



Agile Product Lifecycle Management

Agile PLM EC Web Services - Quick Reference
Guide

v9.3.0.2

Part No. E17318-01

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
Chapter 1 Introduction	1
Chapter 2 EC Object Calls	3
getDesignProperties	3
getItemPropertiesByDesign	4
searchSingleObject	4
retrieveDesignStructure	5
createUpdateDesignStructure	6
createBOM	6
getDesignWhereUsed	7
Chapter 3 EC Orchestrated Calls	9
General	9
Requests	9
Responses.....	9
Alternative Approach	9
Errors.....	10
getDesignPropertiesAndRights	10
getDesignItemPropertiesAndRights	11
createUpdateThenReserve	13
reserveThenCreateUpdate	14
createUpdateThenRelate	15
retrieveDesignStructureAndMetadata	16
Chapter 4 EC Schema Types.....	19

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 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

The Oracle Agile PLM Application Suite enables enterprises to accelerate product innovation and maximize product profitability by managing the information, processes, and decisions about their products throughout the product lifecycle. The Agile PLM product provides comprehensive support for the PLM business process through PC, PQM, PPM, PCM, PGC & EC.

Businesses are starting to realize the benefits of Service Oriented Architecture (SOA) and Web Services. Web Services are rapidly emerging as the standard solution to publish business services, both within corporate firewalls as well as externally to provide integration points with business partners. Customers are also adopting SOA and Web Services technologies to enable more composite applications and enterprise-wide data integrations. SOA is emerging as the premier integration and architecture framework in today's complex and heterogeneous computing environment. SOA can help organizations streamline processes so that they can do business more efficiently, and adapt to changing needs and competition, enabling the *software as a service* concept are the preferred standards-based way to realize SOA.

Web services are designed to support interoperable machine-to-machine interaction over a network. This interoperability is gained through a set of XML-based open standards, such as WSDL, SOAP, and UDDI. These standards provide a common approach for defining, publishing, and using web services.

EC Object Calls

This chapter includes the following:

▪ getDesignProperties	3
▪ getItemPropertiesByDesign	4
▪ searchSingleObject	4
▪ retrieveDesignStructure	5
▪ createUpdateDesignStructure	6
▪ createBOM	6
▪ getDesignWhereUsed	7

The Core service covers most of the basic services, which can also be used in the EC services context. But there are some of the services that the core service does not cover as those are EC specific services. Those EC services will be described in the following respective sections in details. The EC Services are designed as document/wrapped style and uses consistent request response type which in turn uses the types defined in core services data model or extends the one from the core service data model.

getDesignProperties

Description:

The service expects a request with the following parameters from the first set of general parameters. subclass-name and include-tables are the parameters that are read from the request. The value of subclass-name is a subclass of a design for which the properties (Metadata) have to be retrieved. The value of the parameter include-tables is “true”/“false”, which indicates the service to include the metadata for all the tables or not. The default value for include-tables is “false”.

The service reads the Metadata for the given Design subclass and checks if the subclass is of type Design and retrieves the metadata for all the object attributes, Auto number sources, subclass types and table metadata (if include-tables is set to “true”). Then the AgileECClass type is created and filled with the retrieved metadata.

Request Type: getDesignProperties

The request is of type AgileEClassRequestType (see chapter "EC Schema Types").

Properties:

- subclass-name: Name of subclass
- include-tables: true/false

Response Type: getDesignPropertiesResponse

The response is of type AgileEClassResponseType (see chapter "EC Schema Types").

Exceptions:

- If the subclass-name specified is not of type Design
- If there is no subclass of the given subclass type

getItemPropertiesByDesign

Description:

The service expects a request with the following parameters from the first set of general parameters. design-id and include-tables are the parameters that are read from the request. The value of design-id is design ID for which the "where used" items have to be searched. The value of the parameter include-tables is "true"/"false", which would indicate the service to include the metadata for all the tables or not. The default value for include-tables is "false".

The service reads the design-id and retrieves the Design object and parses the WHERE USED table and retrieves the part objects where this design object is used. Then iterates through all the parts and retrieves the Metadata from each unique Part subclass that is found in this table. The metadata retrieved for each Part class found are all the object attributes, Auto number sources, subclass types and table metadata(if include-tables is set to "true"). Then the AgileECClass type is created for each retrieved part class and filled with the retrieved metadata from the part.

Returns an array of AgileECClassType for the part filled in the response object.

Request Type: getItemPropertiesByDesign

The request is of type AgileECObjectRequestType (see EC Schema Types).

Properties:

- design-id: Name of the design object
- include-tables: true/false

Response Type: getItemPropertiesByDesignResponse

The response is of type AgileECObjectResponseType (see EC Schema Types)

Exceptions:

- If the object is not found for the given ID
- If the object found is not of type Design

searchSingleObject

Description:

The service expects a request with the following parameters from the first set of general parameters. file-name, subclass-name and file-category are the parameters that are read from the request. The value of file-name is filename value in the files table of a Design object in Agile PLM, the subclass-name is the subclass name of the Design object searched and the file-category is the

criteria for the file category value in the FILES table in Designs class. The subclass-name and file-category are optional.

The service reads the values and builds an agile query for searching the objects of the type design, which has the specified values for file-name and file-category (optional) in the files table. Executes the query, takes the first object from the result and extracts the AgileObject to put in the response and sends it back.

Returns an AgileBusinessObjectType for Design filled in the response object.

Request Type: searchSingleObject

The request is of type AgileEObjectRequestType (see chapter "EC Schema Types").

Properties:

- file-name: Name of the file
- file-category: Category of the file (file type)

Response Type: searchSingleObjectResponse

The response is of type AgileEObjectResponseType (see chapter "EC Schema Types").

Exceptions:

- If the object is not found for the given ID
- If the object found is not of type Design

retrieveDesignStructure

Description:

The service expects a request to contain a single object with two attributes - Number and Version. The service implementation reads the object Id from object identifier and the version from the attributes. After reading these values successfully the service searches for this object and checks if it is of the type Design. Then the service creates an AgileObject and fills the attributes from the searched object and fill the table and rows from the SDK objects. Once the object is filled the service sends it back to the client through SOAP response.

Returns an AgileBusinessObjectType for Design filled in the response object.

Request Type: retrieveDesignStructure

The request is of type AgileEObjectRequestType (see chapter "EC Schema Types").

Response Type: retrieveDesignStructureResponse

The response is of type AgileEObjectResponseType (see chapter "EC Schema Types").

Exceptions:

- If the object is not found for the given ID
- If the object found is not of type Design

- If the object has no Object identifier

createUpdateDesignStructure

Description:

The service expects a request with a list of object, which is related through the row entry in the respective tables. The service reads the objects and builds the object tree with the relationship information in the table row entries. The service then takes the top object and starts creating the relations specified through table->row entries. The service only processes the Design objects and the STRUCTURE relationship. The service creates only the relationships and not the object itself i.e. the objects should already exist. The service reads the SDK object for the top object in request and then reads the STRUCTURE table with the table ID 2000008318 and iterates through the rows and creates a row entry in SDK object for each found row where the child object referenced exists and is of the type design.

During the row iteration if the request contains the child object referenced by the row, the service makes a createDesignStructure call for this found AgileObject and then creates/updates the row in the SDK object's structure table. With this recursions the service iterates through all the child objects present in the request and executes the call createDesignStructure. The service assumes that the request will contain only one top level object.

After successful completion of creating the relations, the method retrieves the attributes for the objects present in the request and sends it back.

The objects are referenced / related with one another with the table->row entry and its attributes. This is just to avoid the nesting child object that increases the payload by redundantly including the same object in many relations.

Returns an array of AgileBusinessObjectType of type Design filled in the response object.

Request Type: createUpdateDesignStructure

The request is of type AgileEObjectRequestType (see chapter "EC Schema Types").

Response Type: createUpdateDesignStructureResponse

The response is of type AgileEObjectResponse Type (see chapter "EC Schema Types").

Exceptions:

- If the object is not found for the object in request
- If the object found is not of type Design

createBOM

Description:

The service expects a request with a list of object, which is related through the row entry in the respective tables. The service reads the objects and builds the object tree with the relationship information in the table row entries. The service then takef the top object and starts creating the

relations specified through table->row entries. The service only processes the Part objects and the BOM relationship. The service creates only the relationships and not the object itself i.e. the objects should already exist. The service reads the SDK object for the top object in request, then reads the BOM table with the table ID 803, iterates through the rows and creates a row entry in SDK object for each found row where the child object referenced exists and is of the type part.

During the row iteration if the request contains the child object referenced by the row, the service executes createBOM call for this found AgileObject and then creates/updates the row in the SDK object's structure table. With this recursion the service iterates through all the child objects present in the request and executes createBOM call. The service assumes that the request will contain only one top level object.

After successful completion of creating the relations, the method retrieves the attributes for the objects present in the request and sends it back.

The objects are referenced / related with one another with the table->row entry and its attributes. This is just to avoid the nesting child object that increases the payload by redundantly including the same object in many relations.

Returns an array of AgileBusinessObjectType of type Part filled in the response object.

Request Type: createBOM

The request is of type AgileEObjectRequestType (see chapter "EC Schema Types").

Response Type: createBOMResponse

The response is of type AgileEObjectResponseType (see chapter "EC Schema Types").

Exceptions:

- If the object is not found for the given ID
- If the object found is not of type Part

getDesignWhereUsed

Description:

The service expects a request to contain a single object with Object Identifier. The service implementation reads the object Id and subclass from object identifier, with the enhancement to the service it will also read version from object attributes and property INCLUDE-ALL-VERSION. After reading these values successfully the service searches for this object and checks if it is of the type Design. Then the service retrieves the objects referenced by rows in the tables WHERE USED DESIGN and WHERE USED from specified version/Latest Version/ALL Version as per the values set for Version and INCLUDE-ALL-VERSION and creates an AgileObject for each found objects for a row. The object will only be created if the object referenced by row in TABLE_WHEREUSEDDESIGN is of type Design and in WHERE USED is of type Part. The AgileObjects are then filled with the attributes from the referenced SDK object. Once the objects are filled, the array of objects is set in Response and sent back to the client through SOAP response.

If the property value for INCLUDE-ALL-VERSION is true, the service will ignore the value of version

set in object attributes and retrieve whereused for all the version of the target Design object.

Returns an array of AgileBusinessObjectType of type Design or Part filled in the response object.

Request Type: getDesignWhereUsed

The request is of type AgileEObjectRequestType (see chapter "EC Schema Types").

Response Type: getDesignWhereUsedResponse

The response is of type AgileEObjectResponseType (see chapter "EC Schema Types").

Exceptions:

- If the object is not found for the given ID
- If the object found is not of type Design
- If the object has no object identifier

EC Orchestrated Calls

This chapter includes the following:

▪ getDesignPropertiesAndRights.....	10
▪ getDesignItemPropertiesAndRights.....	11
▪ createUpdateThenReserve.....	13
▪ reserveThenCreateUpdate	14
▪ createUpdateThenRelate.....	15
▪ retrieveDesignStructureAndMetadata.....	16

General

Requests

The requests are all defined in ECServicesSchema.xsd. Some of them are empty whereas others are instances of AgileECObjectRequestType.

Responses

The accepted approach of structuring the responses is that the response contains a composite structure of all responses that arrive from the orchestrated call.

- Pros
 - Each response is separately available for analysis.
 - The result of each orchestrated call is clearly visible in the sub-responses.
 - User can easily query for the interesting portion of the sub-response(s).
 - Avoid complicated and time consuming transformations of AgileECResponseType.
 - Saves creation of new complex types in EC Schema for response types that already are there in the core schema. So if the user comfortably understands the core web services as well as the EC services, it will be easier to handle the results of orchestrated calls.
 - Future changes of core services response types are automatically reflected to the end user by the process, since we deal with responses on an as is basis.
- Cons
 - Overall success must be inferred by a union of all responses statuses.

Alternative Approach

Use instances of AgileECObjectResponseType consistently. Append all exception, warnings, notes one after the other in the response header. If some response contains Agile Object Data, put that back in the AgileObject's table part of final response to make the response contain more meaningful data.

- Pros
 - User consistently deals with one response type.
- Cons
 - Hammering the various response structures of the core service (attachment, table, metadata and business) would be time consuming and may not make sense with the different type of responses. The AgileEObjectResponseType would need to become more comprehensive to handle everything.
 - Maintaining all the transformations from various responses into AgileEObjectResponseType would be too cumbersome. A change in one response type would entail changing various xsl sheets to reflect that.
 - Some data in response types may be lost during the transformation to an AgileEObjectResponseType object. For example, checkOutFF returns the status of success in the header for a FF object that is already checked out. But if you look at the actual response structure “checkOutFFResponse”, there is an additional element “success” which would have a boolean value of false. Thus the intent of the operation is fulfilled, i.e. a checkout but the second part says that this was because the folder was already checked out.

Errors

Orchestrated calls return an error which is passed to them by the core web services in the respective response elements of the calls being orchestrated. Any errors or exceptions that the process itself recognizes will be present under the AbstractResponseType under the root response.

getDesignPropertiesAndRights

Description:

This orchestrated call works on the properties present under the first “generalParameters” element block. Any other generalParameters element is ignored.

Objects: No objects are required to be present in the request.

The following are the orchestrated services description done by this process.

- **getTableMetadata (AdminMetadata Service):** This call is based on the property “classIdentifier” provided in the generalParameters element. This is compulsory. If it is not present then this call will be skipped. It will assume that this is the id of either a Design Class or a subclass of the Design class. Based on this assumption it will then query for the metadata for the tables listed below. This assumption is necessary as table-ids vary depending on what type of DataObject we are talking about. If a non Design Class value is used here then the getTableMetadataResponse can still contain metadata but only for applicable table ID’s to the non Design classIdentifier provided.

The following table metadata is fetched for the Design class or Design type subclasses. These values are hard coded in an XSL sheet. Currently you cannot choose to exclude tables:

- TitleBlock (table id = 6146)
- Page Two (table id = 810)

- Page Three (table id = 1501)
 - Where Used (table id = 6149)
 - Files (table id = 6150)
 - Relationships (table id = 2000007761)
 - History (table id = 815)
 - Structure (table id = 2000008318)
 - Where Used Design (table id = 2000008495)
- **checkPrivilege (Business Service):** The second part of the web service queries the user for privileges. All required elements of this core service call must be provided as property value pairs in the “generalParameters” element. Refer to the business service schema for values these properties can take. Possible properties are:
- userIdentification (single instance expected, otherwise first found node value is used)
 - classIdentifier (single instance expected, otherwise first found node value is used)
 - objectName (single instance expected, otherwise first found node value is used)
 - privilege (multiple instances can be present for querying multiple privileges, values must be valid integer values)

Request Type: getDesignPropertiesAndRights

The request is of type AgileECObjectRequestType (see EC Schema Types).

Properties:

- userIdentification: User ID
- classIdentifier: 2000008310 (Class ID)
- objectNumber: Number of object
- privilege: 2, 15, 17, 38

Response Type: getDesignPropertiesAndRightsResponse

The response is of type getDesignPropertiesAndRightsResponse (see EC Schema Types).

getDesignItemPropertiesAndRights

Description:

This orchestrated call works on the properties present on the first two “generalParameters” element block. Any other generalParameters element is ignored.

Objects: No objects are required to be present in the request.

The following services are orchestrated by this process:

- **getTableMetadata (AdminMetadata Service):** The call uses the “classIdentifier” property along with the “type” property under the same “generalParameters” element. Any “classIdentifier” that is of Item class type must specify the value of the “type” property to “item” and any “classIdentifier” that is of Design class type must specify the value of the “type” property to “design”. Both these options are compulsory as table-ids vary depending on what type of class we are talking about. If anyone of these are not present in one of the first two “generalParameters” element then the request will skip that particular “generalParameters” element in the getTableMetadata call. The “generalParameters” element is not limited to one being of type “item” and other to design. We can mix up the parameters and also make them same.

If a non Design Class value or a non Item Class is used here then the getTableMetadataResponse can still contain metadata but only for applicable table ID’s for the given classIdentifier.

The following table metadata is fetched for the Design class or Design type subclasses. This values are hard coded in an xsl sheet. You cannot currently choose to exclude tables:

- TitleBlock (table id = 6146)
- Page Two (table id = 810)
- Page Three (table id = 1501)
- Where Used (table id = 6149)
- Files (table id = 6150)
- Relationships (table id = 2000007761)
- History (table id = 815)
- Structure (table id = 2000008318)
- Where Used Design (table id = 2000008495)

The following table metadata is fetched for the Item class or Item type subclasses:

- TitleBlock (table id = 801)
 - Page Two (table id = 810)
 - Page Three (table id = 1501)
 - Where Used (table id = 805)
 - Attachments (table id = 807)
 - Relationships (table id = 2000007761)
 - History (table id = 814)
 - BOM (table id = 803)
 - Pending Changes (table id = 804)
 - Change History (table id = 802)
 - Redline BOM (table id = -803)
- **checkPrivilege (Business Service):** The second part of the web service queries the user for privileges. All required elements of this core service call must be provided as property value pairs in the “generalParameters” element. Refer to the business service schema for the values these properties can take. Possible properties are:

- userIdentification (single instance expected, otherwise first found node value is used)
- classIdentifier (single instance expected, otherwise first found node value is used)
- objectName (single instance expected, otherwise first found node value is used)
- privilege (multiple instances can be present for querying multiple privileges, values must be valid integer values)

Request Type: getDesignItemPropertiesAndRights

The request is of type AgileEObjectRequestType (see EC Schema Types).

Properties:

- First set:
 - userIdentification: User ID
 - classIdentifier: 2000008310 (Class ID)
 - objectNumber: Number of object
 - privilege: 2, 15, 17, 38
 - type: design
- Second set:
 - userIdentification: User ID
 - classIdentifier: 10141 (Class ID)
 - objectNumber: Number of object
 - privilege: 2, 15, 17, 38
 - type: item

Response Type: getDesignItemPropertiesAndRightsResponse

The response is of type getDesignItemPropertiesAndRightsResponse (see EC Schema Types).

createUpdateThenReserve

Description:

This orchestrated call will try to do the following in the order below.

- **createObject (Business Service):** This call is capable of creating an object and if the “updateExistingObject” property is provided than the object is updated if it already existed. It is an error in createObject call to call a create on an existing object. But if the above mentioned flag is present on an AgileObject, then it is not an error. The createObject call will then update the existing object and return successfully. The option(s) on each AgileObject is individually translated to option(s) on the subrequests in createObject call corresponding to each AgileObject.

Additionally the createObject call can be invoked in 2 ways. The number for the to-be created object should be present in the “objectIdentifier/objectName” element’s text value. The same number is required to be present with the actual attribute id of the number field if the object already exists and needs to be updated. Otherwise the createObject call fails to find the object

for updating it.

If no value is found at the “objectIdentifier/objectName” element’s text value then an “autoNumberSrc” property/value element pair should be used under the options element for that particular AgileObject element. A record of created objects can be seen in the element “createObjectResponse” of the response (see response section below).

- **checkoutFF (Attachment Service):** As a last step all Design/FF objects are used to make a last call to the core attachment service “checkoutFF”. The checkoutFF request is formulated from the createObject response and all successfully created Design/FF Objects are checkout candidates.

Each AgileObject can specify a “Number” attribute to be created together with its value. Alternatively, if an autonumber has to be generate, then an option must be set on each “AgileObject” specifying an autonumber source.

An additional option should be present for File Folders or Design objects. This specifies the version of the Design object to be checked out.

Request Type: createUpdateThenReserve

The request is of type AgileECObjectRequestType (see EC Schema Types).

Properties:

- auto-number-src: Source for auto number
- number-id: Base ID of the number field
- checkout-version: Version to be checked out (default is LATEST)

Response Type: createUpdateThenReserveResponse

The response is of type createUpdateThenReserveResponse (see EC Schema Types).

reserveThenCreateUpdate

Description:

This orchestrated call does the following in the order below:

1. **checkoutFF (Attachment Service):** All Design/FF objects are used to make a call to the core attachment service “checkoutFF”. To exclude a certain Design/FF object from this call the checkoutFlag property value can be set to false and used to do so. The absence of this flag defaults this property value to true.
2. **createObject (Business Service):** This call is capable of creating an object and if the “updateExistingObject” property is provided, then the object is updated if it already existed. It is an error in the createObject call to call a create on an existing object. But if the above mentioned flag is present on an AgileObject, then it is not an error. The createObject call will then update the existing object and return successfully. The option(s) on each AgileObject is individually translated to option(s) on the subrequests in the createObject call corresponding to each AgileObject.

Additionally the `createObject` call can be invoked in 2 ways. The number for the to-be created object should be present in the “objectIdentifier/objectName” element’s text value. The same number is required to be present with the actual attribute id of the number field if the object already exists and needs to be updated. Otherwise the `createObject` call fails to find the object for updating it.

If no value is found at the “objectIdentifier/objectName” element’s text value, then an “autoNumberSrc” property/value element pair should be used under the options element for that particular `AgileObject` element. A record of created objects can be seen in the element “createObjectResponse” of the response (see response section below).

Each `AgileObject` can specify a “Number” attribute to be created with a value. Alternatively if an autonumber should be generated, then an option must be set on each “`AgileObject`” specifying an autonumber source.

An additional option should be present for File Folders or design objects. This specifies the version of the Design object to be checked out.

Request Type: `reserveThenCreateUpdate`

The request is of type `AgileEObjectRequestType` (see EC Schema Types).

Properties:

- `auto-number-src`: Source for auto number
- `number-id`: Base Id of the number field
- `checkout-version`: Version to be checked out (default is LATEST)

Response Type: `reserveThenCreateUpdateResponse`

The response is of type `reserveThenCreateUpdateResponse` (see EC Schema Types).

createUpdateThenRelate

Description:

The input can have any number of objects that need to be updated or created as the first step. As the next step only Design objects will get picked up to create relation to items. This is as creating relation in one direction is enough to make it appear in the other one. So Design object is taken for convenience as EC cases will mostly deal with Design objects. For all Design objects that need to relate to items, there will be a table that corresponds to the “Relationships” table having an id of “2000007761” in the request. Then the rows in that table will contain the data necessary for the core service to successfully complete the call.

This orchestrated call does the following in the order below:

1. **createObject (Business Service):** This call is capable of creating an object and if the “updateExistingObject” property is provided then the object is updated if it already existed. It is an error in `createObject` call to call a create on an existing object. But if the above mentioned flag is present on an `AgileObject` then it is not an error. The `createObject` call will then update the existing object and return successfully. The option(s) on each `AgileObject` is individually translated to option(s) on the subrequests in `createObject` call corresponding to each

AgileObject.

Additionally the createObject call can be invoked in 2 ways. The number for the to-be created object should be present in the “objectIdentifier/objectName” element’s text value. The same number is required to be present with the actual attribute id of the number field if the object already exists and needs to be updated. Otherwise the createObject call fails to find the object for updating it.

If no value is found at the “objectIdentifier/objectName” element’s text value then an “autoNumberSrc” property/value element pair should be used under the options element for that particular AgileObject element. A record of created objects can be seen in the element “createObjectResponse” of the response (see response section below).

2. **addRows (Table Service):** This call then relates designs to items or vice versa as the request specifies. To relate objects the number of the object is really important to know. The relationship table element can refer to objects that were created in the same call to process if these objects were created with known numbers. The process cannot relate objects if the objects to be related were created with the “autoNumberSrc” option. Then the request could not possibly know the numbers of such objects before hand and thus this scenario is not possible. The client can itself draw an autoNumber itself and populate the request with it instead of an autoNumberSrc for this scenario to always work. A separate request is needed for each AgileObject as per core service restriction of core web service. Thus the response can contain multiple “addRowsResponse” elements.

```
<table>
  <tableIdentifier>
    <tableId>2000007761</tableId>
    <tableName>Relationships</tableName>
  </tableIdentifier>
  <row rowId="0">
    <key2000007767 id="2000007767">P00001</key2000007767>
    <key id="2000007762">1</key>
    <key id="2000008523">1</key>
  </row>
</table>
```

Request Type: createUpdateThenRelate

The request is of type AgileECObjectRequestType (see EC Schema Types).

Properties:

- auto-number-src: Source for auto number
- number-id: Base ID of the number field

Response Type: createUpdateThenRelateResponse

The response is of type createUpdateThenRelate (see EC Schema Types).

retrieveDesignStructureAndMetadata

Description:

This orchestrated call does the following in the order below. It works similarly to the

“getDesignPropertiesAndRights” call on the properties present under the first “generalParameters” element block for the “getTableMetadata” and “checkPrivilege” call. Any other generalParameters element is ignored.

1. **retrieveDesignStructure (EC Service):** The whole input provided to the main call is passed to this EC service call. Behavior and input is as defined in this document for this call under EC Object calls.
2. **getTableMetaData (AdminMetadata Service):** This call is based on the property “classIdentifier” provided in the generalParameters element. This is compulsory. If it is not present, the call is skipped. It assumes that this is the id of either a Design Class or a subclass of the Design class. Based on this assumption it queries for the metadata for the tables listed below. This assumption is necessary as the table-ids vary depending on the type of DataObject. If a non Design Class value is used here, then the getTableMetadataResponse can still contain metadata but only for applicable table ID’s to the non Design classIdentifier provided.

The following table metadata is fetched for the Design class or Design type subclasses. These values are hard coded in an xsl sheet. Currently you cannot choose to exclude tables:

- TitleBlock (table id = 6146)
 - Page Two (table id = 810)
 - Page Three (table id = 1501)
 - Where Used (table id = 6149)
 - Files (table id = 6150)
 - Relationships (table id = 2000007761)
 - History (table id = 815)
 - Structure (table id = 2000008318)
 - Where Used Design (table id = 2000008495)
- **checkPrivilege (Business Service):** The second part of the web service would queries the user for privileges. All required elements of this core service call must be provided as property value pairs in the “generalParameters” element. Refer to the business service schema to see what values these properties can take. Possible properties are:
- userIdentification (single instance expected, otherwise first found node value is used)
 - classIdentifier (single instance expected, otherwise first found node value is used)
 - objectName (single instance expected, otherwise first found node value is used)
 - privilege (multiple instances can be present for querying multiple privileges, values must be valid integer values)

Request Type: retrieveDesignStructureAndMetadata

The request is of type AgileECObjectRequestType (see EC Schema Types).

Properties:

- userIdentification: User ID
- classIdentifier: 2000008310 (Class ID)
- objectNumber: Number of object
- privilege: 2, 15, 17, 38

Response Type: retrieveDesignStructureAndMetadataResponse

The response is of type retrieveDesignStructureAndMetadataResponse (see EC Schema Types).

EC Schema Types

AgileEObjectRequestType

Description: It defines the structure of request header for EC services.

<i>Contains</i>	▫	
	▫ req	Request
	▫	Refer to AgileECRequestExtType

AgileECRequestExtType

Description: It defines the structure of request header for EC services.

<i>Extends</i>	▫	
	▫ AbstractRequestType	Base Request Type
	▫	Refer to common:Abstract RequestExtType in core services documentation.
	▫	
<i>Contains</i>	▫ AgileObject	List of objects
	▫	Refer to bobjs:AgileBusinessObjectType in core services documentation.

AgileEObjectResponseType

Description: It defines the structure of response header for EC services.

<i>Contains</i>	▫	
	▫ res	Response
	▫	Refer to AgileECResponseExtType

AgileECResponseExtType

Description: It defines the structure of response header for EC services.

<i>Extends</i>	▫	
	▫ AbstractResponseType	Base Response Type
	▫	Refer to common:AbstractResponseType in core services documentation
	▫	
<i>Contains</i>	▫ AgileClass	List of classes

- Refer to AgileECClassType
-
- AgileObject List of objects
- Refer to bobjs:AgileBusinessObjectType in core services documentation

AgileECClassType

Description: It defines the classes for EC services.

Extends

-
- ClassType Class Type
- Refer to common:ClassType in core service documentation.
-

Contains

- Properties Property list
- Refer to common:PropertyListType in core services documentation.
-
- autoNumber Auto number source
- Refer to common:AdminNodeType in core services documentation.
-
- attributeMetaData Meta Data for Attribute
- Refer to common:AttributeType in core service documentation.
-
- tableMetaData Meta Data for Table
- Refer to common:TableType in core service documentation

getDesignPropertiesAndRightsResponse

Description: Response type for getDesignPropertiesAndRightsResponse call.

Element

-
- res Response element
-

Extends

-
- AbstractResponseType Base response type
- Refer to common:AbstractResponseType in core services documentation.

Contains	□	
	□	
	□	getTableMetadataResponse Response for getTableMetaData call
	□	Refer to adminMetadata:getTableMetadataResponse in core services documentation.
	□	
	□	hasPrivilegeResponse Response for hasPrivilege call
	□	Refer to bobjs:hasPrivilegeResponse in core services documentation.

getDesignItemPropertiesAndRightsResponse

Description: Response type for getDesignItemPropertiesAndRightsResponse call.

<i>Element</i>	□	
	□	res Response element
	□	
Extends	□	
	□	AbstractResponseType Base response type
	□	Refer to common:AbstractResponseType in core services documentation.
Contains	□	
	□	getTableMetadataResponse Response for getTableMetaData call
	□	Refer to adminMetadata:getTableMetadataResponse in core services documentation.
	□	
	□	hasPrivilegeResponse Response for hasPrivilege call
	□	Refer to bobjs:hasPrivilegeResponse in core services documentation.

createUpdateThenReserveResponse

Description: Response type for createUpdateThenReserveResponse call.

<i>Element</i>	□	
	□	
	□	res Response element

Extends	▫		
	▫		
	▫	AbstractResponseType	Base response type
	▫		Refer to common:AbstractResponseType in core services documentation.
Contains	▫		
	▫	createObjectResponse	Response for createObject call
	▫		Refer to bobjs:createObjectResponse in core services documentation.
	▫		
	▫	checkOutFFResponse	Response for checkOutFF call
	▫		Refer to attach:checkOutFFResponse in core services documentation.

reserveThenCreateUpdateResponse

Description:

Response type for reserveThenCreateUpdateResponse call

<i>Element</i>	▫		
	▫	res	Response element
	▫		
Extends	▫		
	▫	AbstractResponseType	Base response type
	▫		Refer to common:AbstractResponseType in core services documentation.
Contains	▫		
	▫	CheckOutFFResponse	Response for checkOutFF call
	▫		Refer to attach:checkOutFFResponse in core services documentation.
	▫		
	▫	createObjectResponse	Response for createObject call
	▫		Refer to bobjs:createObjectResponse in core services documentation.

createUpdateThenRelateResponse

Description:	Response type for createUpdateThenRelateResponse call		
	□		
<i>Element</i>	□		
	□	res	Response element
	□		
<i>Extends</i>	□		
	□	AbstractResponseType	Base response type
	□		Refer to common:AbstractResponseType in core services documentation.
	□		
<i>Contains</i>	□		
	□	CreateObjectResponse	Response for createObject call
	□		Refer to bobjs:createObjectResponse in core services documentation.
	□		
	□	addRowsResponse	Response for addRows call
	□		Refer to table:addRowsResponse in core services documentation.

retrieveDesignStructureAndMetadataResponse

Description:	Response type for retrieveDesignStructureAndMetadataResponse call		
	□		
<i>Element</i>	□		
	□	res	Response element
	□		
<i>Extends</i>	□		
	□	AbstractResponseType	Base response type
	□		Refer to common:AbstractResponseType in core services documentation.
	□		
<i>Contains</i>	□		
	□	getTableMetadataResponse	Response for getTableMetada call

▫		Refer to adminMetadata:getTableMetadataResponse in core services documentation.
▫		
▫	retrieveDesignStructureResponse	Response for retrieveDesignStructure call
▫		Refer to retrieveDesignStructure
▫		
▫	hasPrivilegeResponse	Response for hasPrivilege call
▫		Refer to bobjs:hasPrivilegeResponse in core services documentation.