>ECONumber</apiName>
  typeNODE</type>
  <displayName>ECO Number</displayName>
  <Properties>
    <propertyId>30</propertyId>
    <apiName>Name</apiName>
    <displayName>Name</displayName>
    <readOnly>false</readOnly>
    <Name xsi:type="xs:string"
Number</Name>
    xmlns:xs="http://www.w3.org/2001/XMLSchema">ECO
  </Properties>
  <Properties>
    <propertyId>38</propertyId>
    <apiName>Description</apiName>
    <displayName>Description</displayName>
    <readOnly>false</readOnly>
    <Description xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema">ECO Number</Description>
  </Properties>
</childNodes>
</node>
</nodeResponse>
</response>
</getNodeResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      [getAttributes](#), [getTableMetadata](#), [getLists](#)

## getLists

<b>Service</b>	To retrieve Agile Lists for a given List Identifier.
<b>Usage</b>	The request object contains relevant details for the same. The retrieved lists are obtained in the response.
<b>Basic Steps</b>	<p>To get lists for a given list identifier:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the getLists operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to retrieve lists.</li> <li>3. For each batched request, specify the type of object whose lists will be retrieved.</li> <li>4. Setting the <code>allLists</code> field to true will retrieve all available lists.</li> <li>5. The request objects are set and the Agile Stub is used to make the getLists Web Service call. The status code obtained from the response object is printed to verify the success of the getLists operation.</li> <li>6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>7. If the Web Service call was successful, then display the lists retrieved.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getLists
xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
      <request xmlns="">
        <listsRequest>
          <listIdentifier>PartCategory</listIdentifier>
        </listsRequest>
      </request>
    </getLists>
  </soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getListsResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
        <statusCode>SUCCESS</statusCode>
        <listsResponse>
          <list>
            <id>311</id>
            <apiName>PartCategory</apiName>
          </list>
        </listsResponse>
      </response>
    </getListsResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

```
<displayName>Part_Getresponse</displayName>
typeSIMPLELIST</type>
<value xsi:nil="true"/>
<entry>
  <id>2</id>
  <apiName>ELECTRICAL</apiName>
  typeSIMPLELIST</type>
  <value>Electrical</value>
</entry>
<entry>
  <id>1</id>
  <apiName>MECHANICAL</apiName>
  typeSIMPLELIST</type>
  <value>Mechanical</value>
</entry>
</list>
<listIdentifier>311</listIdentifier>
</listsResponse>
</response>
</getListsResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      [getNode](#), [getTableMetadata](#), [getAttributes](#)

## getAttributes

<b>Service</b>	To retrieve Agile attributes for a particular Class and Attribute Identifier.
<b>Usage</b>	The request object contains relevant details for the same. The retrieved list attributes are obtained in the response.
<b>Basic Steps</b>	<p>To retrieve attributes:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the getAttributes operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to retrieve several attributes.</li> <li>3. For each batched request, specify the type of object whose attributes will be retrieved. Also specify the attribute which will be retrieved.</li> <li>4. The request objects are set and the Agile Stub is used to make the getAttributes Web Service call. The status code obtained from the response object is printed to verify the success of the getAttributes operation.</li> <li>5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>6. If the Web Service call was successful, then display the attributes retrieved.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getAttributes
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <request xmlns="">
          <attributesRequests>
            <classIdentifier>Specification</classIdentifier>
            <attributeIdentifier>description</attributeIdentifier>
          </attributesRequests>
        </request>
      </getAttributes>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getAttributesResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <attributesResponses>
            <attributes>
              <nodeId>2000001968</nodeId>
              <apiName>description</apiName>
              <type>ATTRIBUTE</type>
            </attributes>
          </attributesResponses>
        </response>
      </getAttributesResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

```

</display:Name>Description</display:Name>
<dataType>2</dataType>
<searchable>true</searchable>
<visible>true</visible>
<required>false</required>
<maxLength>100</maxLength>
<properties>
  <propertyId>1</propertyId>
  <apiName>AttType</apiName>
  <displayName>AttType</displayName>
  <readOnly>false</readOnly>
  <AttType xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"></AttType>
</properties>
<properties>
  <propertyId>3</propertyId>
  <apiName>Max System Length</apiName>
  <displayName>Max System Length</displayName>
  <readOnly>true</readOnly>
  <MaxSystemLength xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema">500</MaxSystemLength>
</properties>
<relationalOperators>EQ</relationalOperators>
<relationalOperators>NEQ</relationalOperators>
<relationalOperators>ISNULL</relationalOperators>
<relationalOperators>ISNOTNULL</relationalOperators>
<relationalOperators>LIKE</relationalOperators>
<relationalOperators>NOTLIKE</relationalOperators>
<relationalOperators>STARTSWITH</relationalOperators>
<relationalOperators>NOTSTARTSWITH</relationalOperators>
<relationalOperators>CONTAINS</relationalOperators>
<relationalOperators>NOTCONTAINS</relationalOperators>
</attributes>
<classIdentifier>Specification</classIdentifier>
</attributesResponses>
</response>
</getAttributesResponse>
</soapenv:Body>
</soapenv:Envelope>

```

**See also**      [getLists](#), [getNode](#), [getTableMetadata](#)



## getTableMetadata

<b>Service</b>	To retrieve the metadata information of an Agile table in the PLM system
<b>Usage</b>	The request object contains <code>classIdentifier</code> and <code>tableIdentifier</code> . The table metadata information retrieved is obtained through the response.
<b>Basic Steps</b>	<p>To retrieve the table metadata:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the <code>getTableMetaData</code> operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating several requests to obtain metadata information about several tables in one go.</li> <li>3. For each batched request, specify the type of object whose attributes will be retrieved. Also specify the attribute that has to be retrieved.</li> <li>4. The request objects are set and the Agile Stub is used to make the <code>getTableMetaData</code> Web Service call. The status code obtained from the response object is printed to verify the success of the <code>getTableMetaData</code> operation.</li> <li>5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>6. If the Web Service call was successful, then display the attributes retrieved.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getTableMetadata
xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
      <request xmlns="">
        <requests>
          <classIdentifier>Part</classIdentifier>
          <tableIdentifier>807</tableIdentifier>
        </requests>
        <requests>
          <classIdentifier>ECO</classIdentifier>
          <tableIdentifier>808</tableIdentifier>
        </requests>
      </request>
    </getTableMetadata>
  </soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getTableMetadataResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
      </response>
    </getTableMetadataResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

```
<statusCode>SUCCESS</statusCode>
<responses>
  <attributes>
    <nodeId>3669</nodeId>
    <apiName>list19</apiName>
    typeATTRIBUTE</type>
    <displayName>List19</displayName>
    <dataType>4</dataType>
    <possibleValues>
      <id xsi:nil="true"/>
      <apiName>list19</apiName>
      <displayName>List19</displayName>
      typeSIMPLELIST</type>
      <value xsi:nil="true"/>
    </possibleValues>
    <searchable>true</searchable>
    <visible>false</visible>
    <required>false</required>
    <maxLength>2147483647</maxLength>
    <properties>
      <propertyId>1</propertyId>
      <apiName>AttType</apiName>
      <displayName>AttType</displayName>
      <readOnly>false</readOnly>
      <AttType xsi:type="xs:string"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"></AttType>
    </properties>
    <properties>
      <propertyId>5</propertyId>
      <apiName>DefaultValue</apiName>
      <displayName>DefaultValue</displayName>
      <readOnly>false</readOnly>
      <DefaultValue xsi:type="common:AgileListEntryType"
        xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"/>
    </properties>
  </responses>
</response>
</getTableMetadataResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      [getAttributes](#)

## getAutoNumbers

<b>Service</b>	To retrieve a suitable AutoNumber for an Agile object.
<b>Usage</b>	The request object contains the Class and AutoNumber identifiers of the object. The AutoNumber for the object fetched by the Web Service is obtained through the response.
<b>Basic Steps</b>	<p>To retrieve an AutoNumber:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the getAutoNumbers operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating several requests to obtain several AutoNumbers simultaneously.</li> <li>3. For each batched request, specify the type of object for which AutoNumbers have to be obtained.</li> <li>4. Set <code>includeAllAutoNumberSource</code> boolean field to use all available AutoNumber sources. If this boolean is set to <code>false</code>, then a unique <code>autoNumberIdentifier</code> should be specified.</li> <li>5. Also specify the number of AutoNumbers required by setting the <code>size</code> attribute.</li> <li>6. If an AutoNumber source is available, for example, from a custom AutoNumber source, then instead of including all available AutoNumber sources, include only a particular set of AutoNumbers by specifying <code>AutoNumberIdentifiers</code> as an array of string values.</li> <li>7. The request objects are set and the Agile Stub is used to make the AutoNumber Web Service call. The status code obtained from the response object is printed to verify the success of the getAutoNumbers operation.</li> <li>8. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>9. If the Web Service call was successful, then display the AutoNumbers retrieved through the SOAP response and based on the number of AutoNumbers requested, iterate through the response and display results.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getAutoNumbers
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <request xmlns="">
          <requests>
            <classIdentifier>Part</classIdentifier>
            <includeAllAutoNumberSource>true</includeAllAutoNumberSource>
            <size>3</size>
          </requests>
        </request>
      </getAutoNumbers>
    </soapenv:Body>
  </soapenv:Envelope>

```

```

        <classIdentifier>ECO</classIdentifier>
        <includeAllAutoNumberSource>true</includeAllAutoNumberSource>
        <size>2</size>
      </requests>
    </request>
  </getAutoNumbers>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getAutoNumbersResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <autoNumberResponses>
            <classIdentifier>Part</classIdentifier>
            <autoNumbers>
              <nodeId>12416</nodeId>
              <apiName>PartNumber</apiName>
              <type>AUTONUMBER</type>
              <displayName>Part Number</displayName>
              <autoNumber>P00580</autoNumber>
              <autoNumber>P00581</autoNumber>
              <autoNumber>P00582</autoNumber>
            </autoNumbers>
          </autoNumberResponses>
          <autoNumberResponses>
            <classIdentifier>ECO</classIdentifier>
            <autoNumbers>
              <nodeId>990</nodeId>
              <apiName>ECONumber</apiName>
              <type>AUTONUMBER</type>
              <displayName>ECO Number</displayName>
              <autoNumber>C00186</autoNumber>
              <autoNumber>C00187</autoNumber>
            </autoNumbers>
          </autoNumberResponses>
        </response>
      </getAutoNumbersResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

**See also**      `getAllClasses`, `getSubClasses`

## getUsers

<b>Service</b>	To retrieve the information of Agile PLM Users.
<b>Usage</b>	Obtains a list of users through the response object of the Web Service. The request object does not contain any element while the response consists of <code>AgileUserType</code> objects, which contain message elements carrying information pertaining to an Agile user.
<b>Basic Steps</b>	<p>To retrieve user information:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the <code>getUsers</code> operation.</li> <li>2. The request object for the <code>getUsers</code> operation does not contain any element. A direct Web Service call is made using this request object.</li> <li>3. The <code>getUsers</code> Web Service call is made. The status code obtained from the response object is printed to verify the success of the <code>getUsers</code> operation.</li> <li>4. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>5. If the Web Service call was successful, then display the users retrieved.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getUsers
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <request xmlns=""/>
      </getUsers>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getUsersResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <users>
            <objectIdentifier>
              <classId>11610</classId>
              <className>User</className>
              <classDisplayName>User</classDisplayName>
              <objectId>704</objectId>
              <objectName>admin</objectName>
            </objectIdentifier>
            <userID xsi:type="xs:string"
              xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11617"
              readOnly="False">admin</userID>
          </users>
        </response>
      </getUsersResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

```

        <status xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="11643" readOnly="False">
        <selection>
        <id>1</id>
        <apiName>ACTIVE</apiName>
        <value>Active</value>
        </selection>
        </status>
        <firstName xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11614"
readOnly="False">admin</firstName>
        <lastName xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11616"
readOnly="False">Administrator</lastName>
        <title xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="12235"
readOnly="False"></title>
        <address xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11625"
readOnly="False"></address>
        <geography xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="8840" readOnly="False"/>
        <city xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11626"
readOnly="False"></city>
        <postalZipCode xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11629"
readOnly="False"></postalZipCode>
        <businessPhone xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11619"
readOnly="False"></businessPhone>
        <homePhone xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11620"
readOnly="False"></homePhone>
        <mobilePhone xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11621"
readOnly="False"></mobilePhone>
        <fax xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11622"
readOnly="False"></fax>
        <pager xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11623"
readOnly="False"></pager>
        <email xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="11624"
readOnly="False">admin@admin.com</email>
        <secondaryEmail xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="12350"
readOnly="False"></secondaryEmail>
        <profile xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="8815"
readOnly="True"></profile>
        <roleS xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="11640" readOnly="False">
        </users>
    </getUsersResponse>
</soapenv:Body>
</soapenv:Envelope>

```

**See also**      [getUserGroups](#)

## getUserGroups

<b>Service</b>	To retrieve the information of Users Groups in Agile PLM.
<b>Usage</b>	Gets a list of user groups obtained through the response object of the Web Service. The request does not contain any element while the response consists of <code>AgileUserGroupType</code> objects, which carry message elements that contain information pertaining to an Agile user group.
<b>Basic Steps</b>	<p>To get the User Groups:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the <code>getUserGroups</code> operation.</li> <li>2. The request object for the <code>getUserGroups</code> operation does not contain any element. The Web Service call is made directly using this request object.</li> <li>3. The <code>getUsers</code> Web Service call is made. The status code obtained from the response object is printed to verify the success of the <code>getUserGroups</code> operation.</li> <li>4. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>5. If the Web Service call was successful, then display the user groups retrieved.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getUserGroups
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <request xmlns=""/>
      </getUserGroups>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getUserGroupsResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </getUserGroupsResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

**See also** [getUsers](#)

## convertCurrency

<b>Service</b>	To convert a certain denomination and amount of money to a desired currency.
<b>Usage</b>	It converts a currency from one type to another given a certain <code>date</code> . The <code>money</code> is expressed as an object of type <code>AgileMoneyType</code> . The converted currency is obtained through the response object.
<b>Basic Steps</b>	<p>To convert a currency:</p> <ol style="list-style-type: none"><li>1. Create the request object for the <code>convertCurrency</code> operation.</li><li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to retrieve several attributes.</li><li>3. For each batched request, declare the specifications for which currency will be converted. Money is specified using an object of <code>AgileMoneyType</code>. <code>Date</code> and new currency are also specified.</li><li>4. The request objects are set and the Agile Stub is used to make the <code>convertCurrency</code> Web Service call. The status code obtained from the response object is printed to verify the success of the <code>convertCurrency</code> operation.</li><li>5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li><li>6. If the Web Service call was successful, then display the results.</li></ol>

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <convertCurrency
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
        <request xmlns="">
          <requests>
            <money>
              <amount>100.0</amount>
              <currency>INR</currency>
            </money>
            <toCurrency>GBP</toCurrency>
            <date>2009-05-05T13:37:35.555Z</date>
          </requests>
        </request>
      </convertCurrency>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <convertCurrencyResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/AdminMetadata/V1">
```



```
</message xmlns="">
  <messageId xsi:nil="true"/>
  <messageName xsi:nil="true"/>
  <statusCode>SUCCESS</statusCode>
  <responses>
    <money>
      <amount>1.4843087362171332</amount>
      <currency>GBP</currency>
    </money>
    <date>2009-05-05T13:37:35.555Z</date>
  </responses>
</response>
</convertCurrencyResponse>
</soapenv:Body>
</soapenv:Envelope>
```



## Attachment Web Services

This chapter includes the following:

▪ getFileFF .....	105
▪ getFileAttachment .....	107
▪ addFileAttachment .....	109
▪ checkOutFF .....	111
▪ checkInFF .....	113
▪ cancelCheckOutFF .....	115
▪ checkOutAttachment .....	117
▪ checkInAttachment .....	119
▪ addFileFF .....	121

### getFileFF

**Service** To retrieve a list of files from a specific Agile file folder object.

**Usage** The request object contains the specifications that identify the file to be downloaded. An array of bytes is obtained in the response object, which also provides comprehensive information about the files retrieved, including content, file type, size, and row identifiers.

**Basic Steps** To get a file from a File Folder:

1. Create the request object for the GetFileFF operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to obtain several files from different folders simultaneously.
3. For each batched request, specify the unique folder name and version from which files will be retrieved. Use the element `files` to define the attachment to be downloaded.
4. The attachment to be downloaded is defined as an element in the request type. The `rowId` field of this object is set to specify the file that has been downloaded.
5. You can utilize the Table Web Services to issue the operation `loadTable` after which the message elements of all rows are searched to find a match. A string input with values of either `getRowId` or `getFileId` is also passed a parameter input. Depending on the value of this string, either the `rowId` or `fileId` is returned.
6. The request objects are set and the Agile Stub is used to make the GetFileFF Web Service call. The status code obtained from the response object is printed to verify the success of the getFileFF operation.

7. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
8. If the Web Service call was successful, then display information about the file(s) retrieved.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getFileFF
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <request xmlns="">
          <requests>
            <folderNumber>FOLDER00232</folderNumber>
            <folderVersion xsi:nil="true"/>
            <files>
              <rowId>6112773</rowId>
            </files>
          </requests>
        </request>
      </getFileFF>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getFileFFResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <responses>
            <folderNumber>FOLDER00232</folderNumber>
            <folderVersion xsi:nil="true"/>
            <files>
              <rowId>6112773</rowId>
              <fileId>6112630</fileId>
              <name>FOLDER00232_File123.txt</name>
              <description>Description for file 1</description>
              <fileType>txt</fileType>
              <fileSize>19</fileSize>
              <content>RmlsZSBDb250ZW50Li4uZmlsZQ==</content>
            </files>
          </responses>
        </response>
      </getFileFFResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      [loadTable](#)

## getFileAttachment

<b>Service</b>	To retrieve a specific file from the Attachments Tab of a particular Agile object.
<b>Usage</b>	The request object contains the specifications that identify the attachment to be downloaded. An array of bytes is obtained in the response object, which also provides comprehensive information about the file retrieved.
<b>Basic Steps</b>	<p>To get a File Attachment:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the getFileAttachment operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to retrieve several files simultaneously.</li> <li>3. For each batched request, specify the unique object from whose attachment tab the required files will be retrieved. Supply its class identifier and object number.</li> <li>4. The exact specification of the attachment to be downloaded is defined as an object in the request. This object includes information about rowId, a boolean to indicate whether all the files of the object are to be downloaded and finally provision for fileId to be used in special cases.</li> </ol> <p>You can download a file from the attachment tab using its rowId. This is applicable in a case when the file queried for is present in a single and separate row. In other cases, when there are several files in the same row and the desired file is one of them, the fileId must also be specified.</p> <p>Obtain the rowId or fileId of a particular row when its filename is given, utilizing the Table Web Services to issue the call for the operation loadTable. After this, the message elements of all rows are searched to find a match. A string input, with values of getRowId or getFileId, is also passed a parameter input. Depending on the value of this string, either the rowId or the fileId is returned by the method.</p> <ol style="list-style-type: none"> <li>5. The request objects are set and the Agile Stub is used to make the getFileAttachment Web Service call. The status code obtained from the response object is printed to verify the success of the getFileAttachment operation.</li> <li>6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>7. If the Web Service call was successful, then display information about the file(s) retrieved.</li> </ol>

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
```

```

    <getFileAttachment
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
  <request xmlns="">
    <requests>
      <classIdentifier>Part</classIdentifier>
      <objectNumber>P00734</objectNumber>
      <allFiles>false</allFiles>
      <downloadUrl>true</downloadUrl>
      <attachments>
        <rowId>6112830</rowId>
      </attachments>
    </requests>
  </request>
</getFileAttachment>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getFileAttachmentResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <responses>
            <classIdentifier>Part</classIdentifier>
            <objectNumber>P00734</objectNumber>
            <attachment>
              <rowId>6112830</rowId>
              <fileId>6112635</fileId>
              <name>P00734_file123.txt</name>
              <description>Description for file 1</description>
              <fileType>txt</fileType>
              <fileSize>19</fileSize>
              <fileDownloadURL>http://DTP-VSREEDHA-
WF:8877/webfs/DownloadServlet?token=D6A8C1A41AA40B5AE29A2CFD33D0BEE143E7ABBBBC00245
67A16EBD62A195193C8FE6B3FBCC3131B58BC18A774412F759D648C6D0EC5D579A9E0B660217EA744A
24220B54E0583A1C569F6E9722B6&vault=&fileID=3DB9227A1EFF6DC2EB</fileDownloa
dURL>
            </attachment>
          </responses>
        </response>
      </getFileAttachmentResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

**See also**      loadTable

## addFileAttachment

- Service** To add a new file to the Attachment Tab of an Agile Object.
- Usage** This is facilitated by specifying relevant details of the new file through the request object.
- Basic Steps** To add a file in the attachment tab of an Agile Object:
1. Create the request object for the addFileAttachment operation.
  2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to add several files to different objects simultaneously.
  3. For each batched request, specify the unique object to whose attachment tab the files shall be added. Supply the class identifier and object number information.
  4. The exact specification of the attachment to be added is defined as an object of the request type. This object includes information about the name of the file and its description and content.
  5. The request objects are set and the Agile Stub is used to make the addFileAttachment Web Service call. The status code obtained from the response object is printed to verify the success of the addFileAttachment operation.
  6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
  7. If the Web Service call was successful, then state the same.

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <addFileAttachment
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <request xmlns="">
          <requests>
            <classIdentifier>Part</classIdentifier>
            <objectNumber>P00720</objectNumber>
            <singleFolder>false</singleFolder>
            <attachments>
              <name>Filename.txt</name>
              <description>Description for file </description>
              <content>RmlsZSBDb250ZW50Li4uZmlsZQ==</content>
            </attachments>
          </requests>
        </request>
      </addFileAttachment>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>

```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
  <soapenv:Body>
    <addFileAttachmentResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
        <statusCode>SUCCESS</statusCode>
        <responses>
          <classIdentifier>Part</classIdentifier>
          <objectNumber>P00720</objectNumber>
        </responses>
      </response>
    </addFileAttachmentResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

**See also** [AddFileSOAPAttachment Method](#) on page 207



## checkOutFF

**Service** To check-out an Agile File Folder object.

**Usage** The request object specifies the file folder that has to be checked out. Subsequent operations such as adding a file to this file folder and then checking in the folder back are possible after this step.

**Basic Steps** To check-out an Agile File Folder object:

- ☞ Before adding a new file to a folder, the folder object must be checked out prior to any file operation. The checkout Web Service is used to achieve this.
- 1. Create the request object for the CheckOutFF operation.
- 2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to add checkout several folders simultaneously.
- 3. For each batched request, specify the unique folder that will be checked out by the Web Service operation. Supply the folder number for the same.
- 4. The request objects are set and the Agile Stub is used to make the CheckOutFF Web Service call. The status code obtained from the response object is printed to verify the success of the CheckOutFF operation.
- 5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
- 6. If the Web Service call was successful, then list the folders checked out.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <checkOutFF
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <request xmlns="">
          <requests>
            <folderNumber>FOLDER00214</folderNumber>
          </requests>
        </request>
      </checkOutFF>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <checkOutFFResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </checkOutFFResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

```
</responses>
  <folderNumber>FOLDER00214</folderNumber>
</responses>
</response>
</checkOutFFResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      [loadTable](#)

## checkInFF

<b>Service</b>	To check-in an Agile File Folder Object.
<b>Usage</b>	The request object specifies the file folder that has already been checked out and needs to be checked in by the Web Service operation.
<b>Basic Steps</b>	<p>To check in an Agile File Folder:</p> <ol style="list-style-type: none"> <li>1. After a file folder has been checked out and desired modifications have been carried out, the file folder object must be checked in to reflect the changes in it. The checkInFF Web Service is used to achieve the same.</li> <li>2. Create the request object for the CheckInFF operation.</li> <li>3. Create an array of requests. Batch operations may be performed by populating as many request objects as required to check-in several folders simultaneously.</li> <li>4. For each batched request, specify the unique folder that will be checked in by the Web Service operation. Supply the folder number for the same.</li> <li>5. The request objects are set and the agile Stub is used to make the CheckInFF Web Service call. The status code obtained from the response object is printed to verify the success of the CheckInFF operation.</li> <li>6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>7. If the Web Service call was successful, then list the folders checked out</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <checkInFF
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <request xmlns="">
          <requests>
            <folderNumber>FOLDER00220</folderNumber>
          </requests>
        </request>
      </checkInFF>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <checkInFFResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </checkInFFResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

```
</responses>
  <folderNumber>FOLDER00220</folderNumber>
</responses>
</response>
</checkInFFResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      [checkOutFF](#)

## cancelCheckOutFF

**Service** To cancel the 'checked-out' status of an Agile File Folder object, which was earlier checked out using the checkout operation.

**Usage** The request object specifies the file folder for which the 'checked-out' status has to be annulled.

**Basic Steps** To cancel the checked-out File Folder:

☞ Before adding a new file to a folder, the folder object must be checked out prior to any file operation. The checkout Web Service is used to achieve the same, as shown in the sample code.

1. Create the request object for the CancelCheckOutFF operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to add checkout several folders simultaneously.
3. For each batched request, specify the unique folder that will be checked out by the Web Service operation. Supply the folder number for the same.
4. The request objects are set and the Agile Stub is used to make the CancelCheckOutFF Web Service call. The status code obtained from the response object is printed to verify the success of the CancelCheckOutFF operation.
5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
6. If the Web Service call was successful, then list the folders checked out.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <cancelCheckOutFF
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <request xmlns="">
          <requests>
            <folderNumber>FOLDER00217</folderNumber>
          </requests>
        </request>
      </cancelCheckOutFF>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <cancelCheckOutFFResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
        </response>
      </cancelCheckOutFFResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

```
<messageName xmlns="http://www.agileplm.com/2009/01/10/PLMCoreWebServices/">
  <statusCode>SUCCESS</statusCode>
  <responses>
    <folderNumber>FOLDER00217</folderNumber>
  </responses>
</response>
</cancelCheckOutFFResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      [checkOutFF](#)

## checkOutAttachment

<b>Service</b>	To check out a specific file from the Attachment Tab of an Agile object and to retrieve its contents.
<b>Usage</b>	The request object specifies the object and file to be retrieved while the desired content is received through the response object.
<b>Basic Steps</b>	<p>To check out an attachment:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the checkOutAttachment operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to add several files to different objects simultaneously.</li> <li>3. For each batched request, specify the unique object from whose attachment tab files will be checked out. Supply the class identifier and object number information.</li> <li>4. Provide the exact specification of the file attachment that has to be checked out by the Web Service. Use an object to define an Agile file using its rowId or a combination of its rowId and fileId as required. A boolean element to checkout all files is also available.</li> <li>5. Obtain the rowId or fileId of a particular row when given its filename, utilizing the Table Web Services to issue a call for the operation loadTable. After this, the message elements of all rows are searched to find a match. A string input with values of either getRowId or getFileId is also passed a parameter input. Depending on the value of this string, either the rowId or fileId is returned by the method.</li> <li>6. Set the rowId of the file that has to be checked out.</li> <li>7. The request objects are set and the Agile Stub is used to make the checkOutAttachment Web Service call. The status code obtained from the response object is printed to verify the success of the checkOutAttachment operation.</li> <li>8. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>9. If the Web Service call was successful, then display information about the file(s) checked out.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <checkOutAttachment
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <request xmlns="">
          <requests>

```

```

        </classIdentifier>Part</classIdentifier>
        <objectNumber>P00726</objectNumber>
        <attachments>
            <rowId>6112534</rowId>
        </attachments>
    </requests>
</request>
</checkOutAttachment>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <checkOutAttachmentResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
            <response xmlns="">
                <messageId xsi:nil="true"/>
                <messageName xsi:nil="true"/>
                <statusCode>SUCCESS</statusCode>
                <responses>
                    <classIdentifier>Part</classIdentifier>
                    <objectNumber>P00726</objectNumber>
                    <files>
                        <rowId>6112534</rowId>
                        <fileId>6112352</fileId>
                        <fileType>txt</fileType>
                        <fileSize>19</fileSize>
                        <name>null_File123.txt</name>
                        <description>Description for file 1</description>
                        <content>RmlsZSBDb250ZW50Li4uZmlsZQ==</content>
                    </files>
                </responses>
            </response>
        </checkOutAttachmentResponse>
    </soapenv:Body>
</soapenv:Envelope>

```

**See also**      `loadTable, checkInAttachment`



## checkInAttachment

<b>Service</b>	To check-in an attachment that was previously checked out
<b>Usage</b>	This attachment could possibly have been modified after checking out and has to be checked in back into the Agile system. Details of the modified file and its content, and the parent object, are specified in the request object.
<b>Basic Steps</b>	<p>To check-in an attachment:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the checkInAttachment operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to checkIn several files to different objects simultaneously.</li> <li>3. For each batched request, specify the unique object from whose attachment tab files will be checked out. Supply the class identifier and object number information.</li> <li>4. The exact specification of the file that has to be checked in by the Web Service is achieved by using an object, which defines the file to be checked in using elements that refer to the new file name and also the modified / new file content. It is to be noted that the file name of the file being checked in must have the same extension as that of the file that was checked out.</li> <li>5. Obtain the rowId or fileId of a particular row when its filename is given, utilizing the Table Web Services to issue a call for the operation loadTable after which the message elements of all rows are searched to find a match. A string input with values of either getRowId or getFileId is also passed a parameter input. Depending on the value of this string, either the rowId or fileId is returned by the method.</li> <li>6. Set the rowId into which the file will be checked in.</li> <li>7. The request objects are set and the Agile Stub is used to make the checkInAttachment Web Service call. The status code obtained from the response object is printed to verify the success of the checkInAttachment operation.</li> <li>8. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>9. If the Web Service call was successful, then state the same.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <checkInAttachment
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">

```

```
<request xmlns="">
  <request>
    <classIdentifier>Part</classIdentifier>
    <objectNumber>P00724</objectNumber>
    <rowId>6112462</rowId>
    <attachments>
      <fileName>Modified_P00724_File123.txt</fileName>
    </attachments>
    <fileContent>TW9kaWZpZWQgZmlsZSBpbmZvcmlhdGlvbiBhZGRlZCBhZnRlcib0aGUgY2hlY2tpbg==<
    /fileContent>
  </request>
</request>
</checkInAttachment>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <checkInAttachmentResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </checkInAttachmentResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      `checkOutAttachment`

## addFileFF

<b>Service</b>	To add a new file to the Files Tab of an Agile File Folder object.
<b>Usage</b>	The request object specifies details of the file folder that was previously checked out to facilitate the add file process.
<b>Basic Steps</b>	<p>To add a new file to the Files tab of an object:</p> <ul style="list-style-type: none"> <li>☞ Before adding a new file to a folder, the folder object must be checked out prior to any file operation.</li> </ul> <ol style="list-style-type: none"> <li>1. Create the request object for the AddFileFF operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to add several files to different folders simultaneously.</li> <li>3. For each batched request, specify the unique folder to whose files tab the files have to be added and supply the folder number details.</li> <li>4. The exact specification of the attachment to be added is defined as an object that declares information about the name of the file and its description and content.</li> <li>5. The request objects are set and the Agile Stub is used to make the AddFileFF Web Service call. The status code obtained from the response object is printed to verify the success of the AddFileFF operation.</li> <li>6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>7. If the Web Service call was successful, then state the same.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <addFileFF
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
        <request xmlns="">
          <request>
            <folderNumber>FOLDER00220</folderNumber>
            <files>
              <fileName>File_FOLDER00220.txt</fileName>
              <fileContent>RmlsZSBDb250ZW50Li4uZmlsZQ==</fileContent>
              <description>Description for file 1</description>
            </files>
          </request>
        </request>
      </addFileFF>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>

```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
  <soapenv:Body>
    <addFileFFResponse
      xmlns="http://xmlns.oracle.com/AgileObjects/Core/Attachment/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
        <statusCode>SUCCESS</statusCode>
      </response>
    </addFileFFResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

**See also**      [checkOutFF](#)





## Business Web Services

This chapter includes the following:

▪ createObject .....	125
▪ getObject .....	129
▪ updateObject .....	132
▪ deleteObject.....	134
▪ undeleteObject.....	136
▪ isDeletedObject .....	138
▪ sendObject.....	140
▪ saveAsObject.....	142
▪ checkPrivilege.....	144

### createObject

**Service** To create a specific Object in Agile PLM system.

**Usage** The object specifications are detailed in the request object where the class type, unique object number and other primary data may be configured, apart from more specific options for the created object.

**Basic Steps** To create an Agile Object:

1. Create the request object for the createObject operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to create several new objects, simultaneously.
3. Specify the type of object to be created in each of the request objects.
4. Create a row to set the data for the request objects.
5. Create an array of message elements to specify various attributes for the new object.
6. The API name `Number` is used to identify that this message pertains to the object number for the new object. A `textNode` is then added with the actual value of the object number. Similarly, the `Description` for the new object is specified.
7. The Agile row is updated with the content of these message elements and each request object is updated with the values of their respective Agile rows. Similarly, the next createObject request is also populated with data.
8. The request objects are set and the Agile Stub is used to make the createObject Web Service call. The status code obtained from the response object is printed to verify the success of the createObject operation.

9. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.

## Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <createObject
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <request xmlns="">
          <requests>
            <classIdentifier>Part</classIdentifier>
            <data rowId="0">
              <number>P00585</number>
              <description>Object Desc</description>
            </data>
          </requests>
        </request>
      </createObject>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <createObjectResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <responses>
            <agileObject>
              <objectIdentifier>
                <classId>10141</classId>
                <className>Part</className>
                <classDisplayName>Part</classDisplayName>
                <objectId>6110466</objectId>
                <objectName>P00585</objectName>
              </objectIdentifier>
              <number xsi:type="xs:string"
                xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="1001"
                readOnly="False">P00585</number>
              <itemType xsi:type="common:AgileListEntryType"
                xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
                attributeId="1081" readOnly="False">
                <selection>
                  <id>10141</id>
                  <apiName>PART</apiName>
                  <value>Part</value>
                </selection>
              </itemType>
              <lifecyclePhase xsi:type="common:AgileListEntryType"
                xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
                attributeId="1084" readOnly="True">
                <selection>
                  <id>976</id>
                  <apiName>PRELIMINARY</apiName>
                  <value>Preliminary</value>
                </selection>
              </lifecyclePhase>
            </agileObject>
          </responses>
        </response>
      </createObjectResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```



```

        <description xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="1002"
readOnly="True"><Object Desc/Description>
        <itemCategory xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="1082" readOnly="False"/>
        <size xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="1068" readOnly="False"/>
        <productLineS xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="1004" readOnly="False"/>
        <rev xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="1014" readOnly="False">
        <selection>
        <id>0</id>
        <apiName>Rev</apiName>
        <value>Introductory</value>
        </selection>
        </rev>
        <revIncorpDate xsi:type="xs:date"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="1017" readOnly="True"/>
        <revReleaseDate xsi:type="xs:date"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="1016" readOnly="True"/>
        <effectivityDate xsi:type="xs:date"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="12089" readOnly="True"/>
        <shippableItem xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000002781" readOnly="False">
        <selection>
        <id>0</id>
        <apiName>NO</apiName>
        <value>No</value>
        </selection>
        </shippableItem>
        <excludeFromRollup xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000002859" readOnly="False">
        <selection>
        <id>0</id>
        <apiName>NO</apiName>
        <value>No</value>
        </selection>
        </excludeFromRollup>
        <complianceCalculatedDate xsi:type="xs:date"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="2000004143"
readOnly="True"/>
        <partFamily xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000004416" readOnly="False"/>
        <mass xsi:type="common:AgileUnitOfMeasureType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000004612" readOnly="False"/>
        <overallCompliance xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000004891" readOnly="True"/>
        <itemGroupS xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000008520" readOnly="True"/>
        <thumbnail xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000008549" readOnly="True"/>
    </agileObject>
</responses>
</response>
</createObjectResponse>
</soapenv:Body>
</soapenv:Envelope>

```

**See also**      `getAutoNumbers`, `getAttributes`, `getAllClasses`, `getSubClasses`

## getObject

<b>Service</b>	To retrieve a specific Agile object from the Agile PLM system.
<b>Usage</b>	The specifications of the object to be retrieved are detailed in the request object where the class type, unique object number and relevant data may be specified. Comprehensive information about the object is retrieved in the response object after successful execution of the Web Service call.
<b>Basic Steps</b>	<p>To get an Agile Object:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the getObject operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to retrieve several objects, simultaneously.</li> <li>3. For each request, set the class Identifier to denote the type of object to be retrieved and the object identifier to specify the object number of the object to be retrieved.</li> <li>4. A Table type request is used to specify the tables pertaining to the Agile object that will be retrieved by the operation getObject. The <code>table</code> identifier is used for this purpose. If the meta data is also required, then the boolean <code>loadCellMetaData</code> is set to 'true'.</li> <li>5. The request objects are set and the Agile Stub is used to make the getObject Web Service call. The status code obtained from the response object is printed to verify the success of the getObject operation.</li> <li>6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>7. If the object, was successfully retrieved, then examine its contents. From this object, obtain a list of tables from the agile object.</li> <li>8. From these tables, obtain a set of rows and print the values of the objects within these rows using their respective message elements.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getObject xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <request xmlns="">
          <requests>
            <classIdentifier>ManufacturerPart</classIdentifier>
            <objectNumber>MANUF_PART1241535324230</objectNumber>
            <tableRequests>
              <tableIdentifier>807</tableIdentifier>
              <loadCellMetaData>>false</loadCellMetaData>
            </tableRequests>
            <options>
              <propertyName>manufacturer_name</propertyName>
            </options>
          </requests>
        </request>
      </getObject>
    </soapenv:Body>
  </soapenv:Envelope>

```

```

        </options>
    </requests>
</request>
</getObject>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <getObjectResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
            <response xmlns="">
                <messageId xsi:nil="true"/>
                <messageName xsi:nil="true"/>
                <statusCode>SUCCESS</statusCode>
                <responses>
                    <agileObject>
                        <objectIdentifier>
                            <classId>1488</classId>
                            <className>ManufacturerPart</className>
                            <classDisplayName>Manufacturer Part</classDisplayName>
                            <objectId>6110515</objectId>
                            <objectName>MANUF_PART1241535324230</objectName>
                        </objectIdentifier>
                        <table>
                            <tableIdentifier>
                                <classId>1488</classId>
                                <className>ManufacturerPart</className>
                                <objectId>6110515</objectId>
                                <objectName>MANUF_PART1241535324230</objectName>
                                <tableId>807</tableId>
                                <tableName>Attachments</tableName>
                                <tableDisplayName>Attachments</tableDisplayName>
                            </tableIdentifier>
                        </table>
                        <manufacturerPartNumber xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeId="1648">MANUF_PART1241535324230</manufacturerPartNumber>
                        <manufacturerName xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeId="1647">MANUF1241535323652</manufacturerName>
                        <description xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeId="3566">Description</description>
                        <lifecyclePhase xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="1649">
                            <selection>
                                <id>1517</id>
                                <apiName>ACTIVE</apiName>
                                <value>Active</value>
                            </selection>
                        </lifecyclePhase>
                        <mfrPartType xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2543">
                            <selection>
                                <id>1488</id>
                                <apiName>MANUFACTURER_PART</apiName>
                                <value>Manufacturer Part</value>
                            </selection>
                        </mfrPartType>
                        <partFamily xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000004417"/>

```

```

        <mass xsi:type="common:AgileUnitOfMeasureType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000004612"/>
        <complianceCalculatedDate xsi:type="xs:date"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="2000004735"/>
        <overallCompliance xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000005405"/>
        <thumbnail xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000008559"/>
        <itemGroupS xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="2000008566"/>
    </agileObject>
</responses>
</response>
</getObjectResponse>
</soapenv:Body>
</soapenv:Envelope>

```

**See also**      quickSearch, advancedSearch, createObject

## updateObject

<b>Service</b>	To update a specific object in the Agile PLM system.
<b>Usage</b>	The revised object specifications are detailed in the request object where data specific to an Agile object may be expressed.
<b>Basic Steps</b>	<p>To update an object:</p> <ol style="list-style-type: none"><li>1. Create the request object for the updateObject operation.</li><li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to update several objects.</li><li>3. For each of the requests, set the class identifier to specify the type of object that will be updated and an object number to identify the unique object.</li><li>4. The actual data to be updated is specified in the form of message elements, which are set into the <code>_any</code> field of row type elements.</li><li>5. Based on this message element, the Description field of the specified agile object will be updated with a new value.</li><li>6. The data field of the request object is updated with the new row data.</li><li>7. The request objects are set and the Agile Stub is used to make the updateObject Web Service call. The status code obtained from the response object is printed to verify the success of the updateObject operation.</li><li>8. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li></ol>

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <updateObject
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <request xmlns="">
          <requests>
            <classIdentifier>ManufacturerPart</classIdentifier>
            <objectNumber>MANUF_PART1241535380057</objectNumber>
            <data rowId="0">
              <Message_Desc attributeId="3566">Updated value of Manuf part
Description</Message_Desc>
            </data>
            <options>
              <propertyName>manufacturer_name</propertyName>
              <propertyValue>MANUF1241535379620</propertyValue>
            </options>
          </requests>
        </request>
      </updateObject>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ns="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <updateObjectResponse
      xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
        <statusCode>SUCCESS</statusCode>
      </response>
    </updateObjectResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

**See also** `getAttributes`

## deleteObject

<b>Service</b>	To delete a specific object in the Agile PLM system.
<b>Usage</b>	The specifications of the object to be deleted are given in the request object.
<b>Basic Steps</b>	<p>To delete an object:</p> <ol style="list-style-type: none"><li>1. Create the request object for the deleteObject operation.</li><li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to delete several objects.</li><li>3. For each of the requests, set the class identifier to specify the type of object that will be deleted and an object number to identify the unique object that will be delete from Agile PLM.</li><li>4. The request objects are set and the Agile Stub is used to make the deleteObject Web Service call. The status code obtained from the response object is printed to verify the success of the deleteObject operation.</li><li>5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li></ol>

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <deleteObject
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
      <request xmlns="">
        <requests>
          <classIdentifier>Part</classIdentifier>
          <objectNumber>P00589</objectNumber>
        </requests>
      </request>
    </deleteObject>
  </soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <deleteObjectResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
        <statusCode>SUCCESS</statusCode>
        <responses>
          <isDeleted>true</isDeleted>
        </responses>
      </response>
    </deleteObjectResponse>
  </soapenv:Body>
```



```
</soapenv:Envelope>
```

**See also**

quickSearch, advancedSearch

## undeleteObject

- Service** To revoke the 'deleted' status of a specific object that was previously deleted from the Agile PLM system.
- Usage** The specifications of the object to be undeleted are given in the request object.
- Basic Steps** To undelete a deleted object:
1. Create the request object for the undeleteObject operation.
  2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to 'undelete' several objects.
  3. For each of the requests, set the class identifier to specify the type of object that will be undeleted and an object number to identify the unique object whose deletion will be revoked.
  4. The request objects are set and the Agile Stub is used to make the undeleteObject Web Service call. The status code obtained from the response object is printed to verify the success of the undeleteObject operation.
  5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <undeleteObject
      xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
      <request xmlns="">
        <requests>
          <classIdentifier>Part</classIdentifier>
          <objectNumber>P00600</objectNumber>
        </requests>
      </request>
    </undeleteObject>
  </soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <undeleteObjectResponse
      xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
        <statusCode>SUCCESS</statusCode>
      </response>
    </undeleteObjectResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

**See also**      `advancedSearch`, `quickSearch`, `deleteObject`

## isDeletedObject

<b>Service</b>	To check whether a specific Agile object in the Agile PLM system has been deleted or not.
<b>Usage</b>	The object specifications are detailed in the request object where the class type, unique object number may be specified. From the response object, it is possible to ascertain whether the Agile object queried for still exists in the Agile PLM or if it was previously deleted.
<b>Basic Steps</b>	<p>To check the delete state of an object:</p> <ol style="list-style-type: none"><li>1. Create the request object for the isDeletedObject operation.</li><li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to query the deletion status of several objects simultaneously.</li><li>3. For each of the requests, set the class identifier to specify the type of object whose delete status and an object number to identify the unique object is to be determined.</li><li>4. The request objects are set and the Agile Stub is used to make the isDeletedObject Web Service call. The status code obtained from the response object is printed to verify the success of the isDeletedObject operation.</li><li>5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li><li>6. If the web service call was successful, then determine the status of deletion.</li></ol>

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <isDeletedObject
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <request xmlns="">
          <requests>
            <classIdentifier>Part</classIdentifier>
            <objectNumber>P00593</objectNumber>
          </requests>
        </request>
      </isDeletedObject>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <isDeletedObjectResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
        </response>
      </isDeletedObjectResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

```
<messageName xmlns="http://schemas.xmlsoap.org/wsdl/">
  <statusCode>SUCCESS</statusCode>
  <responses>
    <isDeleted>false</isDeleted>
  </responses>
</response>
</isDeletedObjectResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      [advancedSearch](#), [quickSearch](#), [deleteObject](#)

## sendObject

<b>Service</b>	To send a specific Agile object to the Agile PLM system.
<b>Usage</b>	The object specifications of the object to be sent are detailed in the request object.
<b>Basic Steps</b>	To send an object :

☞ To send the objects through Web Services or web user interface, you must first enable the notification feature. In Java client, under Admin > Server settings > Database, set the **Notification Enabled to Yes**.

1. Create the request object for the sendObject operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to send several objects simultaneously.
3. For each of the requests, set the class identifier to specify the type of object and an object number to identify the unique object that will be 'send'.
4. Define a group of users.
5. The request objects are set and the Agile Stub is used to make the sendObject Web Service call. The status code obtained from the response object is printed to verify the success of the sendObject operation.
6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <sendObject
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <request xmlns="">
          <requests>
            <classIdentifier>Part</classIdentifier>
            <objectNumber>P00599</objectNumber>
            <sendTo>
              <classIdentifier>User</classIdentifier>
              <objectIdentifier>User1241535366448</objectIdentifier>
            </sendTo>
            <comments>Test comments</comments>
          </requests>
        </request>
      </sendObject>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <sendObjectResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <response xmlns="">
```

```
</messageName xsi:nil="true"/>
<messageName xsi:nil="true"/>
<statusCode>SUCCESS</statusCode>
<responses/>
</response>
</sendObjectResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      `getUsers`

## saveAsObject

- Service** To save a specific Agile object as a new object in the Agile PLM system.
- Usage** The object specifications are detailed in the request object where the class type, unique object number and other primary data may be specified. The response object contains information identifying the object that was saved.
- Basic Steps** To save an object as another object:
1. Create the request object for the saveAsObject operation.
  2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to save several objects simultaneously.
  3. For each of the requests, set the class identifier to specify the type of object that will be saved and an object number to identify the unique object and the class type of the new object.
  4. This can be carried out in two ways:  
**Using the element `xsd:any`** to specify the new Object Number - The object number of the new object can be specified through the `xsd:any` attribute of a row type by using MessageElements. The new object number information in this row is stored in the request object using the setData method.  
**Using an `AutoNumber` Source** to specify the new Object Number - The autonumber source is retrieved from a Web Service call to the the service 'getAutoNumber'.
  5. The request objects are set and the Agile Stub is used to make the SaveAsObject Web Service call.
  6. The status code obtained from the response object is printed to verify the success of the saveAsObject operation.
  7. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <saveAsObject
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <request xmlns="">
          <saveAsObjectRequest>
            <parentClassIdentifier>Part</parentClassIdentifier>
            <parentObjectNumber>P00594</parentObjectNumber>
            <newClassIdentifier>Part</newClassIdentifier>
            <data rowId="0">
              <Message_Num attributeId="1001">P00596</Message_Num>
            </data>
          </saveAsObjectRequest>
        </saveAsObjectRequest>
      </saveAsObjectRequest>
    </soapenv:Body>
  </soapenv:Envelope>
</pre>
```



```

        <parentObjectNumber>P00595</parentObjectNumber>
        <newClassIdentifier>Part</newClassIdentifier>
        <autoNumberSource>Part Number</autoNumberSource>
    </saveAsObjectRequest>
</request>
</saveAsObject>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <saveAsObjectResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <saveAsObjectResponse>
            <classIdentifier>Part</classIdentifier>
            <objectId>6110572</objectId>
            <objectNumber>P00596</objectNumber>
          </saveAsObjectResponse>
          <saveAsObjectResponse>
            <classIdentifier>Part</classIdentifier>
            <objectId>6110579</objectId>
            <objectNumber>P00598</objectNumber>
          </saveAsObjectResponse>
        </response>
      </saveAsObjectResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

**See also** [getAutoNumbers](#), [getAttributes](#)

## checkPrivilege

<b>Service</b>	To check whether a specific Agile user holds the privileges to perform a specific action in the Agile PLM system.
<b>Usage</b>	The user and privilege specifications are detailed in the request object. The request object confirms whether or not the specified agile user has the privilege to perform the Web Service operation.
<b>Basic Steps</b>	<p>To check a user's privileges:</p> <ol style="list-style-type: none"><li>1. Create the request object for the checkPrivilege operation.</li><li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to check for several privileges simultaneously.</li><li>3. For each request, specify the user whose privileges are to be checked as objects of user Identifier type. Set the privilege that has to be queried from the privileges available to the specified user. AgilePrivilegeType.value1 in the given sample code refers to the 'Comment' privilege, which is a constant defined in Business Services Schema.</li><li>4. As an optional specification, the class identifier and object number may also be specified to identify a particular Agile Object for whose specific privileges the user details are queried.</li><li>5. The request objects are set and the Agile Stub is used to make the CheckPrivilege Web Service call. The status code obtained from the response object is printed to verify the success of the checkPrivilege operation.</li><li>6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li><li>7. Obtain the object CheckPrivilegeType from the response object. The getCheckPrivilege() method on this object will yield a boolean value indicating whether the user has the specified privilege.</li></ol>

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <checkPrivilege
      xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
      <request xmlns="">
        <requests>
          <userIdentification>
            <userIdentifier>admin</userIdentifier>
          </userIdentification>
          <privilege>l</privilege>
          <classIdentifier>Part</classIdentifier>
          <objectNumber>P00585</objectNumber>
        </requests>
      </request>
    </checkPrivilege>
  </soapenv:Body>
</soapenv:Envelope>
```

```

        </checkPrivilege>
    </soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <checkPrivilegeResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Business/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <responses>
            <userIdentification>
              <userIdentifier>admin</userIdentifier>
            </userIdentification>
            <privilege>
              <privilege>1</privilege>
              <checkPrivilege>true</checkPrivilege>
              <classIdentifier>Part</classIdentifier>
              <objectNumber>P00585</objectNumber>
            </privilege>
          </responses>
        </response>
      </checkPrivilegeResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

**See also**      `getUsers`, `copyTable`



## Collaboration Web Services

This chapter includes the following:

▪ getWorkflows .....	147
▪ getStatus .....	149
▪ auditRObjct .....	151
▪ getApprovers .....	153
▪ changeStatus .....	155
▪ approveRObjct .....	157
▪ rejectRObjct .....	159
▪ setWorkFlow .....	161
▪ addApprovers .....	163
▪ removeApprovers .....	165
▪ commentRObjct .....	167

### getWorkflows

**Service** To retrieve the valid workflows of an Agile routable object.

**Usage** When you create a new change, package, product service request, or quality change order, you must select a workflow. Otherwise, the object remains in an unassigned state and cannot progress through a workflow process.

Agile system can have multiple workflows defined for each type of routable object. To retrieve the valid workflows for an object, use getWorkflows service.

**Basic Steps** To get the workflow of a routable object:

1. Create the request object for the getWorkflows operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to obtain several workflows.
3. For each batched request, specify the type of object whose workflows are to be retrieved and its unique object number.
4. The request objects are set and the Agile Stub is used to make the getWorkflows Web Service call. The status code obtained from the response object is printed to verify the success of the getWorkflows operation.
5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
6. If the Web Service call was successful, display the list of workflows retrieved.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getWorkFlows
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
      <request xmlns="">
        <workflowRequest>
          <classIdentifier>ECO</classIdentifier>
          <objectNumber>C00034</objectNumber>
        </workflowRequest>
      </request>
    </getWorkFlows>
  </soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getWorkFlowsResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <workflowResponse>
            <identifier>
              <classId>6141</classId>
              <className>ECO</className>
              <classDisplayName>ECO</classDisplayName>
              <objectId>6128130</objectId>
              <objectName>C00034</objectName>
            </identifier>
            <workflow>
              <workflowId>3752</workflowId>
              <workflowName>DefaultChangeOrders</workflowName>
              <workflowDisplayName>Default Change
Orders</workflowDisplayName>
            </workflow>
          </workflowResponse>
        </response>
      </getWorkFlowsResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      [setWorkflow](#)

## getStatus

<b>Service</b>	To get the current and the next workflow status of an routable object.
<b>Usage</b>	<p>To determine the status of a change whether it's pending or released. The getStatus service returns a Status object.</p> <p>Workflow functionalities that are made available to users for a particular routable object, depends on the status of the routable object and the user's privileges. The workflow actions available for a pending change are different from those for a released change.</p>
<b>Basic Steps</b>	<p>To get the status of a change:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the getStatus operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to obtain several workflow status objects.</li> <li>3. For each batched request, specify the type of object whose status has to be retrieved and its unique object number.</li> <li>4. The request objects are set and the Agile Stub is used to make the getStatus Web Service call. The status code obtained from the response object is printed to verify the success of the getStatus operation.</li> <li>5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>6. If the Web Service call was successful, display the status information retrieved by the getStatus operation. This status will include the current workflow state and a list of states prior to and beyond that particular workflow state.</li> </ol> <p>A given workflow state is expressed as an object of status type. Further information about the status's display name, id may be obtained through the status object.</p>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getStatus
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <request xmlns="">
          <statusRequest>
            <classIdentifier>ECO</classIdentifier>
            <objectNumber>C00034</objectNumber>
          </statusRequest>
        </request>
      </getStatus>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>

```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
  <soapenv:Body>
    <getStatusResponse
      xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
        <statusCode>SUCCESS</statusCode>
        <statusResponse>
          <identifier>
            <classId>6141</classId>
            <className>ECO</className>
            <classDisplayName>ECO</classDisplayName>
            <objectId>6128130</objectId>
            <objectName>C00034</objectName>
          </identifier>
          <currentStatus>
            <statusId>3753</statusId>
            <statusName>Pending</statusName>
            <statusDisplayName>Pending</statusDisplayName>
          </currentStatus>
          <nextDefaultStatus>
            <statusId>3766</statusId>
            <statusName>Submitted</statusName>
            <statusDisplayName>Submitted</statusDisplayName>
          </nextDefaultStatus>
          <nextStatus>
            <statusId>3766</statusId>
            <statusName>Submitted</statusName>
            <statusDisplayName>Submitted</statusDisplayName>
          </nextStatus>
        </statusResponse>
      </response>
    </getStatusResponse>
  </soapenv:Body>
</soapenv:Envelope>
```



## auditRObjct

<b>Service</b>	To audit a routable object.
<b>Usage</b>	At any point in the lifecycle of a Change, you can audit the Change to determine if any required entry cells are incomplete or the change violates any Agile Smart Rules.
<b>Basic Steps</b>	<p>To audit an object:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the auditRObjct operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to obtain several objects.</li> <li>3. For each batched request, specify the type of routable object on which the audit action will be performed and also its unique object number.</li> <li>4. The request objects are set and the Agile Stub is used to make the auditRObjct Web Service call. The status code obtained from the response object is printed to verify the success of the auditRObjct operation.</li> <li>5. If the status code is not 'success', then populate the list of exceptions returned by the Web Service.</li> <li>6. If the Web Service call was successful, display the audit information retrieved by the auditRObjct operation. This information will consist of a list of warnings and errors pertaining to the audit.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <auditRObjct
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <request xmlns="">
          <request>
            <classIdentifier>ECO</classIdentifier>
            <objectNumber>C00035</objectNumber>
            <auditRelease>true</auditRelease>
          </request>
        </request>
      </auditRObjct>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <auditRObjctResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <responses xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </responses>
      </auditRObjctResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

```
</response>
  <identifier>
    <classId>6141</classId>
    <className>ECO</className>
    <classDisplayName>ECO</classDisplayName>
    <objectId>6128165</objectId>
    <objectName>C00035</objectName>
  </identifier>
  <error>
    <exceptionId>60086</exceptionId>
    <message>The following required fields are missing : C00035:
Affected Items.New Rev: P00336</message>
  </error>
</response>
</responses>
</auditRObjectResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      [changeStatus](#)

## getApprovers

<b>Service</b>	To get a list of approvers or observers for Agile's routable objects.
<b>Usage</b>	When a routable object is released in a workflow, it is either sent to a user for approval or for notification. A list of users is required to be selected and added for the workflow to begin. This list is obtained from Agile system by sending a getApprovers request.
<b>Basic Steps</b>	<p>To get a list of approvers:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the getApprovers operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to obtain several objects.</li> <li>3. For each batched request, specify the class identification and and unique object number. The request objects are set and the Agile Stub is used to make the getApprovers Web Service call. The status code obtained from the response object is printed to verify the success of the getApprovers operation.</li> <li>4. If the status code is not 'success', then populate the list of exceptions returned by the Web Service.</li> <li>5. If the Web Service call was successful, display the list of users who are designated as approvers for the particular status queried for in the request.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getApprovers
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <request xmlns="">
          <approversRequest>
            <classIdentifier>ECO</classIdentifier>
            <objectNumber>C00038</objectNumber>
            <statusIdentifier>CCB</statusIdentifier>
          </approversRequest>
        </request>
      </getApprovers>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getApproversResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </getApproversResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

```
</approversResponse>
  <identifier>
    <classId>6141</classId>
    <className>ECO</className>
    <classDisplayName>ECO</classDisplayName>
    <objectId>6128265</objectId>
    <objectName>C00038</objectName>
  </identifier>
</approversResponse>
</response>
</getApproversResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**     [getStatus](#)

## changeStatus

<b>Service</b>	A general purpose service for changing the status of an Agile object.
<b>Usage</b>	To submit, release, or cancel a change.
<b>Basic Steps</b>	<p>To change status of an object:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the changeStatus operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to obtain several workflow status objects.</li> <li>3. For each batched request, specify the type of object whose statuses are to be changed and its unique object number.</li> <li>4. An array of either users, user groups or both, may be used to list the set of approvers that have to added. Such a set of user information is expressed as an object of User Group Identifier type. Object and Class identifiers are set for each of these objects to denote user information.</li> <li>5. The request objects are set and the Agile Stub is used to make the changeStatus Web Service call. The status code obtained from the response object is printed to verify the success of the changeStatus operation.</li> <li>6. If the status code is not 'success', then populate the list of exceptions returned by the Web Service.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <changeStatus
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <request xmlns="">
          <disableAllWarnings>true</disableAllWarnings>
          <changeStatusRequest>
            <classIdentifier>ECO</classIdentifier>
            <objectNumber>C00033</objectNumber>
            <newStatusIdentifier>Submitted</newStatusIdentifier>
            <comment>Comments</comment>
            <password>agile</password>
            <auditRelease>false</auditRelease>
            <urgent>false</urgent>
            <notifyOriginator>true</notifyOriginator>
            <notifyChangeAnalyst>true</notifyChangeAnalyst>
            <notifyCCB>true</notifyCCB>
          </changeStatusRequest>
        </request>
      </changeStatus>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
HTTP/1.1 200 OK
Date: Fri, 10 Apr 2009 11:55:27 GMT
Server: Oracle-Application-Server-10g/10.1.3.4.0 Oracle-HTTP-Server

```

```
Set-Cookie:
JSESSIONID=f25fe9baa0bb29c93561749bc67ea5f9f03c8b907681f2c13908aa6042d0eb88.e34Kc3
Connection: close
Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <changeStatusResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </changeStatusResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      [auditRObjct](#)

## approveRObject

**Service** To see the approval results for an object.

**Usage** It inform the users whether the object has been approved by the approver, or, when an approver is approving the object on behalf of one or more user groups. After a change is routed to group of approvers, the online approval process begins. Users listed in the Workflow table for a change can approve or reject the change.

When you approve a change, the Agile system records the approval in the Workflow table. When all approvers have approved the change, the system sends an email notification to the change analyst or component engineer indicating that the change is ready to be released.

**Basic Steps** To see the approval results:

1. Create the request object for the approveRObject operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to approve several routable objects with a single operation.
3. For each batched request, specify the type of object and unique object number of the routable object which has to be approved. The user with which the Agile Stub has been authenticated is assumed to be the approver. The approval password of that user is specified in the request and pertinent comments are added.
4. If the second signature has been configured in your java client then the value of the same shall have to be set into the request object. The second signature may be either the username or password depending on the Java Client's configuration from the 'Preferences' tab.
5. Additional options may also be elaborated upon using the request object:  
Specify a list of notifiers using an object of User Group Identifier type.  
  
Escalations, transfers and the facility of approving on behalf of a group may also be achieved by the same method. This may be used in cases when the approval on member of a usergroup is sufficient to expedite the approval process.  
  
In this sample code given below, the null values are passed for these fields.
6. If you wish to issue warnings, you may specify a list of notifiers.  
Agile Warnings are commonly encountered while executing Web Services. While these warnings do not cause a Web Service failure by themselves, they need to be resolved in advance for the Web Service to successfully perform the intended objectives.
7. This may be achieved by either disabling all warnings or by specifically resolving a particular warning message. All warnings may be disabled by setting the element as true:

8. A specific warning may be resolved by using a warning resolution type and specifying the warning id. This warning ID is obtained by using the reference to warning resolution, which returns a string value.
9. You can also disable a collaboration warning that states that a particular user has already signed off on the routable object, clarifying whether the user wishes to sign off again. While a warning of this type would not cause a Web Service failure, the routable object will not be approved. To ensure the approval, the warning should not be disabled.
10. The request objects are set and the Agile Stub is used to make the approveRObj Web Service call. The status code obtained from the response object is printed to verify the success of the approveRObj operation.
11. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
12. If the Web Service call was successful, confirm the same.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <approveRObj
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <request xmlns="">
          <approveRObj>
            <classIdentifier>ECO</classIdentifier>
            <objectNumber>C00034</objectNumber>
            <password>agile</password>
            <comment>Comment</comment>
            <signoffForSelf>true</signoffForSelf>
          </approveRObj>
        </request>
      </approveRObj>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <approveRObjResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </approveRObjResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      [getStatus](#), [getApprovers](#)



## rejectRObjct

**Service** To reject a routable object.

This service informs users that the routable object is rejected by an approver, or when an approver has rejected the object on behalf of one or more user groups.

**Usage** After a change is routed to group of approvers, the online approval process begins. Users listed in the Workflow table for a change can approve or reject the change.

When you approve a change, the Agile system records the approval in the Workflow table. When all approvers have approved the change, the system sends an email notification to the change analyst or component engineer indicating that the change is ready to be released.

**Basic Steps** To reject a routable object:

1. Create the request object for the rejectRObjct operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to reject several routable objects, simultaneously.
3. For each batched request, specify the type of object and unique object number of the routable object which has to be rejected. The user with which the Agile Stub has been authenticated is assumed to be the approver. The approval password of that user is specified in the request and pertinent comments are added for the rejecting the routable object.
4. If the second signature has been configured in your java client then the value of the same shall have to be set into the request object. The second signature may be either the username or password depending on the Java Client's configuration from the 'Preferences' tab.
5. Additional options may also be elaborated upon using the request object:  
Specify a list of notifiers using an object of user group identifier type.  
  
Escalations, transfers and the facility of approving on behalf of a group may also be achieved by the same method. This may be used in cases when the approval on member of a usergroup is sufficient to expedite the approval process.
6. Specify a list of notifiers.
7. The request objects are set and the Agile Stub is used to make the rejectRObjct Web Service call. The status code obtained from the response object is printed to verify the success of the rejectRObjct operation.
8. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
9. If the Web Service call was successful, confirm the same.

**Sample Code SOAP**

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <rejectRObject
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
            <request xmlns="">
                <rejectRObject>
                    <classIdentifier>ECO</classIdentifier>
                    <objectNumber>C00041</objectNumber>
                    <password>agile</password>
                    <comment>Comment</comment>
                    <notifiers>
                        <classIdentifier>11610</classIdentifier>
                        <objectIdentifier>User11239364588798</objectIdentifier>
                    </notifiers>
                    <signoffForSelf>true</signoffForSelf>
                </rejectRObject>
            </request>
        </rejectRObject>
    </soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <rejectRObjectResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
            <response xmlns="">
                <messageId xsi:nil="true"/>
                <messageName xsi:nil="true"/>
                <statusCode>SUCCESS</statusCode>
            </response>
        </rejectRObjectResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

**See also**      [getApprovers](#), [getStatus](#), [auditRObject](#), [approveRObject](#)

## setWorkflow

<b>Service</b>	To set the workflow of an object.
<b>Usage</b>	As long as a change is in the Pending status, you have option to set a different workflow. Once a change moves beyond Pending status, you cannot change the workflow. If a routable object has not been assigned a workflow yet, you can use the setWorkflow method to set the workflow.
<b>Basic Steps</b>	<p>To set a workflow:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the setWorkflow operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to set the workflows for several objects simultaneously.</li> <li>3. For each batched request, specify the type of object whose workflow is to be set and its unique object number. The getWorkflows Web Service may be used to obtain a list of Web Services for a given object, using which the workflowIdentifier may be set.</li> <li>4. The request objects are set and the Agile Stub is used to make the setWorkflow Web Service call. The status code obtained from the response object is printed to verify the success of the setWorkflow operation.</li> <li>5. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>6. If the Web Service call was successful, confirm the same.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <setWorkflow
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <request xmlns="">
          <setWorkflowRequest>
            <classIdentifier>ECO</classIdentifier>
            <objectNumber>C00033</objectNumber>
            <workFlowIdentifier>3752</workFlowIdentifier>
          </setWorkflowRequest>
        </request>
      </setWorkflow>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <setWorkflowResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <response xmlns="">

```

```
</messageName xsi:nil="true"/>
<messageName xsi:nil="true"/>
<statusCode>SUCCESS</statusCode>
</response>
</setWorkflowResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also** [getWorkFlows](#)

## addApprovers

- Service** To add a list of Approvers or Observers to a routable object.
- addApprovers is used for adding a set of approvers for a given status in Agile PLM. Details of status and list of approvers can be specified in the request object. Success of the operation can be verified using the status code in the response object.
- Usage** When a change is routed and the online approval process has begun, it may be necessary to add or remove people from the list of approvers or observers. When you use addApprovers services, you specify the lists of approvers and observers, whether the notification is urgent, and an optional comment.
- Basic Steps** To add approvers or observers:
1. Initiate a batch operation to send a request for Approvers.
  2. Specify the type of object to whose workflow the approvers will be added.
  3. Mention the specific workflow status to which the approvers will be added. For example, use "CCB" as a string to specify the CCB workflow status.
  4. An array of Users or Usergroups, or both, may be used to list the set of approvers that have to added. Such a set of user information is expressed as an object of user group identifier type. Object and Class identifiers are set for each of these objects to denote user information.
  5. If you want to define a usergroup here, use the usergroup class as the specification, followed by the name of the usergroup object.
  6. Set the list of approvers as per the user group identifier type objects defined. Set values for observers, a boolean flag, to indicate urgency and pertinent comments.
  7. The request objects are set and the Agile Stub is used to make the addApprovers Web Service call. The status code obtained from the response object is printed to verify the 'success' of the addApprovers operation.
  8. If the status code is not 'success', then populate the list of exceptions returned by the Web Service.
  9. If the Web Service call was successful, then confirm the success of the operation.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <addApprovers
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <request xmlns="">
          <addApproversRequest>
            <classIdentifier>ECO</classIdentifier>
```

```
</objectNumber>C00074</objectNumber>
<statusIdentifier>CCB</statusIdentifier>
<approvers>
  <classIdentifier>11610</classIdentifier>
  <objectIdentifier>User11239100555679</objectIdentifier>
</approvers>
<approvers>
  <classIdentifier>11610</classIdentifier>
  <objectIdentifier>User21239100555679</objectIdentifier>
</approvers>
<urgent>false</urgent>
<comment>Comments</comment>
</addApproversRequest>
</request>
</addApprovers>
</soapenv:Body>
</soapenv:Envelope>

==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <addApproversResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </addApproversResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      `getWorkFlows`, `getStatus`

## removeApprovers

<b>Service</b>	Removes the approvers or observers added to a routable object.
<b>Usage</b>	After a change has been routed and the online approval process has begun, it may be necessary to remove people from the list of approvers or observers. When you use removeApprovers services, you specify the lists of approvers and observers, whether the notification is urgent, and an optional comment.
<b>Basic Steps</b>	<p>To remove approvers from a routable object:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the removeApprovers operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to remove approvers to several objects of different class types.</li> <li>3. For each batched request, specify the type of object to whose workflow approvers will be removed. Additionally, mention the specific workflow status to which the approvers will be removed. For example, use "CCB" as a string to specify the CCB workflow status.</li> <li>4. An array of either users, usergroups or both, may be used to list the set of approvers that have to added. Such a set of user information is expressed as an object of type user group identifier type. Object and Class identifiers are set for each of these objects to denote user information.</li> <li>5. If usergroup has to be defined here, use the usergroup class as the specification, followed by the name of the usergroup object. In the given sample code, a user has been specified.</li> <li>6. Set the list of approvers as per the user group identifier type objects defined. Set values for observers, a boolean flag to indicate urgency and pertinent comments.</li> <li>7. The request objects are set and the Agile Stub is used to make the removeApprovers Web Service call. The status code obtained from the response object is printed to verify the success of the removeApprovers operation.</li> <li>8. If the status code is not 'success', then populate the list of exceptions returned by the Web Service.</li> <li>9. If the Web Service call was successful, then confirm the success of the operation.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <removeApprovers
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">

```

```
<?xml version="1.0" encoding="utf-8"?>
  <request xmlns="">
    <removeApproversRequest>
      <classIdentifier>ECO</classIdentifier>
      <objectNumber>C00042</objectNumber>
      <statusIdentifier>CCB</statusIdentifier>
      <approvers>
        <classIdentifier>11610</classIdentifier>
        <objectIdentifier>admin</objectIdentifier>
      </approvers>
      <comment>Comments</comment>
    </removeApproversRequest>
  </request>
</removeApprovers>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <removeApproversResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </removeApproversResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      [getApprovers](#), [addApprovers](#)



## commentRObjct

**Service** To comment a routable object.

**Usage** When you comment a change, you send a comment to other CCB reviewers during the online approval process. In addition to the comment, you can specify whether to notify the originator, the change analyst, and the change control board.

**Basic Steps** To comment a routable object:

1. Create the request object for the commentRObjct operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to comment several routable objects with a single operation.
3. For each batched request, specify the type of object and unique object number of the routable object which has to be commented upon. Use boolean variables to denote whether the originators, change analysts and CCB need to be notified. 'true' or 'false' may be used for the same.
4. To set the element 'notifyList', use the object user group identifier type to specify the list of notifiers. This populates the list.
5. The request objects are set and the Agile Stub is used to make the commentRObjct Web Service call. The status code obtained from the response object is printed to verify the success of the commentRObjct operation.
6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
7. If the Web Service call was successful, confirm the same.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <commentRObjct
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <request xmlns="">
          <commentRObjctRequest>
            <classIdentifier>ECO</classIdentifier>
            <objectNumber>C00037</objectNumber>
            <comment>Comment</comment>
            <notifyOriginator>true</notifyOriginator>
            <notifyChangeAnalyst>true</notifyChangeAnalyst>
            <notifyCCB>true</notifyCCB>
            <notifyList>
              <classIdentifier>11610</classIdentifier>
              <objectIdentifier>admin</objectIdentifier>
            </notifyList>
          </commentRObjctRequest>
        </request>
      </commentRObjct>
    </soapenv:Body>
```

```

</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <commentRObjectResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Collaboration/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </commentRObjectResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```





## PC Web Services

This chapter includes the following:

▪ setIncorporate .....	171
▪ getRevisions .....	173
▪ undoRedline .....	175
▪ isRedlineModified .....	177

### setIncorporate

**Service** To set the status of an Agile object as 'incorporated' or 'unincorporated'.

**Usage** The request object is formed based on the class and object identifiers of the object and the status of incorporation. Success of the operation is verified using the status code in the response object.

**Basic Steps** To set the status of an Agile Object:

1. Create the request object for the setIncorporate operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to set the state of incorporation for several Agile objects simultaneously.
3. For each batched request, specify the type and number of the object whose state of incorporation is to be modified. Pass a boolean value in the 'incorporate' field to denote whether the specified object should be incorporated or unincorporated.
4. The request objects are set and the Agile Stub is used to make the setIncorporate Web Service call. The status code obtained from the response object is printed to verify the success of the setIncorporate operation.
5. If the status code is not 'success', then populate the list of exceptions returned by the Web Service.
6. If the Web Service call was successful, then display the state of incorporation for all the objects modified.

#### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <setIncorporate xmlns="http://xmlns.oracle.com/AgileObjects/Core/Pc/V1">
        <request xmlns="">
          <requests>
            <classIdentifier>Part</classIdentifier>
```

```

        <subjectNumber>P00735</subjectNumber>
        <incorporate>true</incorporate>
    </requests>
</request>
</setIncorporate>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <setIncorporateResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Pc/V1">
            <response xmlns="">
                <messageId xsi:nil="true"/>
                <messageName xsi:nil="true"/>
                <statusCode>SUCCESS</statusCode>
                <responses>
                    <isIncorporated>true</isIncorporated>
                </responses>
            </response>
        </setIncorporateResponse>
    </soapenv:Body>
</soapenv:Envelope>

```

## getRevisions

<b>Service</b>	To retrieve the revisions of an Agile object given the details of the object and relevant options.
<b>Usage</b>	The request object is formed based this information and revisions of the object are obtained through the response object. Success of the operation is verified using the status code in the response object.
<b>Basic Steps</b>	<p>To get revisions of an object:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the getRevisions operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to obtain revisions for several objects simultaneously.</li> <li>3. Identify the object whose revisions are to be retrieved by specifying the class identifier and the object number. This request will fetch the latest revision.</li> <li>4. Set the request objects and use the PC Web Service Agile Stub to execute the getRevisions Web Service.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getRevisions xmlns="http://xmlns.oracle.com/AgileObjects/Core/Pc/V1">
      <request xmlns="">
        <requests>
          <classIdentifier>Part</classIdentifier>
          <objectNumber>1000-02</objectNumber>
          <allRevisions>false</allRevisions>
        </requests>
        <requests>
          <classIdentifier>Part</classIdentifier>
          <objectNumber>1000-02</objectNumber>
          <allRevisions>true</allRevisions>
        </requests>
      </request>
    </getRevisions>
  </soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getRevisionsResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Pc/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
        <statusCode>SUCCESS</statusCode>
        <responses>
          <currentRev>C</currentRev>
        </responses>
      </response>
    </getRevisionsResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

```

</response>
<responses>
  <currentRev>C</currentRev>
  <revisions>
    <changeIdentifier>
      <classId>6141</classId>
      <className>ECO</className>
      <classDisplayName>ECO</classDisplayName>
      <objectId>45</objectId>
      <objectName>25000</objectName>
    </changeIdentifier>
    <revision>(D)</revision>
  </revisions>
  <revisions>
    <changeIdentifier>
      <classId>6141</classId>
      <className>ECO</className>
      <classDisplayName>ECO</classDisplayName>
      <objectId>44</objectId>
      <objectName>24433</objectName>
    </changeIdentifier>
    <revision>C</revision>
  </revisions>
  <revisions>
    <changeIdentifier>
      <classId>6141</classId>
      <className>ECO</className>
      <classDisplayName>ECO</classDisplayName>
      <objectId>43</objectId>
      <objectName>24020</objectName>
    </changeIdentifier>
    <revision>B</revision>
  </revisions>
  <revisions>
    <changeIdentifier>
      <classId>6141</classId>
      <className>ECO</className>
      <classDisplayName>ECO</classDisplayName>
      <objectId>41</objectId>
      <objectName>23450</objectName>
    </changeIdentifier>
    <revision>A</revision>
  </revisions>
  <revisions>
    <changeIdentifier xsi:nil="true"/>
    <revision>Introductory</revision>
  </revisions>
</responses>
</response>
</getRevisionsResponse>
</soapenv:Body>
</soapenv:Envelope>

```

**See also**      getObject, loadTable



## undoRedline

<b>Service</b>	To revert a redlined entity in Agile PLM by issuing an undo operation on the redline.
<b>Usage</b>	Relevant details are used to form the request object. Success of the operation may be verified using the status code in the response object.
<b>Basic Steps</b>	<p>To undo the redlining of an entity:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the undoRedline operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to undo redlines for several objects with a single Web Service call.</li> <li>3. Identify the object whose redlines will be canceled by specifying the class identifier and the object number.</li> <li>4. The revision of the object may then be specified by using a property type object.</li> <li>5. Obtain and specify the rowIds corresponding to the rows where the undoRedline operation will be performed.</li> <li>6. Specify the redline table on which the undoRedline operation has to be executed. To obtain a list of possible table values, use the object of redline table type.</li> <li>7. Set the requests and execute the Web Service using the PC Web Service stub. Use the status code to ascertain the success of the operation.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <undoRedline xmlns="http://xmlns.oracle.com/AgileObjects/Core/Pc/V1">
        <request xmlns="">
          <requests>
            <classIdentifier>Part</classIdentifier>
            <objectNumber>P1242809159264</objectNumber>
            <redlineTable>TABLE_REDLINEBOM</redlineTable>
            <rowId>6201465</rowId>
            <options>
              <propertyName>revision</propertyName>
              <propertyValue>B</propertyValue>
            </options>
          </requests>
        </request>
      </undoRedline>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>

```

```
<undoRedlineResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/v1">
  <response xmlns="">
    <messageId xsi:nil="true"/>
    <messageName xsi:nil="true"/>
    <statusCode>SUCCESS</statusCode>
    <responses/>
  </response>
</undoRedlineResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**See also**      [loadTable](#)

## isRedlineModified

**Service** To determine whether a particular redlined entity in Agile PLM has been modified or not.

**Usage** Relevant details are used to form the request object. The response object includes information that will denote whether the specified red line was modified.

**Basic Steps** To determine if a redlined entity is modified or not:

1. Create the request object for the isRedlineModified operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to verify the redline status of rows in several objects with a single Web Service call.
3. Identify the object whose redlines will be verified by specifying the class identifier and the object number.
4. The revision of the object may then be specified by using a property type object.
5. Obtain and specify the rowIds for the rows whose redline status is being queried.
6. Specify the redline table on which the undoRedline operation has be executed. To obtain a list of possible table values, use the object redline table type.
7. Set all the requests and use the Agile Stub of the PC Web Service to invoke the isRedlineModified Web Service.
8. Use the status code to ascertain the success of the operation and use the value 'IsRedlineModified' from the response object to obtain the redline statuses.

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <isRedlineModified
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Pc/V1">
        <request xmlns="">
          <requests>
            <classIdentifier>Part</classIdentifier>
            <objectNumber>P1242818906603</objectNumber>
            <redlineTable>TABLE_REDLINEBOM</redlineTable>
            <rowId>6201729</rowId>
            <options>
              <propertyName>revision</propertyName>
              <propertyValue>B</propertyValue>
            </options>
          </requests>
        </request>
        <requests>
          <classIdentifier>Part</classIdentifier>
          <objectNumber>P1242818906603</objectNumber>
          <redlineTable>TABLE_REDLINEBOM</redlineTable>
```

```
<?xml version="1.0" encoding="utf-8"?>
  <options>
    <propertyName>revision</propertyName>
    <propertyValue>B</propertyValue>
  </options>
</requests>
</request>
</isRedlineModified>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <isRedlineModifiedResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Pc/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <responses>
            <isRedlineModified>true</isRedlineModified>
          </responses>
          <responses>
            <isRedlineModified>false</isRedlineModified>
          </responses>
        </response>
      </isRedlineModifiedResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      loadTable





## Search Web Services

This chapter includes the following:

▪ quickSearch .....	181
▪ advancedSearch .....	184
▪ getSearchableAttributes .....	186

### quickSearch

**Service** To retrieve a list of all Agile objects whose specifications match the search criteria specified in the request object.

**Usage** Object name, number (ID), or description may be used to form the criteria. The response object contains a collection of which of Agile objects, which were successfully queried for. Success of the operation is verified by using the status code in the response object.

**Basic Steps** To retrieve a list of desired Agile Objects:

1. Create the request object for the quickSearch operation.
2. Specify the type of object has to be queried for, by providing the class identifier details. Keywords are then set into the element 'keywords', these keywords are used to form the search criteria. Wildcards, such as '\*', may also be used as a part of the keyword.
3. Search for all objects belonging to a class, say 'Part'.
4. Search for all objects starting with certain characters, say 'P0', which is of 'P00001'.
5. Specify whether the Web Service should search for the keyword within files and attachments in addition to searching for objects.
6. The agile Stub is used to make the quickSearch Web Service call. The status code obtained from the response object is printed to verify the success of the quickSearch operation.
7. If the status code is not 'Success', then populate the list of exceptions returned by the Web Service.
8. If the Web Service call was successful, then display the search results.

#### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```

<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <quickSearch xmlns="http://xmlns.oracle.com/AgileObjects/Core/Search/V1">
        <request xmlns="">
          <classIdentifier>Part</classIdentifier>
          <keywords>P0*</keywords>
          <searchFiles>>false</searchFiles>
        </request>
      </quickSearch>
    </soapenv:Body>
  </soapenv:Envelope>
  <?xml version="1.0" encoding="utf-8"?>
    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Body>
        <quickSearchResponse
          xmlns="http://xmlns.oracle.com/AgileObjects/Core/Search/V1">
          <response xmlns="">
            <messageId xsi:nil="true"/>
            <messageName xsi:nil="true"/>
            <statusCode>SUCCESS</statusCode>
            <table>
              <tableIdentifier>
                <objectId>0</objectId>
                <tableId>-102</tableId>
                <tableName xsi:nil="true"/>
              </tableIdentifier>
              <row rowId="13">
                <objectReferentId>
                  <classId>10141</classId>
                  <className>Part</className>
                  <objectId>6098830</objectId>
                  <objectName xsi:nil="true"/>
                </objectReferentId>
                <ItemType xsi:type="xs:string"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="1">Part</ItemType>
                <Number xsi:type="xs:string"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="2">P00003</Number>
                <Description xsi:type="xs:string"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="3"></Description>
                <LifecyclePhase xsi:type="xs:string"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  attributeId="4">Preliminary</LifecyclePhase>
                <Rev xsi:type="xs:string"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="5"></Rev>
              </row>
              <row rowId="14">
                <objectReferentId>
                  <classId>10141</classId>
                  <className>Part</className>
                  <objectId>6098836</objectId>
                  <objectName xsi:nil="true"/>
                </objectReferentId>
                <ItemType xsi:type="xs:string"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="1">Part</ItemType>
                <Number xsi:type="xs:string"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="2">P00005</Number>
                <Description xsi:type="xs:string"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="3"></Description>
                <LifecyclePhase xsi:type="xs:string"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  attributeId="4">Preliminary</LifecyclePhase>
                <Rev xsi:type="xs:string"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="5"></Rev>
              </row>
            </table>
          </response>
        </quickSearchResponse>
      </soapenv:Body>
    </soapenv:Envelope>
  </?xml>

```



**See also**      `getSearchableAttributes`, `advancedSearch`

## advancedSearch

- Service** To retrieve a list of all Agile objects whose specifications match the advanced search criteria specified in the request object.
- Usage** Advanced search provides options for forming complex search criteria. The response object contains a collection of which of Agile objects which were successfully queried for. Success of the operation is verified using the status code in the response object.
- Basic Steps** To retrieve a list of Agile Objects using Advanced Search:
1. Create the request object for the advancedSearch operation.
  2. Specify the type of object has to be queried for, by providing the class identifier details. Specify whether the search is to be case sensitive.
  3. Advanced searches are executed by forming the search criteria using certain query syntax to construct the query details.
  4. The query listed here searches for all parts containing the characters 'P0' and whose description holds a legitimate value.
  5. The attributes that we desire to obtain as a part of the search result set may be explicitly specified.
  6. Advanced search also provides for other options such as usage of parameters as a part of the query, specification of search type, visibility level and usage of attributes. Some of these variations of the advanced search Web Service are set as follows:  

```
advancedSearchRequestType.setParams();  
advancedSearchRequestType.setType(); setSearchType  
advancedSearchRequestType.setVisibility();
```
  7. The Agile Stub is used to make the advancedSearch Web Service call. The status code obtained from the response object is printed to verify the success of the advancedSearch operation.
  8. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
  9. If the Web Service call was successful, then display the search results.

### Sample Code SOAP

```
==== Request ====  
<?xml version="1.0" encoding="UTF-8"?>  
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
    <soapenv:Body>  
      <advancedSearch  
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Search/V1">  
        <request xmlns="">  
          <classIdentifier>Part</classIdentifier>
```

```

        <criteria>[Title Block.Number] contains 'P0' &amp;&amp; [Title
Block Description] contains 'Gimme'</criteria>
        <caseSensitive>>false</caseSensitive>
        <displayName>Search123</displayName>
        <resultAttributes>Title Block.Number</resultAttributes>
        <resultAttributes>Title Block.Description</resultAttributes>
        <resultAttributes>Title Block.Lifecycle Phase</resultAttributes>
    </request>
</advancedSearch>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <advancedSearchResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Search/V1">
            <response xmlns="">
                <messageId xsi:nil="true"/>
                <messageName xsi:nil="true"/>
                <statusCode>SUCCESS</statusCode>
                <table>
                    <tableIdentifier>
                        <objectId>6112848</objectId>
                        <objectName>Search123</objectName>
                        <tableId>-102</tableId>
                        <tableName xsi:nil="true"/>
                    </tableIdentifier>
                    <row rowId="1">
                        <objectReferentId>
                            <classId>10141</classId>
                            <className>Part</className>
                            <objectId>6098826</objectId>
                            <objectName xsi:nil="true"/>
                        </objectReferentId>
                        <number xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="1001">P00001</number>
                        <description xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="1002">Gimme all yer
money, AAAARRRRR ye land-lover!!</description>
                        <lifecyclePhase xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1"
attributeId="1084">
                            <selection>
                                <id>976</id>
                                <apiName>PRELIMINARY</apiName>
                                <value>Preliminary</value>
                            </selection>
                        </lifecyclePhase>
                    </row>
                </table>
            </response>
        </advancedSearchResponse>
    </soapenv:Body>
</soapenv:Envelope>

```

**See also** `getAllClasses`, `getSearchableAttributes`

## getSearchableAttributes

<b>Service</b>	To retrieve a list of all searchable attributes on a baseclass, class or a subclass.
<b>Usage</b>	The request object is formed using relevant details. The response object contains the attributes that were queried for. Success of the operation may be verified using the status code in the response object.
<b>Basic Steps</b>	<p>To retrieve a list of all searchable attributes on a baseclass, class or subclass:</p> <ol style="list-style-type: none"><li>1. Create the request object to obtain a list of searchable attributes for a given class.</li><li>2. The Agile Stub is used to make the getSearchableAttributes Web Service call. The status code obtained from the response object is printed to verify the success of the getSearchableAttributes operation.</li><li>3. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li><li>4. If the Web Service call was successful, then display the attributes retrieved.</li></ol>

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getSearchableAttributes
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Search/V1">
        <request xmlns="">
          <classIdentifier>Part</classIdentifier>
        </request>
        </getSearchableAttributes>
      </soapenv:Body>
    </soapenv:Envelope>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <getSearchableAttributesResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Search/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
          <attributes>
            <nodeId>2000004143</nodeId>
            <apiName>complianceCalculatedDate</apiName>
            typeATTRIBUTE</type>
            <displayName>Compliance Calculated Date</displayName>
            <dataType>3</dataType>
            <searchable>true</searchable>
            <visible>true</visible>
            <required>false</required>
            <maxLength>2147483647</maxLength>
            <properties>
              <propertyId>1</propertyId>
            </properties>
          </attributes>
        </response>
      </getSearchableAttributesResponse>
    </soapenv:Body>
  </soapenv:Envelope>
</soapenv:Envelope>
```

```

        </apiName>AttType</apiName>
        <displayName>AttType</displayName>
        <readOnly>false</readOnly>
        <AttType xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"></AttType>
    </properties>
    <properties>
        <propertyId>9</propertyId>
        <apiName>Visible</apiName>
        <displayName>Visible</displayName>
        <readOnly>true</readOnly>
        <Visible xsi:type="common:AgileListEntryType"
xmlns:common="http://xmlns.oracle.com/AgileObjects/Core/Common/V1">
            <selection>
                <id>1</id>
                <apiName>YES</apiName>
                <value>Yes</value>
            </selection>
        </Visible>
    </properties>
</attributes>
</response>
</getSearchableAttributesResponse>

```

**See also**      `getAttributes`



## Tables Web Services

This chapter includes the following:

▪ isReadOnlyTable .....	189
▪ clearTable .....	191
▪ copyTable .....	193
▪ addRows .....	195
▪ updateRows .....	197
▪ removeRows .....	199
▪ loadTable .....	201

### isReadOnlyTable

**Service** To query a specific Agile Table object and determine if the table status is 'read-only'.

**Usage** The request object consists of class identifier, object id and table identifier that identify the table. The response object returns true or false for read-only status of the table, besides the table name and table display name.

**Basic Steps** To get the read only status of a table:

1. Create the request object for the isReadOnly operation.
2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to query several tables together.
3. For each batched request, specify the table whose status is to be queried.
4. Tables in Agile Web Services are defined as RequestTableType objects. A specific table may be identified by specifying the class identifier and table identifier attributes.
5. The request objects are set and the agile Stub is used to make the isReadOnly Web Service call. The status code obtained from the response object, <isReadOnlyTable>, is printed to verify the success of the isReadOnly operation.
6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.
7. If the Web Service call was successful, then display the readonly status of the tables queried.

#### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
```

```

    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <isReadOnlyTable
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
            <request xmlns="">
                <isReadOnlyTable>
                    <agileTable>
                        <classIdentifier>Part</classIdentifier>
                        <objectNumber>P00711</objectNumber>
                        <tableIdentifier>807</tableIdentifier>
                    </agileTable>
                </isReadOnlyTable>
            </request>
            <isReadOnlyTable>
                <agileTable>
                    <classIdentifier>ECO</classIdentifier>
                    <objectNumber>C00217</objectNumber>
                    <tableIdentifier>809</tableIdentifier>
                </agileTable>
            </isReadOnlyTable>
        </request>
    </isReadOnlyTable>
</soapenv:Body>
</soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <isReadOnlyTableResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
            <response xmlns="">
                <messageId xsi:nil="true"/>
                <messageName xsi:nil="true"/>
                <statusCode>SUCCESS</statusCode>
                <isTableReadOnly>
                    <agileTable>
                        <classId>10141</classId>
                        <className>Part</className>
                        <objectId>6112208</objectId>
                        <objectName>P00711</objectName>
                        <tableId>807</tableId>
                        <tableName>Attachments</tableName>
                        <tableDisplayName>Attachments</tableDisplayName>
                    </agileTable>
                    <isReadOnlyTable>false</isReadOnlyTable>
                </isTableReadOnly>
            </response>
            <isTableReadOnly>
                <agileTable>
                    <classId>6141</classId>
                    <className>ECO</className>
                    <objectId>6112212</objectId>
                    <objectName>C00217</objectName>
                    <tableId>809</tableId>
                    <tableName>AffectedItems</tableName>
                    <tableDisplayName>Affected Items</tableDisplayName>
                </agileTable>
                <isReadOnlyTable>false</isReadOnlyTable>
            </isTableReadOnly>
        </isReadOnlyTableResponse>
    </soapenv:Body>
</soapenv:Envelope>

```



## clearTable

<b>Service</b>	To purge the contents of an Agile Table object by removing all its rows.
<b>Usage</b>	The request object consists of class identifier and table identifier. Success of the operation may be verified using the status code in the response object.
<b>Basic Steps</b>	<p>To clear a table:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the ClearTable operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to clear several tables.</li> <li>3. For each batched request, specify the table whose contents are to be cleared.</li> <li>4. Tables in Agile Web Services are defined as request table type objects. A specific table is identified by specifying the class identifier and table identifier attributes.</li> <li>5. The request objects are set and the agile Stub is used to make the clearTable Web Service call. The status code obtained from the response object is printed to verify the success of the clearTable operation.</li> <li>6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>7. If the Web Service call was successful, then display the tables that were cleared.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <clearTable xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
        <request xmlns="">
          <clearTable>
            <agileTable>
              <classIdentifier>Part</classIdentifier>
              <objectNumber>P00707</objectNumber>
              <tableIdentifier>807</tableIdentifier>
            </agileTable>
          </clearTable>
          <clearTable>
            <agileTable>
              <classIdentifier>ECO</classIdentifier>
              <objectNumber>C00216</objectNumber>
              <tableIdentifier>809</tableIdentifier>
            </agileTable>
          </clearTable>
        </request>
      </clearTable>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>

```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
  <soapenv:Body>
    <clearTableResponse
      xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
        <statusCode>SUCCESS</statusCode>
      </response>
    </clearTableResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

**See also**      [loadTable](#)

## copyTable

<b>Service</b>	To copy the contents of an Agile Table object from a table to another table.
<b>Usage</b>	The request object consists of relevant information that identifies the tables. Success of the operation may be verified using the status code in the response object.
<b>Basic Steps</b>	<p>To copy the table contents:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the copyTable operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to copy several tables together.</li> <li>3. For each batched request, specify the table from which the contents are to be copied and the target table.</li> <li>4. Tables in Agile Web Services are defined as request table type objects. A specific table is identified by specifying the class identifier and table identifier attributes.</li> <li>5. The request objects are set and the Agile Stub is used to make the copyTable Web Service call. The status code obtained from the response object is printed to verify the success of the copyTable operation.</li> <li>6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>7. If the Web Service call was successful, then display the tables which were copied.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <copyTable xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
        <request xmlns="">
          <copyTable>
            <sourceTable>
              <classIdentifier>Part</classIdentifier>
              <objectNumber>P00709</objectNumber>
              <tableIdentifier>2000001404</tableIdentifier>
            </sourceTable>
            <targetTable>
              <classIdentifier>Part</classIdentifier>
              <objectNumber>P00710</objectNumber>
              <tableIdentifier>2000001404</tableIdentifier>
            </targetTable>
          </copyTable>
        </request>
      </copyTable>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>

```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
  <soapenv:Body>
    <copyTableResponse
      xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
      <response xmlns="">
        <messageId xsi:nil="true"/>
        <messageName xsi:nil="true"/>
        <statusCode>SUCCESS</statusCode>
      </response>
    </copyTableResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

**See also**      loadTable, clearTable

## addRows

<b>Service</b>	To add rows in an Agile Table object
<b>Usage</b>	Uses details of the new row and the table. The request object is built using rowId and objectInfo. Success of the operation is verified using the status code in the response object.
<b>Basic Steps</b>	<p>To add a row in a table :</p> <ol style="list-style-type: none"> <li>1. Create the request object for the addRows operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to add several rows.</li> <li>3. For each batched request, specify the table in which the new rows have to be added.</li> <li>4. Using the addRows Web Service, we add a child element to a Part by adding rows to the BOM table of the parent object. Tables in Agile Web Services are defined as request table type objects. A specific table may be identified by specifying the class identifier and table identifier attributes as shown. Rows in Agile are defined as Agile row type objects. Here message elements may be used to specify the row data. This is similar to how the '_any' information is specified in a createObject or getObject Business service call.</li> <li>5. The request objects are set and the Agile Stub is used to make the addRows Web Service call. The status code obtained from the response object is printed to verify the success of the addRows operation.</li> <li>6. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <addRows xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
        <request xmlns="">
          <data>
            <objectInfo>
              <classIdentifier>ECO</classIdentifier>
              <objectNumber>C00218</objectNumber>
              <tableIdentifier>809</tableIdentifier>
            </objectInfo>
            <row rowId="0">
              <key1054 attributeId="1054">P00713</key1054>
            </row>
          </data>
        </request>
      </addRows>
    </soapenv:Body>
  </soapenv:Envelope>

```

```
----- Response -----
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <addRowsResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </addRowsResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      [getTableMetadata](#), [isReadOnlyTable](#)

## updateRows

<b>Service</b>	To updated an existing row in an Agile Table object
<b>Usage</b>	Uses details of the modified row and the table. The request object is built using objectInfo and rowId attributes. Success of the operation may be verified using the status code in the response object.
<b>Basic Steps</b>	<p>To update a row in a table:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the updateRows operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to update several tables at once.</li> <li>3. For each batched request, specify the following details: <ul style="list-style-type: none"> <li>- The table whose rows will be updated</li> <li>- The rowId of the row that will be updated with new values</li> <li>- The new row that will replace the existing row.</li> <li>- The updated content is specified in the form of Message Elements.</li> </ul> </li> <li>4. Using the updateRows Web Service, in this particular case we modify the effective date attribute on the affected items table of a change object by finding and updating a specific row.  <p>Tables in Agile Web Services are defined as request table type objects. A specific table may be identified by specifying the class identifier and table identifier attributes. Rows in Agile are defined as agile row type objects. Here message elements may be used to specify the row data. This is similar to how the <code>_any</code> information is specified in a <code>createObject</code> or <code>getObject</code> Business service call.</p> </li> <li>5. Obtain the rowId of the specified part in the affected items table of the change by issuing a <code>loadTable</code> Web Service on the table and iterating through the rows till it finds the row queried for. The update row is then set with this rowId as shown.</li> <li>6. Specify the field to be updated through a message element attribute.</li> <li>7. The request objects are set and the agile Stub is used to make the updateRows Web Service call. The status code obtained from the response object is printed to verify the success of the updateRows operation.</li> <li>8. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> <li>9. If the Web Service call was successful, then confirm the success of the operation.</li> </ol>

### Sample Code SOAP

```
==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
```

```

    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Body>
        <updateRows xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
          <request xmlns="">
            <data>
              <objectInfo>
                <classIdentifier>ECO</classIdentifier>
                <objectNumber>C00218</objectNumber>
                <tableIdentifier>809</tableIdentifier>
              </objectInfo>
              <row>
                <rowId>6112255</rowId>
                <row rowId="0">
                  <modified_element
attributeId="newRev">2</modified_element>
                </row>
              </row>
            </data>
          </request>
        </updateRows>
      </soapenv:Body>
    </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Body>
        <updateRowsResponse
xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
          <response xmlns="">
            <messageId xsi:nil="true"/>
            <messageName xsi:nil="true"/>
            <statusCode>SUCCESS</statusCode>
          </response>
        </updateRowsResponse>
      </soapenv:Body>
    </soapenv:Envelope>

```

**See also**      [getTableMetadata](#), [isReadOnlyTable](#), [loadTable](#)



## removeRows

<b>Service</b>	To remove an existing row belonging to an Agile Table object
<b>Usage</b>	The request object includes details of the row to be removed and the table identifier. Success of the operation may be verified using the status code in the response object.
<b>Basic Steps</b>	<p>To remove a row from a table:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the removeRows operation.</li> <li>2. Create an array of requests. Batch operations may be performed by populating as many request objects as required to remove several rows.</li> <li>3. For each batched request, specify the following details: <ul style="list-style-type: none"> <li>- The table whose rows will be removed</li> <li>- The rowId of the row that will be updated with new values</li> </ul> </li> <li>4. Tables in Agile Web Services are defined as request table type objects. A specific table may be identified by specifying the class identifier and table identifier attributes.</li> <li>5. Using the removeRows Web Service, for example, remove a child element in the BOM table of its parent object by finding and removing its specific row.</li> <li>6. Obtain the rowId of the specified BOM by issuing a loadTable Web Service on the BOM table of the parent and iterating through the rows till it finds the row queried for. The request object is then set with this rowId. The row will be removed by using this value to identify it.</li> <li>7. The request objects are set and the Agile Stub is used to make the removeRows Web Service call. The status code obtained from the response object is printed to verify the success of the removeRows operation.</li> <li>8. If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <removeRows xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
      <request xmlns="">
        <rows>
          <objectInfo>
            <classIdentifier>Part</classIdentifier>
            <objectNumber>P00717</objectNumber>
            <tableIdentifier>803</tableIdentifier>
          </objectInfo>
          <rowId>6112309</rowId>
        </rows>
      </request>
    </removeRows>
  </soapenv:Body>
</soapenv:Envelope>

```

```
        </request>
      </removeRows>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <removeRowsResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
          <statusCode>SUCCESS</statusCode>
        </response>
      </removeRowsResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

**See also**      isReadOnlyTable, loadTable

## loadTable

<b>Service</b>	To load the content of an existing Agile Table object
<b>Usage</b>	The request object contains identifier of the table to be retrieved and the information to be obtained from it. Success of the operation may be verified using the status code in the response object.
<b>Basic Steps</b>	<p>To load the contents of a table:</p> <ol style="list-style-type: none"> <li>1. Create the request object for the loadTable operation.</li> <li>2. For each request, specify the table(s) whose contents are to be retrieved. Tables in Agile Web Services are defined as request table type objects. A specific table is identified by specifying the class identifier and table identifier attributes.</li> <li>3. The request objects are set and the agile Stub is used to make the loadTable Web Service call. The status code obtained from the response object is printed to verify the success of the loadTable operation.</li> <li>4. If the status code indicates that the Web Service call was not successful, then populate a list of exceptions.</li> <li>5. If the status code is 'SUCCESS', then populate the information retrieved by the Web Service call.</li> <li>6. The method displayTableContents in this loadTable sample receives an array of tables as input and displays their content.</li> </ol>

### Sample Code SOAP

```

==== Request ====
<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <loadTable xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
        <request xmlns="">
          <tableRequest>
            <classIdentifier>ECO</classIdentifier>
            <objectNumber>C00220</objectNumber>
            <tableIdentifier>809</tableIdentifier>
          </tableRequest>
        </request>
      </loadTable>
    </soapenv:Body>
  </soapenv:Envelope>
==== Response ====
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <loadTableResponse
        xmlns="http://xmlns.oracle.com/AgileObjects/Core/Table/V1">
        <response xmlns="">
          <messageId xsi:nil="true"/>
          <messageName xsi:nil="true"/>
        </response>
      </loadTableResponse>
    </soapenv:Body>
  </soapenv:Envelope>

```

```
</statusCode>SUCCESS</statusCode>
<tableContents>
  <tableIdentifier>
    <classId>6141</classId>
    <className>ECO</className>
    <objectId>6112315</objectId>
    <objectName>C00220</objectName>
    <tableId>809</tableId>
    <tableName>AffectedItems</tableName>
    <tableDisplayName>Affected Items</tableDisplayName>
  </tableIdentifier>
  <row rowId="6112318">
    <objectReferentId>
      <classId>10141</classId>
      <className>Part</className>
      <objectId>6112311</objectId>
      <objectName>P00719</objectName>
    </objectReferentId>
    <hasBeenRedlinedImage xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="6350"
readOnly="True">false</hasBeenRedlinedImage>
    <itemNumber xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="1054"
readOnly="False">P00719</itemNumber>
    <attachmentsImage xsi:type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeId="12623"
readOnly="True">false</attachmentsImage>
  </row>
</tableContents>
</response>
</loadTableResponse>
</soapenv:Body>
</soapenv:Envelope>
```





## This Appendix includes the following:

---

▪ Working with Java Samples.....	205
▪ AddFileSOAPAttachment Method .....	207
▪ Helper Methods .....	209
▪ Troubleshooting .....	215

## Working with Java Samples

The Java Sample Codes covered in this book, and those available for download from [eDelivery.oracle.com](http://eDelivery.oracle.com) site, demonstrate various usage characteristics of Agile Web Services. Apart from outlining the basic cases for each Web Service, they also elaborate upon more specific cases that involve usage of options or mandatory message elements.

Download these Sample Codes in 'src' directory. They are categorized into different packages based on the type of Web Service, which are AdminMetaData, Attachment, Business, Collaboration, PC, Search and Table.

Batch files for building and running samples independent of a java IDE are also provided.

## Building Stubs and Compiling the Samples

A batch file **build.bat** has been provided, which is located in the main directory **JavaWeb Servicesamples**. This batch file uses **ant** tasks to generate stubs for Agile Web Services and compiles the java sample files after generation of these stubs.

Along with **build.bat**, you will find a file **custom.properties** file that specifies the basic configuration properties, such as the Agile server URL, username, password of your user and also the URL of DFM. Unless this property file is edited to reflect the values appropriate for your Agile environment, you will not be able to generate stubs or run the samples.

After ensuring that 'custom.properties' has been modified appropriately, run the batch file 'build.bat'. Running the same through a command prompt may help in identifying error statements (if any) that are echoed onto the console. If the message 'BUILD SUCCESSFUL' is displayed on the console then the build process was completed without any errors. However, if the message 'BUILD FAILED' is observed on the console, your 'custom.properties' configuration may be incorrect and you should verify the same.

The stubs and the compiled samples are added to the folder build/built/\*.jar as two jar files 'ws\_samples.jar' and 'ws\_stubs.jar' which are used later while running the sample.

## Executing the Samples using ant Task

After building the stubs and compiling the sample files by following the steps outlined in [Building](#)

[Stubs and Compiling the Samples](#) on page 205 section,

any sample file may be readily executed by using the batch file run.bat and specifying the fully qualified class name as an argument.

Browse the source directory 'JavaWeb Servicesamples/src' to find the package structure of the sample that you are looking for.

For example, to the run the sample 'CreateObjectAPIName' that creates an object using API names, the following command must be executed through command prompt:

```
run business.create.CreateObjectAPIName
```

Similarly, to run a sample from another sample package, say AddRowsSiteSpecific of the table Web Service, use the command:

```
run table.addrows.AddRowsSiteSpecific
```

If no argument is passed to run.bat, the all the available samples will executed sequentially.

---

**Note** While running samples using this ant task, the Agile server URL, username and password properties are retrieved from the same 'custom.properties' file that was used for building stubs.

---

## Executing the Samples using a Java IDE

To run the sample files from a Java IDE, such as JDeveloper or Eclipse, create a new project or workspace in your IDE (as applicable to your IDE) and in your project properties modify / add project source paths to include the 'JavaWeb Servicesamples/src' directory where the sample source code is located.

You will also need to update your project library or classpath information to include all the necessary classes of Axis and other Agile jar files which have been used in the course of sample development.

Ensure that you have added all the jar files under the folders 'JavaWeb Servicesamples/build/axis' and 'JavaWeb Servicesamples/build/lib' to your library / classpath.

Any sample file may now be executed by browsing through the package structure, and running the desired sample.

---

**Note** The static variables relating to Agile server url, username and password in each java sample must be modified if you choose to run the sample through an IDE.

---

## Understanding the Code

Each java sample file contains header documentation at the class level explaining the functionality or usage scenario that the sample demonstrates. A set of static variables relating to server url, username, password and variables specific to that sample, like partNumber or folderName or nextStatus, are declared here.

If the sample is executed using the ant task (via run.bat), then the server configuration related static



variables are overridden by the method `checkArguments(String[] args())`, which obtains arguments from the ant task and reinitializes server URL username, password and DFM URL variables.

In the case of a sample executed through a Java IDE, the server configuration variables must be modified manually to reflect the server settings of your Agile server.

With the exception of `adminMetaData` services, all samples provided here use the method `prepareData()` to prepare all the data prerequisites necessary to create a scenario using which a particular Web Service may be demonstrated meaningfully.

For example, if the sample demonstrates usage of the operation `loadTable` to load a table from a particular version of an Agile object, `prepareData()` will do the following:

1. Create a part object
2. Add a change, modify the part
3. Provide a new version number
4. Release the change.

After the data has been prepared, the operation `loadTable` is now used with the option 'version' to demonstrate the retrieval of a table from a particular version of an Agile object.

If you intend to use your own data or scenario to execute a Web Service sample, comment out the '`prepareData();`' statement in the main method of that sample. After this, edit the static variables at the top of the code and specify your own data.

All the operations performed in data preparation are also achieved using Agile 93 Web Services. To gain a broader understanding of how Web Services are used in conjunction to orchestrate a larger task, examine the file `DataPrepare.java` in the package `src/run/DataPrepare.java`

The sample files are documented with comments at each stage and evince several usage characteristics of these Web Services while illustrating how basic requests are formed and how the responses obtained are used.

## AddFileSOAPAttachment Method

This sample method demonstrates addition of a file attachment to the Attachment Tab of an Agile object using SOAP.

Create the request object `AddFileAttachmentRequestType` for the `addFileAttachment` operation.

Create an array of requests of type `AgileAddFileAttachmentRequestType`. Batch operations may be performed by populating as many request objects as required to add several files to different objects with one single operation.

For each batched request, specify the unique object to whose attachment tab the files are to be added. Supply class identifier and object number information for the same.

The exact specification of the attachment to be added is defined as an object of type `AgileAddFileAttachmentRequestType`. This object includes information about the name of the file and its description and content.

While using SOAP attachments, create a datahandler to specify the file source and add the add the content as a soap attachment to the soap request. Finally set the `contentId` onto

### AgileAddFileAttachmentRequestType.

The request objects are set and the Agile Stub is used to make the addFileAttachment Web Service call. The status code obtained from the response object is printed to verify the success of the addFileAttachment operation.

If the status code is not 'SUCCESS', then populate the list of exceptions returned by the Web Service.

If the Web Service call was successful, then state the same.

```
try {
    setupServerLogin();

    AddFileAttachmentRequestType addFileAttachmentRequestType =
        new AddFileAttachmentRequestType();
    AgileAddFileAttachmentRequest agileAddFileAttachmentRequest[] =
        new AgileAddFileAttachmentRequest[1];
    agileAddFileAttachmentRequest[0] =
        new AgileAddFileAttachmentRequest();
    agileAddFileAttachmentRequest[0].setClassIdentifier("Part");
    agileAddFileAttachmentRequest[0].setObjectNumber(partNumber);
    System.out.println("Adding a SOAP attachment to the part '" +
        partNumber + "...");
    AgileAddFileAttachmentRequestType attachments[] = new
    AgileAddFileAttachmentRequestType[1];
    attachments[0] =
        new AgileAddFileAttachmentRequestType();
    attachments[0].setName("Filename.txt");
    attachments[0].setDescription("Description for file ");
    String filename = "sample123456.txt";
    BufferedWriter out =
        new BufferedWriter(new FileWriter(filename));
    out.write("Test file");
    out.close();

    DataHandler dh =
        new DataHandler(new FileDataSource(filename));
    AttachmentPart ap = new AttachmentPart(dh);
    agileStub.addAttachment(ap);
    attachments[0].setContentId(ap.getContentId());

    agileAddFileAttachmentRequest[0].setAttachments(attachments);
    agileAddFileAttachmentRequest[0].setSingleFolder(false);
    addFileAttachmentRequestType.setRequests(agileAddFileAttachmentRequest);
    AddFileAttachmentResponseType addFileAttachmentResponseType =
        agileStub.addFileAttachment(addFileAttachmentRequestType);
    System.out.println("\nSTATUS CODE: " +
        addFileAttachmentResponseType.getStatusCode());
    if
    (!addFileAttachmentResponseType.getStatusCode().toString().equals(ResponseStatusCo
    de.SUCCESS.getValue())) {
        AgileExceptionListType[] agileExceptionListType =
            addFileAttachmentResponseType.getExceptions();
        if (agileExceptionListType != null)
            for (int i = 0;
                i < agileExceptionListType.length;
                i++) {
                AgileExceptionType exceptions[] =
                    agileExceptionListType[i].getException();
                for (int j = 0;
                    j < exceptions.length; j++)
                    System.out.println("Exception Id:" +
                        exceptions[j].getExceptionId() +
                        "\nMessage: " +
                        exceptions[j].getMessage());
            }
    }
```

```

    addFileAttachmentResponseType.getWarnings();
    if (agileWarningListType != null)
        for (int i = 0;
            i < agileWarningListType.length;
            i++) {
            AgileWarningType warnings[] =
                agileWarningListType[i].getWarning();
            for (int j = 0;
                j < warnings.length; j++)
                System.out.println("Warning Id: " +
                                    warnings[j].getWarningId() +
                                    "\nMessage: " +
                                    warnings[j].getMessage());
        }
    } else {
        AgileAddFileAttachmentResponse responses[] =
            addFileAttachmentResponseType.getResponses();
        if (responses != null)
            for (int i = 0; i < responses.length;
                i++) {
                System.out.println("The specified SOAP attachment was successfully
added to the Attachment tab");
                System.out.println("of the object: " +
                                    responses[i].getObjectNumber());
            }
    }
}
}

```

## Helper Methods

The `getRowId` and `getFileId` are custom helper methods. These are not Agile Web Services operation.

### getRowId Method

**Service**      Obtaining the rowId for a row on an Agile table

**Usage**      Several table and attachment operations require the rowId as input for executing a Web Service. In such cases, the `loadTable` operation is used to load the table that contains the required row and then iterated through the results till the row is found.

To find a particular row in a table, a keyword may be used to search and identify the row. In this example, the filename is used as the key to identify a row.

Search all the rows available in the attachment table and compare all message elements with tag names 'filename' with the filename specified by the client, looking for a match. Once a match is found, the rowId information is derived from the row and returned.

Compare all 'filename' message elements, searching for a match with the filename specified by the user. If a match is found, return either the fileId or rowId based on the requirement.

---

**Note**    `getValueFromSelection` is a method written in this sample that handles all message elements of type `AgileListEntryType`. Since 'filename' is a message element of `AgileListEntryType`, the values are elicited by this method.

---

Handle all `AgileListEntryType` message elements, cycle through the selection element, obtain the actual selection value and the selection Id and add it to a `HashMap`. Here the selection value denotes the filename while the selection Id denotes the fileId.

In the case of Filefolders the message element for file name is not an `AgileListEntryType`. The value may be obtained directly.

**Basic Steps**    To get a Row ID:

1. Create the request object `LoadTableRequestType` for the `loadTable` operation.
2. For each request, specify the table to which the row belongs
3. Tables in Agile Web Services are defined as `RequestTableType` objects. A specific table may be identified by specifying the class identifier and table identifier attributes.
4. The request objects are set and the agile Stub is used to make the `loadTable` Web Service call. The status code obtained from the response object is printed to verify the success of the `loadTable` operation.
5. If the status code is 'SUCCESS', then use the table results to find the required row. Once the row is found, its rowId is returned.
6. Search for the necessary row by using the filename to look for a match and return the rowId.
7. If the status code indicates that the Web Service call was not successful, then populate a list of exceptions.

**Sample Code**    `getRowId`

```
public static int getRowId(String filename, String classIdentifier, String
objectNumber, String tableId){
    try{
        setupServerLogin_LoadTable();
        LoadTableRequestType loadTableRequestType = new LoadTableRequestType();

        RequestTableType table[] = new RequestTableType[1];
        table[0] = new RequestTableType();
        table[0].setClassIdentifier(classIdentifier);
        table[0].setObjectNumber(objectNumber);
        table[0].setTableIdentifier( tableId );

        loadTableRequestType.setTableRequest(table);
        LoadTableResponseType loadTableResponseType =
        agileStub_Table.loadTable(loadTableRequestType);
        System.out.println("Obtaining row Id / fileId information.....");
        System.out.println("STATUS CODE: " + loadTableResponseType.getStatusCode() );

        if( loadTableResponseType.getStatusCode().toString().equals(
        ResponseStatusCode.SUCCESS.getValue() ) ){
            AgileTableType[] tables = loadTableResponseType.getTableContents();
            return findRowId(tables, filename);
        }
        else{
            System.out.println("<Failed to load table information>");
        }
    }
}
```

```

        AgileExceptionListType[] agileExceptionListType =
loadTableResponseTime.getExceptions();
        if (agileExceptionListType != null)
            for (int i=0; i<agileExceptionListType.length; i++){
                AgileExceptionType exceptions[] =
agileExceptionListType[i].getException();
                for (int j=0; j<exceptions.length; j++){
                    System.out.println(exceptions[j].getMessage() );
                }
            }

        AgileWarningListType agileWarningListType[] =
loadTableResponseType.getWarnings();
        if (agileWarningListType != null)
            for (int i=0; i<agileWarningListType.length; i++){
                AgileWarningType warnings[] = agileWarningListType[i].getWarning();
                for (int j=0; j<warnings.length; j++){
                    System.out.println("Warning Id: " + warnings[j].getWarningId() +
"\nMessage: " + warnings[j].getMessage() );
                }
            }
    }
    catch (Exception ex) {
        ex.printStackTrace();
    }
    return -1;
}

public static int findRowId(AgileTableType[] tables, String filename){
    if (tables != null)
        for (int i=0; i<tables.length; i++){
            AgileRowType[] rows = tables[i].getRow();
            if (rows != null)
                for (int j=0; j<rows.length; j++){
                    MessageElement[] messages = rows[j].get_any();
                    for (int m=0; m<messages.length; m++){
                        if (
messages[m].getName().toString().equalsIgnoreCase("filename") ){
                            HashMap fileValues[] =
getValuesFromSelection(messages[m]);
                            for (HashMap fileValue:fileValues)
                                if (fileValue.get("filename").equals(filename) ){
                                    System.out.println("Row Id successfully
retrieved.");
                                    return rows[j].getRowId();
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    return 0;
}

public static HashMap[] getValuesFromSelection(MessageElement element){
    HashMap fileValues[] = null;
    if (element.getType().getLocalPart().equals("AgileListEntryType")){
        AgileListEntryType list = (AgileListEntryType) element.getObjectValue();
        SelectionType selection[] = list.getSelection();
        fileValues = new HashMap [selection.length];
        for (int i=0; i<selection.length; i++){
            fileValues[i] = new HashMap();
            fileValues[i].put("filename", selection[i].getValue() );
            fileValues[i].put("fileid", selection[i].getId() );
        }
    }
    else{
        fileValues = new HashMap [1];
        fileValues[0] = new HashMap();
        fileValues[0].put("filename", element.getFirstChild().getNodeValue() );
        fileValues[0].put("fileid", null );
    }
}

```

```
} return fileIdValue;
```

## getFileId Method

**Service** Obtaining the FileId for a Row on an Agile Table.

**Usage** Several attachment operations require the fileId as input for executing. For example, if a particular attachment row has several files associated with it, the fileId is necessary to differentiate between each file available on that row.

In such cases, the we first use the loadTable Web Service to load the table that contains the required row and then iterate through the results till the row has been found. To find a particular row in a table, a keyword may be used to search and identify the row.

In this example, the filename is used as the key to identify a row. Once the filename message element is obtained as an AgileListEntry Type element, the 'id' value of the SelectionType element may be returned. This 'id' tag corresponds to the fileId of that file attachment.

Search all the rows available in the attachment table and compare all message elements with tag names 'filename' with the filename specified by the client, looking for a match. Once a match is found, fileId information is derived from the filename selection elements, whose 'id' corresponds to the fileId.

Compare all 'filename' message elements, searching for a match with the filename specified by the user. If a match is found, return either fileId.

---

**Note** If both rowId and fileId values are necessary, then a similar approach may be used to obtain both the rowId and fileId for a particular file attachment from an attachment row.

**Note** getValueFromSelection is a method written in this sample that handles all message elements of type AgileListEntryType. Since 'filename' is a message element of AgileListEntryType, the values are elicited by this method.

---

**Usage** Handle all AgileListEntryType message elements, cycle through the selection element, obtain the actual selection value and the selection Id and add it to a HashMap. Here the selection value denotes the filename while the selection Id denotes the fileId.

In the case of Filefolders the message element for file name is not an AgileListEntryType. The value may be obtained directly.

**Basic Steps** To get a File ID:

1. Create the request object LoadTableRequestType for the loadTable operation.

2. For each request, specify the table to which the row belongs. Tables in Agile Web Services are defined as RequestTableType objects. A specific table may be identified by specifying the class identifier and table identifier attributes.
3. The request objects are set and the agile Stub is used to make the loadTable Web Service call. The status code obtained from the response object is printed to verify the success of the loadTable operation.
4. If the status code is 'SUCCESS', then find and retrieve the fileId.
5. Search for the necessary fileId by using the filename to look for a match and return either of the two, as per the requirement specified in the input parameter 'methodType'.
6. If the status code indicates that the Web Service call was not successful, then populate a list of exceptions.

### Sample Code `getFileId`

```
public static int getFileId(String filename, String clazz, String objectNumber,
String tableId){
    try{
        setupServerLogin_LoadTable();
        LoadTableRequestType loadTableRequestType = new LoadTableRequestType();
        RequestTableType table[] = new RequestTableType[1];
        table[0] = new RequestTableType();
        table[0].setClassIdentifier(clazz);
        table[0].setObjectNumber(objectNumber);
        table[0].setTableIdentifier( tableId );

        loadTableRequestType.setTableRequest(table);
        LoadTableResponseType loadTableResponseType =
        agileStub_Table.loadTable(loadTableRequestType);
        System.out.println("Obtaining row Id / fileId information.....");
        System.out.println("STATUS CODE: " + loadTableResponseType.getStatusCode() );

        if( loadTableResponseType.getStatusCode().toString().equals(
ResponseStatusCode.SUCCESS.getValue() ) ){
            AgileTableType[] tables = loadTableResponseType.getTableContents();
            return findFileId(tables, filename);
        }

        else{
            System.out.println("<Failed to load table information>");
            AgileExceptionListType[] agileExceptionListType =
loadTableResponseType.getExceptions();
            if(agileExceptionListType!=null)
                for(int i=0; i<agileExceptionListType.length; i++){
                    AgileExceptionType exceptions[] =
agileExceptionListType[i].getException();
                    for(int j=0; j<exceptions.length; j++)
                        System.out.println(exceptions[j].getMessage() );
                }

            AgileWarningListType agileWarningListType[] =
loadTableResponseType.getWarnings();
            if(agileWarningListType!=null)
                for( int i=0; i<agileWarningListType.length; i++){
                    AgileWarningType warnings[] = agileWarningListType[i].getWarning();
                    for(int j=0; j<warnings.length; j++)
                        System.out.println("Warning Id: " + warnings[j].getWarningId() +
"\nMessage: " + warnings[j].getMessage() );
                }
        }
    } catch (Exception ex) {
```

```
    }
    return -1;
}

public static int findFileId(AgileTableType[] tables, String filename){
    if(tables!=null)
        for(int i=0; i<tables.length; i++){
            AgileRowType[] rows = tables[i].getRow();
            if(rows!=null)
                for(int j=0; j<rows.length; j++){
                    MessageElement[] messages = rows[j].get_any();
                    for(int m=0; m<messages.length; m++){
                        if(
messages[m].getName().toString().equalsIgnoreCase("filename") ){
                            HashMap fileValues[] =
getValuesFromSelection(messages[m]);
                            for(HashMap fileValue:fileValues)
                                if( fileValue.get("filename").equals(filename) ){
                                    System.out.println("File Id successfully
retrieved.");
                                    return (Integer) fileValue.get("fileid");
                                }
                            }
                        }
                    }
                }
            }
        }
    return 0;
}

public static HashMap[] getValuesFromSelection(MessageElement element){
    HashMap fileValues[] = null;
    if(element.getType().getLocalPart().equals("AgileListEntryType")){
        AgileListEntryType list = (AgileListEntryType) element.getObjectValue();
        SelectionType selection[] = list.getSelection();
        fileValues = new HashMap [selection.length];
        for(int i=0; i<selection.length; i++){
            fileValues[i] = new HashMap();
            fileValues[i].put("filename", selection[i].getValue() );
            fileValues[i].put("fileid", selection[i].getId() );
        }
    }
    else{
        fileValues = new HashMap [1];
        fileValues[0] = new HashMap();
        fileValues[0].put("filename", element.getFirstChild().getNodeValue() );
        fileValues[0].put("fileId", null );
    }
    return fileValues;
}
```



## Appendix B

# Troubleshooting

**Issue #7639814:** API names are not available for a huge set of lists.

**Resolution:** GetLists Admin service is supported only for Admin Lists whichever can be found in **JavaClient>Admin>Lists**. Some class specific lists are not exposed in Admin Lists.

To get values for these Lists, use GetAttributes Admin service, which will return AttributeType. This, in turn, has availableValues as AdminListType. This list values can be used for updateObject or createObject.

**Issue #8253064:** Group sign-off option cannot be set from Web Services.

**Resolution:** Group sign-off is not supported in the current release.

