BOND

User Guide:

Adapt V11 Web Services

V1 Interfaces

Adapt V11 Web Services

V1 Interfaces

Date 03-Jul-2012

Version Version 4.9

Document History

Version	Date	Author	Summary
0.1	4-Nov-2008	Dmitriy Rybalsky	Document draft created
1.0	10-Nov-2008	Dmitriy Rybalsky	Document Baselined
1.1	11-Nov-2008	Dmitriy Rybalsky	• Code highlighted with blue Courier New • DocumentV1 data object described • BookingV1 data object described
1.2	17-Nov-2008	Konstantin Vodyaho	• Filter object described • Time zones and locale described • Steps to create Web User added • Sysadmin UI changes described • Passing and getting XML message to/from BO explained • Usage of SearchParameter object explained • logonDefault function described • ColumnFormat description added • Entity XML format description added • ControlValue object description added. • Exceptions and ownership described
2.0	18-Nov-2008	Dmitriy Rybalsky	Document baselined
2.1	1-Apr-2009	Dmitry Shiyan	Fixed method description. "*Type of the task's owner, can be U - user or E – entity*". Replaced with "*Type of the task's owner, can be 0 - user or 1 – entity*"
3.0	1-Apr-2009	Konstantin Vodyaho	Document baselined.
3.1	5-Oct-2009	Dmitry Shiyan	A new method is introduced to CodeGroupService
3.2	12-Oct-2009	Dmitry Shiyan	Updated XML format description for Entity and Search Result
3.3	14-Oct-2009	Sergey Zaika	bookingCodeId was added for BookingV1
3.4	09-Nov-2009	Dmitry Shiyan	Descriptions for CodeBean, CodeGroupBean and LocalName beans were added
3.5	18-Nov-2009	Tejas Rajpura	A new method getAvailableLanguageNames() is introduced to LogonService
3.6	13-Jan-2010	Mikhail Boldinov	getDocumentsInfoByOwnerAndCategory() is introduced for Document Access Service
3.7	5-Feb-2010	Sriram	A new feature added for runQuery()
3.8	4-Feb-2010	Sergey Kalashnik	Minor stylistic changes
4.0	5-Feb-2010	Sergey Kalashnik	Document baselined.
4.1	29-Apr-2010	Mikhail Boldinov	Search Service, ColumnFormat description was updated. New possible values for parameter 'sort' were added.
4.2	10-Aug-2010	Ashok Varma C	getCodeGroups(),updateCode(), updateCodeGroup() are introduced to CodeGroupService
4.3	23-Sep-2010	Ashok Varma C	Modified getCodeGroups(),updateCode(), updateCodeGroup() services in CodeGroupService
4.4	8-Oct-2010	Ramesh Kayithala	The following methods for DataAdmin manage request notification module added: 1. createMergeRequest() 2. createDeleterequest()

4.5	23-Mar-2011	Dmitriy Kruglov	• New information has been added to the section "3.3.2 runQuery" to cover changes, which had been introduced by TTP12625; • New "3.4.4 getEntityData" and "3.4.6 saveEntityData" sections have been added to cover changes, which had been introduced by TTPs 14684 and 14688 accordingly.
4.6	03-Aug-2011	Alexander Kiselyov	• "2. Getting started" section has been supplemented with information regarding TTP#18295 (multiple WS).
4.7	07-Nov-2011	Elena Sviridenko	The following sections updated based on ttp19993: getSearchResult, getSearchResultXML
4.8	17-Jan-2012	Elena Sviridenko	Section 3.2 – Business Objects Execution Interface - Added info according to the ttp18148. Added subsection "Entity Ownership" to the section "Entity Interface" based on the ttp 20750.
4.9	04-Apr-2012	Konstantin Vodyaho	• getFilteredBookings method introduced to the Calendar Service (ttp22814/22529) • getFavouriteEntities method introduced to the Entity Service(ttp22815/22530) • getSearchResultWithRoles method introduced to the Search Service(ttp22813/22528) • IMetaDataServiceV1 service description added(ttp22811/22526) • quickFind method introduced to the Search Service (ttp22812/22527)
4.91	04-July	Konstantin Vodyaho	• getDocumentsInfoByOwner method introduced to the Document service (ttp23766/23782/23783)
4.92	05-July	Konstantin Vodyaho	• getSearchResultsXML updated to allow to retrieve U_PERSONAL data.(ttp23784) • getFavouriteEntities method description updated. (ttp22815/22530)

Table of Contents

Glossary and Abbreviations

Term or Abbreviation	Definition
JWSDP	Java Web Service Development Pack
Axis	Apache Jakarta Axis – SOAP java library
BO	Adapt V11 Business Object
Entity	A “record” within the system. Most of system objects (like Candidate, Job and Client) are stored as Entities.
Role	Defines the Entity properties and how the system will treat this Entity.
Properties	Collective notion inclusive coherent information in form of Attributes
Attribute	An individual piece of data and the least supported by system
Single Property	Only one value per *Entity*
Named Property	Describe fixed number of occurrences that *Entity* can have and they should be differed by name.
Multiple Property	Can have not limited number of occurrences, differed by unique ID

1. Overview

The Bond **AdaptService** Web Service is an XML-based Web Service that allows developers to integrate most of Adapt V11 functionality into their applications and business processes. The AdaptService Web Service provides an extensive set of Adapt V11 system data and functionality, such as executing Business Objects, searching and accessing Entities, Documents, Tasks, Journal entries, Calendar Appointments, Code Groups, and other types of data that can be used within applications.

For example you can:

- Use the IBOExecutorV1 interface of AdaptService to run Business Object. You may call executeBO to run BO, pass XML to it and retrieve XML data back.
- Call the ISearchV1 interface method runQuery to run query, previously created in Query Explorer tool. Result will contain array of found entities and necessary supplementary information.
- Create, update or delete entities using IEntityV1 interface methods saveEntity, deleteEntity

User authentication is provided through IAuthenticationV1 interface. User calls logon function returning session id. This session ID then used to call all other methods of any interface. User name can be omitted at logon. In this case user will be treated as default anonymous WebServiceUser and will be logged under WebService domain profile.

AdaptService is SOAP XML Web Service and can be accessed using various platforms like Java (Axis or JWSDP), .NET, Visual Basic, PHP and other.

2. Getting Started

This section contains an overview of the AdaptService, as well as requirements and other information needed to get started using the AdaptService Web Service.

There are nine endpoint interfaces in AdaptService Web Service: Authentication, BOExecutor, Search, Entity, Document, Code Group, Task, Journal and Calendar. This topic provides an overview of each of these services.

Java Client can be built on **Apache Axis** or on Sun **JWSDP** (Java Web Services Development Kit). Both frameworks have simple means to generate stub classes and data structures by given URL. Client only needs to know URL to a working service. JWSDP contains wscompile tool, and Axis has wsdl2java tool, used to build classes from WSDL file or URL.

Microsoft Visual Studio provides a very simple way to create all required client classes for .NET. Just add web reference (URL of WSDL file) to the required service to project. Visual Studio automatically creates all stubs and structures; you can use them in application.

To retrieve WSDL descriptor of AdaptService web service, you can use link to any endpoint mapping URL, adding request parameter "`wsdl`": http://<host:port>/<WebApp>/<ServiceMapping>?wsdl

List of possible mappings: LogonServiceV1, BOExecServiceV1, SearchServiceV1, EntityServiceV1, DocServiceV1, CodeServiceV1, TaskServiceV1, ManageRequestServiceV1, JournalServiceV1, CalServiceV1.

Also note that there exists a possibility (in several branches) to generate and deploy multiple endpoints instead of one AdaptService. In this case when the end user visits concrete service's URL for WSDL – resulting data will be the description (WSDL) of corresponding service with only one endpoint. For instance, visiting http://<host:port>/<WebApp>/BOExecServiceV1?wsdl will result in description of service with only BOExecutor interface.

For more information regarding this please refer to the **Engine Documentation\Web Services\WebServicesV1_MultipleSingleEndpoint_Configuration_UserGuide**

2.1. Authentication Interface - ILogonV1

You can use Authentication Interface to initiate and close web service session.

The Authentication Interface methods are:

- **logon** - starts new web services session. Accepts username, password, domain profile, domain, [locale, timezone, dateformat, timeformat] and returns long session id, required to call any other method.
- **logonDefault** - starts new web services session. Accepts domain and returns long session id, required to call any other method.
- **logoff** - closes existing session by id.
- **getAvailableTimezones** - returns array of strings with all timezones recognizable by server.
- **getAvailableLocales** - returns array of strings with all locales recognizable by server.
- **getVersion** - returns VersionInfo object with web service version information.
- **getAvailableLanguageNames** - returns array of LanguageBean objects which contains all active language information.

2.2. BOExecutor Interface – IBOExecutorV1

You can use BOExecutor Interface to run Adapt V11 Business Objects.

The BOExecutor Interface methods are:

- **executeBO** - runs BO with given string data, preferably XML and returns result as string, preferably XML. Accepts session id, BO name, and xml string. Returns XML generated by BO.

2.3. Search Interface – ISearchV1

You can use Search Interface to run queries, previously created in Query Explorer tool. You may also access those queries and saved search results of queries.

- **getAllQueriesNames** – retrieves list of all queries available to logged user. Returns array of query names.
- **runQuery** - executes a query and returns the Search Result object (SearchResultV1) containing array of found entities and all supplementary information. Accepts array of search parameters (SearchParameter) and can save or not save search results to Adapt V11 database depending on saveResults boolean parameter. This method accepts firstItemIndex and itemCount parameters to limit size of returning data, to prevent slowdown when retrieving results with very big number of found entities.
- **getSearchResults** – Returns array of IDs of all stored query result sets that match the filter criteria. Filter is specified via array of search parameters (SearchParameter) and can be “Label”, "Created By", "Created Start Date", "Created End Date", "Modified By", "Modified Start Date", "Modified End Date", "Method" and "Highlight" .
- **getSearchResult** – Returns stored query result in form of Search Result object (SearchResultV1). This method also accepts firstItemIndex and itemCount parameters to limit size of returning data.
- **getSearchResultXML** – Returns stored query result in form of XML string. Accepts array of ColumnFormat objects in order to provide convenient formatting of columns in returning data set.
- **getSearchResultEntityCount** – Returns number of records in specified stored query result
- **deleteSearchResult** – Permanently deletes search result by specified id.
- **getSearchResultWithRoles** – This method returns stored query result in form of Search Result object (SearchResultWithRoles). This method is similar to the getSearchResult method except returning value (SearchResultWithRoles instead of SearchResultV1)Entity Interface – IEntityV1
- **quickFind** - This method performs search and returns its result in form of XML string

2.4. Entity Interface – IEntityV1

You can use Entity Interface to retrieve, create, update and delete Entities. It is also possible to retrieve User as Entity.

- **getUser** – returns user data in form of XML, based on “get user as entity” functionality of Adapt V11 core.
- User can be retrieved by search parameter see 3.3.3.1
- **getEntity** – returns Entity XML by entity ID. Formatting rules like time zone, locale, and date and time format is specified at logon.
- **saveEntity** – saves (creates new or updates existing) using given XML data.
- **deleteEntity** – deletes existing entity. Entity can be deleted permanently if “permanently” flag is true.
- **getFavouriteEntities** – retrieves all favourite entities of current user and returns them in XML representation.

2.5. Document Interface – IDocumentV1

You can use Document Interface to retrieve, create, update and delete Document in Adapt V11 database. It is also possible to get IDs of all documents of specified entity.

- **getDocument** – retrieves document from the database.
- **saveDocument** – saves document. If document ID is 0 the new document will be created.
- **deleteDocument** – deletes document. Document can be deleted permanently if "permanently" flag is true.
- **getEntityDocuments** – returns specified entity documents ID. Language and document library category can be specified.

2.6. Calendar – ICalendarV1

Calendar Interface provides access to appointments in Adapt V11 Calendar.

- **getBookingById** – returns booking data by given booking Id.
- **getBookings** – returns all bookings of their creator (entity or user)
- **saveBooking** – saves booking. If bookingID is 0, the new booking will be created.
- **deleteBooking** – deletes booking
- **getFilteredBookings** – returns list of bookings filtered by start and end dates for the specified owner (entity or user).

2.7. Task – ITaskV1

Task Interface provides access to Adapt V11 Tasks and is very similar to Calendar.

- **getTaskById** – returns task by its Id.
- **getTasks** – returns array of task objects filtered and sorted by given criteria
- **saveTask** – saves task
- **deleteTask** – deletes task
- **deleteTasks** – deletes tasks based on criteria.

2.8. Journal – IJournalV1

You can use Journal Interface to retrieve Adapt V11 Journal entries.

- **getJournalEntry** – returns journal entry by ID
- **getJournalEntries** – returns array of journal entries found by given filter

2.9. Code Group – ICodeGroupV1

You can use Code Group Interface to retrieve codes and code groups.

- **getCodeGroup** - Retrieves CodeGroup identified by domain name and display name of CodeGroup with localized descriptions in all active languages in the system
- **getCodeGroupByLanguage** - Retrieves CodeGroup identified by domain name and display name of CodeGroup with localized representation in one specified language
- **getCodeByID** - Retrieves localized representation Code by ID and language name
- **getCodeIDFromPath** - Retrieves Code ID by Code Path.
- **getCodeGroups** – Retrieves all code groups identified by domain name
- **updateCode** – Updates the language specific code details like synonyms, dependents, groups along with code details identified by domain name and CodeBean object

- **updateCodeGroup** – Updates the code group details identified by domain name and CodeGroupBean object

2.10. Meta Data - MetaDataServiceV1

MetaData service interface can be used to retrieve meta data information from the system.
The MetaData service interface methods are:

getRoles - Retrieves information about all Roles in the specified domain.

2.11. Web Service Tools (WSTool)

WSTool is java GUI application used to access AdaptService for testing and debugging purposes. It can be found in AdaptV11/WebServices/wstool folder. Ant build script located in this directory can rebuild WSTool. Resulting WSTool.jar file will be placed to AdaptV11/WebServices/build folder. This jar can be invoked the same way as application, or by calling "java –jar WSTool.jar".

2.12. Configuration

Client Prerequisites

Adapt Web Services Access client bundle contains 3 directories with tests and demos for 3 types of clients: Axis, WSDP and Microsoft .Net. To use them you need following software:

- Apache Ant 1.7.0 (should be accessible in command prompt)
- JWSDP (not required for Axis and .Net clients). http://java.sun.com/webservices/jwsdp
- wscompile tool (from WSDP) should be accessible in command prompt
- Microsoft Visual Studio 2005 (required only for .Net client)
- Axis tool already added as a library to the test project, so it's not needed to manually setup it.

Building and running unit tests

AdaptV11/WebServices/client-axis folder contains Apache Axis client. There are ant build script, unit tests, auxiliary classes, configuration files, few simple demos and all required libraries. Build.xml file contains deployhost property with value="localhost". You should change value localhost to exact host name of your Adapt V11. To build everything and run tests, you should simply call ant target "test":

cd AdaptV11\WebServices\client-axis; enter into dir with build.xml
call ant cleanup-junit-dirs ; wipe out old JUnit reports
call ant test; generate stubs, compile them, run test and generate reports

After successful run of this script, you may check JUnit reports in "junit_reps\ AxisClient" directory. "junit_reps" directory is previous to client-axis. Unit Tests may be used as demo reference to create own client application on Axis.

2.13. User Access Levels

When one performs authentication using logon function, the domain profile should be specified. Access to each Web Service interface is restricted by domain profile settings. Sysadmin UI domain profile settings Web Services tab has 8 checkboxes, one per each web services interface.

You can omit username and domain profile at logon. In this case you will be treated as default anonymous web service user with default web user domain profile. If you have specified username and password, it will be checked at logon. Domain profile is also required when logging in as genuine (not anonymous) user.

2.14. Exceptions

There is a standard way of handling exceptions in SOAP technology, so all exceptions thrown on the server are wrapped into SOAPFault object (<fault> section of SOAP response) and returned back to client. Usually generated client stubs wrap this response into checked exception thrown at the call of service function on client. Alternative SOAPFaultException may be thrown, it will contain <fault> XML accessible inside.

The following types of exceptions are defined in Adapt V11 Web Services and can be thrown via standard SOAPFault. These four exceptions are exposed to web service client through WSDL:

- **InvalidArgumentException** - thrown if arguments passed to web service method are invalid (null objects, empty strings etc.);
- **DataNotFoundException** - thrown if no data can be found matching method arguments;
- **AccessDeniedException** – thrown if there is no access to a service. It can be not accessible if:
    - there is no permission in domain profile for requested service;
    - session ID is invalid;
    - incorrect login/password, etc.
- **ServerErrorException** – thrown if server encounters any error while executing service method not covered by (InvalidArgumentException, DataNotFoundException and AccessDeniedException).

2.15. Time Zones

AdaptService allows you to work in different Time Zones to perform correct operations with **Calendar** and **Task** services. You can provide your Time Zone to the service and expect it to work in it. However if the Time Zone provided is not supported by the server, **InvalidArgumentException** will be thrown. If the Time Zone wasn't provided, service will work in the server's default time zone.

Time Zones are provided by their names, e.g. GMT, GMT0, Greenwich, UTC, Universal, Zulu, Europe/Amsterdam, Europe/Berlin.

2.16. Locales

AdaptService works with different Locales. Locale provided by the user is checked against locales supported by the server (based on available languages in Adapt V11 Configuration Domain used by the service). If locale provided isn't valid, InvalidArgumentException will be thrown.

Locales are provided by their names, e.g. EN, EN_US, RU_RU, ES_ES

2.17. Web User creation

AdaptService can be used with default anonymous web service user and default web user domain profile. In this case __WebServiceUser_ is used.

In order to create __WebServiceUser_, following steps should be taken:

1. Log in Adapt V11 under sys admin user.
2. Expand User node and click on Users.
3. Click Create New User button.
4. Select Web User under User Type
5. Fill Login Name and User Name fields with __WebServiceUser_. Enter some password.
6. Fill out the rest of the fields if necessary and click Create User button
7. On the appeared page specify User Profile and Domain Access settings if needed and press Save User Settings.

8. Find newly created user and click on it.
9. In Domain Access section click on the domain.
10. Under Domain Profiles section click Add/Remove Profiles button and add WebUser profile.
11. Select radio button next to WebUser profile. This will make this profile the default one for user.
12. Click Save User Setting button.

2.18. Ownership

If Entity Web Service is accessed by anonymous user (WebUser), service ignores Entity Ownership. This means that Entity ownership exception is not thrown, when updating Entity owned by another user.

If Entity Access Web Service is accessed by genuine Adapt user (not anonymous WebUser), service throws **DataNotFoundException** when one is trying to update Entity owned by another user. It is not that the Entity is owned by another user, but this means that the server perform a check that the Entity ownership list for the specific Entity has got the User ID OR one of the user groups the user belongs to in the list

2.19. Sysadmin UI Changes

In order to have the ability to restrict access to web services on the domain profile level, the new tab was introduced under domain profile settings in sys admin. This tab contains eight checkboxes that enable or disable access to a particular service for the domain profile:

Web Services	
Service Endpoint	**Allowed?**
Business Objects Executor	☑
Search	☑
Entity	☑
Document	☑
Code Group	☑
Task	☑
Journal	☑
Calendar	☑
DataAdmin Manage Request	☐
Meta Data	☐

3. Using the System

This section includes topics to help you develop applications using the AdaptService Web Service.

3.1 Authentication Interface

Authentication interface is used to initiate and close web service session logon. It also provides supplementary functions used to retrieve versioning information and information about timezones and locales supported by the server. Authentication interface consist of six methods: logon, logonDefault, logoff, getVersion, getAvailableTimezones and getAvailableLocales.

3.1.1 Logon

This method starts new web services session.

```
long logon(String username,
           String password,
           String domain,
           String domainProfile,
           String locale,
           String timeZone,
           int dateformat,
           int timeformat)
throws
      ServerErrorException,
      InvalidArgumentException,
      DataNotFoundException,
      AccessDeniedException;
```

Method arguments:

Argument	Type	Description
Username	String	Adapt V11 User name
password	String	User password
domain	String	Adapt V11 Configuration Domain
domainProfile	String	Adapt V11 Domain Profile
Locale	String	Locale name, one of those retrieved by getAvailableLocales(). Examples: EN, EN_US, RU_RU, ES_ES_Traditional
timeZone	String	Time Zone name, one of returned by getAvailableTimezones (). Few examples: America/Argentina/Buenos_Aires, Canada/Newfoundland, US/Pacific, US/Arizona, US/Mountain, US/Central, US/Eastern, US/Michigan, Europe/Belfast, Europe/London, GB, GB-Eire, GMT, GMT0, Greenwich, UTC, Universal, Zulu, Europe/Amsterdam, Europe/Berlin, Europe/Brussels, Europe/Madrid, Europe/Paris, Europe/Rome, Europe/Vienna, Europe/Kiev, Poland
dateFormat	Integer	Date format – can be Default = 0, Long = 1, Medium = 2, Short = 3 The standard web services user uses the default dateformat (Which is the same as Short=3)
timeFormat	Integer	Time format – can be Full = 0, Long = 1, Medium = 2, Short = 3

Return value:

Type	Description
Long	Id of new session.

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.1.2 logonDefault

This method starts new web services session for default web service user, using default web user login, web user domain profile, default web user language, date/time formats and server's default time zone and locale.

```
long logonDefault(String domainName)
throws
      ServerErrorException,
      InvalidArgumentException,
      DataNotFoundException,
      AccessDeniedException;
```

Method arguments:

Argument	Type	Description
domainName	String	Adapt V11 Configuration Domain

Return value:

Type	Description
Long	New session Id.

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.1.3 Logoff

Closes existing session by id

```
void logoff(long sid)
throws  ServerErrorException,
        DataNotFoundException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session ID

Return value: N/A

Exceptions:

- DataNotFoundException
- ServerErrorException

3.1.4 getAvailableTimezones

Returns array of strings with all time zones server recognizes.

```
String[] getAvailableTimezones();
```

Method arguments: N/A

Return value:

Type	Description
String[]	All time zones server recognizes.

Exceptions: N/A

3.1.5 getAvailableLocales

Returns array of strings with all locales server recognizes.

```
String[] getAvailableLocales(String domainName)
throws ServerErrorException,
       DataNotFoundException,
       InvalidArgumentException;
```

Method arguments:

Argument	Type	Description
domainName	String	Adapt V11 Configuration Domain

Return value:

Type	Description
String[]	All locales recognizable by server.

Exceptions:

- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.1.6 getVersion

Returns VersionInfo object with web service version information

```
VersionInfo getVersion();
```

Method arguments: N/A

Return value:

Type	Description
VersionInfo	Holds service and endpoints version.

Exceptions: N/A

3.1.7 getAvailableLanguageNames

Returns array of LanguageBean which contains all active language information

```
LanguageBean[]getAvailableLanguageNames (String domainName)
throws ServerErrorException,
       DataNotFoundException,
       InvalidArgumentException;
```

Method arguments:

Argument	Type	Description
domainName	String	Adapt V11 Configuration Domain

Return value:

Type	Description
LanguageBean[]	All active languages information.

Exceptions:

- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.2 Business Objects Execution interface

One can run Adapt V11 Business Objects using IBOExecutorV1 interface. Interface consists of one method called executeBO.

If an Entity is created via executeBO method, it gets associated in the ENTITY_OWNERSHIP table not only with the user who executed a BO (via web services) but also with other users that share user groups with him.

3.2.1 executeBO

Runs BO with given string data, preferably XML and returns result as string, preferably XML. Method accepts session id, BO name, and xml string. Returns XML generated by BO

```
String executeBO (long sid,
                  String BOName,
                  String xml,
                  ControlValue[] values)
throws
      ServerErrorException
      InvalidArgumentException,
      DataNotFoundException,
      AccessDeniedException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
BOName	String	Name of Business Object
Xml	String	String value passed to Business Object
values	ControlValue[]	Array of control values that may be used by BO

Return value:

Type	Description
String	String value returned by Business Object

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.2.2 ControlValue

This object is used to populate view values to be used by called BO.

Members:

Name	Type	Description
name	String	Can be **ENTITYID** to store value of the current entity or other client specific name.
controlPath	String	Path of the view control element.
value	String	Control value
dataType	String	Control data type, e.g. NUMERIC, DATE, TIME

3.2.3 Using XML in executeBO

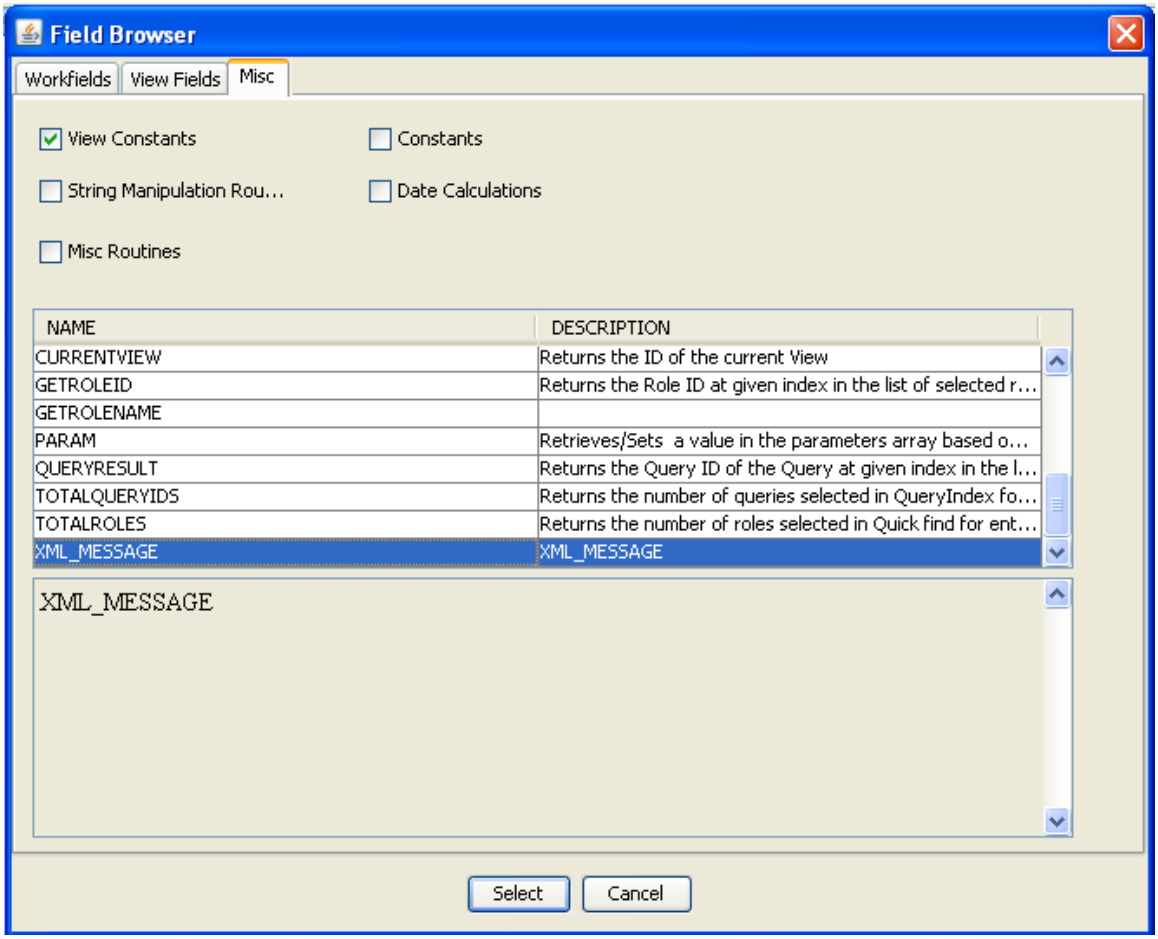
String value passed to business object can be a simple string message or a formatted XML message, e.g.:

web service test data

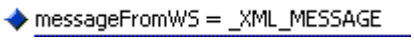
or

<bo-message>web service test data</bo-message>

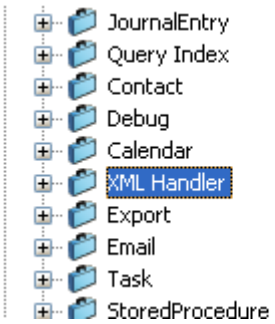
Passed message is obtained through XML_MESSAGE view constant:



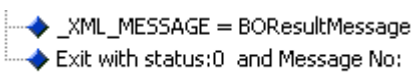
This message then can be assigned to some variable:



To handle xml message passed to BO, XML Handler PO can be used:



Upon BO execution, resulting message is sent back to the service using the same XML_MESSAGE constant:



3.3 Search Interface

Search Interface can be used to run queries previously created in Query Explorer tool. You may also access those queries and saved queries search results.

3.3.1 getAllQueriesNames

This method returns array of names of all queries available to current user.

```
String[] getAllQueriesNames(long sid)
throws AccessDeniedException,
       ServerErrorException,
       DataNotFoundException;
```

Method arguments:

Argument	Type	Description
Sid	Long	Session id

Return value:

Type	Description
String[]	Array of names of all queries available to current user

Exceptions:

- AccessDeniedException
- ServerErrorException
- DataNotFoundException

3.3.2 runQuery

This method executes a query and returns the Search Result object (SearchResultV1) containing array of found entities and all supplementary information. Accepts array of search parameters (SearchParameter) and can save or not save search results to Adapt V11 database depending on saveResults boolean parameter. This method accepts firstItemIndex and itemCount parameters to limit size of returning data, to prevent slowdown when retrieving results with a big number of found entities.

If Business Object is associated with a query in the Query Explorer Tool, it will be executed after the execution of the Query. The parameters which are not used by query will be available to Business Object.

```
SearchResultV1 runQuery (long sid,
                         String queryName,
                         SearchParameter [] parameters,
                         int firstItemIndex,
                         int itemCount,
                         boolean saveResults,
                         String assignToUserName}
throws
        AccessDeniedException,
        InvalidArgumentException,
        DataNotFoundException
        ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
queryName	String	Name of the query
parameters	SearchParameter[]	Parameters of the search
firstItemIndex	Integer	Index of the first item
itemCount	Integer	Maximum number of items to be returned
saveResults	Boolean	Should results be saved or not
assignToUserName	String	User name to assign results to. If not provided, results are assigned to service user.

Return value:

Type	Description
SearchResultV1	Search result data

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

Note: The method runQuery() executes the configured pre-query BO before executing the query. The execution of the query depends upon the status returned by the pre-query BO, as given below:

1. No pre-query BO is configured for a query: The query executes as normal.
2. The pre-query BO returns a status which is greater than 0: The query is not executed
3. The pre-query BO returns a status of -1: The query is not executed
4. A system error occurs while executing the pre-query BO: The query is not executed
5. The pre-query BO returns a status of 0: The query executes as normal.

3.3.2.1 SearchParameter

SearchParameter is used to pass parameters to the query executed by the service, to retrieve search results based on some criteria and to retrieve user in **Entity** service.

Members:

Name	Type	Description
dataType	Integer	Data type of the passed parameter. 1 – String 2 – Numeric 3 – Date 4 - Character
name	String	Name of the query parameter
stringValue	String	Holds value of the string type parameter
longValue	Long	Holds value of the numeric type parameter
dateValue	Date	Holds value of the date type parameter

When retrieving search results (using **getSearchResults** method, see below), valid names and data types of the search parameters are:

Name	dataType	Description
Label	1	The title assigned to the query result
Created By	2	Query result creator
Created Start Date	3	Date range when query result was created
Created End Date	3	
Modified By	2	Modified by
Modified Start Date	3	Date range when query result was modified
Modified End Date	3	
Method	4	Method of the query invocation (e.g. B – Business Object, Q - query)
Highlight	2	Query result highlighted by user

In **getUser** method of **Entity** service, valid parameter names and data type are:

Name	dataType	Description
userName	1	User name
userID	2	User ID
ADName	1	Active Directory user name

3.3.3 getSearchResults

This method returns array of IDs of all stored query results sets that match the filter criteria. Filter is specified via array of search parameters (SearchParameter, see **3.3.2.1**).

```
long[] getSearchResults(long sid,
                        SearchParameter[] parameters)
throws
      InvalidArgumentException,
      ServerErrorException,
      DataNotFoundException,
      AccessDeniedException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
parameters	SearchParameter[]	Array of search parameters

Return value:

Type	Description
Long[]	Array of IDs of all stored query result sets that match the filter criteria

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.3.4 getSearchResult

This method returns stored query result in form of Search Result object (SearchResultV1) according to the formatter object of the search result/user/domain. This method also accepts firstItemIndex and itemCount parameters to limit size of returning data.

```
SearchResultV1 getSearchResult(long sid,
                               long id,
                               int firstItemIndex,
                               int itemCount)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
id	Long	Saved search result ID
firstItemIndex	Integer	Index of the first item
itemCount	Integer	Maximum number of items to be returned

Return value:

Type	Description
SearchResultV1	stored search result

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.3.5 getSearchResultXML

This method returns stored query result in form of XML string with formatter settings obtained by search result (not search definition) id. Method accepts array of ColumnFormat objects in order to provide convenient formatting of columns in returning XML data set.

```
String getSearchResultXML(long sid,
                          long id,
```

```
                              int firstItemIndex,
                              int itemCount,
                              ColumnFormat[] columns)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
id	Long	Saved search result id
firstItemIndex	Integer	Index of the first item
itemCount	Integer	Maximum number of items to be returned
columns	ColumnFormat[]	Array of ColumnFormat objects in order to provide convenient formatting of columns in returning XML data set. See 3.3.5.1

Return value:

Type	Description
String	XML of saved search result

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.3.5.1 ColumnFormat

This class defines path to value returned in search result XML and used to apply sorting.

Members:

Name	Type	Description
referencePath	String	See description below
language	String	Name of any language registered in Adapt (e.g. English, French, German). This field is applicable for code attributes only and used to retrieve description in necessary language.
display	Short	Applicable for code attributes only. Possible values are: 0 – returning raw code value, 1 – code names are returned, 2 – returning code descriptions.

<table>
<tr>
<td>sort</td>
<td>Short</td>
<td>Possible values of sort order are:

<table>
<tr><th rowspan="2">Value</th><th colspan="2">Attribute Data Type</th></tr>
<tr><th>Non CODE</th><th>CODE</th></tr>
<tr><td>0</td><td>no ordering</td><td>no ordering</td></tr>
<tr><td>1</td><td>ascending by display value</td><td>ascending by Code ID</td></tr>
<tr><td>2</td><td>descending by display value</td><td>descending by Code ID</td></tr>
<tr><td>3</td><td>ascending by display value</td><td>ascending by Code Name</td></tr>
<tr><td>4</td><td>descending by display value</td><td>descending by Code Name</td></tr>
<tr><td>5</td><td>ascending by display value</td><td>ascending by Code Description</td></tr>
<tr><td>6</td><td>descending by display value</td><td>descending by Code Description</td></tr>
</table>
</td>
</tr>
</table>

The **referencePath** field contains sequence of reference strings. At least one reference should be specified.

Each reference consists of property name and 1-2 attribute names: optional reference attribute and mandatory value attribute.

The property name is the name of either system (like ENTITY_TABLE) or ordinary property (including “X_” tables) without prefix “PROP_” and could include optional occurrence name for named occurrence property.

The reference attribute is used to link current property with preceding property or with the root table.

The value attribute is used to retrieve reference value and in case of subsequent references to link with following property.

The CREATEDDATE, CREATED_BY, UPDATEDDATE value attribute names are allowed for the ENTITY_TABLE system property.

The USER_ID, USER_NAME, INITIALS, LOGIN_NAME, EMAIL value attribute names are allowed for U_PERSONAL table.

The format of **referencePath** field of **ColumnFormat** is as follows:

REF_ATTR_1.PROP_1(OCC_OF_PROP_1).ATTR_1:REF_ATTR_2.
PROP_2(OCC_OF_PROP_2).ATTR_2:REF_ATTR_3. PROP_3(OCC_OF_PROP_3)…

Where “REF_ATTR_1.PROP_1(OCC_OF_PROP_1).ATTR_1” is the first reference and “REF_ATTR_2.
PROP_2(OCC_OF_PROP_2).ATTR_2” is the second one and so on.

The REF_ATTR_1 is the name of the reference attribute of the first reference.

The PROP_1 is the name of the property.

The OCC_OF_PROP_1 is the name of the occurrence for the first reference property.

The ATTR_1 is the value attribute name.

The given sequence of references means that for each item in the result the instance of PROP_1 will be found where ENTITY_ID of the result item equals to REF_ATTR_1 value.

And for each PROP_1, instance of PROP_2 will be found, where value of ATTR_1 equals to REF_ATTR_2 and so on.

The reference attribute name can be omitted. In this case attribute predefined in Adapt will be used.

The last value attribute is column value. The sorting and display format is applied to it.

Examples of **referencePath**:

PERSON_GEN.FULLNAME

ADDRESS(Primary).COUNTRY

ENTITY_TABLE.UPDATEDDATE

X_ASSIG_CAND.JOB:JOB_GEN.JOB_TITLE

OFFICE.OWN_OFFICE.REFERENCE:REFERENCE.CLIENT_GEN.NAME

REFERENCE.OWN_CONS(Permanent).CONSULTANT:USER_ID.U_PERSONAL.USER_NAME

3.3.5.2 Search result XML

XML of saved search result has the following format:

<SearchResult searchID='SEARCH_ID'>

<Entity id='ENTITY_ID' defaultrole="DEFAULT_ROLE" resultIndex='1'/>

….

<Entity id='ENTITY_ID' defaultrole="DEFAULT_ROLE" resultIndex='N'/>

</SearchResult>

3.3.6 getSearchResultCount

This method returns number of records in specified stored query result.

```
long getSearchResultEntityCount (long sid,
                                 long id)
throws
      AccessDeniedException,
```

```
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
id	Long	Saved search result id

Return value:

Type	Description
Long	Number of records in query result

Exceptions:

- AccessDeniedException
- IDataNotFoundException
- ServerErrorException

3.3.7 deleteSearchResult

This method permanently deletes search result by specified id

```
void deleteSearchResult(long sid,
                        long id)
throws
      AccessDeniedException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
id	Long	Saved search result id

Return value: N/A

Exceptions:

- AccessDeniedException
- IDataNotFoundException
- ServerErrorException

3.3.8 getSearchResultWithRoles

This method returns stored query result in form of Search Result object (SearchResultWithRoles). This method is similar to the getSearchResult method except returning value (SearchResultWithRoles instead of

SearchResultV1). The SearchResultWithRoles object extends the SearchResultV1 by adding new field - array of 'EntityRoleBean' beans, which in their turn contain an entity Id and its default role name. This method also accepts firstItemIndex and itemCount parameters to limit size of returning data.

```
SearchResultWithRoles getSearchResultWithRoles(long sessionId,
                                long id,
                                int firstItemIndex,
                                int itemCount)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Usage	Description
sessionId	Long	Manadatory	Session id
id	Long	Manadatory	Saved search result ID
firstItemIndex	Integer	Manadatory	Number of the Page to represent found entries from
itemCount	Integer	Manadatory	Maximum number of items to be returned

Return value:

Type	Description
SearchResultWithRoles	stored search result

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.3.9 quickFind

This method performs search and returns its result in form of XML string. All method arguments are mandatory. Search results are not limited. Search criteria is taken from the attributeToCompare – it must be a quickfindable attribute and hence it has the criteria defined.

```
String quickFind(long sessionId,
                        String fromProperty,
                        String attributeToCompare,
                        String attributeToReturn,
                        String dataToFind)
throws
      AccessDeniedException,
```

```
InvalidArgumentException,
DataNotFoundException,
ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
fromProperty	String	Config name of the property
attributeToCompare	String	Config name of the attribute to be searched on. Must be quickfindable attribute
attributeToReturn	String	Config name of the attribute whose value will be returned
dataToFind	String	The value to look for in the **attributeToCompare**

Return value:

Type	Description
String	XML result of performed search.

Exceptions:

- AccessDeniedException
- InvalidArgumentException – when a non-quickfindable attribute is supplied
- DataNotFoundException
- ServerErrorException

3.3.9.1 XML of performed search has the following format:

<SearchResult >

<Entity id='id' **defaultrole='DEFAULT_ROLE'>**

<Property name='PROPERTY_NAME'>

<Attribute name='ATTRIBUTE_NAME'>

Attribute value

</Attribute>

</Property>

</Entity>

. . .

</SearchResult>

3.4 Entity Interface

Entity Interface can be used to retrieve, create, update and delete Entities. It is also possible to retrieve User as Entity.

3.4.1 Entity Ownership

Entities can be retrieved and edited with regard to the user account used to authenticate the connection, so that the user can only view and edit Entities which have been created by him personally or by other users who share user groups with him.

The table below shows dependency between Entity Access Service methods and users with different access rights:

		Disable Entity Visibility = YES			Disable Entity Visibility = NO		
		Entity owned by User Group which is User's Full Access Group	Entity owned by User Group which is User's View Only Group	Entity owned by User Group which is neither User's Full Access Group nor View Only	Entity owned by User Group which is User's Full Access Group	Entity owned by User Group which is User's View Only Group	Entity owned by User Group which is neither User's Full Access Group nor View Only
V1							
getUser		+	+	+	+	+	+
getEntity		+	+	+	+	+	x
getEntityData		+	+	+	+	+	x
getEntityDataNoPrimaryReference		+	+	+	+	+	x
saveEntity	create new	+	+	+	+	+	+
saveEntityData	create new	+	+	+	+	+	+
saveEntity	update existing	+	x	x	+	x	x
saveEntityData	update existing	+	x	x	+	x	x
deleteEntity		+	x	x	+	x	x

3.4.2 getUser

This method returns user data in form of XML, based on Adapt V11 core “get user as entity” functionality.

```
String getUser(long sid,
               SearchParameter parameter)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
parameter	SearchParameter	Search Parameter (see 3.3.3.1)

Return value:

Type	Description
String	User XML

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.4.3 getEntity

This method returns Entity XML by entity ID. Formatting rules like time zone, locale, and date and time format are specified at logon

```
String getEntity (long sid,
                  long entityID)
throws
      AccessDeniedException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
id	Long	Entity ID

Return value:

Type	Description
String	Entity XML

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.4.4 Entity XML format

getEntity method returns entity in XML representation. Format for this representation is as follows:

```
<Entity defaultrole='DEFAULT_ROLE'>
                    <Role>Role1</Role>
                    <Role>Role2</Role>
                          ……….
                    <Role>RoleN</Role>
          <Property name='PROPERTY_NAME_1' occurrence='OCCURRENCE_NAME'>
                    <Attribute name='ATTRIBUTE_NAME' sqldatatype='ATTRIBUTE_TYPE'>
                              Value
                    </Attribute>
                    ……
                    <Attribute name='ATTRIBUTE_NAME' sqldatatype='ATTRIBUTE_TYPE'>
```

```
                    Value
                </Attribute>
        </Property>
        ……..
        <Property name='PROPERTY_NAME_N'>
                <Attribute name='ATTRIBUTE_NAME' sqldatatype='ATTRIBUTE_TYPE'>
                    Value
                </Attribute>
                ...
                <Attribute name='ATTRIBUTE_NAME' sqldatatype='ATTRIBUTE_TYPE'>
                    Value
                </Attribute>
        </Property>
</Entity>
```

If referenced attribute in list of property is an entity, it will have the following format:

```
        <Property name='PROPERTY_NAME_1' occurrence='OCCURRENCE_NAME'>
        <Attribute name='ATTRIBUTE_NAME'
                  defaultrole='DEFAULT_ROLE'
                  sqldatatype='ATTRIBUTE_TYPE'>
                Value
        </Attribute>
        </Property>
```

To create a new *Entity,* an XML representation of the entity to be created must be constructed. The format of the XML for *Entity* creation is as follows:

```
<Entity defaultrole='DEFAULT_ROLE'>
                    <Role>Role1</Role>
                    <Role>Role2</Role>
                        ……….
                    <Role>RoleN</Role>
        <Property name='PROPERTY_NAME_1' occurrence='OCCURRENCE_NAME'>
        <Attributename='ATTRIBUTE_NAME' sqldatatype='ATTRIBUTE_TYPE'>
                Value
        </Attribute>
        </Property>
        ……..
        <Property name='PROPERTY_NAME_N'>
        <Attributename='ATTRIBUTE_NAME' sqldatatype='ATTRIBUTE_TYPE'>
                Value
        </Attribute>
        </Property>
</Entity>
```

Where **'OCCURRENCE_NAME'** is occurrence from the *Named Property*.
Note that **'OCCURRENCE_NAME'** only needs to be specified for *Named Properties*

Below is an example of the XML that creates permanent candidate *Entity*.

```
<Entity defaultrole='PERM_CAND'>
  <Role>PERM_CAND</Role>
  <Property name='PERSON_GEN'>
    <Attribute name='MIDDLE_NAME' sqldatatype='NVARCHAR'>Middle</Attribute>
```

```
    <Attribute name='FIRST_NAME' sqldatatype='NVARCHAR'>First</Attribute>
    <Attribute name='LAST_NAME' sqldatatype='NVARCHAR'>Last</Attribute>
    <Attribute name='FULLNAME' sqldatatype='NVARCHAR'>Mr Phillips</Attribute>
    <Attribute name='TITLE' sqldatatype='NUMERIC'>1303812</Attribute>
    <Attribute name='GENDER' sqldatatype='NUMERIC'>1303874</Attribute>
  </Property>
  <Property name='CAND_GEN'>
    <Attribute name='PLACE_OF_B' sqldatatype='NVARCHAR'></Attribute>
    <Attribute name='NATIONAL_NUM' sqldatatype='NVARCHAR'></Attribute>
    <Attribute name='ID_NO' sqldatatype='NVARCHAR'></Attribute>
    <Attribute name='BENEFIT_FROM' sqldatatype='CHAR'>N</Attribute>
    <Attribute name='RIGHT_BENEF' sqldatatype='CHAR'>N</Attribute>
    <Attribute name='OWN_TRANS' sqldatatype='CHAR'>N</Attribute>
    <Attribute name='HOW_FIND_US' sqldatatype='NUMERIC'></Attribute>
     <Attribute name='NO_OF_CHILD' sqldatatype='NUMERIC'></Attribute>
    <Attribute name='DIVISION' sqldatatype='NUMERIC'></Attribute>
    <Attribute name='NO_OF_DEPEND' sqldatatype='NUMERIC'></Attribute>
    <Attribute name='GENDER' sqldatatype='NUMERIC'>1303874</Attribute>
    <Attribute name='LOCATION_CD' sqldatatype='NUMERIC'></Attribute>
    <Attribute name='AVAILABILITY' sqldatatype='NUMERIC'></Attribute>
    <Attribute name='NATIONALITY' sqldatatype='NUMERIC'>20010516</Attribute>
    <Attribute name='MARITAL_STAT' sqldatatype='NUMERIC'></Attribute>
    <Attribute name='DT_OF_BIRTH' sqldatatype='DATETIME'>09/11/1976</Attribute>
  </Property>
<Property name='TELEPHONE' occurrence='Work'>
    <Attribute name='EXTENSION' sqldatatype='NVARCHAR'></Attribute>
    <Attribute name='TEL_NUMBER' sqldatatype='NVARCHAR'>+44 131 312 1331</Attribute>
    <Attribute name='CAN_SMS' sqldatatype='CHAR'>N</Attribute>
    <Attribute name='CAN_CONTACT' sqldatatype='CHAR'>N</Attribute>
    <Attribute name='OCC_ID' sqldatatype='NUMERIC'>2034418</Attribute>
 </Property>
</Entity>
```

To update existing entity, XML representation of the entity is used.
XML format for *Entity* update is as follows:

```
<Entity id='ENTITY_ID_TO_UPDATE' defaultrole='ENTITY_DEFAULT_ROLE'>
  <Role>Role1</Role>
  <Role>Role2</Role>
        ……
  <Role>RoleN</Role>

<Property name=’PROPERTY_NAME_MULTIPLE’ bisuniqueid=’BISUNIQUE_ID’>
        <Attribute name=’ATTRIBUTE_NAME’ sqldatatype=’ATTRIBUTE_TYPE’>
                Value
    </Attribute>
</Property>
<Property name=’PROPERTY_NAME_NAMED’ occurrence=’OCCURRENCE_NAME’>
        <Attribute name=’ATTRIBUTE_NAME’ sqldatatype=’ATTRIBUTE_TYPE’>
                Value
    </Attribute>
</Property>
<Property name=’PROPERTY_NAME_SINGLE’>
        <Attribute name=’ATTRIBUTE_NAME’ sqldatatype=’ATTRIBUTE_TYPE’>
                Value
```

</Attribute>
</Property>

……..
</Entity>

Where **'BISUNIQUE_ID'** is unique ID of the *Multiple Property*
Below are the rules that apply to XML format for *Entity* update:

1. ID of the *Entity* to update must be provided. If there is no *Entity* with such in the system – **DataNotFoundException** will be thrown.
2. In order to update a record in a *Multiple Property*, **'BISUNIQUE_ID'** of the record must be specified. If there is no provided **'BISUNIQUE_ID'** in the system – **DataNotFoundException** will be thrown. If **'BISUNIQUE_ID'** is omitted – new record will be added to the property.
3. To update a N*amed Property*, '**OCCURRENCE_NAME'** or **'BISUNIQUE_ID'** must be specified. If **'OCCURRENCE_NAME'** or **'BISUNIQUE_ID'** is not found – **DataNotFoundException** will be thrown.
4. In order to delete a record from a Multiple Property, the following syntax is used:

   <Property name='PROPERTY_NAME' bisuniqueid='BISUNIQUE_ID'>
   </Property>

5. In order to delete a record from a Named Property, one of the following syntaxes can be used:

   1. **<Property name='PROPERTY_NAME' occurrence='OCCURRENCE_NAME'>**
      </Property>

   2. **<Property name='PROPERTY_NAME' bisuniqueid='BISUNIQUE_ID'>**
      </Property>

6. To delete all records from a property, the following syntax should be used:
   <Property name='PROPERTY_NAME_N' ></Property

Note that you can't update existent property record while deleting all records from the property – as a result property with no records will be received. E.g.:

<Property name='PROPERTY_NAME_N' ></Property>
<Property name='PROPERTY_NAME' occurrence='OCCURRENCE_NAME'>
<Attribute name='ATTRIBUTE_NAME' sqldatatype='ATTRIBUTE_TYPE'>
Value
</Attribute>
</Property>

It is possible however to delete all records from the property while adding new ones.

3.4.5 getEntityData

This method retrieves existing entity data by entity ID and list of entity's properties and returns it in XML representation. The resulting XML will contain entity's data stored in the all attributes from specified properties.

String getEntityData(long sessionId, long entityId, String[] properties) throws RemoteException, ServerErrorException, DataNotFoundException, AccessDeniedException;

Method arguments:

Argument	Type	Description
sessionId	Long	Session id
entityID	Long	Entity ID
properties	String[]	Array of properties' config names to be returned

Return value: String – entity XML

Exceptions:

- AccessDeniedException
- DataNotFoundException
- ServerErrorException

RemoteException

3.4.6 saveEntity

This method performs saving (creates new or updates existing) using given XML data.

```
String saveEntity(long sid,
                  String entityXml)
throws
    AccessDeniedException,
    InvalidArgumentException,
    DataNotFoundException,
    ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
entityXml	String	Entity XML

Return value:

Type	Description
String	Entity XML

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.4.7 saveEntityData

This method performs saving an entity (creates new or updates existing) using given XML data. The return value contains only ID of the saved entity.

```
Long saveEntity(long sid,
                String entityXml)
throws
    AccessDeniedException,
    InvalidArgumentException,
    DataNotFoundException,
    ServerErrorException;
```

Method arguments:

Argument	Type	Description

sid	Long	Session id
entityXml	String	Entity XML

Return value:

Type	Description
Long	Entity ID

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.4.8 deleteEntity

This method deletes existing entity. Entity can be deleted permanently if “permanently” flag is set in true.

```
void deleteEntity(long sid,
                  long entityID,
                  boolean permanently)
throws
        AccessDeniedException,
        DataNotFoundException,
        ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
entityID	Long	Entity ID
permanently	Boolean	Either entity should be deleted permanently or with possibility to restore

Return value: N/A

Exceptions:

- AccessDeniedException
- DataNotFoundException
- ServerErrorException

3.4.9 getFavouriteEntities

This method retrieves all favourite entities of current user and returns them in XML representation.
The resulting XML will contain list of entities with their Id, default role Id and default value.

```
String getFavouriteEntities(long sessionId)
throws
        RemoteException,
        ServerErrorException,
        DataNotFoundException,
        AccessDeniedException;
```

Method arguments:

Argument	Type	Description
sessionId	Long	Session id

Returning value:
<Favourites>
<Entity id='1' defaultrole='1'>Default value1</Entity>
<Entity id='2' defaultrole='2'>Default value2</Entity>
[...etc]
</Favourites>

Exceptions:

- AccessDeniedException
- DataNotFoundException
- ServerErrorException
- RemoteException

3.5 Document Interface

Document interface can be used to retrieve, create, update and delete Document in Adapt V11 database. It is also possible to get IDs of all documents for specified entity.

3.5.1 DocumentV1 data object

This object contains everything related to document in Adapt V11 system.

Field	Type	Read only	Description
documentID	Long		Document ID. Should be zero when creating new document.
owner	String	Yes	Owner name
ownerID	Long		Owner ID
category	String		Path of the document category
name	String		Document name
fileExtension	String		Document file extension
description	String		Document description
createdDate	Date		Creation date
createdBy	String	Yes	Creator name
createdByID	Long		Creator ID
updatedDate	Date		Updating date
updatedBy	String	Yes	Updater name
updatedByID	Long		Updater ID
note	String		Note
content	Byte[]		Binary content of the document
size	Date		Size of the binary content
default	Boolean		Whether document is the default one
active	Boolean	Yes	Whether document is active one(not deleted)
ownerType	Integer		Can be User = 1, Entity = 0, Journal = 2

3.5.2 getDocument

This method retrieves document from database.

```
DocumentV1 getDocument(long sid,
                       long id,
                       boolean withContent)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
```

```
        ServerErrorException;
```

Method arguments:

Argument	Type	Description
Sid	Long	Session ID
Id	Long	Document ID
withContent	Boolean	Whether document content should be downloaded too

Return value:

Type	Description
DocumentV1	Document data with or without document content

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.5.3 saveDocument

This method saves document. If document ID is 0, the new document will be created.

```
long saveDocument(long sid,
                  DocumentV1 document)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session ID
document	DocumentV1	Document to be saved

Return value:

Type	Description
Long	ID of saved Document. Is necessary for new created documents

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.5.4 deleteDocument

This method deletes document. Document can be deleted permanently if “permanently” flag is set in true.

```
void deleteDocument(long sid,
                    long documentID,
                    boolean permanently)
throws
```

AccessDeniedException,
DataNotFoundException,
ServerErrorException;

Method arguments:

Argument	Type	Description
sid	Long	Session ID
documentID	Long	Document ID
permanently	Boolean	Whether entity should be deleted permanently or there should be a possibility to restore it

Return value: N/A

Exceptions:

- AccessDeniedException
- DataNotFoundException
- ServerErrorException;

3.5.5 getEntityDocuments

This method returns ID of specified entity documents. Language and document library category can be specified.

```
long[] getEntityDocuments(long sid,
                          long entityID,
                          String category)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session ID
entityID	Long	Entity ID
Category	String	Category of the document to be retrieved

Return value:

Type	Description
Long[]	Specified entity documents IDs Array

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException;

3.5.6 getDocumentsInfoByOwnerAndCategory

This method is used to retrieve list of documents for specific owner and document library category. Unlike getEntityDocuments method which returns array of document identifiers, this method returns array of document beans which contains all document information (see section 3.5.1 DocumentV1 data object) except document content.

```
DocumentV1[] getDocumentsInfoByOwnerAndCategory(long sessionId,
                                                long ownerID,
                                                String ownerType,
                                                String category)
```

Method arguments:

Argument	Type	Description
sessionId	long	Session id
ownerId	long	Owner ID
ownerType	String	Owner Type ('E' – Entity, 'U' – User, 'J' – Journal Entry, 'T' - Task)
category	String	Category of the document to be retrieved

Return value:

Type	Description
DocumentV1[]	Array of document V1 beans which contains all document information except document content

Exceptions:

- ServerErrorException
- AccessDeniedException
- DataNotFoundException
- RemoteException
- InvalidArgumentException;

3.5.7 getDocumentsInfoByOwner

This method is used to retrieve list of documents for a specific owner. Unlike getEntityDocuments method which returns array of document identifiers, this method returns array of document beans which contains all document information (see section 3.5.1 DocumentV1 data object) except document content.

```
DocumentV1[] getDocumentsInfoByOwner(long sessionId,
                                     long ownerID,
                                     String ownerType)
```

Method arguments:

Argument	Type	Description
sessionId	long	Session id
ownerId	long	Owner ID
ownerType	String	Owner Type ('E' – Entity, 'U' – User, 'J' – Journal Entry, 'T' - Task)

Return value:

Type	Description
DocumentV1[]	Array of document V1 beans which contains all document information except document content

Exceptions:

- ServerErrorException
- AccessDeniedException
- DataNotFoundException
- RemoteException
- InvalidArgumentException;

3.6 Calendar

Calendar Interface provides access to appointments in Adapt V11 Calendar.

3.6.1 BookingV1 data object

This object contains everything related to calendar appointment (Booking) in Adapt V11 system.

Field	Type	Read only	Description
id	Long		Booking ID.
bookingType	Integer		USER = 0 ENTITY = 1
bookingCodeId	Long		Code ID the booking was booked with
endDate	Date		End date
startDate	Date		Start date
note	String		Booking note
subject	String		Booking subject
userIds	Long[]		User IDs
entityIds	Long[]		Entity IDs

3.6.2 getBookingById

This method returns booking data by given booking Id.

```
BookingV1 getBookingById(long sid,
                         long bookingID,
                         int bookingType)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session ID
bookingID	Long	Booking ID
bookingType	Integer	Booking type. Can be USER=0 or ENTITY=0

Return value:

Type	Description
BookingV1	Booking data

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.6.3 getBookings

This method returns all bookings for the owner specified (entity or user).

```
BookingV1[] getBookings(long sid,
                        long ownerID,
                        int ownerType)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session ID
ownerID	Long	Creator ID, can be ID of Entity or User
ownerType	Integer	Can be USER=0 or ENTITY=0

Return value:

Type	Description
BookingV1[]	Array of bookings

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.6.4 saveBooking

This method saves booking. If bookingID is 0, the new booking will be created.

```
long saveBooking(long sid,
                 BookingV1 booking)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
booking	BookingV1	Booking data to be saved

Return value:

Type	Description
long	Saved booking ID.

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.6.5 deleteBooking

This method deletes booking.

```
void deleteBooking(long sid,
                   long bookingID,
                   int bookingType)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
bookingID	Long	Booking ID
bookingType	Integer	0 = Entity Booking or 1 = User Booking

Return value: N/A

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.6.6 getFilteredBookings

This method returns list of bookings filtered by start and end dates for the specified owner (entity or user).

```
BookingV1[] getFilteredBookings(Long sessionId, Filter filter)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sessionId	Long	Session ID
Filter	Filter	Filter. Please refer to the Section '3.7.3.1 Filter' of this Document

Return value:

Type	Description
BookingV1[]	Array of bookings

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
  ServerErrorException

3.7 Task

Task Interface provides access to Adapt V11 Tasks and is very similar to Calendar

3.7.1 TaskV1 data object

This object contains all necessary data related to task in Adapt V11 system.

Field	Type	Read only	Description
id	Long		Task ID.
creatorId	Long		ID of the entity or user who created this task
creatorType	Integer		USER = 0 ENTITY = 1
entityIDs	Long[]		ID of the entities this task is assigned to
taskCreationDate	Date		Task creation date
startDate	Date		Task start date
dueDate	Date		Task end date
completedDate	Date		Factual task completion date
completed	Boolean		Whether the task is completed
completedPercentage	Integer		Task completion percentage
status	String		Can be "All", "N", "P", "C", "H", "D"
priority	String		Can be "H", "M", "L"
subject	String		Subject of the task
description	String		Description of the task

3.7.2 getTaskById

This method returns task by its ID

```
TaskV1 getTaskById (long sid,
                    long taskID)
throws
      AccessDeniedException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session ID
taskID	Long	Task ID

Return value:

Type	Description
TaskV1	Task data

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.7.3 getTasks

This method returns array of task objects filtered and sorted by given criteria.

```
TaskV1[] getTasks(long sid,
                  Filter filter,
                  int firstItemIndex,
                  int itemCount)
throws
      AccessDeniedException,
```

```
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException
```

Method arguments:

Argument	Type	Description
sid	Long	Session ID
filter	Filter	Filter, see **3.7.3.1**
firstItemIndex	firstItemIndex	First item Index
itemCount	itemCount	Maximum number of items to be returned

Return value:

Type	Description
TaskV1 []	Array of Tasks found by given criteria

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.7.3.1 Filter

Helper object that contains filter criteria passed to the getTasks or getFileteredBookings method to sort/filter returned Tasks or Calendar entries.

Members:

Name of the criteria	Type	Usage	Criteria Description for getTasks method	Criteria Description for getFileteredBookings method
startDate	Calendar	Optional	Task start date	Booking start date (if not specified, 01-Jan -1753 is set by default)
endDate	Calendar	Optional	Task end date	Booking end date (if not specified, 31-Dec-9999 is set by default)
sortColumn	String	Mandatory	Column name to sort retrieved tasks by	Ignored
sortOrder	String	Optional	Order of sorting	Ignored
filterText	String	Optional	Text to search upon – applies to Task subject and documents and to Journal documents.	Ignored
ownerID	Long	Mandatory	Task owner ID	Booking owner ID
ownerType	Integer	Mandatory	Type of the task's owner, can be 0 - user or 1 – entity	Booking owner type, can be 0 - user or 1 – entity

3.7.4 saveTask

This method saves task. If taskID is 0, the new booking will be created

```
long saveTask (long sid,
                  TaskV1 task)
throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
```

```
        ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
task	TaskV1	Task to be saved

Return value:

Type	Description
Long	Created/updated task Id

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.7.5 deleteTask

This method deletes task.

```
void deleteTask(long sid,
                long taskID)
throws
        AccessDeniedException,
        InvalidArgumentException,
        DataNotFoundException,
        ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
taskID	Long	ID of the task to be deleted

Return value: N/A

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.7.6 deleteTasks

This method deletes tasks.

```
void deleteTask(long sid,
                long[] taskIds)
throws
        AccessDeniedException,
        InvalidArgumentException,
        DataNotFoundException,
        ServerErrorException;
```

Method arguments:

Argument	Type	Description

sid	Long	Session id
taskID	Long[]	IDs of the tasks to be deleted

Return value: N/A
Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.8 Journal

Journal Interface can be used to retrieve Adapt V11 Journal entries

3.8.1 getJournalEntry

This method returns journal entry by ID

```
JournalBean getJournalEntry(long sid,
                            long id)
 throws
      AccessDeniedException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
Sid	Long	Session id
Id	Long	Journal record ID

Return value:

Type	Description
JournalBean	Journal record data

Exceptions:

- AccessDeniedException
- DataNotFoundException
- ServerErrorException

3.8.2 getJournalEntries

This method returns array of journal entries found by given filter

```
JournalBean[] getJournalEntries(long sid,
                                Filter filter,
                                int firstItemIndex,
                                int itemCount)
 throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description

Sid	Long	Session id
Filter	Filter	Filter (same as for **Task**, see **3.7.3.1**)
firstItemIndex	Integer	First item Index
itemCount	Integer	Maximum number of items to be returned

Return value:

Type	Description
JournalBean[]	Array of journal entries found by given filter

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.9 Code Group

Is used to retrieve codes and code groups

3.9.1 getCodeGroup

This method retrieves CodeGroup identified by domain name and display name of CodeGroup with localized descriptions in all active system languages.

```
CodeGroupBean getCodeGroup (long sid,
                            String groupName)
 throws
      AccessDeniedException,
      InvalidArgumentException,
      DataNotFoundException,
      ServerErrorException
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
groupName	String	Code Group name

Return value:

Type	Description
CodeGroupBean	Code Group data

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.9.2 getCodeGroupByLanguage

This method retrieves CodeGroup identified by domain name and display name of CodeGroup with localized representation for the language specified.

```
CodeGroupBean getCodeGroupByLanguage (long sid,
                                      String groupName,
                                      String language)
throws
      AccessDeniedException,
```

```
        InvalidArgumentException,
        DataNotFoundException,
        ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session id
groupName	String	Code group name
language	String	Code group language

Return value:

Type	Description
CodeGroupBean	Code group data

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.9.3 getCodeByID

This method retrieves localized representation of the Code by ID and language name

```
CodeBean getCodeByID (long sid,
                      long codeID,
                      String language)
throws
        AccessDeniedException,
        InvalidArgumentException,
        DataNotFoundException,
        ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session ID
codeID	Long	Code ID
language	String	The language to retrieve code for

Return value:

Type	Description
CodeBean	Code data

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.9.4 getCodeIDFromPath

This method retrieves Code ID by Code Path

```
 long getCodeIDFromPath(long sid,
                        String path)
throws
        AccessDeniedException,
```

```
        InvalidArgumentException,
        DataNotFoundException,
        ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session ID
path	String	Code path

Return value:

Type	Description
Long	Code ID

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.9.5 getCodeIDFromGroupAndCodeDescr

This method retrieves Code ID by Code Group ID and Code Description

```
 long[] getCodeIDFromPath(long sid, long groupId, String description,
                          String language)
throws
        AccessDeniedException,
        InvalidArgumentException,
        DataNotFoundException,
        ServerErrorException;
```

Method arguments:

Argument	Type	Description
sid	Long	Session ID
groupId	Long	Code Group ID
description	String	Code description
language	String	The language to retrieve code for

Return value:

Type	Description
long[]	Code Ids Array

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException

3.9.6 getCodeGroups

The functionality of this method is to retrieve all the code groups available in the domain specified. CodeGroupBean object will contains all the information related to the Code Group except the codes it contains.

```
CodeGroupBean[] getCodeGroups(long sessionId)
        throws ServerErrorException, DataNotFoundException, AccessDeniedException
```

Method Arguments:

Argument	Type	Description
sessionId	Long	Session ID

Return value:

Argument	Description
CodeGroupBean[]	Code Group data array

Exceptions:

- AccessDeniedException
- ServerErrorException
- DataNotFoundException

Below is an example of XML request that retrieves an array of CodeGroupBean objects.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:cod="http://codegroup.webservice.bis.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <cod:getCodeGroups>
      <String_1>SRxUKCRA</String_1>
    </cod:getCodeGroups>
  </soapenv:Body>
</soapenv:Envelope>
```

3.9.7 updateCodeGroup

The functionality of this method is to update the code group details in the form of CodeGroupBean object passed as parameter to this method along with the domain name. The CodeGroupBean contains all the data related to the code group except the codes.

long updateCodeGroup(long sessionId, CodeGroupBean codeGroupBean)
throws InvalidArgumentException, ServerErrorException, AccessDeniedException, DataNotFoundException, InvalidCodeGroupException

Method arguments:

Argument	Type	Description
sessionId	Long	Session ID
codeGroupBean	CodeGroupBean	Code Group details

Return value:

Type	Description
long	Id of the code group

Exceptions:

- InvalidArgumentException
- AccessDeniedException
- DataNotFoundException
- ServerErrorException
- InvalidCodeGroupException

Below is an example of XML request that updates the passed codegroupbean details.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:cod="http://codegroup.webservice.bis.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <cod:updateCodeGroup>
      <String_1>SRxUKCRA</String_1>
      <CodeGroupBean_2>
        <allDescriptions>
          <description>Authorisation Italian Desc</description>
          <language>Italian</language>
          <name>Authorisation Italian</name>
        </allDescriptions>
        <allowParent>true</allowParent>
        <format>YYYY</format>
        <groupName>Authorisation</groupName>
        <id>7959152</id>
        <localLanguage>English</localLanguage>
        <localisedDesc>Authorisation English Desc</localisedDesc>
        <localisedName>Authorisation English</localisedName>
        <lookupBy>D</lookupBy>
        <namespaceID>0</namespaceID>
        <nodesExpanded>true</nodesExpanded>
        <sortBy>D</sortBy>
      </CodeGroupBean_2>
    </cod:updateCodeGroup>
  </soapenv:Body>
</soapenv:Envelope>
```

3.9.8 updateCode

The functionality of this method is to update the language specific code details in the form of CodeBean object passed as parameter to this method along with the domain name. The CodeBean object contains all the data related to the code.

long updateCode(long sessionId, CodeBean codeBean)
throws InvalidArgumentException, ServerErrorException, AccessDeniedException, DataNotFoundException, InvalidCodeException

Method arguments:

Argument	Type	Description
sessionId	Long	Session ID
codeBean	CodeBean	Code details

Return value:

Type	Description
long	Id of the code

Exceptions:

- InvalidArgumentException
- AccessDeniedException
- DataNotFoundException
- ServerErrorException
- InvalidCodeException

Below is an example of XML request that updates the language specific codebean details. The language name is specified in between 'localLanguage' tag is highlighted.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:cod="http://codegroup.webservice.bis.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <cod:updateCode>
      <String_1>SRxUKCRA</String_1>
      <CodeBean_2>
        <allDescriptions>
          <description>CV Spanish Desc</description>
          <language>Spanish</language>
          <name>CV Spanish</name>
        </allDescriptions>
        <dependants>
          <childCodeID>4611686018435640346</childCodeID>
          <childGroupID>9187343239844111142</childGroupID>
        </dependants>
        <globalEquivId>0</globalEquivId>
        <groups>6624762</groups>
        <hidden>true</hidden>
        <id>8249277</id>
        <localLanguage>Italian</localLanguage>
        <localizedDesc>CV Italian Desc</localizedDesc>
        <localizedName>CV Italian</localizedName>
        <name>CV_TYPE</name>
        <namespaceId>0</namespaceId>
        <parentId>0</parentId>
        <synonyms>
          <matchPercent>55</matchPercent>
          <synonymID>8249602</synonymID>
        </synonyms>
      </CodeBean_2>
    </cod:updateCode>
  </soapenv:Body>
</soapenv:Envelope>
```

3.9.9 CodeBean object

This object contains information related to codes in Adapt V11 system.

Field	Type	Description
Id	Long	Code ID
parentId	Long	Parent code ID. 0 for the top level code
name	String	Code Name
localizedName	String	Language specific name of code
localizedDesc	String	Language specific description of code
localLanguage	String	The name of the local language used for localised settings
allDescriptions	LocalName []	An array of local name objects providing details of multilingiual name and descriptions for the code
hidden	Boolean	Is this code hidden or not
globalEquivId	Long	Language specific name of code
namespaceId	Long	Language specific description of code

synonyms	CodeSynonym[]	An array of CodeSynonym objects defining the synonyms for the code.
groups	Long[]	An array of user group IDs that visibility of this code is restricted.
dependants	DynamicVisibility[]	An array of DynamicVisibility objects defining the codes that are dependent on this code when dynamic code groups are used.

3.9.10 CodeGroupBean object

This object contains information related to code group in Adapt V11 system.

Field	Type	Description
id	Long	Code group ID
groupName	String	Code group name
allDescriptions	LocalName[]	Array of language specific names represented by LocalName objects
codes	CodeBean[]	Array of codes which belong to this code group
localLanguage	String	The name of the local language used for localised settings
localisedName	String	The group name in the local language
localisedDesc	String	The group description in the local language
format	String	The format specifier for the code group
lookupBy	String	Lookup the code group based on either code or description(C – sort by code, D – sort by description)
sortBy	String	Sort the code group based on either code or description(C – sort by code, D – sort by description)
nodesExpanded	boolean	Are nodes expanded or not
allowParent	boolean	Is selection of a parent node allowed or not
namespaceID	Long	ID the namespace this code group is defined in

3.9.11 LocalName object

This object contains information related to Adapt V11 system naming.

Field	Type	Description
language	String	Language name
name	String	Language specific name
description	String	Language specific description

3.9.12 CodeSynonym object

The following new properties added to this object.

Field	Type	Description
synonymID	Long	The ID of the associated (synonym) code
matchPercent	Int	The extent to which it matches the code

3.9.13 DynamicVisibility object

The following new properties added to this object.

Field	Type	Description
childGroupID	Long	The ID of the code group that the dependent code belongs to
childCodeID	Long	The ID of the dependent code in the group.

3.9.14 Exceptions

The following types of exceptions defined in AdaptV11 Web Services and can be thrown via standard SOAPFault. These exceptions exposed to webservice client through WSDL.

- **InvalidCodeException** – thrown if the code id passed to the webservice method is invalid or retrieved code object is null
- **InvalidCodeGroupException** – thrown if the code group id passed to the webservice method is invalid or codegroup object is null

3.10 Meta Data – IMetaDataServiceV1

MetaData service interface can be used to retrieve meta data information from the system.
The MetaData service interface methods are:

- **getRoles** - Retrieves information about all Roles in the specified domain.

3.10.1 getRoles

The purpose of this method is to retrieve all the roles available in the specified domain. RoleBean object will contain all the information related to the Role, including role's ID and config name, parent group ID and config name, and array of role's multilingual descriptions, wrapped into the LanguageDescription objects.

```
RoleBean[] getRoles(long sessionId)
throws
      RemoteException,
      AccessDeniedException,
      DataNotFoundException,
      ServerErrorException;
```

Method arguments:

Argument	Type	Description
sessionId	Long	Session id

Return value:

Type	Description
RoleBean[]	Roles data array

Exceptions:

- InvalidArgumentException
- AccessDeniedException
- DataNotFoundException
- ServerErrorException
- InvalidCodeException

3.10.2 RoleBean

This object contains information related to roles in Adapt V11 system.

Field	Type	Description
id	long	Role ID
name	String	Configuration role name.
roleGroupId	long	Role Group ID (if available)
roleGroupName	String	Role Group confuguration name

descriptionsArray	LanguageDescription[]	An array of role's multilingual descriptions

3.10.3 LanguageDescription

This object contains information related to multilingual names & descriptions in Adapt V11 system.

Field	Type	Description
languageId	long	Language ID
LanguageName	String	Language name
name	String	Item's name
description	String	Item's description

3.11 DataAdmin ManageRequestNotification

Is used to create merge request and create delete request

3.11.1 createMergeRequest

The functionality of this method is to create the merge request and it contains merge request details in the form of ManageRequestBean object passed as parameter to this method along with the session id.

String createMergeRequest(long sessionId, ManageRequestBean manageRequestBean)
throws RemoteException, InvalidArgumentException, ServerErrorException, DataNotFoundException, AccessDeniedException

Method arguments:

Argument	Type	Description
sessionid	long	Session id
manageRequestBean	ManageRequestBean	Merge request details to be saved

Return value:

Type	Description
String	Success/failure create merge request message

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException
- RemoteException

3.11.2 createDeleteRequest

The functionality of this method is to create the delete request and it contains delete request details in the form of DeleteRequestBean object passed as parameter to this method along with the session id.

String createDeleteRequest (long sessionId, DeleteRequestBean deleteRequestBean)
throws RemoteException, InvalidArgumentException, ServerErrorException, DataNotFoundException, AccessDeniedException

Method arguments:

Argument	Type	Description
sessionid	long	Session id
deleteRequestBean	DeleteRequestBean	delete request details to be saved

Return value:

Type	Description

String	Success/failure create delete request message

Exceptions:

- AccessDeniedException
- InvalidArgumentException
- DataNotFoundException
- ServerErrorException
- RemoteException