





Web Services Guide

VoiceObjects 11.1

To ensure that you are using the documentation that corresponds to the VoiceObjects software you are licensed to use, compare this version number with the software version shown in the Help menu of the VoiceObjects software you are using.

Copyright

Copyright © 2001-2011 Voxeo Germany GmbH and its licensors. All rights reserved.

Published in Germany – Legal information 2011

Information in this document is subject to change without notice and does not represent a commitment on the part of Voxeo or any of its affiliated companies (collectively “Voxeo”). The software described in this document is furnished solely under a license agreement. The software may be used or copied only in accordance with the terms of the license agreement. You shall not reverse engineer the software or sub-license, distribute or make the software available to third parties by other means except as specifically allowed in the license agreement or by mandatory law. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Voxeo.

Protected by German, European and US patents. Further patents pending.

Companies, names, and dates used in examples herein are fictitious unless otherwise noted. If such names affect copyrights or trademarks or others, please notify Voxeo by e-mail at support@voxeo.com.

Trademarks

VoiceObjects is a registered trademark. Any other trademarks, trade names or service marks mentioned in this document belong to their respective owners.

The material presented herein is based upon information that we consider reliable, but we do not represent that it is error-free and complete. Voxeo is not making any representation or granting any warranty with respect to such material, and the distribution of such material shall not subject Voxeo to any liability.

Third-Party Products

If Licensed Products are distributed together with third-party software or if this is contained in the Licensed Products, which are subject to additional license provisions, Licensee undertakes to observe the license provisions of the third party.

Explicit Copyright Notice

The VoiceObjects software includes software developed by the Apache Software Foundation (www.apache.org). Copyright © 1999-2011 – The Apache Software Foundation. All rights reserved.

Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S., other countries, or both.

Specific versions of the VoiceObjects software contain copyright material authorized by the Eclipse Foundation (www.eclipse.org), their contributors and others. All rights reserved.

Specific versions of the VoiceObjects software work with Microsoft Excel or make use of copyright material from Microsoft Corporation (www.microsoft.com). All rights to such copyright material rest with Microsoft. Microsoft and Excel are registered trademarks of Microsoft Corporation.

The VoiceObjects software is bundled with the software NuGram IDE Basic Edition developed by Nu Echo Inc. (www.nuecho.com). All title and copyrights in and to the software NuGram IDE Basic Edition, the accompanying printed materials, and any copies of the software are owned by Nu Echo Inc. and its suppliers.

Document Number: E-021-20111215-VO11



Table of Contents

TABLE OF CONTENTS	3
ABOUT THE WEB SERVICES GUIDE	7
Organization of this Guide	7
Typographical Conventions	7
Feedback and Questions	7
1 WELCOME TO THE WEB SERVICES INTERFACE	8
Service-Oriented Architecture (SOA)	8
Web Services	9
2 GETTING STARTED WITH THE WEB SERVICES INTERFACE	10
Configuration	10
How to Connect to the Web Services Interface	10
XML Return Structure	11
Starting from Scratch	13
3 USING THE WEB SERVICES INTERFACE	14
Creating New Objects	14
Configuration objects	14
Dialog objects	14
Modifying Existing Objects	15
Configuration objects	15
Dialog objects	15
Change access control lists	15
Deploying Applications	15
Deploy from file	15
Deploy from string	16
Deploy from VoiceObjects Desktop project	16
Controlling Servers and Services	16
Retrieve a list of active servers	16
Query a specific server	16
State changing commands	17
4 WSI COMMAND REFERENCE	18
Authentication and Security	18
logIn	18
logOut	18
changePassword	19
getSessionList	19
killUserSession	21
getSessionPartitioning	21



Server Management	24
ping	24
startServer	24
stopServer	24
idleServer.....	25
resumeServer	25
resetServer	25
shutdownServer.....	26
queryServer	26
addServiceToServer.....	34
removeServiceFromServer.....	34
reloadServiceList	35
createServer	35
modifyServer.....	36
deleteServer	36
getServerDef.....	37
addLicense	37
getLicense	37
getServerList.....	38
setTracingServer	39
setDBLoggingServer	39
Server Instance Management	40
startInstance	40
stopInstance	40
idleInstance.....	41
resumeInstance	41
Service Management.....	41
startService.....	41
stopService	42
idleService	42
resumeService.....	43
redeployService.....	43
restoreService	44
createService.....	44
modifyService	45
deleteService	45
getServiceDef	45
deployXDKService.....	46



validateXDKApplication	46
redeployXDKApplication	47
setTracingService	48
setDBLoggingService	48
Project and Object Management	49
exportProjectVersion	49
exportObject	50
importObject	51
getObjectDefinition	52
createProject	52
modifyProject	53
getProjectDef	53
deleteProject	54
copyProject	54
publishProjectVersion	54
modifyProjectVersion	55
getProjectVersionDef	56
deleteProjectVersion	56
upgradeProject	56
attachLibraryToVersion	57
removeLibraryFromVersion	57
getObjectList	58
getParentObjectList	60
getUnusedObjectList	61
lockObject	61
unlockObject	62
deleteObject	63
getLockedObjectList	63
getObjectCount	64
addReviewerAnnotation	65
getReviewerAnnotation	66
modifyObject	66
getDialogFlow	67
searchObjects	68
User Management	70
createUser	70
modifyUser	70
deleteUser	71



getUserDef.....	71
addUserToACL.....	72
removeUserFromACL.....	72
Log and Trace File Handling	73
getFileList	73
getFile	74
getLatestTraceFile.....	75
deleteTraceFile	76
Object Type Constants	76
Error Codes	78
A – WSI SECURITY	87



About the Web Services Guide

This *Web Services Guide* contains a detailed feature description of the **Web Services Interface** of VoiceObjects – which provides full access to all server functionalities including deployment of services, administration of server(s) and services.

The *Web Services Guide* includes an overview of the functionality provided, a description of available methods, and a detailed explanation of how to develop and deploy applications through the Web Services Interface.

Organization of this Guide

The *Web Services Guide* consists of the following chapters:

Chapter 1 – *Welcome to the Web Services Interface* – provides an overview of the functionality of the Web Services Interface of VoiceObjects.

Chapter 2 – *Getting Started with the Web Services Interface* – contains a description of the steps required to connect to and communicate with the Web Services Interface of VoiceObjects.

Chapter 3 – *Using the Web Services Interface* – describes usage examples and outlines the process of managing servers and services and deploying applications.

Chapter 4 – *WSI Command Reference* – contains a list of available WSI methods including a detailed description of parameter values and return structures.

Appendix A – *WSI Security* – contains a table of all WSI methods displaying for which user roles each method is available.

Typographical Conventions

This document contains the following typographical conventions:

<i>Italic Font</i>	Used to indicate names of applications, projects, objects, variables, files, and folders; output text, and book titles.
Bold Font	Used to indicate any screen terminology like names of windows, worksheets, editors, sections, boxes, tabs, fields, and menus.
<code>Courier New</code>	Used for grammar code.

All path specifications in this document use slashes (/) to apply to both Linux and Windows. If you work on Windows you may also use backslashes (\).

Feedback and Questions

If you have any comments on this document please send your feedback to support@voxeo.com.

If you have technical difficulties please contact your local VoiceObjects administrator or if you have a valid software support and maintenance contract in place send an email describing your problem to support@voxeo.com.



1 Welcome to the Web Services Interface

Web services were introduced by the W3C as a means to describe a standard way of communication between various software systems, independent of the platforms and frameworks used. With a Web Services Interface (WSI) software applications can exchange data over the network by using standards like XML and HTTP and thus provide interoperability and extensibility.

By providing a Web Services Interface, VoiceObjects Server gains all the aforementioned advantages in regard to extensibility and openness for other software applications. Because the interface leverages the standard technology of Web services, efficient and seamless integration into any alternative front-end, IDE or other kind of applications to create or operate communication services is guaranteed. For example, 3rd party software vendors or application service providers can use this interface to embed VoiceObjects Server into their solutions.

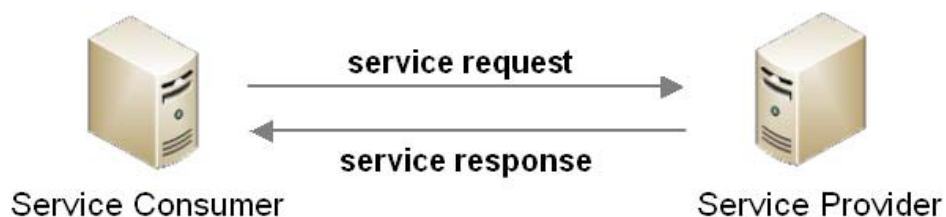
The Web Services Interface exposes a broad range of functionality of VoiceObjects Server. This includes server management functionality, such as starting, stopping or deploying services and servers, but also application development features like creating projects and individual dialog objects.

Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) describes a software architecture concept that makes IT infrastructures more responsive to business needs and helps enterprises manage heterogeneous systems landscapes. It is an open architecture for business solutions abstracting business activities or events from the underlying technical functionality. It allows enterprises to incrementally create building blocks for future business processes and with its use of abstraction makes it easy to combine and recombine functionality. The abstraction allows altering the underlying technical functionality without impacting the applications using the service.

Service-oriented architectures follow a distributed software model that uses independent Web services. These Web services can be used and reused in sequences defined by business logic to form applications that enable specific business processes.

In a service-oriented architecture the different software systems are designed as logical nodes in a network. Each node can provide a different service, being loosely coupled with each other. The service consumers can issue requests to either one of the service provider nodes, without being required to use the same operating system or programming language.





Web Services

A Web service is a self-contained and self-describing piece of functionality accessible through standard mechanisms that accepts one or more requests and returns one or more responses based on Web standards. Web services are self-contained because applications using the service only depend on the service itself, independent of the underlying technical implementation. It is self-describing, because the information how to use the service can be obtained from the service itself. Web services are loosely coupled allowing application developers to easily use existing services across the network inside and outside the corporation. Web services communicate with each other. The communication can involve either simple data passing or it can involve two or more services coordinating some activity.

For example, a phone banking service may leverage a Web service providing access to customer profile data to authorize the caller and to personalize the dialog. Further on in the dialog flow the caller activates a money transfer that again triggers a Web service providing access to the transfer functionality of the banking system. Successful operations of Web services require powerful management capabilities of the service lifecycle including design, development, deployment, operation and ongoing management and improvement. This requires an application infrastructure for building and exposing services and a powerful service infrastructure for operating the service network.



2 Getting Started with the Web Services Interface

The following chapter deals with getting the Web Services Interface up and running and explains the steps necessary to connect to the WSI and issue first commands.

i » **Note:** The Web Services Interface provides an alternative way to issue commands to the server. As such, it is recommended to not use the Command Line Interface in parallel with the Web Services Interface. Doing so may result in unexpected interactions.

Configuration

To issue commands through the Web Services Interface, the functionality must be enabled on all instances throughout a clustered environment. The VoiceObjects license key must allow WSI method calls and the user who is sending the request must be allowed to use WSI.

To enable the WSI functionality, the element `<WSIEnabled>` in the `VOServer_Configuration.xml` and `VODesktop_Configuration.xml` files must be set to `true`.

If this property is set to `false`, all method calls through the WSI will be rejected. However, retrieval of the Web Services Descriptor Language (WSDL) file and the method `ping` will still be available.

To test the current status of the WSI the method `ping` can be used.

How to Connect to the Web Services Interface

The VoiceObjects WSI starts as a dedicated container within each server instance and VoiceObjects Desktop. In a multi-instance cluster environment only the master instance executes method calls. Calling a WSI method on a non-master instance returns a HTTP redirect, specifying the URL of the master instance. This is applicable for all method calls, except for retrieval of the WSDL. The WSDL file will also be returned by non-master instances.

i » **Note:** If VoiceObjects Desktop is running, it can also serve method calls. However, a Desktop instance can only execute metadata method calls. Server management functionality can only be handled by a server instance.

The Web services session handling is local to the master instance. Consequently, if the instance that is currently master becomes unavailable, a new session must be opened on the new master instance.

A list of all methods, their parameters and the Web services endpoint is available through a Web Services Descriptor Language (WSDL) file. To access the WSDL file the following URL must be used:

```
http://<server host>:<server port>/VoiceObjects/Services/WSPProvider?wsdl
```

The Web Services Interface will dynamically generate the WSDL file and return it to the client.

Before any method can be called the user must successfully authenticate against the server and retrieve a valid session ID by using the `login` method. The WSI uses the VoiceObjects internal security and authentication mechanisms. All security relevant restrictions, based on either user role, site or access control list, will be applied through the WSI. In situations that require a high level of security, a secure socket connection (HTTPS) should be used to communicate with the WSI.



The session ID uniquely identifies this user's session and must be included in all subsequent method calls, which are then executed within the given user context.

Each method returns exactly one result value. This result value contains a predefined XML structure, which encloses general information about the command that was executed, potential warnings and errors, and the actual result, depending on the command that was executed.

XML Return Structure

The following table shows a detailed description of the generic XML return structure. Information that is specific to certain commands, such as the definition of an object, lists of files, etc., are enclosed in the `commandResult` element. A documentation of the command specific return structure can be found in the documentation of the corresponding method.

Element	Description (incl. Example)
<code><root></code>	Root element of the XML structure.
<code><creationDetails></code>	Includes general information about the creation of this content (see elements below for the details).
<code><user></code>	ID of the user who created this content, e.g. <i>voadmin</i> .
<code><role></code>	Role of the user who created this content. Can be one of: reviewer designer serviceController siteController siteAdministrator serverController serverAdministrator
<code><counter></code>	Numeric counter that is incremented for each session.
<code><timestamp></code>	The timestamp when this content was created. This information is presented in the ISO8601 format YYYY-MM-DDThh:mmTZD, e.g. 1997-07-16T19:20+01:00.
<code><serverVersion></code>	The current version of the VoiceObjects platform, e.g. 7.3.0.1530.
<code><encoding></code>	The current encoding setting of the server.
<code><repositoryName></code>	The name of the server's Metadata Repository as it was assigned during installation.
<code><repositoryID></code>	The unique ID of the server's Metadata Repository.
<code><siteID></code>	The site ID of the currently logged in user.



Element	Description (incl. Example)
</creationDetails>	Closing element for general information about the creation of this content.
<commandDetails>	Includes general information about the processed command (see elements below for the details).
<command>	Shows the processed command, e.g. <i>stopServer</i> .
<executionResult>	Allowed values are: 0 indicates that the command was processed successfully > 0 indicates that an error occurred during command processing
<message>	Shows the message that was returned by the WSI. This can either be a specific error message or SUCCESS if the command was processed successfully.
<errors>	Opening element for errors that occurred during command processing. Each error is wrapped in a single <error> tag. If no errors occurred, this element is missing.
<error>	Opening element for an error.
<errorCode>	Contains the internal error code. If no appropriate code could be found, -1 (unknown) is returned.
<errorMessage>	Contains the original error message as returned from the server.
</error>	Closing element for an error.
</errors>	Closing element for the errors collection.
<warnings>	Opening element for warnings that occurred during command processing. Each warning is wrapped in a single <warning> tag. If no warnings occurred, this element is missing.
<warning>	Opening element for a warning.
<warningCode>	Contains the internal error code of the warning. If no appropriate code could be found, -1 (unknown) is returned.
<warningMessage>	Contains the original warning message as returned from the server.
</warning>	Closing element for a warning.
</warnings>	Closing element for the warnings collection.



Element	Description (incl. Example)
<code></commandDetails></code>	Closing element for general information about the processed command.
<code><commandResult></code>	Contains the result of the specified command. Contents of this element depend on the command that was issued. For some commands, such as starting a server, this element is left empty.
<code></root></code>	Closing element for this XML structure.

Starting from Scratch

To enable the WSI functionality on top of a freshly installed VoiceObjects Metadata Repository, do the following:

1. Start VoiceObjects Desktop. VoiceObjects Desktop will start up, although no valid license is in the Metadata Repository and will allow adding licenses to the repository.
2. Execute the `login` method with the parameters `userID` set to **voadmin** and `password` set to **manager** (or the password you have selected during the installation process).
3. Use the method `addLicense` to add a valid Server and/or Desktop license to the repository. If this license allows access through the Web Services Interface, you can now issue Web services method calls to VoiceObjects Desktop.
4. Execute the method `createServer` to create a Server object in the Metadata Repository. The reference ID of the server must correspond to the logical server name that is specified in the `VOServer_Configuration.xml` file.
5. Execute the method `logout` to end your session. Then start the server. The server will start up and provide access to all available WSI methods.



3 Using the Web Services Interface

This chapter describes the actions that can be performed using the Web Services Interface (WSI). All object representations and modifications require knowledge of VoiceObjectsXML. It is therefore strongly recommended to read through the *XDK Guide* and the relevant parts of the *Object Reference* when working with the WSI.

Creating New Objects

New objects can be created through the Web Services Interface by supplying the definition in VoiceObjectsXML. Through appropriate methods, configuration objects as well as dialog objects can be built and imported into the Metadata Repository. All import and create methods provide a parameter `isURI`. If this parameter is set to `false`, the WSI expects the object definition as a character string. Alternatively you can provide the URI to a file containing the definition. In this case the parameter `isURI` must be set to `true`. Valid URIs can be specified by using either the file or the HTTP protocol:

```
file://\<hostname>\<directory>\<filename>
```

```
http://<hostname>:<port>/<directory>/<filename>
```



Note: As the server will use the given URL to retrieve the VoiceObjectsXML document, you have to make sure that the given file is accessible from the machine hosting the server.

Configuration objects

The WSI provides dedicated create methods for every type of configuration object. The create methods can be used to create new objects in the Metadata Repository. If you want to create a new object you have to make sure that no object of the same type and name, or of a different type but the same reference ID exists. If it does, a corresponding error message will be returned and the new object is not created.

To prevent overwriting of existing objects the parameter `overwrite` should be set to `false`. If this parameter is set to `true`, an existing object with the same reference ID will be deleted before the new object can be created.



Caution: If the new object cannot be created (e.g. because of an invalid XML definition) the existing object is still removed from the Metadata Repository.

Dialog objects

New dialog objects (i.e. objects of all object types within the categories Components, OSDMs, Resources, Logic, Action, Layer, and Business Task; for details see the *Object Reference*) can be imported into an existing project version by using the `importObject` method. If a naming conflict occurs during the import because an object of the same type and name already exists in the project, VoiceObjects automatically appends a time stamp to the name of the newly imported object. By setting the parameter `overwriteByName` to `true`, existing objects of the same name and type will be overwritten.



Modifying Existing Objects

Modifying an object will only change specific properties that are given in the VoiceObjectsXML definition. All other values will stay unchanged. It is not possible to change an object's GUID.

Configuration objects

There are dedicated modify methods for every type of configuration object. The modify methods will change the definition of existing configuration objects. The modified object definition must be specified in VoiceObjectsXML. Properties that are not specified in the XML will stay unchanged.

Dialog objects

To modify existing dialog objects or complete applications, the method *importObject* should be used. If your application definition contains GUIDs, objects with the same GUID will be overwritten. By setting the parameter *generateNewGuids* to *true*, new GUIDs will be created for the imported objects.

If your application does not contain GUIDs and the parameter *overwriteByName* is set to *true*, existing objects with the same name and type will be overwritten.

Change access control lists

Access control lists (ACLs) define which users are allowed to access and modify individual objects. To add or remove user entries from the ACL of an object the methods *addUserToACL* and *removeUserFromACL* can be used, respectively.

To modify the complete ACL at once, the modify methods or the method *importObject* can be used. In this case the object definition must contain a valid ACL in its VoiceObjectsXML definition. Due to security restrictions the user executing the modify or *importObject* command might not be allowed to view existing ACL entries. When importing, these entries will not be removed but only the new entries will be added.

Deploying Applications

Depending on the location of the application definition, there are several ways to deploy a service on a server.

Deploy from file

If both the service definition and the application definition are available as XDK files, the method *deployXDKService* can be used. This method creates a Service object from the given service definition and adds it to the list of services hosted on the server. The method *reloadServiceList* must be executed afterwards to reload the service list on the server.

If a service with the given VSN already exists in the Metadata Repository, the existing Service object will be overwritten. If, in addition, this service already was on the service list of the server, the method *redeployService* has to be executed instead of *reloadServiceList*.



Deploy from string

If the application definition is not persisted as a file or the server does not have access to the location of the file, the method `redeployXDKApplication` can be used to create a “volatile” service.

Before the method `redeployXDKApplication` can be used, a Service object must be created and added to the list of services hosted on the server. The start object of this service can be left empty. Once the service is shown on the list of services (returned by the `queryServer` method) the method `redeployXDKApplication` can be called. The parameter `applicationDefinition` must contain the application as a VoiceObjectsXML string.



Note: A service can only act as a volatile service if it has the option `retainOnShutdown` set to `true`. Otherwise, the method `redeployXDKApplication` will return an error message (with code 800400004).

A volatile service can only be redeployed by using the `redeployXDKApplication` command. Executing the method `redeployService` on such a service has no effect.

Deploy from VoiceObjects Desktop project

If you want to deploy a service from a VoiceObjects Desktop project, you can also do so by using the appropriate WSI methods.

When creating the service you have to specify the start object by using the following syntax:

```
project://<Project name>/<version name>/#<Start Object Reference ID>
```

This URL must point to an existing object inside the specified project version. After the service is created successfully, you need to add this service to the list of services hosted on the server and reload the services list.

Controlling Servers and Services

To control and monitor the states of your servers and services, the Web Services Interface provides dedicated methods.

Retrieve a list of active servers

To get a list of active logical servers in your environment the method `getServerList` can be used. This method returns all currently running servers that belong to the same cluster group as the Web services provider that is called.

From the returned XML structure the reference ID of one or multiple servers can be retrieved to query the servers and modify their states.

Query a specific server

The `queryServer` method can be used to get an overview of a specific server, its server instances and deployed services. The return XML shows detailed information about the currently active settings of the server, server instances and services, such as the status of e.g. DB logging and tracing. It also includes statistics information about active, finished, aborted and rejected sessions per server, service and server instance.

By using this command you can monitor the status of your environment and check the current load of your server instances and services.



State changing commands

There are several methods available to modify the states of servers and services. These commands are executed in a synchronous manner. This means that upon execution the method will poll the server until the specified state has been reached or an error occurred. For example the `idleServer` method will only return after the server has reached the idled state.

Especially when using server management methods, such as `resetServer` or `stopServer`, you should execute the method `queryServer` to make sure all server instances have reached the expected state.



4 WSI Command Reference

The following chapter lists all available commands for the Web Services Interface (WSI) along with their parameters, the returned values and a detailed description for each command.

For a description of the generic XML return structure refer to XML Return Structure in Chapter 2 – *Getting Started with the Web Services Interface*.

The return structure for information that is specific to certain commands is described in the paragraph of the corresponding method below.

Appendix A – *WSI Security* contains a table of all WSI methods displaying the availability of each method for the various user roles in VoiceObjects.

Authentication and Security

The following paragraph describes the methods that are available to maintain your and other user's sessions, to get an overview of the current activity on the WSI and the session partitioning settings.

logIn

Parameter Name	Data Type	IN / OUT	Description
userID	STRING	IN	User ID.
siteID	STRING	IN	ID of the site. This parameter is ignored at the moment.
password	STRING	IN	Password of the user.
language	STRING	IN	Specifies the locale of the session. Can be one of: en = English de = German
logInReturn	STRING	OUT	Contains a predefined XML return structure.

The *logIn* method creates a session on VoiceObjects Server with the given username and password. The parameter *language* specifies the locale of the user session. The parameter *siteID* is not used at the moment and thus can be left empty. If the authentication attempt was successful, a session ID is returned in the *commandResult* section of the result XML.

logOut

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.



Parameter Name	Data Type	IN / OUT	Description
logOutReturn	STRING	OUT	Contains a predefined XML return structure.

The *logOut* method ends the current WSI session, marks it as invalid and frees associated resources on the server. Subsequent method calls using this session ID will return an invalid session error. To ensure minimal resource usage on the server, all sessions should be explicitly closed when not used anymore.

If a WSI session is not terminated using the *logOut* method, it times out after a configurable period of time (6 hours by default).

To end sessions by other users, the method *killUserSession* can be used.

changePassword

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
oldPassword	STRING	IN	Old password of the user.
newPassword	STRING	IN	New password.
changePasswordReturn	STRING	OUT	Contains a predefined XML return structure.

The *changePassword* method changes the password of the currently logged in user. To change passwords of other users, the method *modifyUser* can be used.

In case a user needs to change the password at first login, this method needs to be executed by setting the sessionID parameter to the user ID.

getSessionList

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
getSessionListReturn	STRING	OUT	Contains a predefined XML return structure.

The *getSessionList* method retrieves a list of all currently active Web services and Desktop sessions. This command is only available for Server Administrators and Site Administrators, restricted to sessions belonging to their site.

The session list will be returned in the following XML structure, enclosed in the `<commandResults>` tag.



Element	Description (incl. Example)
<sessions>	Opening element for the session list.
<session>	Includes detailed information about each user session (see elements below for the details).
<userName>	Name of the user who opened the session, e.g. <i>VoiceObjects Administrator</i> .
<userID>	ID of the user who opened the session, e.g. <i>voadmin</i> .
<counter>	Numeric counter that is increased for each session.
<userRole>	Role of the user, e.g. <i>ServerAdmin</i> .
<userRoleID>	Numerical representation of the user role. Can be one of: 0 = Reviewer 1 = Designer 2 = Service Controller 3 = Site Controller 4 = Site Administrator 5 = Server Controller 6 = Server Administrator
<clientIP>	IP address of the client.
<lastActivity>	Timestamp representing the last activity of the user. This information is stored following the ISO8601 format YYYY-MM-DDThh:mmTZD.
<sessionType>	Type of the session. Can be one of: Desktop WebService
<sessionTypeID>	Numeric representation of the session type. Can be one of: 0 = Desktop for Web 1 = WebServices / Desktop for Eclipse
</session>	Closing element for detailed information about the user session.
</sessions>	Closing element for the session list.



killUserSession

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
targetUserName	STRING	IN	ID of the user whose session will be ended.
targetCounter	STRING	IN	Counter of the user session to be ended.
killUserSessionReturn	STRING	OUT	Contains a predefined XML return structure.

The *killUserSession* method closes the session of the specified user. The user ID and associated counter can be retrieved with the *getSessionList* method. This method cannot be used to end your own session. To do this the *logOut* method should be used.

getSessionPartitioning

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
getSessionPartitioning Return	STRING	OUT	Contains a predefined XML return structure.

The *getSessionPartitioning* method returns information about the requested and granted session partitioning settings.

The session partitioning information is returned on three levels: per server, per site and per service. The following table describes the structure of the returned XML, which is enclosed in the `<commandResult>` element.

Element	Description (incl. Example)
<code><sessionPartitioning></code>	Opening element for session partitioning information.
<code><sites></code>	Opening element for site information.
<code><site></code>	Opening element for each site.
<code><siteID></code>	The ID of the site.



Element	Description (incl. Example)
<grantedGuarantee>	The effective number of guaranteed sessions available for this site.
<grantedPool>	The effective number of floating sessions for this site.
<grantedLimit>	The maximum number of concurrent sessions actually available for this site.
<requestedGuarantee>	The requested number of guaranteed sessions for this site.
<requestedLimit>	The requested maximum number of sessions for this site.
<requestedDistribution>	Currently not applicable for sites.
<warning>	Shows a warning message for this site's session partitioning settings, enclosed in a <![CDATA[]]> tag. If no warning occurred, this element is missing.
</site>	Closing element for each site.
</sites>	Closing element for site information.
<servers>	Opening element for servers.
<server>	Opening element for each server.
<serverRefID>	The reference ID for the server.
<siteID>	The ID of the site that the server belongs to.
<grantedGuarantee>	The effective number of guaranteed sessions available for this server.
<grantedPool>	The effective number of floating sessions for this server.
<grantedLimit>	The maximum number of concurrent sessions actually available for this server.
<requestedGuarantee>	The requested number of guaranteed sessions for this server.
<requestedLimit>	The requested maximum number of sessions for this server.
<requestedDistribution>	Currently not applicable for servers.



Element	Description (incl. Example)
<warning>	Shows a warning message for this server's session partitioning settings, enclosed in a <![CDATA[]]> tag. If no warning occurred, this element is missing.
</server>	Closing element for each server.
</servers>	Closing element for servers.
<services>	Opening element for services.
<service>	Opening element for each service.
<vsn>	VoiceObjects Service Name.
<serverRefID>	Reference ID of the server that this service is hosted on.
<siteID>	ID of the site that this service belongs to.
<grantedGuarantee>	The effective number of guaranteed sessions available for this service.
<grantedPool>	The effective number of floating sessions for this service.
<grantedLimit>	The maximum number of concurrent sessions actually available for this service.
<requestedGuarantee>	The requested number of guaranteed sessions for this service.
<requestedLimit>	The requested maximum number of sessions for this service.
<requestedDistribution>	The session distribution percentage for this service. If no session distribution setting is specified '-' is returned.
<warning>	Shows a warning message for this service's session partitioning settings, enclosed in a <![CDATA[]]> tag. If no warning occurred, this element is missing.
</service>	Closing element for each service.
</services>	Closing element for services.
<sessionPartitioning>	Closing tag for session partitioning information.



Server Management

This paragraph shows a list of all available methods for retrieving a list of active servers, checking the status of the WSI, and querying and changing the state of a specific server.

ping

Parameter Name	Data Type	IN / OUT	Description
pingReturn	STRING	OUT	Contains one of: <disabled/> <noLicense/> <enabled/>

The *ping* method checks the current status of the WSI. This method is always available, even if the WSI is disabled and no session ID is required. Depending on the status of the WSI, this method will return one of the following values:
<disabled/> = The WSI is disabled in the configuration files.
<noLicense/> = The WSI is disabled due to missing license information.
<enabled/> = WSI functionality is enabled for this server.

startServer

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
startServerReturn	STRING	OUT	Contains a predefined XML return structure.

The *startServer* method starts the server with the specified reference ID.

stopServer

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
stopServerReturn	STRING	OUT	Contains a predefined XML return structure.



The *stopServer* method stops the server with the specified reference ID. All active sessions will be terminated.

idleServer

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
idleServerReturn	STRING	OUT	Contains a predefined XML return structure.

The *idleServer* method idles the server with the specified reference ID.

resumeServer

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
resumeServerReturn	STRING	OUT	Contains a predefined XML return structure.

The *resumeServer* method resumes the server with the specified reference ID. Resuming a server will change its status from idled to started.

resetServer

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
resetServerReturn	STRING	OUT	Contains a predefined XML return structure.

The *resetServer* method resets the server with the specified reference ID. The reset command stops the server, reloads the server and its associated service definitions from the Metadata Repository and restarts the server.



shutdownServer

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
shutdownServerReturn	STRING	OUT	Contains a predefined XML return structure.

The *shutdownServer* method shuts down the server. All active sessions will be terminated immediately. Once shut down, the server cannot be restarted by using the WSI.

queryServer

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server
aggregateData	BOOLEAN	IN	If set to <code>true</code> , aggregates data for services on server level.
queryServerReturn	STRING	OUT	Contains a predefined XML return structure.

The *queryServer* method returns the current status of the specified server, including information about the deployed services and server instances. If the parameter *aggregateData* is set to `false`, the service specific information is repeated for each server instance.

The XML structure contained in *queryServerReturn* looks as follows:

Element	Description (incl. Example)
<code><server></code>	Opening element for query server information.
<code><serverDetails aggregated="true/false"></code>	Opening element for server information. The attribute <i>aggregated</i> contains the value of the parameter <i>aggregateData</i> .
<code><referenceID></code>	Unique reference ID of this server e.g. <i>VOServer</i> .



Element	Description (incl. Example)
<status>	Status of this server. Allowed values are: STARTED STARTING STOPPED STOPPING IDLED REQUEST_IDLE
<startTime>	Start time of this server. This information is stored following the ISO8601 format YYYY-MM-DDThh:mmTZD, e.g. 1997-07-16T19:20+01:00.
<activeSessions>	Shows the number of currently active sessions for this server.
<finishedSessions>	Shows the number of finished sessions for this server.
<abortedSessions>	Shows the number of aborted sessions for this server.
<rejectedSessions>	Shows the number of rejected sessions for this server.
<totalSessions>	Shows the total number of sessions for this server (addition of active, finished, aborted and rejected sessions).
<tracing>	Status of tracing for this server. Allowed values are: Enabled Disabled Unavailable
<systemDBLogging>	Status of System DB logging for this server. Allowed values are: Enabled Disabled Unavailable Enabled / Not available Not licensed
<customDBLogging>	Status of Custom DB logging for this server. Allowed values are: Enabled Disabled Unavailable Enabled / Not available Not licensed



Element	Description (incl. Example)
<utteranceRecording>	Status of utterance recording for this server. Allowed values are: Enabled Disabled
<siteID>	Identifies the site this server belongs to. The site is identified either by an explicitly defined site ID or by the name of the root Site Administrator. If the server is from the system site <i>System</i> is returned.
<owner>	Shows the user ID of the user who created this server.
<locked>	Identifies the lock status of this server. Allowed values are: true false
<lockAccountID>	Shows the GUID of the user who locked this server.
<lockAccountName>	Shows the name of the user who locked this server.
<sessionLimit>	Shows the concurrent session limit for this server. The string <i>unlimited</i> is used if the number of sessions is not limited.
<sessionGuarantee>	Shows the guaranteed number of sessions for this server. 0 indicates that no sessions are guaranteed for this server.
<group>	Shows the cluster group this server belongs to.
<licenseExpireDate>	Shows the date when the license for this server expires. This information is stored following the ISO8601 format YYYY-MM-DDThh:mmTZD, e.g. 1997-07-16T19:20+01:00.
</serverDetails>	Closing element for server information.
<serverInstances>	Opening element for the list of server instances.
<serverInstance>	Opening element for server instance information.
<instanceName>	Shows the name for this server instance; this name has to be unique within the cluster.



Element	Description (incl. Example)
<configuredIP>	Shows the IP address configured for this server instance in the configuration file, e.g. 127.0.0.1.
<configuredPort>	Shows the port that is used by this server instance, e.g. 8099.
<detectedIP>	Shows the detected IP address for this server instance, e.g. 172.22.23.188.
<master>	Identifies if this server instance is currently the master instance. Allowed values are: true false
<status>	Status of this server instance. Allowed values are: STARTED STARTING STOPPED STOPPING IDLED REQUEST_IDLE
<clusterMode>	Cluster mode of this server instance. Allowed values are: SINGLETON STANDALONE NETWORK
<startTime>	Start time of this server instance. This information is stored following the ISO8601 format YYYY-MM-DDThh:mmTZD, e.g. 1997-07-16T19:20+01:00.
<activeSessions>	Shows the number of currently active sessions for this server instance.
<finishedSessions>	Shows the number of finished sessions for this server instance.
<abortedSessions>	Shows the number of aborted sessions for this server instance.
<rejectedSessions>	Shows the number of rejected sessions for this server instance.
<totalSessions>	Shows the total number of sessions for this server instance (addition of active, finished, aborted and rejected sessions).
<usedMemoryMB>	Shows the amount of the currently used memory in megabytes.



Element	Description (incl. Example)
<freeMemoryMB>	Shows the amount of the currently free memory in megabytes.
<error>	Lists error messages for this server instance, e.g. <i>Missing database connection for logging.</i>
<infostoreErr>	Lists the System DB error messages for this server instance.
<customDBErr>	Lists the Custom DB error messages for this server instance.
<systemDBLogging>	Status of System DB logging for this server instance. Allowed values are: Enabled Disabled Unavailable Enabled / Not available Not licensed
<customDBLogging>	Status of Custom DB logging for this server instance. Allowed values are: Enabled Disabled Unavailable Enabled / Not available Not licensed
<services>	If the parameter <i>aggregateData</i> is set to <i>false</i> , the list of existing services is shown here. For a list of all child elements, see the <services> element below.
</serverInstance>	Closing element for server instance information.
</serverInstances>	Closing element for the list of server instances.
<services>	Shows the list of existing services. This element is only shown if the parameter <i>aggregateData</i> is set to <i>false</i> . Otherwise this element and its child elements will be repeated for each server instance.
<service>	Opening element for service information.
<vsn>	Shows the VSN (VSN = VoiceObjects Service Name) for this service; this name has to be unique within the cluster.
<name>	Shows the name of this service.



Element	Description (incl. Example)
<source>	Used to identify if this service is deployed from file, by string or from a project: file string project
<status>	Status of this service. Allowed values are: STARTED STARTING STOPPED STOPPING IDLED REQUEST_IDLE Unavailable (only used for the Invalid Requests service) Validating Loading_Cache
<startupMode>	Startup mode of this service. Allowed values are: Automatic Manual Disabled
<startTime>	Start time of this service. This information is stored following the ISO8601 format YYYY-MM-DDThh:mmTZD, e.g. 1997-07-16T19:20+01:00.
<activeSessions>	Shows the number of currently active sessions for this service.
<finishedSessions>	Shows the number of finished sessions for this service.
<abortedSessions>	Shows the number of aborted sessions for this service.
<rejectedSessions>	Shows the number of rejected sessions for this service.
<totalSessions>	Shows the total number of sessions for this service (addition of active, finished, aborted and rejected sessions).
<tracing>	Status of tracing for this service. Allowed values are: Enabled Disabled Overridden



Element	Description (incl. Example)
<tracingANIs>	Shows a comma separated list of ANIs, e.g. +492204845196,+492204845197. Tracing will be restricted to these ANIs.
<systemDBLogging>	Status of System DB logging for this service. Allowed values are: Enabled Disabled Overridden
<customDBLogging>	Status of Custom DB logging for this service. Allowed values are: Enabled Disabled Overridden
<utteranceRecording>	Status of utterance recording for this service. Allowed values are: Enabled Disabled Overridden
<utteranceRecordingFiltering>	Show the utterance recording filtering for this service, e.g. a value of 15% means that for 15% of all logged sessions utterance recordings will be available.
<infostoreFiltering>	Shows the Infostore filtering for this service, e.g. a value of 30% means that log data will be written for 30% of all sessions.
<filterScope>	Shows the filter scope for this service. Allowed values are: InputState SystemDB SystemDBCUSTOMDB
<project>	Shows the name of the project this service belongs to, e.g. <i>Prime Insurance</i> . For an XDK service, the entry shows <i>Unavailable</i> .
<projectVersion>	Shows the project version, e.g. 1.0 or <i>Unavailable</i> .
<startObject>	Shows the name of the start object for this service, e.g. <i>PrimeInsurance Portal</i> or <i>Unavailable</i> .
<error>	Lists error messages for this service, e.g. <i>Skipping configuration of service 'PI' due to missing start object.</i>



Element	Description (incl. Example)
<activeCacheTimestamp>	Timestamp of the currently used cache of this service. This information is stored following the ISO8601 format YYYY-MM-DDThh:mmTZD, e.g. 1997-07-16T19:20+01:00.
<restoreCacheTimestamp>	Timestamp of the restorable cache of this service. This information is stored following the ISO8601 format YYYY-MM-DDThh:mmTZD, e.g. 1997-07-16T19:20+01:00.
<activeCacheSize>	Shows the number of objects of the currently used cache of this service.
<restoreCacheSize>	Shows the number of objects of the restorable cache of this service.
<siteID>	Identifies the site this service belongs to. If the service is from the system site <i>System</i> is returned.
<locked>	Identifies the lock status of this service. Allowed values are: true false
<lockAccountID>	Shows the GUID of the user who locked this service.
<lockAccountName>	Shows the name of the user who locked this service.
<sessionLimit>	Shows the concurrent session limit for this service. The string <i>unlimited</i> is used if the number of sessions is not limited.
<sessionGuarantee>	Shows the guaranteed number of sessions for this service. 0 indicates that no sessions are guaranteed for this service.
<sessionDistribution>	Shows the distribution of sessions over different services, e.g. 0.3 means that 30% of all available sessions are assigned to this service. The string <i>Disabled</i> is used if session distribution is disabled for this service.
</service>	Closing element for service information.
</services>	Closing element for the list of existing services.



Element	Description (incl. Example)
<code></server></code>	Closing element for this XML structure.

addServiceToServer

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
vsn	STRING	IN	VSN of the Service that should be added.
addServiceToServer Return	STRING	OUT	Contains a predefined XML return structure.

The *addServiceToServer* method adds the specified service to the specified server. This command changes the definition of the server in the Metadata Repository. The *reloadServiceList* method must be called to activate the changes.

removeServiceFromServer

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
vsn	STRING	IN	VSN of the service that should be removed.
removeServiceFromServerReturn	STRING	OUT	Contains a predefined XML return structure.

The *removeServiceFromServer* method removes the specified service from the specified server. The specified Service object will not be deleted. This command changes the definition of the server in the Metadata Repository. The *reloadServiceList* command must be issued to activate the changes.

**reloadServiceList**

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
reloadServiceListReturn	STRING	OUT	Contains a predefined XML return structure.

The method *reloadServiceList* reloads the list of services for the specified server from the Metadata Repository.

createServer

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverDef	STRING	IN	Server definition in VoiceObjectsXML.
overwrite	BOOLEAN	IN	If set to <code>true</code> , overwrites an existing definition.
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>serverDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
createServerReturn	STRING	OUT	Contains a predefined XML return structure.

The method *createServer* creates a new Server object in the Metadata Repository. If the parameter *isURI* is set to `false`, *serverDef* must contain the definition of the server in VoiceObjectsXML structure. If this parameter is set to `true`, *serverDef* must contain a URI pointing to a file containing the server definition. A description of the structure can be found in [Configuration Objects](#) in the *XDK Guide*. If the parameter *overwrite* is set to `true`, an already existing server definition with the same reference ID will be overwritten.



modifyServer

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the server to be modified.
serverDef	STRING	IN	Server definition in VoiceObjectsXML.
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>serverDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
modifyServerReturn	STRING	OUT	Contains a predefined XML return structure.

The *modifyServer* method changes the definition of the specified server in the Metadata Repository. If the parameter *isURI* is set to `false`, *serverDef* must contain the definition of the server in VoiceObjectsXML structure. If this parameter is set to `true`, *serverDef* must contain a URI pointing to a file, containing the server definition. By using this command all properties of the Server object except for the GUID, creation date, modification date, owner and last modifier can be modified. Properties that are not specified in the XML structure will remain unchanged.

i ▶ **Note:** Depending on the role of the user executing this command, not all properties of the Server object will be changed. For example a Site Administrator may be able to change the list of services, but not the server's name.

deleteServer

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the server to be deleted.
deleteServerReturn	STRING	OUT	Contains a predefined XML return structure.

The *deleteServer* method deletes the specified server from the Metadata Repository. This method is available to Server Administrators, Server Controllers, Site Administrators and Site Controllers.

**getServerDef**

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
getServerDefReturn	STRING	OUT	Contains a predefined XML return structure.

The method *getServerDef* returns the definition of the specified server in VoiceObjectsXML. This structure is enclosed in the `<commandResult>` tag of the general return structure.

addLicense

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
licenseType	STRING	IN	Specifies the type of the license. Can be one of: server desktop
license	STRING	IN	License in XML format.
overwrite	BOOLEAN	IN	Overwrites the definition if set to <code>true</code> .
addLicenseReturn	STRING	OUT	Contains a predefined XML return structure.

The method *addLicense* stores the specified license file in the Metadata Repository. If the parameter *overwrite* is set to `true`, an already existing license will be overwritten. The parameter *licenseType* specifies if the parameter *license* contains a Desktop or a Server license.

getLicense

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.



Parameter Name	Data Type	IN / OUT	Description
licenseType	STRING	IN	Defines the type of the license. Can be one of: server desktop
getServerLicenseReturn	STRING	OUT	Contains a predefined XML return structure.

The *getLicense* method retrieves the current Server or Desktop license and returns it in an XML structure. This structure contains the encrypted license key as well as human readable information about the licensed components. The parameter *licenseType* specifies if the parameter *license* contains a Desktop or a Server license.

getServerList

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
getServerListReturn	STRING	OUT	Contains a predefined XML return structure.

The *getServerList* method retrieves a list of currently active servers and associated server instances. The resulting XML structure is a subset of the *queryServer* command, containing only information about the server and the server instances.

Element	Description (incl. Example)
<servers>	Opening element for the xml structure.
<server referenceID=>	Opening element for the server. The attribute <i>referenceID</i> contains the reference ID of the server.
<instances>	Opening element for the instances.
<instance>	Opening element for each instance.
<instanceName>	Name of the instance.
<instanceIP>	IP address of the instance.
<instancePort>	Port of the instance.
<instanceMaster>	Indicates, if the current instance is the master instance.
</instance>	Closing element for each instance.



Element	Description (incl. Example)
<code></instances></code>	Closing element for the instances.
<code></server></code>	Closing element for the server.
<code></servers></code>	Closing element for the XML structure.

setTracingServer

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the server.
traceEnabled	BOOLEAN	IN	Sets the tracing status on the server.
setTracingServer Return	STRING	OUT	Contains a predefined XML return structure.

The *setTracingServer* method enables or disables tracing on the specified server. If the parameter *traceEnabled* is set to `true`, tracing will be activated. If it is set to `false`, tracing functionality will be unavailable.

setDBLoggingServer

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the server.
logSettings	STRING	IN	Can be one of: none system custom both
setDBLoggingServer Return	STRING	OUT	Contains a predefined XML return structure.

The *setDBLoggingServer* method enables or disables System and Custom DB logging for the specified server. Changes made here have immediate effect, but they do not change the permanent settings made in the Server object. Thus when resetting or restarting a server, the default values defined in the Server object are restored.



Depending on the value of the parameter *logSettings*, DB logging of the server will be changed to one of the following states:

none = DB logging is completely disabled for the server

system = System DB logging is enabled, Custom DB logging is disabled

custom = Custom DB logging is enabled, System DB logging is disabled

both = System and Custom DB logging are both enabled

Server Instance Management

This paragraph gives a list of available methods for modifying the states of individual server instances.

startInstance

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
serverInstanceName	STRING	IN	Specifies the name of the server instance.
startInstanceReturn	STRING	OUT	Contains a predefined XML return structure.

The *startInstance* method starts the specified server instance. The instance must be identified by its name as specified in the return structure of the *getServerList* method.

stopInstance

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
serverInstanceName	STRING	IN	Specifies the name of the server instance.
stopInstanceReturn	STRING	OUT	Contains a predefined XML return structure.

The *stopInstance* method stops the specified server instance. When stopping an instance, all active sessions will be terminated immediately.

***idleInstance***

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
serverInstanceName	STRING	IN	Specifies the name of the server instance.
idleInstanceReturn	STRING	OUT	Contains a predefined XML return structure.

The *idleInstance* method idles the specified server instance. Once idled, all currently active sessions will be processed but no new sessions will be accepted.

resumeInstance

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
serverInstanceName	STRING	IN	Specifies the name of the server instance.
resumeInstanceReturn	STRING	OUT	Contains a predefined XML return structure.

The *resumeInstance* method resumes the specified server instance.

Service Management

This paragraph shows a list of all available methods for modifying the states of services, redeploying running services and creating and modifying Service objects in the Metadata Repository.

startService

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.



Parameter Name	Data Type	IN / OUT	Description
serverRefID	STRING	IN	Reference ID of the logical server.
vsn	STRING	IN	VSN of the service.
startServiceReturn	STRING	OUT	Contains a predefined XML return structure.

The *startService* method starts the specified service on the specified server. The service that should be started has to be identified by its VoiceObjects Service Name (VSN). If this method is called on a service that is already started, it will return successfully but include a corresponding warning message.

stopService

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
vsn	STRING	IN	VSN of the service.
stopServiceReturn	STRING	OUT	Contains a predefined XML return structure.

The *stopService* method stops the specified service. The service that should be stopped has to be identified by its VoiceObjects Service Name (VSN). Stopping a service will immediately terminate all active sessions. If this method is called on a service that is already stopped, it will return successfully but include a corresponding warning message.

idleService

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
vsn	STRING	IN	VSN of the service.
idleServiceReturn	STRING	OUT	Contains a predefined XML return structure.



The *idleService* method idles the specified service. The service that should be idled has to be identified by its VoiceObjects Service Name (VSN). Idling a service does not terminate currently active sessions but stops accepting new sessions. If this method is called on a service that is already idled, it will return successfully but include a corresponding warning message.

resumeService

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
vsn	STRING	IN	VSN of the service.
resumeServiceReturn	STRING	OUT	Contains a predefined XML return structure.

The *resumeService* method resumes the specified service. The service that should be resumed has to be identified by its VoiceObjects Service Name (VSN). If this method is called on a service that is not idled, it will return an error code.

redeployService

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
vsn	STRING	IN	VSN of the service.
redeployServiceReturn	STRING	OUT	Contains a predefined XML return structure.

The *redeployService* method redeploys the specified service. A redeploy reloads the service definition from the Metadata Repository. The service that should be redeployed has to be identified by its VoiceObjects Service Name (VSN). Redeploying a service does not impact currently active sessions.



restoreService

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
vsn	STRING	IN	VSN of the service.
restoreServiceReturn	STRING	OUT	Contains a predefined XML return structure.

The *restoreService* method restores the specified service to its state before the most recent redeploy. The service that should be restored has to be identified by its VoiceObjects Service Name (VSN). Restoring a service does not impact currently active sessions.

createService

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
serviceDef	STRING	IN	Definition of the service.
overwrite	BOOLEAN	IN	Overwrites an existing definition if set to <code>true</code> .
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>serviceDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
createServiceReturn	STRING	OUT	Contains a predefined XML return structure.

The *createService* method creates a new Service object in the Metadata Repository. If the parameter *isURI* is set to `false` *serviceDef* must contain the definition of the service in VoiceObjectsXML structure. If this parameter is set to `true`, *serviceDef* must contain a URI pointing to a file containing the service definition. A description of the structure can be found in Configuration Objects in the *XDK Guide*. If the parameter *overwrite* is set to `true`, an already existing service with the same VSN will be overwritten.

**modifyService**

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
vsn	STRING	IN	VSN of the service to be modified.
vsDef	STRING	IN	Definition of the service.
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>serverDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
modifyServiceReturn	STRING	OUT	Contains a predefined XML return structure.

The *modifyService* method changes the definition of the specified Service object in the Metadata Repository. If the parameter *isURI* is set to `false` *vsDef* must contain the definition of the service in VoiceObjectsXML structure. If this parameter is set to `true`, *serviceDef* must contain a URI pointing to a file containing the service definition. A description of the structure can be found in Configuration Objects in the *XDK Guide*. By using this command all properties of the Service object except for GUID, creation date, modification date, owner and last modifier can be changed. Properties that are not specified in the XML structure will remain unchanged.

deleteService

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
vsn	STRING	IN	VSN of the service.
deleteServiceReturn	STRING	OUT	Contains a predefined XML return structure.

The *deleteService* method permanently deletes the specified service from the Metadata Repository.

getServiceDef

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.



Parameter Name	Data Type	IN / OUT	Description
vsn	STRING	IN	VSN of the service.
getServiceDefReturn	STRING	OUT	Contains a predefined XML return structure.

The method *getServiceDef* retrieves the definition of the specified service in VoiceObjectsXML. The definition is enclosed in the `<commandResults>` tag of the return structure.

deployXDKService

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
serviceDef	STRING	IN	The service definition. Can either be the definition itself or a URI pointing to a file.
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>serviceDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
deployXDKService Return	STRING	OUT	Contains a predefined XML return structure.

Deploys the specified XDK service on the specified server. This method creates a Service object from the definition specified under *serviceDef* and adds it to the list of hosted services on the server. If the service already exists, the definition will be overwritten. A redeploy command has to be executed on the service to reload the modified definition. If the service was not on the list of hosted services, the service list has to be reloaded.

validateXDKApplication

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
applicationDef	STRING	IN	A string representation of the VoiceObjectsXML file.



Parameter Name	Data Type	IN / OUT	Description
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>serverDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
validateXDKApplication Return	STRING	OUT	Contains a predefined XML return structure.

The *validateXDKApplication* validates the specified XDK application for syntactical and referential correctness.

redeployXDKApplication

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
vsn	STRING	IN	VSN of the service.
startObjectRefID	STRING	IN	Reference ID of the start object.
applicationDefinition	STRING	IN	Application definition in VoiceObjectsXML.
redeployXDKApplication Return	STRING	OUT	Contains a predefined XML return structure.

The *redeployXDKApplication* method redeploys the specified service as a volatile service. Before calling this method, the service must be on the list of hosted services for the specified server. The application definition must be supplied as a string in the *applicationDefinition* parameter. The parameter *startObject* specifies the reference ID of the start object. This reference ID must point to an object available in the attached application definition.

For debugging purposes the application definition is dumped to a file in a temporary directory (WEB-INF/dump).



Note: This method can only be used on services that have the option *retainOnShutdown* set to `true`. Otherwise, an error message (with code 800400004) is returned.

**setTracingService**

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the server.
vsn	STRING	IN	VSN of the service.
restrictANI	STRING	IN	If specified, restricts the ANIs for which tracing will be enabled.
traceEnabled	STRING	IN	Specifies the new tracing status of the service. Can be one of: true false overwrite
setTracingService Return	STRING	OUT	Contains a predefined XML return structure.

The *setTracingService* method enables or disables tracing on the specified service. If the parameter *traceEnabled* is set to `true`, tracing will be activated. If it is set to `false`, tracing functionality will be unavailable. If it is set to `overwrite`, only one trace file per service and server instance will be generated. The latest trace file (i.e. the file with the most recent timestamp) will always be deleted before a new one is created. The parameter *restrictANI* can contain a comma-separated list of ANIs. If this parameter is left empty, tracing will be enabled for all calls.

setDBLoggingService

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the server.
vsn	STRING	IN	VSN of the service.
logSettings	STRING	IN	Can be one of: none system custom both
setDBLoggingService Return	STRING	OUT	Contains a predefined XML return structure.



The *setDBLoggingService* setting enables or disables System and Custom DB logging for the specified service. Changes made here have immediate effect, but they do not change the permanent settings made in the Service object. Thus when redeploying a service or restarting a server, the default values defined in the Service object are restored. Depending on the value of the parameter *logSetting*, DB logging of the service will be changed to one of the following states:

none = DB logging is completely disabled for the server

system = System DB logging is enabled, Custom DB logging is disabled

custom = Custom DB logging is enabled, System DB logging is disabled

both = System and Custom DB logging are both enabled

Project and Object Management

exportProjectVersion

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
format	STRING	IN	Defines the format. Can be one of: ExportXML VoiceObjectsXML VoiceObjectsXMLwithoutIDs
zipped	BOOLEAN	IN	Specifies if the application should be zipped and uuencoded.
exportLib	BOOLEAN	IN	Specifies if library objects should be contained in the export.
exportProjectVersion Return	STRING	OUT	Contains a predefined XML return structure.

The *exportProjectVersion* method exports the complete application of the specified project version in the specified format. The exported application definition is enclosed in the `<commandResults>` tag of the result structure. If the parameter *zipped* is set to `true`, the exported application will be zipped, using a gzip algorithm, and uuencoded before it is sent to the client. If the *exportLib* parameter is set to `true`, all objects from attached libraries will also be included in the export. For details on exporting project versions with libraries see Chapter 3 – *Repository Browser* in the *Desktop for Eclipse Guide* or Chapter 2 – *Working with Projects* in the *Desktop for Web Guide*.

**exportObject**

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
objectRefID	STRING	IN	Reference ID of the object.
objectType	STRING	IN	Specifies the type of the object that should be exported.
format	STRING	IN	Defines the format. Can be one of: ExportXML VoiceObjectsXML VoiceObjectsXMLWithoutIDs
zipped	BOOLEAN	IN	Specifies if the VoiceObjectsXML document should be zipped and uuencoded.
exportLib	BOOLEAN	IN	Specifies if library objects should be contained in the export.
exportObjectReturn	STRING	OUT	Contains a predefined XML return structure.

The *exportObject* method exports the specified object and all linked objects in the specified format. It can also be used to export a complete application definition from a given project version. Depending on the type of the object that is exported, the parameter *objectType* needs to contain different values.

- Export a single object and its subtree from a project version
The parameters *projectName* and *versionName* must point to the respective project version. *objectRefID* must contain the reference ID of the object to be exported. *objectType* must contain the corresponding object type (see [Object Type Constants](#) for a complete list).
- Export a configuration object from the repository
The parameters *projectName*, *projectVersion* and *exportLib* are ignored and thus can be left empty. The parameter *objectRefID* must contain the reference ID of the configuration object, the *objectType* parameter must contain either "server", "service", or "user".
For configuration objects, only the format *VoiceObjectsXML* is allowed.
- Export a project definition
The parameters *versionName*, *objectRefID* and *exportLib* are ignored and thus



can be left empty. The parameter *projectName* must contain the name of the project, the *objectType* parameter must contain the string "project".

- Export a project version definition
The parameters *objectRefID* and *exportLib* are ignored and thus can be left empty. The parameters *projectName* and *versionName* must point to the respective project version. The parameter *objectType* must contain the string "projectVersion".
- Export a complete application from a project version
The parameters *projectName* and *versionName* must point to the respective project version. The parameter *objectType* must contain the string "object" to indicate a complete export of the application. In addition the parameter *objectRefID* must be left empty.

importObject

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
generateNewGuids	BOOLEAN	IN	Can either be <code>true</code> or <code>false</code> . If set to <code>true</code> , new GUIDs will be created for all imported objects.
overwriteByName	BOOLEAN	IN	Indicates whether objects with the same name should be overwritten.
objectDef	STRING	IN	Object definition in VoiceObjectsXML.
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>objectDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
importObjectReturn	STRING	OUT	Contains a predefined XML return structure.

The *importObject* method imports the specified object definition into the Metadata Repository. The parameter *objectDef* can either contain a single object or multiple objects. This method allows to import configuration objects (server, services, users, and projects) as well as dialog objects (objects contained in the categories Components, OSDMs, Resources, Logic, Actions, or Layers; for details see the *Object Reference*). However, it is not allowed to mix configuration and dialog objects in one file. For configuration objects, the parameter *projectName* and *versionName* are ignored and thus can be left empty.



getObjectDefinition

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
objectRefID	STRING	IN	Reference ID of the object.
objectType	STRING	IN	Specifies the type of the object that should be retrieved.
getObjectDefinition	STRING	OUT	Contains a predefined XML return structure.

The *getObjectDefinition* method retrieves the specified object and its directly linked objects in VoiceObjectsXML format. This method should only be used for dialog objects. There are dedicated methods (*getProjectDef*, *getServerDef*, *getServiceDef*, *getUserDef*) to retrieve the definition of configuration objects.

createProject

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
projectDef	STRING	IN	Project definition in VoiceObjectsXML.
overwrite	BOOLEAN	IN	If set to <code>true</code> , an existing object with the same name will be overwritten.
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>projectDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
createProjectReturn	STRING	OUT	Contains a predefined XML return structure.

The *createProject* method creates a new project with a first initial project version. The project definition in VoiceObjectsXML must contain at least one valid project version definition. If the parameter *isURI* is set to `false`, *projectDef* must contain the definition of the project in VoiceObjectsXML structure. If this parameter is set to `true`, *projectDef* must contain a URI pointing to a file containing the project definition. A



description of the structure can be found in Configuration Objects in the *XDK Guide*. If the parameter *overwrite* is set to `true`, an already existing project with the same name will be overwritten. By overwriting an existing project all attached project versions and application definitions are lost.

modifyProject

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
projectDef	STRING	IN	Project definition in VoiceObjectsXML.
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>serverDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
modifyProjectReturn	STRING	OUT	Contains a predefined XML return structure.

The *modifyProject* method modifies the specified project object in the Metadata Repository. If the parameter *isURI* is set to `false`, *projectDef* must contain the definition of the project in VoiceObjectsXML structure. If this parameter is set to `true`, *projectDef* must contain a URI pointing to a file containing the project definition. A description of the structure can be found in Configuration Objects in the *XDK Guide*. With this method all properties except for GUID, creation date, modification date, owner and last modifier can be changed. Additionally the type of the project (library or non-library) cannot be changed, once the project has been created.

i **Note:** This method cannot be used to modify project versions or the content of a project version. To modify the properties of a project version the method *modifyProjectVersion* should be used. The methods *importObject* or *modifyObject* provide a means to alter objects inside a project version.

Properties that are not explicitly specified in the project definition XML will remain unchanged.

getProjectDef

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
getProjectDefReturn	STRING	OUT	Contains a predefined XML return structure.



The method *getProjectDef* retrieves the definition of the specified project in VoiceObjectsXML. The definition is enclosed in the `<commandResults>` tag of the return structure.

deleteProject

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
deleteProjectReturn	STRING	OUT	Contains a predefined XML return structure.

The *deleteProject* method permanently deletes the specified project and all its project versions from the Metadata Repository.

copyProject

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project to be copied.
targetProjectName	STRING	IN	Name of the new project.
copyProjectReturn	STRING	OUT	Contains a predefined XML return structure.

The *copyProject* method copies a complete project, along with all attached project versions and applications.

publishProjectVersion

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version to be published.



Parameter Name	Data Type	IN / OUT	Description
newVersionName	STRING	IN	Name of the new project version.
createEmpty	BOOLEAN	IN	If set to <code>true</code> , the new project version will initially be empty.
publishProjectVersion Return	STRING	OUT	Contains a predefined XML return structure.

The *publishProjectVersion* method creates a new project version of the specified project. If the parameter *createEmpty* is set to `false`, the new version will contain a copy of all objects from the version specified under *versionName*. If *createEmpty* is set to `true`, the parameter *versionName* is ignored.

modifyProjectVersion

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
versionDef	STRING	IN	Definition of the project version in XML format.
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>serverDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
modifyProjectVersion Return	STRING	OUT	Contains a predefined XML return structure.

The *modifyProjectVersion* method modifies the specified project version in the Metadata Repository. If the parameter *isURI* is set to `false`, *versionDef* must contain the definition of the project version in VoiceObjectsXML structure. If this parameter is set to `true`, *versionDef* must contain a URI pointing to a file containing the project definition. A description of the structure can be found in Configuration Objects in the *XDK Guide*. With this method all properties except for the GUID, creation date, modification date, owner and last modifier can be changed. Properties that are not explicitly specified in the project version definition XML will remain unchanged.

***getProjectVersionDef***

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
getProjectVersionDef Return	STRING	OUT	Contains a predefined XML return structure.

The method *getProjectVersionDef* retrieves the definition of the specified project version in VoiceObjectsXML. The definition is enclosed in the `<commandResults>` tag of the return structure.

deleteProjectVersion

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
deleteProjectVersion Return	STRING	OUT	Contains a predefined XML return structure.

The *deleteProjectVersion* permanently deletes the specified project version from the Metadata Repository.

upgradeProject

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
projectName	STRING	IN	Name of the project.
upgradeProjectReturn	STRING	OUT	Contains a predefined XML return structure.

The *upgradeProject* method upgrades the specified project in the Metadata Repository to the current version of the server. If the specified project already is in the current



version, the method will indicate successful execution but a corresponding warning message will be returned. If the parameter *projectName* is left empty, all configuration objects (servers, services and users) will be upgraded.

attachLibraryToVersion

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
libraryName	STRING	IN	Name of the library.
libraryVersionName	STRING	IN	Name of the library version.
attachLibraryToVersion Return	STRING	OUT	Contains a predefined XML return structure.

The *attachLibraryToVersion* method attaches the specified library to the specified project version.

removeLibraryFromVersion

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
libraryName	STRING	IN	Name of the library.
libraryVersionName	STRING	IN	Name of the library version.
removeLibraryFrom Version	STRING	OUT	Contains a predefined XML return structure.

The *removeLibraryFromVersion* method removes the specified library from the specified project version.

**getObjectList**

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
objectType	STRING	IN	Specifies the object type(s) in a comma-separated list.
getObjectListReturn	STRING	OUT	Contains a predefined XML return structure.

The *getObjectList* method returns a list of all objects of the specified type available in the specified project version. Depending on the type of the objects that should be listed, the parameters need to be set in the following way:

- Get a list of dialog objects
The parameters *projectName* and *versionName* must point to the respective project version. *objectType* must contain either one object type or a comma-separated list of dialog object types (see [Object Types Constants](#) for a complete list). To get a list of all dialog objects, this parameter must contain the string "object".
- Get a list of configuration objects and/or projects
The parameters *projectName* and *projectVersion* must be left empty. The parameter *objectType* must contain the string "server", "service", "project" or "user". To get a list of all configuration objects, this parameter must contain the string "object".
- Get a list of project versions
The parameter *projectName* must point to the respective project. The parameter *versionName* must be left empty. The parameter *objectType* must contain the string "projectVersion".

The result list is enclosed in the `<commandResults>` tag and shows the following structure:

Element	Description (incl. Example)
<code><objectList></code>	Opening element for the object list.
<code><object></code>	Opening element for each object.
<code><name></code>	Name of the object, enclosed in a <code><![CDATA[]]></code> tag.
<code><guid></code>	GUID of the object.



Element	Description (incl. Example)
<refID>	Reference ID of the object.
<typeisLibrary=true/false>	Type of the object. See Object Type Constants for a complete list of available object types. The attribute <i>isLibrary</i> is only returned, if the type equals to project. It specifies if the project is a library project.
<isWriteProtected>	Specifies if the object is write-protected, e.g. for system expressions.
<layerState>	Only part of the return structure if type equals to layer. Specifies all available states for the layer.
<object>	Repeats the complete <object> element for each layer state.
</layerState>	Closing element for the layer states.
<shortDesc>	Short description of the object, enclosed in a <![CDATA[]]> tag.
<disabled>	True if the object is disabled, false otherwise.
<deleted>	True if the object is deleted, false otherwise.
<lock>	Opening element for information about locks. If the object is not locked, this element will be empty.
<lockUser>	ID of the user who locked this object.
<lockUserName>	Name of the user who locked this object.
<lockTimeStamp>	Timestamp of the lock in W3C format.
</lock>	Closing element for information about locks.
<creator>	Opening element for information about the creator of the object.
<name>	ID of the user who created this object.
<userName>	Name of the user who created this object.
<creatorTimeStamp>	Timestamp of the creation in UNIX format.
</creator>	Closing element for information about the creator of the object.



Element	Description (incl. Example)
<code><modifier></code>	Opening element for information about the last modification of the object.
<code><name></code>	ID of the user who last modified the object.
<code><userName></code>	Name of the user who last modifies the object.
<code><modificationTimestamp></code>	Timestamp of the last modification of the object.
<code></modifier></code>	Closing element for information about the last modification.
<code><version></code>	The current metadata version of the object.
<code></object></code>	Closing element for each object.
<code><libraries></code>	Opening element for objects available in attached libraries. This element is only part of the return structure if the library contains objects of the specified type.
<code><library name= version= ></code>	Opening element for each library. The attributes name and version contain the name and version of the library respectively.
<code><object></code>	Repeats the complete <code><object></code> element for each object found in the project library.
<code></library></code>	Closing element for each library.
<code></libraries></code>	Closing element for library objects.
<code></objectList></code>	Closing element for the object list.

If no object of the specified type could be found the `<objectList>` element will be empty.

getParentObjectList

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
objectType	STRING	IN	Specifies the object type.



Parameter Name	Data Type	IN / OUT	Description
objectRefID	STRING	IN	Reference ID of the object whose parents should be retrieved.
restrictedObjectType	STRING	IN	Restricts the resulting object list to only contain objects of that type.
getParentObjectList Return	STRING	OUT	Contains a predefined XML return structure.

The *getParentObjectList* method retrieves a list of objects that contain a reference to the specified object. For configuration objects the parameters *projectName* and *versionName* must be left empty. The parameter *restrictedObjectType* can be used to only show parent objects of certain types. Multiple object types can be specified by providing them in a comma-separated list. To retrieve a list of all object types this parameter must be set to "object".

The return structure of this method is the same as for the *getObjectList* method.

getUnusedObjectList

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
getUnusedObjectList Return	STRING	OUT	Contains a predefined XML return structure.

The *getUnusedObjectList* method retrieves a list of objects that are not used inside the specified project version. It is only possible to search for dialog objects. This method does not search or return objects that are inside attached libraries of project. When executed for a library project, this method only looks into the library project itself, but not into projects that use this library.

The return structure of this method is the same as for the *getObjectList* method.

lockObject

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
objectType	STRING	IN	Type of the object.



Parameter Name	Data Type	IN / OUT	Description
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
objectRefID	STRING	IN	Reference ID of the object.
lockObjectReturn	STRING	OUT	Contains a predefined XML return structure.

The *lockObject* method locks the specified object of the specified type in the Metadata Repository. Depending on the type of the object that should be locked, the parameters need to be set in the following way:

- Locking of dialog objects
The parameters *projectName* and *versionName* must point to the respective project version. *objectType* must contain the type of the object (see [Object Types Constants](#) for a complete list) and the parameter *objectRefID* must contain the reference ID of the object that should be locked.
- Locking of configuration objects
The parameters *projectName* and *projectVersion* must be left empty. The parameter *objectType* must contain the string "server", "service", or "user". The parameter *objectRefID* must contain the reference ID of the configuration object that should be locked.
- Locking projects
The parameters *versionName* and *objectRefID* must be left empty. The parameter *objectType* must contain the string "project" and the parameter *projectName* must contain the name of the project that should be locked.
- Locking project versions
The parameters *projectName* and *versionName* must point to the respective project version. The parameter *objectRefID* must be left empty. The parameter *objectType* must contain the string "projectVersion".

unlockObject

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
objectType	STRING	IN	Type of the object.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
objectRefID	STRING	IN	Reference ID of the object.
unlockObjectReturn	STRING	OUT	Contains a predefined XML return structure.



The *unlockObject* method unlocks the specified object in the Metadata Repository. The parameters of this method must be set as described for the method *lockObject*.

deleteObject

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
objectRefID	STRING	IN	Reference ID of the object.
objectType	STRING	IN	Type of the object.
deleteObjectReturn	STRING	OUT	Contains a predefined XML return structure.

The *deleteObject* method permanently deletes the specified object from the Metadata Repository. For configuration objects the parameters *projectName* and *versionName* are ignored and thus can be left empty.

getLockedObjectList

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
restrictedObjectType	STRING	IN	Restricts the resulting object list to only contain objects of that type.
getLockedObjectList Return	STRING	OUT	Contains a predefined XML return structure.

The *getLockedObjectList* method returns a list of currently locked objects. Depending on the type of the object, the parameters need to be set in the following way:

- Get a list of locked dialog objects
The parameters *projectName* and *versionName* must point to the respective project version. *restrictedObjectType* must contain either one object type or a comma-separated list of dialog object types (see [Object Types Constants](#) for a complete list). To get a list of all locked dialog objects, this parameter must contain the string "object".



- Get a list of locked configuration objects
The parameters *projectName* and *projectVersion* must be left empty. The parameter *restrictedObjectType* must contain the string "server", "service", or "user". To get a list of all locked configuration objects, this parameter must contain the string "object".
- Get a list of locked projects
The parameters *projectName* and *versionName* must be left empty. The parameter *restrictedObjectType* must contain the string "project".
- Get a list of locked project versions
The parameter *projectName* must point to the respective project. The parameter *versionName* must be left empty. The parameter *restrictedObjectType* must contain the string "projectVersion".

getObjectCount

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
objectType	STRING	IN	Restricts the resulting object list to only contain objects of that type.
getObjectCountReturn	STRING	OUT	Contains a predefined XML return structure.

The *getObjectCount* method retrieves the number of objects available per object type. Depending on the type of the object, the parameters need to be set in the following way:

- Get the number of dialog objects
The parameters *projectName* and *versionName* must point to the respective project version. *objectType* must contain either one object type or a comma-separated list of dialog object types (see [Object Types Constants](#) for a complete list). To get the number of all dialog objects, this parameter must contain the string "object".
- Get the number of configuration objects and/or projects
The parameters *projectName* and *projectVersion* must be left empty. The parameter *objectType* must contain the string "server", "service", "project", or "user". To get a list of all locked configuration objects, this parameter must contain the string "object".
- Get the number of project versions
The parameter *projectName* must point to the respective project. The parameter *versionName* must be left empty. The parameter *objectType* must contain the string "projectVersion".



If there are no objects of a specific type, the respective element will be suppressed in the return structure.

Element	Description
<objectCount>	Opening element for the object list. For each object type, one child element will be returned. The value of the element describes the number of objects available to the user.
<project isLibrary=true/false>	The number of projects is divided into library and non-library projects. Library projects are counted under the element <code><project isLibrary= true ></code> whereas non-library objects are counted under the element <code><project isLibrary=false></code> .
<server>	Contains the number of servers.
...	Contains one element per object type. For a complete list of available object types see Object Types Constants .
<libraries>	Opening element for information from attached libraries. If no libraries are attached to the project, this element and its child elements are omitted.
<library name= version=	Opening element for each attached library.
...	Contains one element per object type.
</library>	Closing element for each attached library.
</libraries>	Closing element for information about attached libraries.
</objectCount>	Closing element for the object list.

i » **Note:** Depending on the access rights of the user, the returned list might not resemble the complete list of all objects in the Metadata Repository. For example, a user with a Designer role will retrieve a count of 0 for servers, although there might exist servers.

addReviewerAnnotation

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.



Parameter Name	Data Type	IN / OUT	Description
projectName	STRING	IN	The name of the project.
versionName	STRING	IN	The name of the project version.
annotation	STRING	IN	The annotation to be added to the version.
addReviewerAnnotation Return	STRING	OUT	Contains a predefined XML return structure.

The *addReviewerAnnotation* method adds a reviewer annotation to the specified project version. Once entered, a reviewer annotation cannot be removed or edited.

getReviewerAnnotation

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	The name of the project.
versionName	STRING	IN	The name of the project version.
getReviewerAnnotation Return	STRING	OUT	Contains a predefined XML return structure.

The *getReviewerAnnotation* method returns all reviewer annotations for the specified project version. The annotations will be included in the `<commandResult>` tag and wrapped with the `<![CDATA[]]>` tag.

modifyObject

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	The name of the project
versionName	STRING	IN	The name of the project version.
objectRefID	STRING	IN	Reference ID of the object.
objectDef	STRING	IN	Object definition in VoiceObjectsXML.



Parameter Name	Data Type	IN / OUT	Description
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>objectDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
modifyObjectReturn	STRING	OUT	Contains a predefined XML return structure.

The *modifyObject* method changes the definition of the specified object in the Metadata Repository. If the parameter *isURI* is set to `false`, *objectDef* must contain the definition of the object in VoiceObjectsXML structure. If this parameter is set to `true`, *objectDef* must contain a URI pointing to a file, containing the object definition. By using this command all properties of the object except for the GUID, creation date, modification date, owner and last modifier can be modified. Properties that are not specified in the XML structure will remain unchanged.

getDialogFlow

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	The name of the project.
versionName	STRING	IN	The name of the project version.
objectRefID	STRING	IN	Reference ID of the object.
objectType	STRING	IN	Type of the object.
startLevel	STRING	IN	start level – parameter currently not used.
endLevel	STRING	IN	end level – parameter currently not use.
getDialogFlowReturn	STRING	OUT	Contains a predefined XML return structure.

The method *getDialogFlow* exports the specified object and all linked objects in VoiceObjectsXML. The export can be used to display the dialog flow of the selected start object. Information that is not required for the dialog flow is not part of the return structure. For a complete export of the object and its children the method *exportObject* should be used.

**searchObjects**

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	The name of the project.
versionName	STRING	IN	The name of the project version.
searchCondition	STRING	IN	The search condition.
getDialogFlowReturn	STRING	OUT	Contains a predefined XML return structure.

The *searchObjects* method performs a search in the specified project version and its attached libraries and returns a list of objects matching the search criteria. The return structure is the same as for the method *getObjectList*. This method cannot be used to search for configuration objects.

The search restrictions should be supplied through the parameter *searchCondition* represented by the following XML structure:

Attribute	Description
<searchCondition>	Opening element for the search filter.
<contains exactMatch= field=>	Specifies a string that should be searched for. The attribute <i>field</i> specifies the object property that should be searched on. It can contain one of the following values: all comment errorDescription keywords name objectID referenceID shortDescription versionDescription The optional attribute <i>exactMatch</i> specifies if the search should only return objects matching the exact specified string. It can either be <i>true</i> or <i>false</i> . If this attribute is omitted, it defaults to <i>false</i> .
<modificationDate qualification= unixTS= >	Opening element for filtering on modification date. The format should follow the ISO8601 standard YYYY-MM-DDThh:mm:ssTZD. The attribute <i>qualification</i> can be one of: after before



Attribute	Description
	between The attribute <i>unixTS</i> can contain the modification date in Unix Time format (milliseconds since Jan 1 st , 1970). If both the attribute value and the element value contain a date, the attribute value has precedence.
<and unixTS= />	Required element when specifying qualification="between". The attribute <i>unixTS</i> can contain the modification date in Unix Time format (milliseconds since Jan 1 st , 1970). If both the attribute value and the element value contain a date, the attribute value has precedence.
</modificationDate>	Closing element for filtering on modification date.
<creationDate qualification= unixTS= >	Opening element for filtering on creation date. The format should follow the ISO8601 standard YYYY-MM-DDThh:mm:ssTZD. The attribute <i>qualification</i> can be one of: after before between The attribute <i>unixTS</i> can contain the creation date in Unix Time format (milliseconds since Jan 1 st , 1970). If both the attribute value and the element value contain a date, the attribute value has precedence.
<and unixTS />	Required element when specifying qualification="between". The attribute <i>unixTS</i> can contain the creation date in Unix Time format (milliseconds since Jan 1 st , 1970). If both the attribute value and the element value contain a date, the attribute value has precedence.
</creationDate>	Closing element for filtering on creation date.
<objectType>	Can be one of all available object types. Multiple object types can be specified by comma-separating them.
<objectStatus>	Can be one of: locked = all objects that are locked. lockedByMe = all object that are locked by the yourself. lockedByOthers = all objects that are locked by other users.



Attribute	Description
<code></searchCondition></code>	Closing element for the search filter.

All root elements of this structure are optional. If one element is not specified in the XML, no restriction on that criterion will be taken. For example, only supplying the `<searchCondition />` element will return an unfiltered list of all objects.

User Management

createUser

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
userDef	STRING	IN	User definition in VoiceObjectsXML.
overwrite	BOOLEAN	IN	If set to <code>true</code> , overwrites existing object.
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>serverDef</i> is interpreted as a URI pointing to a VoiceObjectsXML file.
createUserReturn	STRING	OUT	Contains a predefined XML return structure.

The *createUser* method creates a new user with the specified definition in the Metadata Repository. If the parameter *isURI* is set to `false`, *userDef* must contain the definition of the user in VoiceObjectsXML structure. If this parameter is set to `true`, *userDef* must contain a URI pointing to a file containing the user definition. A description of the structure can be found in Configuration Objects in the *XDK Guide*. If the parameter *overwrite* is set to `true`, an already existing user with the same user ID will be overwritten. By overwriting, the existing user will be marked as deleted.

modifyUser

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
userID	STRING	IN	User ID of the user.



Parameter Name	Data Type	IN / OUT	Description
userDef	STRING	IN	User definition in VoiceObjectsXML.
isURI	BOOLEAN	IN	If set to <code>true</code> , the parameter <i>userDef</i> is interpreted as a URI pointing to a VoiceObjectsXMLX file.
modifyUserReturn	STRING	OUT	Contains a predefined XML return structure.

The *modifyUser* method modifies the specified User object in the Metadata Repository. If the parameter *isURI* is set to `false`, *userDef* must contain the definition of the project in VoiceObjectsXML structure. If this parameter is set to `true`, *userDef* must contain a URI pointing to a file containing the user definition. A description of the structure can be found in Configuration Objects in the *XDK Guide*. With this method all properties except for GUID, creation date, modification date, owner and last modifier can be changed. Properties that are not explicitly specified in the user definition XML will remain unchanged.

Due to security restrictions, this method cannot be used to modify your own user object.

deleteUser

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
userID	STRING	IN	User ID of the user.
deleteUserReturn	STRING	OUT	Contains a predefined XML return structure.

The *deleteUser* method permanently deletes the specified user from the Metadata Repository.

getUserDef

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
userID	STRING	IN	User ID of the user.
getUserDefReturn	STRING	OUT	The definition of the user.



The method *getUserDef* retrieves the definition of the specified user in VoiceObjectsXML. The definition is enclosed in the `<commandResults>` tag of the return structure.

addUserToACL

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
userID	STRING	IN	User ID of the user.
objectRefID	STRING	IN	Reference ID of the object.
objectType	STRING	IN	Specifies the type of the object. Can be one of: project server service user
addUserToACLReturn	STRING	OUT	Contains a predefined XML return structure.

The *addUserToACL* method adds the specified user to the access control list of the specified object. Depending on the type of the object on which the ACL should be modified, the parameters need to be set in the following way.

- Add a user to ACL of projects
The parameter *projectName* must contain the name of the project. The parameter *objectType* must contain the string "project" and *userID* must contain the ID of the user who should be added to the ACL. The parameters *versionName* and *objectRefID* must be left empty.
- Add a user to ACL of servers, services and users
The parameter *objectRefID* must contain the reference ID of the object of which the ACL should be modified and *objectType* must contain one of "server", "service" or "user" respectively. The parameter *userID* must contain the ID of the user who should be added to the ACL. The parameters *projectName* and *versionName* must be left empty.

removeUserFromACL

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.



Parameter Name	Data Type	IN / OUT	Description
projectName	STRING	IN	Name of the project.
versionName	STRING	IN	Name of the project version.
userID	STRING	IN	User ID of the user.
objectRefID	STRING	IN	Reference ID of the object.
objectType	STRING	IN	Specifies the type of the object. Can be one of: project server service user
removeUserFromACL Return	STRING	OUT	Contains a predefined XML return structure.

The *removeUserFromACL* method removes the specified user from the access control list of the specified object. The parameters of this method must be set as described for the method *addUserToACL*.

Log and Trace File Handling

getFileList

Parameter Name	Data Type	IN / OUT	Description
sessionID	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
fileType	STRING	IN	Defines the type of the files to list. Can be one of: server service desktop tracing
getFileListReturn	STRING	OUT	Contains a predefined XML return structure.

The *getFileList* method retrieves a list of available log or trace files from the server. The parameter *fileType* specifies the type of the files. The list of files is enclosed in the `<commandResults>` tag. The following table lists the detailed structure:



Element	Description (incl. Example)
<fileSet>	Opening element for the file list.
<instanceFileSet>	Opening element for the file list per server instance.
<instance>	Opening element for each server instance.
<instanceName>	The name of the server instance.
</instance>	Closing element for each server instance.
<files>	Opening element for the list of files.
<file>	Opening element for each file.
<name>	Name of the file.
<date>	Last modified timestamp of the file in W3C format.
<size>	Size of the file in bytes.
<path>	File path, starting from the OVAPRoot.
</file>	Closing element for each file.
</files>	Closing element for the list of files.
</instanceFileSet>	Closing element for the file list per server instance.
</fileSet>	Closing element for the file list.

This structure lists all files for each server instance. In a multi-instance environment the element *instanceFileSet* is repeated for each available server instance.

getFile

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
instanceName	STRING	IN	The name of the server instance.
fileName	STRING	IN	The name of the file.



Parameter Name	Data Type	IN / OUT	Description
fileType	STRING	IN	The type of the file that should be retrieved. Can be one of: server service desktop tracing
zipped	BOOLEAN	IN	Specifies if contents of the file should be zipped and uuencoded.
getFileReturn	STRING	OUT	Contains a predefined XML return structure.

The *getFile* method retrieves the contents of the specified file from the server. The contents of the file are uuencoded by using the Base64 encoding (RFC 2050) and enclosed in the `<commandResult>` tag of the result structure. If the parameter *zipped* is set to `true`, the file contents are zipped before encoding using the gzip algorithm.

getLatestTraceFile

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>login</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
instanceName	STRING	IN	The name of the server instance.
vsn	STRING	IN	VSN of the service.
zipped	BOOLEAN	IN	Specifies if contents of the file should be zipped and uuencoded.
getLatestTraceFile Return	STRING	OUT	Contains a predefined XML return structure.

The *getLatestTraceFile* method retrieves the latest trace file (i.e. the file with the most recent timestamp) from the specified server instance. The parameter *vsn* specifies the service for which the trace file should be retrieved.

**deleteTraceFile**

Parameter Name	Data Type	IN / OUT	Description
sessionId	STRING	IN	Session ID returned by the <i>logIn</i> method.
serverRefID	STRING	IN	Reference ID of the logical server.
instanceName	STRING	IN	The name of the server instance.
fileNames	STRING	IN	The name of the trace file to be deleted. Can also be a comma-separated list for deleting multiple files at once.
deleteTraceFile Return	STRING	OUT	Contains a predefined XML return structure.

The *deleteTraceFile* method tries to delete the specified trace file from the specified server instance. This method can also be used to delete multiple trace files at once. If at least one of the specified trace files was successfully deleted, the method's return structure indicates success. If problems occurred during deletion, they will be listed in the <warnings> element of the return structure.

Object Type Constants

The following table lists all possible values for the *objectType* parameter, including a description. Refer to the description of each method to find out which values are allowed for which method:

Value	Object Category	Description
server	Configuration	Server object
service	Configuration	Service object
user	Configuration	User object
project	Configuration	Project
projectVersion	Configuration	Project Version
module	Dialog (Components)	Module object
input	Dialog (Components)	Input object
output	Dialog (Components)	Output object



Value	Object Category	Description
sequence	Dialog (Components)	Sequence object
menu	Dialog (Components)	Menu object
confirmation	Dialog (Components)	Confirmation object
list	Dialog (Components)	List object
osdmAlphanumeric	Dialog (OSDMs)	OSDM Alphanumeric object
osdmCreditCard Expiration	Dialog (OSDMs)	OSDM Credit Card Expiration object
osdmCreditCard Number	Dialog (OSDMs)	OSDM Credit Card Number object
osdmCurrency	Dialog (OSDMs)	OSDM Currency object
osdmCustomContext	Dialog (OSDMs)	OSDM Custom Context object
osdmDate	Dialog (OSDMs)	OSDM Date object
osdmDigits	Dialog (OSDMs)	OSDM Digits object
osdmNumber	Dialog (OSDMs)	OSDM Number object
osdmPhoneNumber	Dialog (OSDMs)	OSDM Phone Number object
osdmPostalCode	Dialog (OSDMs)	OSDM Postal Code object
osdmSocialSecurity Number	Dialog (OSDMs)	OSDM Social Security Number object
osdmTime	Dialog (OSDMs)	OSDM Time object
osdmYesNo	Dialog (OSDMs)	OSDM Yes/No object
audio	Dialog (Resources)	Audio object
video	Dialog (Resources)	Video object
format	Dialog (Resources)	Format object
silence	Dialog (Resources)	Silence object
grammar	Dialog (Resources)	Grammar object
connector	Dialog (Resources)	Connector object
script	Dialog (Resources)	Script object



Value	Object Category	Description
plugin	Dialog (Resources)	Plug-in object
log	Dialog (Resources)	Log object
resourceLocator	Dialog (Resources)	Resource Locator object
variable	Dialog (Logic)	Variable object
collection	Dialog (Logic)	Collection object
expression	Dialog (Logic)	Expression object
if	Dialog (Logic)	If object
case	Dialog (Logic)	Case object
loop	Dialog (Logic)	Loop object
goto	Dialog (Logic)	Goto object
hyperlink	Dialog (Actions)	Hyperlink object
pause	Dialog (Actions)	Pause object
recording	Dialog (Actions)	Recording object
transfer	Dialog (Actions)	Transfer object
exit	Dialog (Actions)	Exit object
layer	Dialog (Layers)	Layer object
businessTask	Dialog (Business Task)	Business Task object
object	Dialog / Configuration	Placeholder for all object types that are allowed in the parameter.

Error Codes

The following table gives a list of all possible error codes and their corresponding descriptions.

Error Code	Error Message	Description
0	Success	Command has been executed successfully.
100100001	No such object	Object does not exist.



Error Code	Error Message	Description
100600001	Internal error	Internal error.
100600002	No such object	Object does not exist.
100600003	Project version already present	Project version already exists.
100600004	No such project version	Project version doesn't exist.
100600005	No such project	Project doesn't exist.
100600006	Already unlocked	Object is already unlocked.
100600007	Already locked by 'user'	Object is already locked by 'user'.
100600008	You cannot lock or unlock your own user object.	Existing lock couldn't be altered.
100600009	Could not create system expressions	Could not create system expressions.
100600010	Failed to create project version	Failed to create project version.
100600011	Failed to save objects	Failed to save objects.
100600012	Failed to delete object	Failed to delete object.
100600013	Failed to get user	Failed to get user.
400000001	Failed to load the license from the metadata repository	License could not be retrieved from the Metadata Repository.
500100001	Invalid response 'response' for 'command'	Invalid response 'response' for 'command'.
500300001	Failed to retrieve file	Failed to retrieve file
500300002	Failed to get log file content	Failed to get the contents of a log file.
500300003	No information available for instance 'instance' on logical server 'server'	No information available for instance 'instance' on logical server 'server'.
600000001	Invalid login	User ID and/or password are wrong.
600000002	Wrong password	Wrong current password when changing the password.



Error Code	Error Message	Description
600000003	Insufficient privileges	Insufficient privileges for the requested command.
600000004	Invalid session ID	No session exists for the given ID.
600000005	User account is disabled	User account is disabled.
600000006	No license available	Either there is no license for the site or the Desktop available.
600000007	Web Services Interface not licensed	Web Services Interface not licensed.
600000008	DB Connection not available	DB Connection not available.
600000010	Invalid Session ID	The used session ID is invalid.
600100001	Access denied	Access denied.
600100003	You have insufficient privileges to access this object type.	You have insufficient privileges to access this object type.
800000001	Failed to create service cache	No cache could be created in the Object Server.
800000002	Exception: No response available	No response was received from all other cluster members.
800000003	Unknown error	The exact error message could not be retrieved.
800000004	Session ID missing	Generic error when the session ID is not available.
800000005	The server is not in a state that allows the execution of server commands	The server is not in a state that allows the execution of server commands.
800000006	No instance with name 'instance' available	No instance with name 'instance' available.
800000007	No restore cache available for service 'service'	No restore cache available for service 'service'.
800000008	The service is not in a state that allows the execution of service commands.	The service is not in a state that allows the execution of service commands.



Error Code	Error Message	Description
80000009	Invalid instance name	Invalid instance name.
80000010	Timeout occurred, stop waiting for event	Timeout occurred, stop waiting for event.
80000011	Service is in error state, aborting command	Service is in error state, aborting command.
80000012	Service is in wrong state, aborting command	Service is in wrong state, aborting command.
80000013	Service is in wrong state, aborting command	Service is in wrong state, aborting command.
80000014	No such server available	No such server available.
800100001	Cannot stop service due to wrong server state	Service can only be stopped in matching server states.
800100002	Cannot start service due to wrong server state	Service can only be started in matching server states.
800100003	Failed to deploy service	Failed to deploy service.
800100005	The own user session cannot be killed	The own user session cannot be killed.
800100006	No valid session available for user 'user'	Either the user is already logged out or his session has been killed.
800100007	No user session available for killing	The session is from a different site.
800100008	Cannot change the status of a stopped service to idle	Cannot change the status of a stopped service to idle.
800100009	No such object	Object does not exist.
800200001	No such server	Server does not exist.
800200002	No such service	Service does not exist.
800200003	No such project	Project does not exist.
800200004	No such project version	Project version does not exist.
800200005	The project has been created with a previous version of VoiceObjects. Please upgrade to the	Upgrade warning.



Error Code	Error Message	Description
	latest version	
800200006	Error while adding the license to the repository	There is no license object available in the metadata and it could also not be created.
800200007	Could not load the license object from the repository	Could not load the license object from the repository.
800200008	Could not save the license as overwrite is false	Could not save the license as overwrite is false.
800200009	License is invalid	License is invalid.
800200011	Project name missing	Missing project name in the project related commands.
800200012	Project version name missing	Missing project version name in the project version related commands
800200013	Reviewer annotation missing	No reviewer annotation for the annotate command.
800200014	No such project version	Found no project version with the given name.
800200015	Failed to lookup required component(s)	General service error when looking up a system component.
800200016	Object type for filtering missing	Used when the filter criteria is missing, e.g. in getObjectList.
800200017	VoiceObjectsXML export for service 'service' returned an empty string	VoiceObjectsXML export for service 'service' returned an empty string.
800200018	Error occurred while exporting the service definition	General error when exporting a service.
800200019	Insufficient privileges to publish the project version	Insufficient privileges to publish the project version.
800200020	Error while exporting the user definition	Error while exporting the user definition.
800200021	Insufficient privileges to change the password	Insufficient privileges to change the password.



Error Code	Error Message	Description
800200022	Existing password is not correct	Existing password is not correct.
800200023	Failed to change the password	Failed to change the password.
800200024	The specified user does not exist	The specified user does not exist.
800200025	Error while modifying the user definition	Thrown when the user definition could not be modified.
800200026	Error while modifying the server definition	Thrown when the server definition could not be modified.
800200027	Error while modifying the service definition	Thrown when the service definition could not be modified.
800200028	Error while modifying the project definition	Thrown when the project definition could not be modified.
800200030	Error while modifying the project version definition	Thrown when the project version definition could not be modified.
800200031	Error while modifying the object definition	Thrown when the object definition could not be modified.
800200032	Could not create a configuration set	Could not create a configuration set.
800200033	Service already linked to the server	Service already linked to the server.
800200034	Internal error	Internal error.
800200035	No configuration set available for the server	No configuration set available for the server.
800200036	No service for linking available	No service for linking available.
800200037	Service could not be saved:	Used in case when the service could not be saved while deploying an XDK application.
800200038	Object type should not be null or empty	Object type should not be null or empty.
800200039	No upgrade required	No upgrade is required.
800200040	Error while upgrading	An upgrade has not been successful.



Error Code	Error Message	Description
800200041	Standard Export requires project name and version	Standard Export requires project name and version.
800200042	Error while exporting the project version content	Error while exporting the project version content.
800200043	Invalid format specified for export. Valid formats are ExportXML, VoiceObjectsXML, VoiceObjectsXMLWithoutIDs	Invalid format specified for export. Valid formats are ExportXML, VoiceObjectsXML, VoiceObjectsXMLWithoutIDs.
800200044	Error while zipping export result	Error while zipping export result.
800200045	Export returned an empty string	Export returned an empty string.
800200046	Error occurred while exporting the object definition	Error occurred while exporting the object definition.
800200047	Reference ID missing	Reference ID missing.
800200048	Insufficient privileges to export the server definition	Insufficient privileges to export the server definition.
800200049	Cannot attach library to another library	Cannot attach library to another library.
800200050	Library is already attached to the project	Library is already attached to the project.
800200051	Incompatible library version 'version' upgrade necessary	Incompatible library version 'version' upgrade necessary.
800200052	Referenced project is not a library	Referenced project is not a library.
800200053	Either library with name 'library' is not available or is not a valid library	Either library with name 'library' is not available or is not a valid library.
800200054	No such library	Library does not exist.
800200055	Failed to attach library	Failed to attach library.
800200056	Invalid license type	Invalid license type.



Error Code	Error Message	Description
800200057	Object type missing	Object type missing.
800200058	Object with name/reference id not found in repository	Object with name/reference ID not found in repository.
800200059	Error while copying project	Error while copying project.
800200060	New project cannot be empty	New project cannot be empty.
800200061	Insufficient privileges to create or copy project	Insufficient privileges to create or copy project.
800200062	Error while saving objects	Error while saving objects.
800200063	Error while checking user privileges	Error while checking user privileges.
800200064	Invalid import/export path specified, the site path is 'path'	Invalid import/export path specified, the site path is 'path'.
800200065	Unable to delete version 'version'	Unable to delete version 'version'.
800200066	No project version 'version' for project 'project'	No project version 'version' for project 'project'.
800200067	Unsupported search type	Unsupported search type.
800200068	User ID missing	User ID missing.
800200069	Adding users to a project version is not supported	Adding users to a project version is not supported.
800200070	Object ACL could not be modified	Object ACL could not be modified.
800200071	No such user	User does not exist.
800200072	Only users can be added to the ACL	Only users can be added to the ACL.
800200073	No object of type 'type' found with reference ID 'refID'	No object of type 'type' found with reference ID 'refID'.



Error Code	Error Message	Description
800200074	Logical server name not found	The specified logical server name was not found.
800200075	Libraries cannot be published	Library projects cannot be published.
800200076	Cannot resolve references in import	References in an import file point to non-existing objects.
800200077	Cannot remove library since there are objects referenced from the project	Library project cannot be deleted while there are projects referencing it.
800200078	Library is not attached to project	The library is not attached to the project, so no objects can be referenced.
800200079	Failed to import content	Generic import error.
800400001	VSN must be specified for this command	Warning given when any service command called without VSN.
800400002	Log settings cannot be empty	Warning message issued when calling setDBLoggingService.
800400003	Invalid log settings for service. Supported log settings for service are 'system', 'custom', 'both', 'none'	Warning issue incase some invalid string is entered as log settings.
800400004	redeployXDKApplication can only be called on services that have the option retainOnShutdown set to true	redeployXDKApplication can only be called on services that have the option retainOnShutdown set to true.



A – WSI Security

The following table lists all available WSI methods and shows the availability of each method for the various user roles in VoiceObjects.

Method	Server Administrator	Server Controller	Service Controller	Site Administrator	Site Controller	Designer	User Manager	Reviewer	Observer
<i>login</i>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<i>logout</i>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<i>changePassword</i>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<i>getSessionList</i>	✓	✗	✗	Site	✗	✗	✗	✗	✓
<i>killUserSession</i>	✓	✗	✗	Site	✗	✗	✗	✗	✗
<i>getSLASummary</i>	✓	✓	✗	Site	Site	✗	✗	✗	✓
<i>writeAuditTrail</i>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<i>ping</i>	✓	✓	✓	✓	✓	✓	✓	✓	✓
<i>startServer</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗
<i>stopServer</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗
<i>idleServer</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗
<i>resumeServer</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗



Method	Server Administrator	Server Controller	Service Controller	Site Administrator	Site Controller	Designer	User Manager	Reviewer	Observer
<i>resetServer</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗
<i>shutdownServer</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗
<i>queryServer</i>	✓	✓	ACL	Site / ACL*	Site / ACL*	✗	✗	✗	✓
<i>addServiceToServer</i>	✓	✓	✗	Site / ACL	Site / ACL	✗	✗	✗	✗
<i>removeServiceFromServer</i>	✓	✓	✗	Site / ACL	Site / ACL	✗	✗	✗	✗
<i>reloadServiceList</i>	✓	✓	✗	Site / ACL*	Site / ACL*	✗	✗	✗	✗
<i>createServer</i>	✓	✓	✗	✓	✓	✗	✗	✗	✗
<i>modifyServer</i>	✓	✓	✗	Site / ACL	Site / ACL	✗	✗	✗	✗
<i>deleteServer</i>	✓	✓	✗	Site / ACL	Site / ACL	✗	✗	✗	✗
<i>getServerDef</i>	✓	✓	✗	Site / ACL	Site / ACL	✗	✗	✗	✗
<i>addLicense</i>	✓	✓	✗	✗	✗	✗	✗	✗	✗
<i>getLicense</i>	✓	✓	✗	✗	✗	✗	✗	✗	✗
<i>getServerList</i>	✓	✓	✗	Site / ACL*	Site / ACL*	✗	✗	✗	✓
<i>startInstance</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗



Method	Server Administrator	Server Controller	Service Controller	Site Administrator	Site Controller	Designer	User Manager	Reviewer	Observer
<i>stopInstance</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗
<i>idleInstance</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗
<i>resumeInstance</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗
<i>startService</i>	✓	✓	ACL	Site	Site	✗	✗	✗	✗
<i>stopService</i>	✓	✓	ACL	Site	Site	✗	✗	✗	✗
<i>idleService</i>	✓	✓	ACL	Site	Site	✗	✗	✗	✗
<i>resumeService</i>	✓	✓	ACL	Site	Site	✗	✗	✗	✗
<i>redeployService</i>	✓	✓	ACL	Site	Site	✗	✗	✗	✗
<i>restoreService</i>	✓	✓	ACL	Site	Site	✗	✗	✗	✗
<i>createService</i>	✓	✓	✓	Site	Site	✗	✗	✗	✗
<i>modifyService</i>	✓	✓	ACL	Site	Site	✗	✗	✗	✗
<i>deleteService</i>	✓	✓	ACL	Site	Site	✗	✗	✗	✗
<i>getServiceDef</i>	✓	✓	ACL	Site	Site	✗	✗	✗	✓
<i>exportProjectVersion</i>	✓	✗	✗	Site	✗	ACL	✗	ACL	✗
<i>exportObject</i>	✓	✓	✓	Site	Site	ACL	✗	ACL	✗



Method	Server Administrator	Server Controller	Service Controller	Site Administrator	Site Controller	Designer	User Manager	Reviewer	Observer
<i>importObject</i>	✓	✓	ACL	Site	Site	ACL		✗	✗
<i>createProject</i>	✓	✗	✓	Site	✗	✓	✗	✗	✗
<i>modifyProject</i>	✓	✓	ACL	Site	Site	ACL	✗	✗	✗
<i>getProjectDef</i>	✓	✓	ACL	Site	Site	ACL	✗	✗	✓
<i>deleteProject</i>	✓	✓	ACL	Site	Site	ACL	✗	✗	✗
<i>copyProject</i>	✓	✓	ACL	Site	Site	ACL	✗	✗	✗
<i>publishProjectVersion</i>	✓	✓	ACL	Site	Site	ACL	✗	✗	✗
<i>modifyProjectVersion</i>	✓	✓	ACL	Site	Site	ACL	✗	✗	✗
<i>getProjectVersionDef</i>	✓	✓	ACL	Site	Site	ACL	✗	✗	✓
<i>deleteProjectVersion</i>	✓	✗	ACL	Site	Site	ACL	✗	✗	✗
<i>upgradeProject</i>	✓	✗	ACL	Site	✗	ACL	✗	✗	✗
<i>attachLibraryToVersion</i>	✓	✗	ACL	Site	✗	ACL	✗	✗	✗
<i>removeLibraryFromVersion</i>	✓	✗	ACL	Site	✗	ACL	✗	✗	✗
<i>deployXDKService</i>	✓	✓	✗	Site / ACL *	Site / ACL*	✗	✗	✗	✗
<i>validateXDKApplication</i>	✓	✓	✓	✓	✓	✓	✗	✗	✗



Method	Server Administrator	Server Controller	Service Controller	Site Administrator	Site Controller	Designer	User Manager	Reviewer	Observer
<i>redeployXDKApplication</i>	✓	✓	ACL	Site	Site	✗	✗	✗	✗
<i>getObjectList</i>	✓	✓	ACL	Site	Site	ACL		✗	✓
<i>getParentObjectList</i>	✓	✓	ACL	Site	Site	ACL	✗	✗	✓
<i>lockObject</i>	✓	✓	ACL	Site	Site	ACL	✓	✗	✗
<i>unlockObject</i>	✓	✓	ACL	Site	Site	ACL	✓	✗	✗
<i>deleteObject</i>	✓	✓	ACL	Site	Site	ACL	✓	✗	✗
<i>getLockedObjectList</i>	✓	✓	ACL	Site	Site	ACL	✓	✗	✗
<i>setTracingServer</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗
<i>setTracingService</i>	✓	✓	✓	Site	Site	✗	✗	✗	✗
<i>setDBLoggingServer</i>	✓	✓	✗	Site	Site	✗	✗	✗	✗
<i>setDBLoggingService</i>	✓	✓	✓	Site	Site	✗	✗	✗	✗
<i>addReviewerAnnotation</i>	✗	✗	✗	✗	✗	✗	✗	✓	✗
<i>getReviewerAnnotation</i>	✓	✓	✓	Site	Site	✓	✗	✓	✗
<i>createUser</i>	✓	✗	✗	✓	✗	✗	✓	✗	✗
<i>modifyUser</i>	✓	✗	✗	Site	✗	ACL	✓	✗	✗



Method	Server Administrator	Server Controller	Service Controller	Site Administrator	Site Controller	Designer	User Manager	Reviewer	Observer
<i>deleteUser</i>	✓	✗	✗	Site	✗	ACL	✓	✗	✗
<i>getUserDef</i>	✓	✗	✗	Site	✗	ACL	✓	✗	✗
<i>addUserToACL</i>	✓	✓	✗	Site	Site	ACL	✗	✗	✗
<i>removeUserFromACL</i>	✓	✓	✗	Site	Site	ACL	✗	✗	✗
<i>getFileList</i>	✓	✓	✓	Site	Site	✗	✗	✗	✓
<i>getFile</i>	✓	✓	✓	Site	Site	✗	✗	✗	✓
<i>deleteTraceFile</i>	✓	✓	✓	Site	Site	✓	✗	✗	✗
<i>getLatestTraceFile</i>	✓	✓	✓	Site	Site	✓	✗	✗	✗

✓	=	Available
✗	=	Not available
Site	=	Available to all objects in the site
Site / ACL	=	Available to objects in the site, but only if the user is listed on the object's ACL
ACL	=	Available to objects, but only if the user is listed on the object's ACL
Configuration	=	Available to configuration objects only
* The user must be on the ACL of the server and will only see a list of services from his site		