

Oracle® Cloud

Developer's Guide for Oracle Analytics Cloud



G12245-16
September 2026



Oracle Cloud Developer's Guide for Oracle Analytics Cloud,

G12245-16

Copyright © 2024, 2026, Oracle and/or its affiliates.

Contributing Authors: Stefanie Rhone, Hemala Vivek, Adam Donald

Contributors: Oracle Analytics development, product management, and quality assurance teams

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Contents

Preface

Audience	i
Related Documents	i
Conventions	i

Part I Overview of Oracle Analytics Developer Resources

1 About Extending and Integrating Oracle Analytics

Introduction to Oracle Analytics Developer Resources	1
Obtain OAuth 2.0 Access Tokens for Oracle Analytics Cloud	1

Part II Create and Manage Custom Extensions

2 Create Custom Data Action Extensions

About Data Action Extensions and the Data Actions Framework	1
Data Action Categories	2
Data Action Context	3
Data Action Code Design	4
Data Action Model Classes	4
Data Action Service Classes	6
Data Action Code Interactions	7
Example Data Action plugin.xml File	8
Data Action Extension Files and Folders	9
Choose the Best Data Action Class to Extend	9
AbstractDataAction Class	10
DataActionKOModel Class	11
CanvasDataAction Class	12
EventDataAction Class	12
AbstractHTTPDataAction Class	13
URLNavigationDataAction Class	13

HTTPAPIDataAction Class	14
Generate Data Action Extensions from a Template	14
Generated Folders and Files	15
Extend a Data Action Base Class	16
Choose Which Data Action Inherited Methods to Override	17
Test, Package, and Install Your Data Action	20
Use an Upgrade Handler for Knockout Model Changes	21
Upgrade Data Action Extensions	21
Manage Data Action Extensions	22
Data Action Extension File Reference	22
Data Action plugin.xml File Example	23
Data Action plugin.xml File Properties Section - tns:obiplugin	24
Data Action plugin.xml File Resources Section - tns:resources	24
Data Action plugin.xml File Extensions Section - tns:extension	26

Part III Embed Content

3 Get Started Embedding Content into Applications and Web Pages

About Embedding Oracle Analytics Content into Applications and Web Pages	1
Register an Application as a Safe Domain	1

4 Embed Oracle Analytics Content With iFrames

Considerations for Embedding Oracle Analytics Content With iFrame	1
Use iFrame to Embed Analytics Content into an Application or Web Page	1

5 Embed Oracle Analytics Content With the JavaScript Embedding Framework

About the JavaScript Embedding Framework V2	1
Typical Workflow to Use the JavaScript Embedding Framework with Oracle Analytics Content	2
Enable Oracle Analytics Developer Options	3
Find the Javascript and HTML for Embedding Oracle Analytics Content	3
Prepare the HTML Page for Embedded Oracle Analytics Content	4
Pass Filters to the HTML Page for Embedded Oracle Analytics Content	8
About Filters and the JavaScript Embedding Framework V2	9
Use the filters Property to Pass Filters	9
Use the filters-v2 Property to Pass Filters	11
Find the JSON Objects for the filters-v2 Property	15
Pass Parameters to the HTML Page for Embedded Oracle Analytics Content	16

Refresh Data in the HTML Page for Embedded Oracle Analytics Content	17
Embed Oracle Analytics Content into a Custom Application that Uses Oracle JET	18
Embed Oracle Analytics Content into a Custom Application That Doesn't Use Oracle JET	20
Add Authentication to an Application or Web Page Containing Embedded Oracle Analytics Content	22
Use Login Prompt Authentication With Embedded Oracle Analytics Content	22
Use 3-Legged OAuth Authentication With Embedded Oracle Analytics Cloud Content	23
Use Token Authentication With Embedded Oracle Analytics Cloud Content	24

6 Embed Oracle Analytics AI Assistant With the JavaScript V2 Embedding Framework

Prepare the HTML Page for Embedding the Oracle Analytics AI Assistant	1
Permission Required to Use Embedded AI Assistant	4

Part IV Connect Oracle Analytics to AI

7 Get Started with Oracle Analytics Cloud MCP Tools (Preview)

About oac-mcp-connect (Preview)	2
Available Oracle Analytics Cloud MCP Tools (Preview)	3
Visible MCP Tool Metadata (Preview)	4
JSON-RPC Shape (Preview)	7
HTTP Session Initialization (Preview)	7
Common Workflows (Preview)	9
Discover Data and Run a Query Workflow (Preview)	9
Safety Rules (Preview)	10
Troubleshooting (Preview)	11
Quick Reference (Preview)	12
Deprecated MCP Server Tools (Preview)	12

8 Oracle Analytics Cloud MCP Tools Developer's Reference (Preview)

ANSI SQL Execution (Preview)	1
Catalog Discovery (Preview)	3
Copy Catalog Item (Preview)	6
Data Model Metadata (Preview)	8
Delete Catalog Item (Preview)	10
Folder Creation (Preview)	12
Logical SQL Execution (Preview)	13
Move Catalog Item (Preview)	15
Natural Language Data Source Matching (Preview)	17

Save Catalog Content (Preview)	18
Update Catalog ACL (Preview)	22
Workbook Export (Preview)	25

Part V Use APIs

9 REST APIs

10 SOAP APIs

Introduction to Oracle Analytics Web Services	1
About Oracle Analytics Web Services	1
What are the Oracle Analytics Session-Based Web Services?	1
Description of Services and Methods in Oracle Analytics Web Services	2
AdministrationService Service	2
deleteCSPWhitelist() Method	3
getCSPDefaultAllowList() Method	3
getCSPWhitelist() Method	3
updateCSPWhitelist() Method	4
AnalysisExportViewsService Service	4
completeAnalysisExport() Method	4
initiateAnalysisExport() Method	5
ConditionService Service	5
evaluateCondition() Method	6
evaluateInlineCondition() Method	6
getConditionCustomizableReportElements() Method	6
HtmlViewService Service	7
About HtmlViewService Bridging and Callback URLs	8
addReportToPage() Method	8
endPage() Method	9
getCommonBodyHTML() Method	9
getHeadersHTML() Method	9
getHtmlforPageWithOneReport() Method	10
getHTMLForReport() Method	10
setBridge() Method	11
startPage() Method	12
iBotService Service	12
deleteIBot() Method	13
enableIBot() Method	13
executeIBotNow() Method	14
getAgentPaths() Method	14

getAgents() Method	14
getIBotStatus() Method	15
moveIBot() Method	15
purgeAlerts() Method	15
sendMessage() Method	16
subscribe() Method	16
unsubscribe() Method	17
writeIBot() Method	17
MetadataService Service	18
clearQueryCache() Method	18
describeColumn() Method	18
describeSubjectArea() Method	19
describeTable() Method	20
getSubjectAreas() Method	21
reloadLogConfiguration() Method	21
reloadMetadata() Method	21
ReportEditingService Service	22
applyReportDefaults() Method	22
applyReportParams() Method	22
getPromptElements() Method	23
generateReportSQL() Method	24
getReportColumns() Method	24
getReportElements() Method	24
SAWSessionService Service	25
getCurUser() Method	25
getSessionEnvironment() Method	26
getSessionVariables() Method	26
impersonate() Method	26
impersonateex() Method	27
keepAlive() Method	28
logoff() Method	28
logon() Method	28
logonex() Method	29
SchedulerService Service	29
getJobReferences() Method	30
getJobInstanceReferences() Method	30
getJob() Method	31
getJobInstance() Method	32
cancelJobInstance() Method	32
removeJobs() Method	33
purgeJobInstances() Method	33
Examples of Using the SchedulerService API	34

SecurityService Service	36
forgetAccountsEx() Method	36
getAccounts() Method	37
getGlobalPrivilegeACL() Method	37
getGlobalPrivileges() Method	38
getPermissions() Method	38
getPermissionsEx() Method	38
getPrivilegesStatus() Method	39
renameAccountsEx() Method	39
updateGlobalPrivilegeACL() Method	40
UserPersonalizationService Service	40
addFavorite() Method	40
deleteFavorite() Method	41
getFavorites() Method	41
updateFavorites() Method	42
getMostRecents() Method	42
WebCatalogService Service	42
copyItem() Method	43
copyItem2() Method	44
createFolder() Method	44
createLink() Method	45
deleteItem() Method	45
getItemInfo() Method	45
getMaintenanceMode() Method	46
getObjectCategories() Method	46
getObjectCreateList() Method	47
getObjectTypes() Method	47
getSubItems() Method	47
getUnarchiveJobState() Method	48
getUserHomeDirPath() Method	48
maintenanceMode() Method	49
moveItem() Method	49
pasteItem2() Method	49
readObjects() Method	50
removeFolder() Method	50
setItemAttributes() Method	51
setItemProperty() Method	51
setOwnership() Method	52
unarchive() Method	52
updateCatalogItemACL() Method	52
writeObjects() Method	53
XMLViewService Service	54

XMLQueryOutputFormat Enumeration	54
cancelQuery() Method	54
executeSQLQuery() Method	55
executeXMLQuery() Method	55
fetchNext() Method	56
getPromptedFilters() Method	56
upgradeXML() Method	57
Description of Structures in Oracle Analytics Web Services	57
AccessControlToken Structure	59
Account Structure	59
ACL Structure	60
AnalysisExportExecutionOptions Structure	60
AnalysisExportResult Structure	61
AuthResult Structure	61
CatalogObject Structure	61
CSPWhitelist Structure	62
CSPWhitelistXml Structure	62
ErrorInfo Structure	63
FavoriteItem Structure	63
ForgetAccount Structure	64
ForgetAccountResult Structure	64
ForgetAccountsStatus Structure	64
GetSubItemsParams Structure	65
ItemInfo Structure	65
Job Structure	66
JobFilter Structure	67
JobInstance Structure	67
JobInstanceFilter Structure	68
JobInstanceStatus Enumeration	68
JobReferenceAndInstanceReferences Structure	69
MRUItem Structure	69
NameValuePair Structure	69
ReportHierarchicalColumn Structure	70
Prompt Structures	70
PromptsObjectModel Structure	71
PromptCollectionRunTimeInfo Structure	71
PromptStepObjectModel Structure	72
PromptStepRunTimeInfo Structure	72
IndividualPromptObjectModel Structure	73
IndividualPromptRunTimeInfoLimitedByInfo Structure	73
IndividualPromptRunTimeInfo Structure	74
IndividualPromptRunTimeInfoDataTypeHierarchyLevelSQLInfo Structure	74

IndividualPromptRunTimeInfoDataTypeHierarchyLevelInfo Structure	75
IndividualPromptRunTimeInfoDataTypeHierarchyLevels Structure	75
IndividualPromptRunTimeInfoDataTypeHierarchyFormulaLevels Structure	76
IndividualPromptRunTimeInfoDataTypeHierarchyInfo Structure	76
IndividualPromptRunTimeInfoDataTypeDoubleColumnInfo Structure	77
IndividualPromptRunTimeInfoDataType Structure	77
IndividualPromptRunTimeInfoSingleValueType Structure	78
IndividualPromptRunTimeInfoValuesType Structure	78
IndividualPromptRunTimeInfoCurrentValues Structure	79
IndividualPromptRunTimeInfoAvailableOptions Structure	79
IndividualPromptRunTimeInfoDataTypeHierarchyInfo Structure	80
IndividualPromptRunTimeInfoLimitedByPromptReference Structure	80
IndividualPromptRunTimeInfoLimitedByPromptRefGroups Structure	80
Privilege Structure	80
PurgeJobInstancesFilter Structure	81
QueryResults Structure	81
RenameAccount Structure	81
RenameAccountResults Structure	82
RenameAccountsStatus Structure	82
ReportADFPParameters Structure	82
ReportHTMLOptions Structure	83
ReportHTMLLinksMode Enumeration	83
ReportParams Structure	83
ReportRegularColumn Structure	84
ColumnAggregationRule Values	85
ReportRef Structure	85
SAColumn Structure	86
SADatatype Values	86
AggregationRule Values	87
SASubjectArea Structure	87
SATable Structure	88
SAWLocale Structure	88
SAWSessionParameters Structure	88
SegmentationOptions Structure	89
SessionEnvironment Structure	90
StartPageParams Structure	90
UpdateACLParams Structure	91
UpdateACLMode Enumeration	91
UpdateCatalogItemACLParams Structure	91
Variable Structure	92
XMLQueryExecutionOptions Structure	92

11 Schemas for Validating XML Documents

analysis_customization.xsd	1
analysis_ibot.xsd	3
condition.xsd	12

Preface

Learn how to develop and extend your Oracle Analytics instance with embedded content, and SDKs.

Topics:

- [Audience](#)
- [Related Documents](#)
- [Conventions](#)

Audience

This guide is intended for application developers and integrators who want to programmatically access and use the Oracle Analytics components to create applications or integrations with other components. You need to have knowledge of the following:

- Oracle Analytics Cloud
- Oracle Cloud Infrastructure

Related Documents

For a full list of guides, refer to the Books tab on Oracle Analytics Cloud Help Center.

<http://docs.oracle.com/en/cloud/paas/analytics-cloud/books.html>

Conventions

Conventions used in this document are described in this topic.

Text Conventions

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Videos and Images

Your company can use skins and styles to customize the look of the Oracle Analytics Cloud, dashboards, reports, and other objects. It is possible that the videos and images included in the product documentation look different than the skins and styles your company uses.

Even if your skins and styles are different than those shown in the videos and images, the product behavior and techniques shown and demonstrated are the same.

Part I

Overview of Oracle Analytics Developer Resources

This part introduces you to the Oracle Analytics developer resources.

Chapters:

- [About Extending and Integrating Oracle Analytics](#)

1

About Extending and Integrating Oracle Analytics

This chapter contains information about the available developer resources. It also explains how to obtain security tokens.

Topics:

- [Introduction to Oracle Analytics Developer Resources](#)
- [Obtain OAuth 2.0 Access Tokens for Oracle Analytics Cloud](#)

Introduction to Oracle Analytics Developer Resources

Oracle Analytics provides resources to help you augment and expand how your company uses analytics.

Developer Resources

Developer Resource	See
REST APIs	Oracle Analytics Cloud REST APIs REST APIs for Oracle Analytics Publisher REST APIs in Oracle Analytics Cloud OCI REST APIs for managing your analytics instance
SOAP APIs	SOAP APIs Oracle Analytics Publisher SOAP APIs
Custom data action extensions	Create Custom Data Action Extensions
Analytics content embedding	Embed Content
AI client connection	Connect Oracle Analytics to AI
Command line interfaces (CLI)	OCI CLI Commands

Obtain OAuth 2.0 Access Tokens for Oracle Analytics Cloud

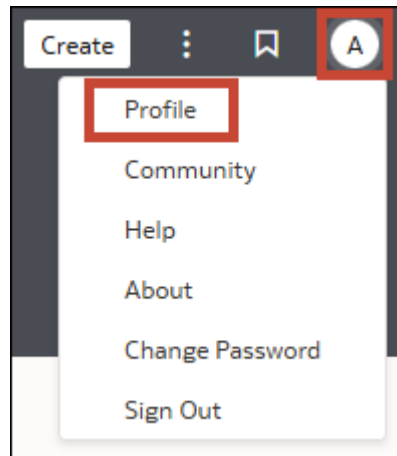
You need an OAuth 2.0 access token to authorize access to the Oracle Analytics Cloud REST APIs, test analytics content embedded into applications, or call the Oracle Analytics Cloud MCP Server endpoints (Preview).

If the Access Tokens tab isn't displayed in your Profile window, go to the Advanced tab in your profile window, click the **Enable Developer Options** button, and click **Save**.

These options are available from your Profile page:

- **Access Token** - Use this token to make Oracle Analytics Cloud REST API calls, test embedded analytics content, or use an MCP server (Preview). Refer to the text at the top of the Access Tokens tab to learn how long the access token is valid for. To obtain a new token, click **Refresh tokens**.

- **Refresh Token** - Use this token with the valid access token to programmatically get a new access token before the access token expires.
1. Go to your Oracle Analytics Cloud home page, and click the name badge with your initials.



2. Click **Profile**.
3. In My Profile, click **Access Tokens**.
4. Click **Download tokens** to download the tokens .json file containing the access token and refresh token. Or, click the access or refresh token's **Copy** button to copy the token.

Part II

Create and Manage Custom Extensions

This part explains how to create and manage custom data action extensions.

Topics:

- [Create Custom Data Action Extensions](#)

2

Create Custom Data Action Extensions

You can create custom data action extensions to use in Oracle Analytics.

Data action extensions extend Oracle Analytics and enable users to select data-points in visualizations and to invoke specific actions. Oracle Analytics provides a core set of data actions that cover many common use cases, but by writing your own data action extension, you can extend this functionality even further.

You must have a basic understanding of the following to create custom data action extensions:

- JavaScript
- RequireJS
- JQuery
- KnockoutJS

Topics:

- [About Data Action Extensions and the Data Actions Framework](#)
- [Choose the Best Data Action Class to Extend](#)
- [Generate Data Action Extensions from a Template](#)
- [Generated Folders and Files](#)
- [Extend a Data Action Base Class](#)
- [Choose Which Data Action Inherited Methods to Override](#)
- [Test, Package, and Install Your Data Action](#)
- [Use an Upgrade Handler for Knockout Model Changes](#)
- [Upgrade Data Action Extensions](#)
- [Manage Data Action Extensions](#)
- [Data Action Extension File Reference](#)

About Data Action Extensions and the Data Actions Framework

Data action extensions leverage the data actions framework to provide custom, data-driven actions that are tightly integrated into the Oracle Analytics user interface.

When a user invokes a data action, the Data Action Manager passes the request context (for example, qualified data reference, measure values, filters and metadata) to the data action extension which is responsible for handling the request. Oracle provides four types of data action extensions: `CanvasDataAction`, `URLNavigationDataAction`, `HTTPAPIDataAction` and `EventDataAction`. You can extend these data action extension types along with their abstract base classes to provide your own data actions.

Topics:

- [Data Action Categories](#)

- [Data Action Context](#)
- [Data Action Code Design](#)
- [Data Action Model Classes](#)
- [Data Action Service Classes](#)
- [Data Action Code Interactions](#)
- [Example Data Action plugin.xml File](#)
- [Data Action Extension Files and Folders](#)

Data Action Categories

The data action categories include Navigate to URL, HTTP API, Navigate to Canvas, and Event actions:

- **Navigate to URL:** Opens the specified URL in a new browser tab.
- **HTTP API:** Uses the GET/POST/PUT/DELETE/TRACE commands to target an HTTP API and doesn't result in a new tab. Instead the HTTP status code is examined and a transient success or failure message is displayed.
- **Navigate to Canvas:** Enables the user to navigate from a source canvas to a target canvas in either the same or a different visualization. Any filters that are in effect in the source canvas are passed to the target canvas as external filters. When the target canvas opens, it attempts to apply the external filters to the visualization. The mechanism by which external filters are applied isn't described here.
- **Event Actions:** Publishes an event using the Oracle Analytics event router. Any JavaScript code (for example, a third-party extension) can subscribe to these events and handle their custom response accordingly. This provides the maximum flexibility because the extension developer can choose how the data action responds. For example, they can choose to display a user interface or pass data to multiple services at once.

Both the **Navigate to URL** and **HTTP API** data action category types can use a token syntax to inject data or metadata from the visualization into the URL and POST parameters.

URL Token Replacement

HTTP data actions can replace tokens in URLs with values from the context passed to the data action. For example, qualified data reference values, filter values, username, workbook path, and canvas name.

Token	Notes	Replace With	Example	Result
\$ {valuesForColumn:C OLUMN}	NA	Column display values from the qualified data reference.	\${valuesForColumn: "Sales"."Products" ."Brand"}	BizTech, FunPod

Token	Notes	Replace With	Example	Result
<code>\${valuesForColumn:COLUMN, separator:"/"}</code>	Any token that can potentially be replaced with multiple values supports the optional separator option. The separator defaults to a comma (,) but you can set it to any string. You can escape double quotes inside this string by using a backslash (\).	Column display values from the qualified data reference.	<code>\${valuesForColumn:"Sales"."Products"."Brand"}</code>	BizTech, FunPod
<code>\${valuesForColumn:COLUMN, separationStyle:individual}</code>	Any separationStyle defaults to delimited but you can set it to individual if the user needs to generate separate URL parameters for each value.	Column display values from the qualified data reference.	<code>&myParam=\${valuesForColumn:"Sales"."Products"."Brand"}</code>	<code>&myParam=BizTech&myParam=FunPod</code>
<code>\${keyValuesForColumn:COLUMN}</code>	NA	Column key values from the qualified data reference.	<code>\${keyValuesForColumn:COLUMN}</code>	10001, 10002
<code>\${env:ENV_VAR}</code>	Supported environment variables are: sProjectPath, sProjectName, sCanvasName, sUserID, and sUserName.	An environment variable.	<code>\${env:'sUserID'}</code>	myUserName

Data Action Context

You can define a context that is passed when the user invokes a data action.

You define how much of the context is passed to the data action when you create the data action.

Qualified Data Reference

When the data action is invoked a qualified data reference is generated for each marked data point using an array of `LogicalFilterTree` objects. A `LogicalFilterTree` consists of multiple `LogicalFilterNode` objects arranged in a tree structure. This object includes:

- The attributes on the row or column edges of the data layout.
- The specific measure on the measure edge that addresses each marked cell.
- The specific measure value for each marked cell.
- Key values and display values.

Environment Variables

In addition to the data and metadata describing each marked data point, certain data actions may need further context describing the environment from where the data action is invoked. Such environment variables include:

- Project Path
- Project Name
- Canvas Name
- User ID
- User Name

Data Action Code Design

You create data actions using API classes.

- There are four concrete classes of data action that inherit from the `AbstractDataAction` class:
 - `CanvasDataAction`
 - `URLNavigationDataAction`
 - `HTTPAPIDataAction`
 - `EventDataAction`
- You can create new types of data actions using the data action extension API.
- The registry of data action types is managed by the `DataActionPluginHandler`.
- Code that creates, reads, edits, deletes, or invokes instances of data actions does so by publishing events.
- Events are handled by the `DataActionManager`.

Data Action Model Classes

There are several different types of data action model classes.

AbstractDataAction

This class is responsible for:

- Storing the Knockout Model (subclasses are free to extend this with their own properties).
- Defining the abstract methods that subclasses must implement:
 - `+ invoke(oActionContext: ActionContext, oDataActionContext: DataActionContext) <<abstract>>`
Invokes the data action with the passed context - should only be called by the `DataActionManager`.
 - `+ getGadgetInfos(oReport): AbstractGadgetInfo[] <<abstract>>`
Constructs and returns the `GadgetInfos` responsible for rendering the user interface fields for editing this type of data action.
 - `+ validate() : DataActionError`
Validates the data action and returns null if valid or a `DataActionError` if it's invalid.
- Providing the default implementation for the following methods used to render generic parts of the data action user interface fields:
 - `+ getSettings(): JSON`
Serializes the data action's Knockout Model to JSON ready to be included in the report (uses `komapping.toJS(_koModel)`).

- + createNameGadgetInfo(oReport) : AbstractGadgetInfo
Constructs and returns the GadgetInfo that can render the data action's **Name** field.
- + createAnchorToGadgetInfo(oReport) : AbstractGadgetInfo
Constructs and returns the GadgetInfo that can render the data action's **Anchor To** field.
- + createPassValuesGadgetInfo(oReport) : AbstractGadgetInfo
Constructs and returns the GadgetInfo that can render the data action's **Pass Values** field.

Subclasses may not need all of the GadgetInfos that the base class provides so they may not need to call all of these methods. By separating out the rendering of each field in this way, subclasses are free to pick and choose the gadgets they need. Some subclasses may even choose to provide a different implementation of these common data action gadgets.

CanvasDataAction, URLNavigationDataAction, HTTPAPIDataAction, EventDataAction

These are the concrete classes for the basic types of data actions. These classes work by themselves to provide the generic user interface for these types of data action. They can also act as convenient base classes for custom data action plug-ins to extend.

- **CanvasDataAction**: Used to navigate to a canvas.
- **URLNavigationDataAction**: Used to open a web page in a new browser window.
- **HTTPAPIDataAction**: Used to make a GET/POST/PUT/DELETE/TRACE request to an HTTP API and handle the HTTP Response programmatically.
- **EventDataAction**: Used to publish JavaScript events through the Event Router.

Each class is responsible for:

- Implementing the abstract methods from the base class.
 - invoke(oActionContext: ActionContext, oDataActionContext:DataActionContext)
This method should invoke the data action by combining the properties defined in the KOModel with the specified DataActionContext object.
 - getGadgetInfos(oReport): AbstractGadgetInfo[]
This method should:
 - * Create an array containing AbstractGadgetInfos.
 - * Call individual createXXXGadgetInfo() methods pushing each AbstractGadgetInfo into the array.
 - * Return the array.
- Providing the additional methods for creating the individual gadgets that are specific to the particular subclass of data action.

Subclasses of these concrete classes may not need to use all of the gadgets provided by their superclasses in their custom user interfaces. By separating out the construction of each gadget in this way, subclasses are free to pick and choose the gadgets they need.

DataActionKOModel, ValuePassingMode

The DataActionKOModel class provides the base KOModel shared by the different subclasses of AbstractDataAction. See [DataActionKOModel Class](#).

Data Action Service Classes

There are several different data action service classes.

DataActionManager



All communication with DataActionManager uses ClientEvents.DataActionManager which implements event handlers for:

- Managing the set of data actions defined in the current workbook.
- Invoking a data action.
- Retrieving all the data actions defined in the current workbook.
- Retrieving all the data actions that are applicable to the current marked data points.

DataActionContext, EnvironmentContext

When a data action is invoked, the DataActionContext class contains the context that's passed to the target.

- getColumnValueMap()
Returns a map of attribute column values keyed by attribute column names. These define the qualified data reference for the data points that the data action is invoked from.
- getLogicalFilterTrees()
Returns a LogicalFilterTrees object describing the qualified data references for the specific data points that the data action is invoked from (see the InteractionService for details).
- getEnvironmentContext()
An instance of the EnvironmentContext class describing the source environment such as:
 - getProjectPath()
 - getCanvasName()
 - getUserID()
 - getUsername()
- getReport()
Returns the report that the data action is invoked from.

DataActionHandler

The DataActionHandler class registers the various data action extensions. Its API is broadly consistent with the other extension handlers (for example, VisualizationHandler).

The DataActionHandler class provides the following public methods:

- getClassname(sPluginType:String) : String

Returns the fully qualified class name for the specified data action type.

- `getDisplayName(sPluginType:String) : String`
Returns the translated display name for the specified data action type.
- `getOrder(sPluginType:String) : Number`
Returns a number used to sort lists of the types of data action into the preferred order.

The `DataActionHandler` class provides the following static methods:

- `getDependencies(oPluginRegistry:Object) : Object.<String, Array>`
Returns a dependency map covering all the registered data action types.
- `getHandler(oPluginRegistry:Object, sExtensionPointName:String, oConfig:Object) : DataActionPluginHandler`
Constructs and returns a new instance of the `DataActionHandler` class.

DataActionUpgradeHandler

The `DataActionUpgradeHandler` class is called by the `UpgradeService` when a report is opened.

The `DataActionHandler` class provides two main methods:

- `deferredNeedsUpgrade(sCurrentVersion, sUpgradeTopic, oDataActionJS, oActionContext) : Promise`
Returns a `Promise` that resolves to a `Boolean` indicating whether the specified data action must be upgraded (`true`) or not (`false`). The method decides whether the data action must be upgraded by comparing the data action instance with the data action's constructor.
- `performUpgrade(sCurrentVersion, sUpgradeTopic, oDataActionJS, oActionContext, oUpgradeContext) : Promise`
Carries out the upgrade on the specified data action and resolves the `Promise`. The upgrade itself is carried out by calling the `upgrade()` method on the data action (only the specific subclass of data action being upgraded is qualified to upgrade itself).
- `getOrder(sPluginType:String) : Number`
Returns a number used to sort lists of the types of data action into the preferred order.

Data Action Code Interactions

A data action interacts with Oracle Analytics code when it creates a user interface field, and when a user invokes a data action.

Create the Field for a New Data Action Instance

This interaction starts when Oracle Analytics wants to render a data action user interface field. To do so, it:

1. Creates a `PanelGadgetInfo` that acts as the parent `GadgetInfo` for the `GadgetInfos` that the data action returns.
2. Calls `getGadgetInfos()` on the data action.
3. Adds the data action's `GadgetInfos` as children of the `PanelGadgetInfo` created in the first step.
4. Creates the `PanelGadgetView` that renders the `PanelGadgetInfo`.
5. Sets the `HTMLElement` that's the container of the `PanelGadgetView`.

6. Registers the `PanelGadgetView` as a child `HostedComponent` of a `HostedComponent` that's already attached to the `HostedComponent` tree.
This renders the data action's gadgets inside the `Panel` gadget in the order they appear in the array returned by `getGadgetInfos()`.

Invoke a Data Action

This interaction starts when the user invokes a data action through the Oracle Analytics user interface (for example, from the context menu on a data point in a visualization).

In response to the user interaction, the code:

1. Publishes an `INVOKE_DATA_ACTION` event containing the data action's ID, the `DataVisualization` that the data action is invoked from, and a `TransientVizContext` object.
2. The `DataActionManager` handles this event by:
 - a. Obtaining the data action instance from its ID.
 - b. Obtaining the `LogicalFilterTrees` for the marked data points in the specified `DataVisualization`.
 - c. Constructing a `DataActionContext` that contains all the information to pass to the data action's target.
 - d. Calling `invoke(oDataActionContext)` on the data action.

Example Data Action plugin.xml File

This topic shows an example `plugin.xml` file for a `CanvasDataAction` data action.

Example plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:obiplugin xmlns:tns="http://plugin.frameworks.tech.bi.oracle"
  xmlns:viz="http://plugin.frameworks.tech.bi.oracle/extension-
points/visualization"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  id="obitech-currencyconversion"
  name="Oracle BI Currency Conversion"
  version="0.1.0.@qualifier@"
  optimizable="true"
  optimized="false">

  <tns:resources>
    <tns:resource id="currencyconversion" path="scripts/
currencyconversion.js" type="script" optimizedGroup="base"/>
    <tns:resource-folder id="nls" path="resources/nls" optimizable="true">
      <tns:extensions>
        <tns:extension name="js" resource-type="script"/>
      </tns:extensions>
    </tns:resource-folder>
  </tns:resources>

  <tns:extensions>
    <tns:extension id="oracle.bi.tech.currencyconversiondataaction" point-
```

```

id="oracle.bi.tech.plugin.dataaction" version="1.0.0">
  <tns:configuration>
    {
      "resourceBundle": "obitech-currencyconversion/nls/messages",
      "properties":
      {
        "className": "obitech-currencyconversion/
currencyconversion.CurrencyConversionDataAction",
        "displayName": { "key" : "CURRENCY_CONVERSION", "default" :
"Currency Conversion" },
        "order": 100
      }
    }
  </tns:configuration>
</tns:extension>
</tns:extensions>

</tns:obiplugin>

```

Data Action Extension Files and Folders

The following files and folders are used to implement data action extensions.

bitech/client/plugins/src/

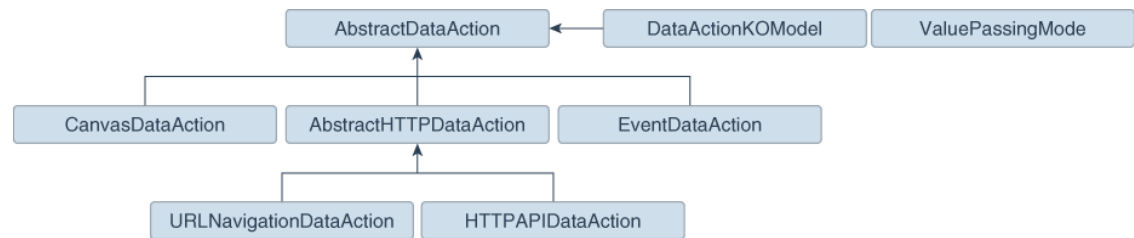
- report
 - obitech-report
 - * scripts
 - * dataaction
 - * dataaction.js
 - * dataactiongadgets.js
 - * dataactionpanel.js
 - * dataactionupgradehandler.js
- obitech-reportservice
 - scripts
 - * dataaction
 - * dataactionmanager.js
 - * dataactionhandler.js

Choose the Best Data Action Class to Extend

Before you start writing your custom data action extension, decide which of the existing data action classes you want to extend. Choose the data action class that provides functionality that most closely matches what you want your data action to do.

Each data action inherits from the `AbstractDataAction` class as shown in the class diagram. The class diagram shows the two abstract data action classes (`AbstractDataAction` and `AbstractHTTPDataAction`) and the four concrete data action classes (`CanvasDataAction`,

URLNavigationDataAction, HTTPAPIDataAction, and EventDataAction) that you can extend. Each data action that you provide must extend one of these classes. Which class you extend depends on the behavior you want to implement when you invoke your data action. Most third-party data actions are likely to extend either URLNavigationDataAction, HTTPAPIDataAction or EventDataAction.



Regardless of which class you extend, when your data action is invoked, you're provided with metadata describing the full context of the data-point from which the data action is invoked. See [Data Action Context](#).

AbstractDataAction Class

AbstractDataAction is the abstract base class from which all types of data action inherit. It's responsible for providing common functionality and default behavior that the subclasses can use.

AbstractDataAction

All types of data action are subclasses of the **AbstractDataAction** base class. It provides the core set of functionality common to all data actions. Unless you're creating a complex data action that carries out multiple types of action when invoked, or you need to do something not supported by the concrete classes, you shouldn't extend this class directly. If you need to create a complex data action then consider extending the concrete class that most closely provides the functionality you require.

AbstractDataAction Syntax

```

+ AbstractDataAction(oKOModel)

+ getKOViewModel():DataActionKOModel

+ createFromJS(fDataActionConstructor, sClassName, oDataActionKOModelUS) :
AbstractDataAction

+ invoke(oActionContext, oDataActionContext)
+ getGadgetInfos(oReport) : AbstractGadgetInfo[]
+ validate() : DataActionError

+ getSettings() : Object
+ requiresActionContextToInvoke() : Boolean
+ isAllowedHere() : Boolean

# createNameGadgetInfo(oReport) : AbstractGadgetInfo
# createAnchorToGadgetInfo(oReport) : AbstractGadgetInfo
# createPassValuesGadgetInfo(oReport) : AbstractGadgetInfo

```

DataActionKOModel Class

Each subclass of AbstractDataAction is likely to create its own subclass of DataActionKOModel. The DataActionKOModel base class provides the following properties:

DataActionKOModel, ValuePassingMode

- **sID:String**
The unique ID given to the data action instance.
- **sClass:String**
The class name of this specific type of data action.
- **sName:String**
The display name given to the data action instance.
- **sVersion**
- **sScopeID**
- **eValuePassingMode:ValuePassingMode**
The mode used when passing context values. The mode can be one of the ValuePassingMode values (ALL, ANCHOR_DATA, NONE, CUSTOM).
- **aAnchorToColumns: ColumnKOViewModel[]**
The columns that this data action is anchored to. This is optional. If not supplied, then the data action is available on all columns.
- **aContextColumns : ColumnKOViewModel[]**

The columns that this data action includes in the context passed to the data action target when the data action is invoked. If not supplied, all marked columns are included in the context.

CanvasDataAction Class

CanvasDataAction is a subclass of the AbstractDataAction base class. You can extend this concrete class to provide the functionality you require.

CanvasDataAction

Use the CanvasDataAction class to navigate from a data point in a visualization to a different canvas. The canvas you're navigating to can be in the same workbook or a different one. All the active filters for the source visualization are passed to the target canvas along with new filters that describe the Qualified Data Reference of the data point itself. If your data action needs to navigate to a different canvas then this is the class your data action should extend.

```
+ CanvasDataAction(oKOModel)

+ create(s)ID_sName) : CanvasDataAction
+ upgrade(oOldDataActionJS) : Object

+ invoke(oActionContext: ActionContext, oDataActionContext:DataActionContext)
+ getGadgetInfos(oReport) : AbstractGadgetInfo[]
+ validate() : DataActionError

# createProjectGadgetInfo(oReport) : AbstractGadgetInfo
# createCanvasGadgetInfo(oReport) : AbstractGadgetInfo
```

EventDataAction Class

EventDataAction is a subclass of the AbstractDataAction base class. You can extend this concrete class to provide the functionality you require.

EventDataAction

Use the EventDataAction class to publish a client-side event. You can then register one or more subscribers that listen for that event and perform their own actions. Use this type of data action in more complex use cases where you've a large amount of code and can benefit from

keeping your data action code loosely coupled to the code that performs the necessary actions when the data action is invoked.

```
+ EventDataAction(oKModel)

+ create(sID_sName) : EventDataAction
+ upgrade(oOldDataActionJS) : Object

+ invoke(oActionContext: ActionContext, oDataActionContext:DataActionContext)
+ getGadgetInfos(oReport) : AbstractGadgetInfo[]
+ validate() : DataActionError

# createEventGadgetInfo(oReport) : AbstractGadgetInfo
```

AbstractHTTPDataAction Class

AbstractHTTPDataAction is the abstract base class that the URLNavigationDataAction and HTTPAPIDataAction subclasses inherit common functionality and default behavior from.

AbstractHTTPDataAction

The AbstractHTTPDataAction abstract base class is shared by both the URLNavigationDataAction and HTTPAPIDataAction classes. If your data action needs to open a web page in a new browser tab you must extend URLNavigationDataAction. If your data action needs to invoke an HTTP API then you should extend HTTPAPIDataAction. You may decide it's better to extend AbstractHTTPDataAction directly.

```
+ HTTPDataAction(oKModel)

+ validate() : DataActionError

# createURLGadgetInfo(oReport) : AbstractGadgetInfo
```

URLNavigationDataAction Class

URLNavigationDataAction is a subclass of the AbstractHTTPDataAction base class.

URLNavigationDataAction

Use the URLNavigationDataAction class to open a specific URL in a new browser tab. You compose the URL using tokens that are replaced with values derived from data points that the user selects when they invoke the data action. The data point values are passed as part of the data action context to the external web page. For example, create a data action invoked using

a CustomerID column that opens a customer's web page in your Customer Relations Management application such as Oracle Sales Cloud.

```
+ URLNavigationDataAction(oKModel)

+ create(sID_sName) : URLNavigationDataAction
+ upgrade(oOldDataActionJS) : Object

+ invoke(oActionContext: ActionContext, oDataActionContext:DataActionContext)
+ getGadgetInfos(oReport) : AbstractGadgetInfo[]
```

HTTPAPIDataAction Class

HTTPAPIDataAction is a subclass of the AbstractHTTPDataAction base class. You can extend this concrete class to provide the functionality you require.

HTTPAPIDataAction

Use the HTTPAPIDataAction class to invoke HTTP APIs by creating an asynchronous XMLHttpRequest (XHR) and submitting it to the specified URL. The HTTP response code enables a message to be displayed briefly on the canvas. For example, you can customize the request to send JSON or XML payloads to a REST or SOAP server and you can customize the response handler to show a custom user interface.

For the HTTPAPIDataAction data action to work, you must add the URL of the HTTP API you want to access to your list of Safe Domains and grant it **Connect** access. See Register Safe Domains.

```
+ HTTPAPIDataAction(oKModel)

+ create(sID_sName) : HTTPAPIDataAction
+ upgrade(oOldDataActionJS) : Object

+ invoke(oActionContext: ActionContext, oDataActionContext:DataActionContext)
+ getGadgetInfos(oReport) : AbstractGadgetInfo[]

# createHTTPMethodGadgetInfo(oReport) : AbstractGadgetInfo
# createPostParamGadgetInfo(oReport) : AbstractGadgetInfo
```

Generate Data Action Extensions from a Template

You use a series of commands to generate a development environment and populate it with a HTTP API Data Action along with the necessary folders and files that you need to create a custom data action extension.

All extensions files follow the same basic structure. You can manually create the files and folders or you can generate them from a template. The tools to do this are part of the Oracle Analytics Desktop software development kit (SDK) which is included with Oracle Analytics Desktop.

Use these commands to generate your development environment and populate it with a HTTP API data action.

1. At a command prompt, specify the root folder of your Oracle Analytics Desktop installation:
`set DVDESKTOP_SDK_HOME=C:\Program Files\Oracle Analytics Desktop`
2. Specify the location to store your custom extensions:
`set PLUGIN_DEV_DIR=C:\temp\dv-custom-plugins`
3. Add the SDK command line tools to your path using:
`set PATH=%DVDESKTOP_SDK_HOME%\tools\bin;%PATH%`
4. Create a folder for the directory used to store the custom extensions using:
`mkdir %PLUGIN_DEV_DIR%`
5. Change the directory to the folder for storing custom extensions:
`cd %PLUGIN_DEV_DIR%`
6. Create the environment variables:
`bicreateenv`
7. Create the template files needed to start developing a custom HTTP API data action, for example:
`bicreateplugin -pluginxml dataaction -id company.mydataaction -subType httpapi`
Use the `-subType` option to specify the data action type that you want to create from: `httpapi`, `urlNavigation`, `canvasNavigation`, `event`, or `advanced`. The advanced option extends from the `AbstractDataAction` base class.

Generated Folders and Files

Your newly generated data action development environment contains these folders and files:

```
1  %PLUGIN_DEV_DIR%\src\customdataaction
2      company-mydataaction\
3          extensions\
4              oracle.bi.tech.plugin.dataaction\
5                  company.mydataaction.json
6          nls\
7              root\
8                  messages.js
9                  messages.js
10                 mydataaction.js
11                 mydataactionstyles.css
12                 plugin.xml
```

- **Line 2:** The `company-mydataaction` folder is the ID that you specify.
- **Line 6:** The `nls` folder contains the files for externalizing strings that enable your extension to provide Native Language Support.
- **Line 7:** The strings in the files under the `nls\root` folder are the default strings used when translations for a requested language aren't available.

- **Line 8:** The `messages.js` file contains externalized strings for your extension that you can add.
- **Line 9:** The `messages.js` file must contain an entry that you add for each additional language that you want to provide localized strings for. You must add a corresponding folder under the `nls` folder for each locale that you want to add translations for. Each folder must contain the same set of files, with the same file names as those added under the `nls\root` folder.
- **Line 10:** The `mydataaction.js` file is the newly generated JavaScript module template that provides a starting point to develop your custom data action.
- **Line 11:** The `mydataactionstyles.css` file can contain any CSS styles that you want to add, and which your data action's user interface can use.
- **Line 12:** The `plugin.xml` file registers your extension and its files with Oracle Analytics.

Extend a Data Action Base Class

Once you've chosen the subclass of data action that you want to extend and have generated the necessary folders and files, you're ready to start writing the code specific to your new data action.

You can find your newly generated data action code under `%PLUGIN_DEV_DIR%\src\dataaction`. See [Generated Folders and Files](#) for an explanation of the files and folder structure. The main file you must edit is the JavaScript file. For example, if your custom data action ID is `company.MyDataaction`, then the file you're looking for is `%PLUGIN_DEV_DIR%\src\dataaction\company-mydataaction\mydataaction.js`.

Extending Your Data Action's Knockout Model

If your data action has additional properties that need to be stored, then you must add them as observable properties to the Knockout Model. If your data action is given the ID `company.MyDataaction`, then the Knockout Model is called `mydataaction.MyDataActionKOModel` which is located near the top of `mydataaction.js`. By default, this Knockout Model is configured to extend the Knockout Model used by your data action's superclass so you only need to add additional properties to the model.

For a data action that's extending the `HTTPAPIDataAction` base class, use code similar to the following:

```
1 - mydataaction.MydataactionKOModel = function (sClass, sID, sName,
sVersion, sScopeID, aAnchorToColumns, eValuePassingMode, sURL,
    eHTTPMethod, sPOSTParams)
2 - {
3 - mydataaction.MydataactionKOModel.baseConstructor.call(this, sClass, sID,
sName, sVersion, sScopeID, aAnchorToColumns, eValuePassingMode, sURL,
eHTTPMethod, sPOSTParams);
4 - };
5 - jsx.extend(mydataaction.MydataactionKOModel,
dataaction.HTTPAPIDataActionKOModel);
```

- **Line 1:** This is the constructor for your Knockout Model. It accepts the properties that the model needs to store.
- **Line 3:** This is the superclass's constructor, otherwise known as the `baseConstructor` to which you pass the values for all of the properties that are handled by one of the Knockout Model's superclasses.

- **Line 5:** This sets the superclass for this Knockout Model class.

Use code similar to the following to add a string and an array to set properties that are persisted by the data action.

```

1  mydataaction.MydataactionKOModel = function (sClass, sID, sName,
sVersion, sScopeID, aAnchorToColumns, eValuePassingMode, sURL, eHTTPMethod,
sPOSTParams)
2  {
3  mydataaction.MydataactionKOModel.baseConstructor.call(this, sClass, sID,
sName, sVersion, sScopeID, aAnchorToColumns, eValuePassingMode, sURL,
eHTTPMethod, sPOSTParams);
4
5
6  // Set Defaults
7  sMyString = sMyString || "My default string value";
8  aMyArray = aMyArray || [];
9
10
11 // Asserts
12 jsx.assertString(sMyString, "sMyString");
13 jsx.assertArray(aMyArray, "aMyArray");
14
15
16 // Add observable properties
17 this.sMyString = ko.observable(sMyString);
18 this.aMyArray = ko.observableArray(aMyArray);
19 };
20 jsx.extend(mydataaction.MydataactionKOModel,
dataaction.HTTPAPIDataActionKOModel);

```

Choose Which Data Action Inherited Methods to Override

Each data action must implement various methods in order to function properly, so you only need to override those methods that implement behavior that you want to change.

Generic Methods

If you're extending one of the concrete data actions classes, for example HTTPAPIDataAction, then most of the required methods are already implemented and you only need to override the methods that implement the behavior you want to change.

This section describes the various methods and what's expected of them.

All types of data action must implement the methods that are described here.

create(sID, sName)

The create() static method is called when you're creating a new data action and select a **Data Action Type** from the drop-down menu. This method is responsible for:

- Constructing the Knockout Model class that your data action uses.
The Knockout Model class must have the ID and name that's passed to the create() method along with sensible defaults for all other properties. For example, for a currency conversion data action you might want to set the default currency to convert into Dollars. The Knockout Model is the correct place to provide your default values.
- Constructing an instance of your data action from the Knockout Model.

- Returning the instance of your data action.

invoke(oActionContext, oDataActionContext)

The `invoke()` method is called when the user invokes your data action from the context menu for a data point in a visualization. The method passes the `DataActionContext` argument which contains metadata describing the selected data points, visualization, filters, workbook, and session. See [Data Action Service Classes](#).

validate()

The `validate()` method is called on each data action when the user clicks **OK** in the Data Actions dialog. The `validate()` method returns a null to indicate that everything is valid or a `DataActionError` if something is invalid. If there's an error in one of the data actions in the dialog, the error prevents the dialog from closing and an error message is displayed to the user. This method validates the name of the data action using the `this.validateName()` method.

getGadgetInfos(oReport)

The `getGadgetInfos()` method is called to enable the user interface to display data action property fields. The method returns an array of `GadgetInfos` in the order you want them to appear in the user interface. Gadgets are provided for all of the most common types of fields (for example, text, drop-down, password, multi-select, radio button, check box) but you can create custom gadgets if you want more complicated fields (for example, where multiple gadgets are grouped together, or where different gadget fields display depending on which option you select). It's a best practice to create a method that constructs each `GadgetInfo` you want in your array, as it makes it easier for potential subclasses to pick and choose from the `GadgetInfos` you've provided. If you follow this best practice there are already various methods implemented by the different data action base classes that can return a `GadgetInfo` for each of the fields that they use in their user interfaces. If you also need one of these `GadgetInfos` then you call the corresponding `create****GadgetInfo()` method and push its return value into your array of gadgets.

isAllowedHere(oReport)

The `isAllowedHere()` method is called when the user right-clicks on a data-point in a visualization and the user interface starts to generate the context menu. If a data action exists that's relevant to the selected data-points, then the method returns `true` and the data action appears in the context menu. If the method returns `false`, then the data action doesn't appear in the context menu. Consider accepting the default behavior inherited from the superclass.

upgrade(oOldDataActionJS)

If you're creating your first data action then don't use the `upgrade(oOldDataActionJS)` method. Only use this method after you've created your first Knockout Model and are making significant changes to properties for a second version of your Knockout Model. For example, if the first version of your data action stores a URL in its Knockout Model, but you decide that the next version will store URL component parts in separate properties (for example, protocol, hostname, port, path, queryString and bookmark).

The second version of your Knockout Model code would request to open a data action that had been saved with the first version of your Knockout Model code which can cause problems. To resolve this issue, the system identifies that your current data action code version is newer than that of the data action being opened and it calls the `upgrade()` method on your new data action class and passes in the old data action Knockout Model (serialized to a JSON object). You can then use the old JSON object to populate your new Knockout Model and return an upgraded version of the JSON object. This ensures that old data action metadata continues to work as you improve your data action code.

HTTPAPIDataAction Methods

If you're extending the HTTPAPIDataAction class, then it provides the following additional method that you may choose to override:

getAJAXOptions(oDataActionContext)

The getAJAXOptions() method is called by the data action's invoke() method. The getAJAXOptions() method creates the AJAX Options object that describes the HTTP request that you want your data action to make. The getAJAXOptions() method is passed the oDataActionContext object that contains the metadata describing the selected data-points, visualization, filters, workbook, and session. Set the AJAX Options as required by the HTTP API you're trying to integrate with and specify the functions you want to be called when the HTTPRequest is successful or results in an error. See the JQuery website for an explanation of the jquery.ajax object and its properties.

The following implementation is inherited from the HTTPAPIDataAction class. You need to rewrite the inherited method to specify requirements. For example, forming the HTTP request, and the code that handles the HTTP response. This implementation is useful as it shows the parameters passed to the getAJAXOptions() function, the object that it's expected to return, and gives a clear example of how to structure the code inside the method.

```

1 /**
2  * This method returns an object containing the AJAX settings used when the
3  * data action is invoked.
4  * Subclasses may wish to override this method to provide their own
5  * behavior.
6  * @param {module:obitech-reportservices/
7  * dataactionmanager.DataActionContext} oDataActionContext The context metadata
8  * describing where the data action was invoked from.
9  * @returns {?object} A JQuery AJAX settings object (see http://
10 * api.jquery.com/jquery.ajax/ for details) - returns null if there is a
11 * problem.
12 */
13 dataaction.HTTPAPIDataAction.prototype.getAJAXOptions = function
14 (oDataActionContext)
15 {
16     jsx.assertInstanceOfModule(oDataActionContext, "oDataActionContext",
17     "obitech-reportservices/dataactionmanager", "DataActionContext");
18
19     var oAJAXOptions = null;
20     var oKOVViewModel = this.getKOVViewModel();
21     var sURL = oKOVViewModel.sURL();
22     if (sURL)
23     {
24         // Parse the URL
25         var sResultURL = this._parseURL(sURL, oDataActionContext);
26         if (sResultURL)
27         {
28             // Parse the POST parameters (if required)
29             var eHTTPMethod = oKOVViewModel.eHTTPMethod()[0];
30             var sData = null;
31             if (eHTTPMethod ===
32 dataaction.HTTPDataActionK0Model.HTTPMethod.POST)
33             {
34                 var sPOSTParams = oKOVViewModel.sPOSTParams();

```

```

26         sData =
sPOSTParams.replace(dataaction.AbstractHTTPDataAction.RegularExpressions.LINE_
END, "&");
27         sData = this._parseURL(sData, oDataActionContext, false);
28     }
29     oAJAXOptions = {
30         type: eHTTPMethod,
31         url: sResultURL,
32         async: true,
33         cache: false,
34         success: function (/*oData, sTextStatus, oJQXHR*/)
35         {
36
oDataActionContext.getReport().displaySuccessMessage(messages.HTTP_API_DATA_AC
TION_INVOCATION_SUCCESSFUL.format(oKOVModel.sName()));
37         },
38         error: function (oJQXHR/*, sTextStatus, sError*/)
39         {
40
oDataActionContext.getReport().displayErrorMessage(messages.HTTP_API_DATA_ACTI
ON_INVOCATION_FAILED.format(oKOVModel.sName(), oJQXHR.statusText,
oJQXHR.status));
41         }
42     };
43     if (sData)
44     {
45         oAJAXOptions.data = sData;
46     }
47 }
48 }
49 return oAJAXOptions;
50 };

```

Test, Package, and Install Your Data Action

You use Oracle Analytics Desktop to test your data action from its source location before you install it.

1. If Oracle Analytics Desktop is currently running, close it.
2. If you're working behind a proxy, set the proxy settings in %PLUGIN_DEV_DIR%\gradle.properties. For information about accessing the web through HTTP proxy, see Gradle User Manual.
3. Run Oracle Analytics Desktop in SDK mode by using the command prompt you started in [Choose Which Data Action Inherited Methods to Override](#) and enter the following commands:

```

cd %PLUGIN_DEV_DIR%
.\gradlew run

```

Oracle Analytics Desktop starts in SDK mode. Your data action extension appears in the Console | Extensions page.

Create a workbook and test your data action. If you find any issues, you can debug your code using your browser's built-in developer tools.

4. If you created an HTTP API data action:
 - a. Go to the Console and display the Safe Domains page.
 - b. Add each domain that you want to access.

For example, if you need access to the `apilayer.com` APIs, add `apilayer.net` to the list of safe domains.
 - c. Click the **Connect** column checkbox for the selected domain.
 - d. Reload the Safe Domains page in your browser for the changes to take effect.
5. If you want to prepare your data action extension to distribute to other people or to install in Oracle Analytics:
 - Package all of the files into a single ZIP file containing the `%PLUGIN_DEV_DIR%\src\customdataaction` folder and its contents.
 - Name the zip using the same ID you gave to your data action extension when you created it.
6. Install your data action extension. See [Manage Data Action Extensions](#).

Use an Upgrade Handler for Knockout Model Changes

For some Knockout Model changes you need to upgrade your data action extension using an upgrade handler.

When you're making improvements to your data action extension without making changes to the Knockout Model you normally edit your JavaScript or CSS files, create a new ZIP file, and replace the existing data action extension with the new ZIP file. However, if you've made changes to your data action's Knockout Model then you might need to change the data action `VERSION` property and provide an upgrade handler.

Decide whether you need to use an upgrade handler:

Upgrade Handler Required

- If you rename a property in your Knockout Model.
- If you combine multiple properties into a single property in your Knockout Model.
- If you split a single property into multiple properties in your Knockout Model.
- If you add a new property to the Knockout Model and the correct default value for it depends on other values in the Knockout Model.

Upgrade Handler Not Required

- If you add a new property to the Knockout Model and can provide a default value that's correct for all existing usages of your data action.
- If you remove a property from the Knockout Model because it's no longer used by your data action code.

Upgrade Data Action Extensions

Upgrade your data action extensions to improve the data action code or upgrade the metadata to enable existing data actions to work with new data action code.

Use an upgrade handler to upgrade a data action extension.

1. Increase the version number of your data action.

For example, if your data action is called `company.MyDataAction`, then search `mydataaction.js` for the `mydataaction.MyDataAction.VERSION` property. If it's currently set to `1.0.0` then change it to `1.0.1`.

2. Add a static `upgrade(oOldDataActionJS)` method to your data action's class.
If the `VERSION` property differs from the `sVersion` value stored in the data action metadata then the Data Action Manager calls the static `upgrade()` method on your data action's class.
3. Implement your `upgrade()` method by calling the `upgrade()` method on the superclass and capture its response.
4. Continue to implement your `upgrade()` method by making further edits to the partially upgraded data action JSON returned by the superclass, until the object matches the correct set of properties required by your latest Knockout Model.
5. To finish call `var oUpgradedDataAction = dataaction.AbstractDataAction.createFromJS(fDataActionClass, sFullyQualifiedDataActionClassName, oUpgradedDataActionJS)`.

This command constructs a new instance of your data action from the upgraded data action JSON and returns `oUpgradedDataAction.getSettings()`.

Manage Data Action Extensions

You can upload, download, search for, and delete data action extensions in Oracle Analytics. Developers create data action extensions externally and then import them into Oracle Analytics.

1. On the Home page, click the **Navigator**, and then click **Console**.
2. Click **Extensions**.
3. To upload an extension, click **Upload Extension**, browse to the extension ZIP file, and click **Open** to upload the extension.
4. Perform any of the following tasks.
 - To search for an extension, enter your search criteria in the **Search** field and click **Return** to display search results.
 - To delete an extension, click **Options** on the extension and select **Delete**, and click **Yes** to delete the extension.
 - To download an extension, click **Options** on the extension and select **Download**.

Data Action Extension File Reference

Each data action extension requires a `plugin.xml` file and each `plugin.xml` file can contain any number of data actions.

Topics:

- [Data Action plugin.xml File Example](#)
- [Data Action plugin.xml File Properties Section - tns:obiplugin](#)
- [Data Action plugin.xml File Resources Section - tns:resources](#)
- [Data Action plugin.xml File Extensions Section - tns:extension](#)

Data Action plugin.xml File Example

The plugin.xml file has three main sections, tns:obiplugin, tns:resources, and tns:extension.

Example plugin.xml

This example shows a typical plugin.xml file for one data action.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <tns:obiplugin xmlns:tns="http://plugin.frameworks.tech.bi.oracle"
3     id="obitech-currencyconversion"
4     name="Oracle BI Currency Conversion"
5     version="0.1.0.@qualifier@"
6     optimizable="true"
7     optimized="false">
8
9
10  <tns:resources>
11      <tns:resource id="currencyconversion" path="scripts/
currencyconversion.js" type="script" optimizedGroup="base"/>
12      <tns:resource-folder id="nls" path="resources/nls" optimizable="true">
13          <tns:extensions>
14              <tns:extension name="js" resource-type="script"/>
15          </tns:extensions>
16      </tns:resource-folder>
17  </tns:resources>
18
19
20  <tns:extensions>
21      <tns:extension id="oracle.bi.tech.currencyconversiondataaction" point-
id="oracle.bi.tech.plugin.dataaction" version="1.0.0">
22          <tns:configuration>
23              {
24                  "host": { "module": "obitech-currencyconversion/
currencyconversion" },
25                  "resourceBundle": "obitech-currencyconversion/nls/messages",
26                  "properties":
27                      {
28                          "className": "obitech-currencyconversion/
currencyconversion.CurrencyConversionDataAction",
29                          "displayName": { "key" : "CURRENCY_CONVERSION", "default" :
"Currency Conversion" },
30                          "order": 100
31                      }
32                  }
33          </tns:configuration>
34      </tns:extension>
35  </tns:extensions>
36
37 </tns:obiplugin>

```

Data Action plugin.xml File Properties Section - tns:obiplugin

The `tns:obiplugin` section defines properties common to all types of extensions.

Extension Properties

The `tns:obiplugin` section defines properties common to all types of extensions.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <tns:obiplugin xmlns:tns="http://plugin.frameworks.tech.bi.oracle"
3     id="obitech-currencyconversion"
4     name="Oracle BI Currency Conversion"
5     version="0.1.0.@qualifier@"
6     optimizable="true"
7     optimized="false">
```

- **Line 1:** The XML declaration.
- **Line 2:** The opening tag for the extension's root `XMLElement` and the declaration for the `tns` namespace that's used throughout `plugin.xml` files.
- **Line 3:** The extension's unique ID.
- **Line 4:** The extension's default display name (used when a localized version isn't available).
- **Line 5:** The extension's version number.
- **Line 6:** A boolean indicating whether or not the JS/CSS can be optimized (compressed).
- **Line 7:** A boolean indicating whether or not the JS/CSS has been optimized (compressed).

Data Action plugin.xml File Resources Section - tns:resources

The `tns:resources` section registers all of the files that contribute to your extension.

Resources

```

1 <tns:resources>
2   <tns:resource id="currencyconversion" path="scripts/
currencyconversion.js" type="script" optimizedGroup="base"/>
3   <tns:resource-folder id="nls" path="resources/nls" optimizable="true">
4     <tns:extensions>
5       <tns:extension name="js" resource-type="script"/>
6     </tns:extensions>
7   </tns:resource-folder>
8 </tns:resources>
```

You need to register each JavaScript, CSS, Image, and Translation Resource File here. The section is contained within the `<tns:resources>` element and contains any number of the following elements:

- `<tns:resource>`
These elements are used to register a single file (for example, a JavaScript or CSS file).

- `<tns:resource-folder>`
These elements are used to register all the files under a specified folder at the same time. For example, an image folder or the folder containing the resource files for Native Language Support.

More information on how to register each type of file is provided in the following sections.

JavaScript Files

Each JavaScript file in your extension must be registered with a line similar to the one shown below.

```
<tns:resource id="currencyconversion" path="scripts/currencyconversion.js"
type="script" optimizedGroup="base"/>
```

Where:

- **id** is the ID given to the file.
Set the ID to match the JavaScript filename without the .js extension.
- **path** is the relative path to the JavaScript file from the plugin.xml file. JavaScript files should be stored under your extension's `scripts` directory.
Use all lowercase for your JavaScript files with no special characters (for example, underscore, hyphen).
- **type** is the type of file being registered. It must be set to `script` for JavaScript files.
- **optimizedGroup** groups multiple JavaScript files into a single compressed file. Third-party extensions must leave this set to `base`.

CSS Files

Each CSS file in your extension must be registered with a line similar to the one shown below.

```
<tns:resource id="currencyconversionstyles" path="resources/
currencyconversion.css" type="css"/>
```

Where:

- **id** is the ID given to the file.
Set the ID to match the CSS filename without the .css extension.
- **path** is the relative path to the CSS file from the plugin.xml file. CSS files should be stored under your extension's `resources` directory.
Use all lowercase for your CSS files with no special characters (for example, underscore, hyphen).
- **type** is the type of file being registered. It should always be set to `css` for CSS files.

Image Folders

If your extension has images that you need to refer to from within your JavaScript code, then put them in a `resources/images` directory within your extension's directory structure and add a `<tns:resource-folder>` element to your `plugin.xml` as follows:

```
<tns:resource-folder id="images" path="resources/images" optimizable="false"/>
```

If your images are only referenced by your CSS files, then you don't need to add this `<tns:resource-folder>` element to your `plugin.xml` file. In this case, you must still add

them to the `resources/images` directory so that you can then refer to them using a relative path from your CSS file.

Native Language Support Resource Folders

Oracle Analytics implements Native Language Support. This requires developers to externalize the strings they display in their user interface into separate JSON resource files. You can then provide different localized versions of those files in a prescribed directory structure and Oracle Analytics automatically uses the correct file for the user's chosen language. You can provide as many translated versions of the resource files as needed. A Native Language Support resource folder points Oracle Analytics to the root of the prescribed Native Language Support directory structure used by your extension. All extensions that use Native Language Support resource files must have a `<tns:resource-folder>` entry that looks exactly like the example below.

```
1 <tns:resource-folder id="nls" path="resources/nls" optimizable="true">
2   <tns:extensions>
3     <tns:extension name="js" resource-type="script"/>
4   </tns:extensions>
5 </tns:resource-folder>
```

See [Generated Folders and Files](#) for details about the contents of the files and the prescribed directory structure that you should follow.

Data Action plugin.xml File Extensions Section - `tns:extension`

For each data action you want your extension to provide, you must register a data action extension using a `<tns:extension>` element similar to this:

```
<tns:extension id="oracle.bi.tech.currencyconversiondataaction" point-
id="oracle.bi.tech.plugin.dataaction" version="1.0.0">
  <tns:configuration>
    {
      "host": { "module": "obitech-currencyconversion/currencyconversion" },
      "resourceBundle": "obitech-currencyconversion/nls/messages",
      "properties":
      {
        "className": "obitech-currencyconversion/
currencyconversion.CurrencyConversionDataAction",
        "displayName": { "key" : "CURRENCY_CONVERSION", "default" :
"Currency Conversion" },
        "order": 100
      }
    }
  </tns:configuration>
</tns:extension>
```

Where:

- **id** is the unique ID you give to your data action.
- **point-id** is the type of extension you want to register. For data action extensions, this must be set to `oracle.bi.tech.plugin.dataaction`.

- **version** is the extension API version that your extension definition uses (leave this set to **1.0.0**).

The `<tns:configuration>` element contains a JSON string that defines:

- **host.module** - This is the fully qualified name of the module containing your data action. This fully qualified module name is formulated as `%PluginID%/%ModuleName%`, where:
 - `%PluginID%` must be replaced with the extension ID you specified in the `id` attribute of the `<tns:obiplugin>` element.
 - `%ModuleName%` must be replaced with the resource ID you specified in the `id` attribute of the `<tns:resource>` element for the JavaScript file containing your data action.
- **resourceBundle** - This is the Native Language Support path to the resource file that contains this data action's localized resources. If your resource files are named `messages.js` and stored correctly in the prescribed `nls` directory structure, then set this property to `%PluginID%/nls/messages` (where `%PluginID%` must be replaced with the extension ID you specified in the `id` attribute of the `<tns:obiplugin>` element at the top of the `plugin.xml` file).
- **properties.className** - This is the fully qualified class name given to the data action you're registering. This fully qualified class name is formulated as `%PluginID%/%ModuleName%.%ClassName%`, where:
 - `%PluginID%` must be replaced with the extension ID you specified in the `id` attribute of the `<tns:obiplugin>` element.
 - `%ModuleName%` must be replaced with the resource ID you specified in the `id` attribute of the `<tns:resource>` element for the JavaScript file containing your data action.
 - `%ClassName%` must be replaced with the name you gave to the data action class in your JavaScript file.
- **properties.displayName** - This property contains an object and two further properties:
 - **key** is the Native Language Support message key that can be used to lookup the data action's localized display name from within the specified `resourceBundle`.
 - **default** is the default display name to use if for some reason the localized version of the display name can't be found.
- **properties.order** - This property enables you to provide a hint that's used to determine the position that this data action should appear when shown in a list of data actions. Data actions with lower numbers in their `order` property appear before data actions with higher numbers. When there's a tie, the data actions are displayed in the order they're loaded by the system.

Part III

Embed Content

This part explains how to embed content into applications and web pages.

Topics:

- [About Embedding Oracle Analytics Content into Applications and Web Pages](#)
- [Embed Oracle Analytics Content With iFrames](#)
- [Embed Oracle Analytics Content With the JavaScript Embedding Framework](#)
- [Embed Oracle Analytics AI Assistant With the JavaScript V2 Embedding Framework](#)

3

Get Started Embedding Content into Applications and Web Pages

This chapter contains information that you need to know before you embed content into applications and web pages.

Topics:

- [About Embedding Oracle Analytics Content into Applications and Web Pages](#)
- [Register an Application as a Safe Domain](#)

About Embedding Oracle Analytics Content into Applications and Web Pages

You can embed Oracle Analytics content into an application, custom application, or portal web page.

When you embed analytics, you put information where users need it to make business decisions. Embedded analytics delivers fast time-to-insight and increases user productivity.

There are two analytics content embedding methods:

- Use the analytics content item's URL. Typically this method uses an iFrame. See [Embed Oracle Analytics Content With iFrames](#).
- Use the JavaScript embedding framework when you need an integrated way to embed analytics content. This method provides greater flexibility than the iFrame embedding method. For example, use this method when you want to embed visualizations into a custom web application. See [Typical Workflow to Use the JavaScript Embedding Framework with Oracle Analytics Content](#).

Register an Application as a Safe Domain

Before you can embed Oracle Analytics content into another application, your administrator must register the application's domain as safe.

For security reasons, you're not allowed to add analytics content to an applications unless your administrator considers it safe to do so.

See Register Safe Domains.

Web browsers have become more restrictive about dealing with third party cookies. This restriction can impact embedding projects where the browser won't display your embedded analytics content.

To work around this issue you can use a vanity URL for the Oracle Analytics instance so that it appears to be on the same domain as the domain where you're embedding analytics content. See Set Up a Custom Vanity URL.

Use this information if you're using JavaScript to embed analytics content:

- Due to CORS safeguarding, you can't open your HTML file containing embedded analytics content directly in a browser. To work around this issue you must register the web server (either localhost or another web server) as a safe domain.
- If you use a localhost web server for testing, then you may need to add references to `http://localhost:<port>` and `http://127.0.0.1:<port>`.

You must be an administrator to perform this task.

1. Go to Oracle Analytics, click **Navigator**, and click **Console**.
2. Click **Safe Domains**.
3. Click **Add Domain** and enter the domain.
4. Select **Embedding**.
5. If using compatibility mode with embedding, select **Allow Frames**.

4

Embed Oracle Analytics Content With iFrames

This chapter explains how to use iFrames to embed Oracle Analytics content into applications and web pages.

Topics:

- [Considerations for Embedding Oracle Analytics Content With iFrame](#)
- [Use iFrame to Embed Analytics Content into an Application or Web Page](#)

Considerations for Embedding Oracle Analytics Content With iFrame

This topic describes issues that you might encounter when you use iFrame to embed Oracle Analytics content into applications and web pages.

Typically when users open embedded analytics content from an application, they'll be prompted to log into Oracle Analytics. To avoid this issue, set up single sign-on or user federation between Oracle Analytics and the application hosting the embedded analytics content.

If you're using the Safari browser and the embedded analytics content doesn't display as expected, try disabling Safari's **Prevent cross-site tracking** preference.

Use iFrame to Embed Analytics Content into an Application or Web Page

You can embed your analytics content into an application or web page by adding the target analytics content's URL into an application or portal's iFrame. For example, you can use this method to embed analytics content into Microsoft Teams.

Note

If you need an integrated way to embed analytics content, then use the JavaScript embedding framework. This method provides greater flexibility than the iFrame embedding method. See [Typical Workflow to Use the JavaScript Embedding Framework with Oracle Analytics Content](#).

Before you perform this task, confirm that you've registered the domain that you want to embed your analytics content into as a safe domain. See [Register an Application as a Safe Domain](#).

If you need to manually build the URL, for example to create a URL that includes parameters, be sure to properly escape any characters. All special characters in the URL need to be URL-encoded. For example, use %2C to encode commas and %20 to encode spaces.

1. On the Home page, click **Navigator**, and then click **Catalog**.

2. Locate the item that you want to embed and click **Actions**. Click **Open**.
3. Go to the browser's address bar and copy the item's URL. These are examples of URLs:
 - **Report** - `http://example.com/analytics/saw.dll?PortalGo&path=%2Fshared%2FRevenuehttp://example.com/analytics/saw.dll?PortalGo&Action=prompt&path=%2Fshared%2FSaled%2FSales%20by%20Brand`
 - **Dashboard** - `http://example.com/analytics/saw.dll?Dashboard&PortalPath=%2Fshared%2FSales%2F_portal%2FQuickStart&page=Top%20Products`
 - **Workbook** - `http://example.com/ui/dv/home.jsp?pageid=visualAnalyzer&reportmode=full&reportpath=%2Fshared%2FMySalesWorkbook`
 - **Canvas** - `https://example.com:8080/ui/dv/?pageid=visualAnalyzer&reportmode=full&reportpath=%2F%40Catalog%2Fusers%2Fadmin%2F0AC%20Demo%20Samples%2FCost%20Management%20Analytics%20copy&canvasname=canvas!2.`
See Share a Workbook URL with a Specific Canvas Selected.
4. Alternatively, manually build and then copy the URL to insert into an iFrame.

This is an example of how to construct a URL containing parameters:

```
https://example.com/ui/dv/ui/project.jsp?
pageid=visualAnalyzer&reportmode=full&reportpath=%2F%40Catalog%2Fshared&p1n=pCustomerSegment&p1v=Corporate&p2n=pCity&p2v=Bristol%2CCardiff%2CAustin
```
5. Open the target application or portal, locate an iFrame, and paste the analytics content's URL into it.

5

Embed Oracle Analytics Content With the JavaScript Embedding Framework

This chapter explain how to use the JavaScript embedding framework to embed Oracle Analytics content into applications and web pages.

Topics:

- [About the JavaScript Embedding Framework V2](#)
- [Typical Workflow to Use the JavaScript Embedding Framework with Oracle Analytics Content](#)
- [Enable Oracle Analytics Developer Options](#)
- [Find the Javascript and HTML for Embedding Oracle Analytics Content](#)
- [Prepare the HTML Page for Embedded Oracle Analytics Content](#)
- [Pass Filters to the HTML Page for Embedded Oracle Analytics Content](#)
- [Pass Parameters to the HTML Page for Embedded Oracle Analytics Content](#)
- [Refresh Data in the HTML Page for Embedded Oracle Analytics Content](#)
- [Embed Oracle Analytics Content into a Custom Application That Doesn't Use Oracle JET](#)
- [Embed Oracle Analytics Content into a Custom Application that Uses Oracle JET](#)
- Add Authentication

About the JavaScript Embedding Framework V2

JavaScript Embedding Framework V1 was the original framework for embedding Oracle Analytics content in applications and web pages. JavaScript Embedding Framework V2 is the preferred framework for new integrations.

Note

Oracle Analytics continues to support JavaScript Embedding Framework V1. V1 integrations don't require migration to JavaScript Embedding Framework V2.

V2 provides a lightweight initial loader, avoids adding Oracle Analytics dependencies to the host application namespace, supports automatic selection of the embedding mode, and provides a framework-independent initialization API.

In V1, the embedding page receives RequireJS configuration, a registry, and loader dependencies. V1 integrations also require developers to load Knockout and Oracle JET modules before embedded content can render. V2 retrieves the required implementation as needed and removes that initialization requirement.

V2 supports a language override for embedded user interface elements and messages, so applications can select the embedded-content language rather than always using the browser locale.

Existing V1 integrations continue to use the V1 procedures until they are migrated. The V1 APIs that require RequireJS and Knockout are not used by V2.

For information specific to how to pass filters in the JavaScript Embedding Framework V2, see [About Filters and the JavaScript Embedding Framework V2](#).

Typical Workflow to Use the JavaScript Embedding Framework with Oracle Analytics Content

If you're using the JavaScript embedding framework to embed Oracle Analytics content into an application or web page, then follow these tasks as a guide.

Note

You can also embed Oracle Analytics content using the analytic content item's URL. Typically this method uses an iFrame. See [Embed Oracle Analytics Content With iFrames](#).

Task	Description	More Information
Register the host application	Register development, test, and production host domains as safe domains. If the integration uses an iframe or sandbox option, enable Allow Frames .	Register an Application as a Safe Domain
Enable Developer options and copy the embed markup	Enable the Developer option. In the workbook's Developer page, copy the <oracle-dv> markup and workbook properties that you want to embed.	Enable Oracle Analytics Developer Options Find the Javascript and HTML for Embedding Oracle Analytics Content
Create the HTML page	Create or update the host page. Include <!DOCTYPE html>, set the dir attribute as required, add appropriately sized containers for the embedded content, and reference the V2 embedding.js script.	Prepare the HTML Page for Embedded Oracle Analytics Content
Select a V2 embedding mode	Use jet for an Oracle JET host application, standalone for a non-JET host application, or auto to let Oracle Analytics determine the appropriate mode.	Prepare the HTML Page for Embedded Oracle Analytics Content
Initialize V2	Wait for <code>oracle.oa.embedding.ready()</code> , set page-level configuration as required, and call <code>applyBindings()</code> after the <oracle-dv> elements are present.	Prepare the HTML Page for Embedded Oracle Analytics Content

Task	Description	More Information
Configure authentication	Configure login prompt, 3-legged OAuth, or token authentication as required.	Use Login Prompt Authentication With Embedded Oracle Analytics Content Use 3-Legged OAuth Authentication With Embedded Oracle Analytics Cloud Content Use Token Authentication With Embedded Oracle Analytics Cloud Content
Add embedded content and test	Add <oracle-dv> elements to the host page and test rendering, filters, parameters, refresh, locale, authentication, and frame configuration.	Pass Filters to the HTML Page for Embedded Oracle Analytics Content Pass Parameters to the HTML Page for Embedded Oracle Analytics Content Refresh Data in the HTML Page for Embedded Oracle Analytics Content

Enable Oracle Analytics Developer Options

Enable the developer's options to access the Oracle Analytics Developer's page. Use the Developer's page to find the <script> tag, HTML, and column's expression that you need to embed Oracle Analytics content into an application or web page.

1. Go to the top toolbar and click your user name.
2. Click **Profile** and in the Profile window, click **Advanced**.
3. Click the **Enable Developer Options** icon and click **Save**.

Find the Javascript and HTML for Embedding Oracle Analytics Content

Oracle Analytics generates the <oracle-dv> HTML for you to copy into your custom application or portal page. Use the V2 script URL and initialization pattern in this chapter rather than copying a V1 script reference or RequireJS bootstrap.

If the **Developer** option isn't displayed in the workbook's **Menu**, then you need to enable it. See [Enable Oracle Analytics Developer Options](#).

1. Go to Oracle Analytics and open the workbook containing the analytics content you want to embed.
2. Click **Edit** to enter the workbook in author mode.
3. Click the workbook's **Menu** and then click **Developer**.
4. In the Developer window, click the Embed tab.
5. Locate the **Embedding Script to Include** field and click **Copy** to copy the <script> tag to paste in to the HTML page.
6. Optional: If you want the embedded workbook to show the workbook's default view, then locate the **Default** field, click **Copy** to copy the HTML, and paste it in to the HTML page.

- Optional: If you want the embedded workbook to show an item such as a specific canvas, then locate the item's field, click **Copy** to copy the HTML, and paste it in to the HTML page.

Prepare the HTML Page for Embedded Oracle Analytics Content

To embed Oracle Analytics content, create or update an HTML page that includes the required DOCTYPE declaration, the `dir` global attribute, the JavaScript Embedding Framework V2 script, and one or more `<oracle-dv>` elements. You must select an embedding mode, configure authentication as required, initialize the V2 framework, and add any attributes for the embedded workbook.

This topic contains the following information:

- [DOCTYPE Declaration](#)
- [Dir Global Attribute](#)
- [<script> Tag and JavaScript Source Reference](#)
- [Authentication](#)
- [<oracle-dv> Element](#)
- [Example](#)

Doctype Declaration

Set the doctype declaration to `<!DOCTYPE html>`. This page might not render correctly if you use a different doctype declaration or omit the doctype declaration.

Dir Global Attribute

Set the `dir` global attribute as required by the web page's locale. The `dir` global attribute indicates the layout direction for embedded analytics content.

Note

If you support multiple locales, use JavaScript to set the attribute.

The attribute's value are:

- `rtl` - Use for right to left layout direction.
- `ltr` - Use for left to right layout direction.
- `auto` - Don't use. Oracle Analytics doesn't support this value.

<script> Tag and JavaScript Source Reference

Note

Oracle Analytics generates the `<script>` tag and JavaScript source's URL that you need to include.

Add a `<script>` tag that references the JavaScript Embedding Framework V2 `embedding.js` file to your HTML page.

Use the following structure:

```
<script
  src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/<embedding-mode>/embedding.js"
  type="text/javascript"></script>
```

Specify one of the following values for `<embeddingMode>`:

- `jet`: Use when the host application uses Oracle JET. Oracle Analytics extends the host application with the components required to embed Oracle Analytics content. See [Embed Oracle Analytics Content into a Custom Application that Uses Oracle JET](#).

The application's Oracle JET version must match the same major version of Oracle JET that Oracle Analytics uses.

To find the version of JET that Oracle Analytics uses, log into Oracle Analytics, open the browser console, and run this command:

```
requirejs('ojs/ojcore').version
```

- `standalone` Use when the host application doesn't use Oracle JET. Oracle Analytics loads the implementation and components required to embed Oracle Analytics content. See [Embed Oracle Analytics Content into a Custom Application That Doesn't Use Oracle JET](#).
- `auto`: Use when you want Oracle Analytics to determine if the host application uses Oracle JET and select the appropriate embedding mode.

Language

Optionally append the `lang` query parameter to the V2 script URL to specify the language used for embedded user interface elements and messages. Specify the language as an IETF language tag.

If you don't specify `lang`, Oracle Analytics uses the browser locale.

For example, this script URL requests Canadian French:

```
<script
  src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/auto/embedding.js?lang=fr-CA"
  type="text/javascript"></script>
```

Authentication

You need an authenticated session to view the embedded analytics content. You can use logon prompt, 3-Legged OAuth authentication, or token authentication.

For V2, configure 3-legged OAuth and token authentication through `setEmbeddingConfig()` after the framework is ready. Don't use the deprecated `IDCS_OAUTH3LEGGED=true` or `TOKEN=true` query parameters in V2 script URLs.

See [Add Authentication](#).

<oracle-dv> Element

To embed a workbook, you must add an <oracle-dv> element with the required attribute values inside an appropriately sized element. Oracle Analytics generates the HTML that you need to include.

```
<oracle-dv
  project-path=""
  active-page=""
  active-tab-id=""
  filters="">
</oracle-dv>
```

Supported attributes —

The <oracle-dv> element supports static attribute values and, where supported, values supplied through the embedding application's binding model.

- **project-path:** Specifies the path of the workbook that you want to render.
- **active-page:** (Optional) Specifies if to render an insight other than the default. When you specify **active-page**, you also use **active-tab-id** to specify the exact canvas to show. Valid value is **insight**.

Note

The **canvas** value is deprecated. Update existing embedding code to use **insight**. Existing embeds that use **canvas** continue to work, but a warning displays in the browser console.

- **active-tab-id:** (Optional) Specifies the ID of the canvas to show.
- **filters:** (Optional) Passes filter values to an embedded workbook. This attribute supports only list, number range, and date filters.

Note

Oracle recommends that you use the **filters-v2** attribute instead of **filters**. The **filters-v2** attribute supports more filter types and doesn't reload the entire workbook when filters are modified.

- **filters-v2:** (Optional) Passes filter values to an embedded workbook without reloading the entire workbook when filter values change. Supports all filter types.
- **project-options:** (Optional) In this attribute, *project* refers to *workbook*. Allows you to pass these options:
 - **bDisableMobileLayout:** Enables or disables the mobile layout. Mobile layout refers to the summary card layout available only on phone devices. Specify **true** or **false**.
 - **bShowFilterBar:** Shows or hides the filter bar. Specify **true** or **false**.
 - **showCanvasNavigation:** Shows or hides the canvases in the workbook according to the canvas navigation setting in the workbook tab of the presentation settings page. Specify **true** or **false**.

- `showVizToolbar`: Shows the floating visualization toolbar when users hover over a visualization on a canvas. Specify true or false. If true, the floating toolbar header bar only displays if it's enabled in the presentation settings.
- `showHeaderBar`: Shows the workbook header. Specify true or false. If true, the header bar only displays if it's enabled in the presentation settings.
- `brushing-type`: Controls brushing behavior. The value you specify overrides all other settings, including system defaults and settings stored in the workbook. Value should be the string on, off, or auto.
 - `on`: Issues brushing queries with normal priority. Brushing queries and visualization queries are mixed and run at the same time.
 - `auto`: Default. Issues brushing queries with low priority. When a user interacts with a visualization, there may be a delay showing marks in other visualizations until the brushing queries complete.
- `compatibility-mode`: Use when different major versions of Oracle JET are present. This creates an iFrame at runtime to sandbox the embedded analytics content. Value should be the string yes, no, or auto.

Note

When setting this attribute, note these two items:

If using compatibility mode, confirm that **Allow Frames** is selected for the application your administrator registered as a safe domain. See [Register an Application as a Safe Domain](#).

To find the version of JET that Oracle Analytics uses, log into Oracle Analytics, open the browser console, and run this command:

```
requirejs('ojs/ojcore').version
```

- `yes`: Use when you always want to sandbox the analytics embedded content. This is useful when embedding into Oracle APEX applications.
- `no`: Default. Use when you don't want to create an iFrame.
- `auto`: Use to automatically detect major differences in Oracle JET version between the host embedding application and Oracle Analytics. You can use this when embedding into Oracle APEX.

V2 Initialization Guidance

After the `<oracle-dv>` elements are present in the page, wait for the V2 framework to be ready. Then configure the embedding application as required and call `applyBindings()`.

```
<script>
  oracle.oa.embedding.ready().then((application) => {
    application.applyBindings();
  });
</script>
```

With no arguments, `applyBindings()` processes all `<oracle-dv>` elements in the document. You can optionally provide a model for bindings or a root element to process only the `<oracle-dv>` elements beneath that element.

Example

In this example, all instances of *project* refer to *workbook*.

```
<!DOCTYPE html>
<html dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Embedded Oracle Analytics Workbook Example</title>

    <script
      src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/auto/embedding.js?lang=en-US"
      type="text/javascript"></script>
    </head>

    <body>
      <h1>Embedded Oracle Analytics Workbook</h1>

      <div style="border: 1px solid black; width: calc(100% - 40px); height:
calc(100vh - 120px)">
        <oracle-dv
          project-path="<project path>"
          active-page="insight"
          active-tab-id="snapshot!canvas!1">
        </oracle-dv>
      </div>

      <script>
        oracle.oa.embedding.ready().then((application) => {
          application.applyBindings();
        });
      </script>
    </body>
  </html>
``
```

For examples that use 3-legged OAuth or token authentication, call `setEmbeddingConfig()` before `applyBindings()`.

Pass Filters to the HTML Page for Embedded Oracle Analytics Content

You can pass filters to the HTML pages where you're embedding Oracle Analytics content.

Topics:

- [About Filters and the JavaScript Embedding Framework V2](#)

- [Use the filters Property to Pass Filters](#)
- [Use the filters-v2 Property to Pass Filters](#)
- [Find the JSON Objects for the filters-v2 Property](#)

About Filters and the JavaScript Embedding Framework V2

JavaScript Embedding Framework V1 was the original framework for embedding Oracle Analytics content in applications and web pages. JavaScript Embedding Framework V2 is the preferred framework for new integrations. You can use both the `filters` and `filters-v2` properties with JavaScript Embedding Framework V2.

Keep in mind the following when working with filters in JavaScript Embedding Framework V2:

- Initialize the V2 framework as described in [Prepare the HTML Page for Embedded Oracle Analytics Content](#). Call `application.applyBindings(model)` to provide the property values to an `<oracle-dv>` element.
- The name `filters-v2` identifies the newer filter-payload property. It does not identify the version of the JavaScript Embedding Framework.
- For new integrations, use `filters-v2` when it meets your requirements. It supports more filter types and updates embedded content without reloading the entire workbook.
- If the embedded page uses 3-Legged OAuth or token authentication, call `setEmbeddingConfig()` before `applyBindings()`.

Use the `filters` Property to Pass Filters

You can use the `filters` property to pass only list, number range, and date filters to the HTML page where you're embedding Oracle Analytics content. This property reloads the entire workbook when filters are modified.

Note

Oracle recommends that you use the `filters-v2` property instead of the `filters` property. The `filters-v2` property supports more filter types and doesn't reload the entire workbook when filters are modified. See [Use the filters-v2 Property to Pass Filters](#).

The filters payload is a Javascript array containing one filter Javascript object per array item.

In this example, all instances of *project* refer to *workbook*. Rendering a workbook while applying filters looks like this:

```
<oracle-dv
  project-path="{{projectPath}}"
  filters="{{filters}}">
</oracle-dv>

<script>
  const model = {
    projectPath: "/users/weblogic/EmbeddingStory",
    filters: [
      {
```

```

        "sColFormula": "\"A - Sample Sales\".\"Products\".\"P2 Product
Type\"",
        "sColName": "P2 Product Type",
        "sOperator": "in",
        "isNumericCol": false,
        "bIsDoubleColumn": false,
        "aCodeValues": [],
        "aDisplayValues": ["Audio", "Camera", "LCD"]
    },
    {
        "sColFormula": "\"A - Sample Sales\".\"Base Facts\".\"1- Revenue\"",
        "sColName": "Rev",
        "sOperator": "between",
        "isNumericCol": true,
        "bIsDoubleColumn": false,
        "aCodeValues": [],
        "aDisplayValues": [0, 2400000]
    }
]
};

oracle.oa.embedding.ready().then((application) => {
    application.applyBindings(model);
});
</script>

```

Supported attributes — Each filter object within the filters payload must contain the following attributes:

- **sColFormula:** Specifies the three-part formula of the column to filter. The column formula must include three parts.

If you're unsure of the formula, create a workbook that uses that column, and then in the workbook editor, click the workbook's **Menu** and then click **Developer**. In the Developer page, click the JSON tab to view the column's expression. For example, `sColFormula`: `"\"A - Sample Sales\".\"Base Facts\".\"1- Revenue\""`.

If the **Developer** option isn't displayed in the workbook's **Menu**, then you need to enable it. See [Enable Oracle Analytics Developer Options](#).
- **sColName:** (Required) Specifies a unique name for this column.
- **sOperator:** Use `in`, `notIn`, `between`, `less`, `lessOrEqual`, `greater`, or `greaterOrEqual`.
 - `in` and `notIn` - Apply to list filters.
 - `between`, `less`, `lessOrEqual`, `greater`, and `greaterOrEqual` - Apply to numeric filters.
- **isNumericCol:** Specifies if the filter is numeric or list. Value should be `true` or `false`.
- **isDateCol:** (Required) Indicates whether the column is a date column. Value should be `true` or `false`. Use `true` if the column is a date, but not for year, month, quarter, and so on. If set to `true`, then specify date or dates in the `aDisplayValues` attribute.
- **bIsDoubleColumn:** Specifies if the column has double column values behind the display values. Value should be `true` or `false`.
- **aCodeValues:** When `bIsDoubleColumn` is `true`, this array is used.
- **bHonorEmptyFilter:** (Optional) Indicates whether an empty filter is honored (for example, empty `aCodeValues`/`aDisplayValues` based on the `bIsDoubleColumn` flag). This attribute

applies to all column filters: list filters, number range filters, and date range filters. Value should be true or false.

- If set to true and the user passes empty `aCodeValues/aDisplayValues`, then all values are part of the filter.
- If set to false and user passes empty `aCodeValues/aDisplayValues`, then the attribute won't be applied and there is no change in filter values.
- If this attribute isn't present, then the default value is false.
- `aDisplayValues`: When `bIsDoubleColumn` is false, then this array is used to filter and to display values within the user interface.

When `bIsDoubleColumn` is true, then the values in this array are used for display in the user interface while the values in `aCodeValues` are used for filtering. When `bIsDoubleColumn` is true, there must be the same number of entries in this array as there are in the `aCodeValues` array and the values must line up. For example, suppose `aCodeValues` has two values 1 and 2, then `aDisplayValues` must have two values a and b, where 1 is the code value for a, and 2 is the code value for b.

If `isDateCol` attribute is set to true, then specify the `aDisplayValues` array with dates. If either no time zone in the time stamp or no time stamp is provided, then time is set with the local time zone. Use any of the following formats:

- mm/dd/yyyy (For example, 12/31/2011.)
- yyyy-mm-dd (For example, 2011-12-31.)
- yyyy/mm/dd (For example, 2011/12/31.)
- mm/dd/yyyy or yyyy/mm/dd, hh:mm:ss (For example, 12/31/2011 or 2011/12/31, 23:23:00.)
Note: Use a 24 hour format. You can use a space as the separator.
- mm/dd/yyyy or yyyy/mm/dd, hh:mm:ss AM/PM (For example, 12/31/2011 or 2011/12/31, 11:23:00 PM.)
Note: Use a 12 hour format. You can use a space as the separator.
- <3 letter month name> dd yyyy (For example, Mar 25 2015.)
- dd <3 letter month name> yyyy (For example, 25 Mar 2015.)
- Fri Sep 30 2011 05:30:00 GMT+0530 (India Standard Time)
- ISO Date Format - 2011-10-05T14:48:00.000Z

Use the `filters-v2` Property to Pass Filters

You can use the `filters-v2` property to programmatically pass filter values to the HTML page where you embed Oracle Analytics content. This option passes filter values immediately to embedded content without reloading the entire workbook. You can use this property to filter any type of data.

The `filters-v2` property requires an array of `FilterControl` JSON objects. You can obtain this array from a workbook that contains existing filters of the types you want to embed. Or you can build temporary filters in a workbook to generate the needed array, and after you copy the array for embedding, you can delete the filter from the workbook.

For more information, see About Filters and [Find the JSON Objects for the `filters-v2` Property](#)

The `filters-v2` property can modify filters that already exist in the workbook. If filters don't already exist in the workbook, this property creates them.

In this example, all instances of *project* refer to *workbook*. Rendering a workbook while applying filters looks like this:

```
<oracle-dv
  project-path="{{projectPath}}"
  filters-v2="{{filtersV2}}">
</oracle-dv>

<script>
  const model = {
    projectPath: "/users/weblogic/EmbeddingStory",
    filtersV2: [
      /* Paste the FilterControl JSON objects obtained from the workbook
here. */
    ]
  };

  oracle.oa.embedding.ready().then((application) => {
    application.applyBindings(model);
  });
</script>
```

FilterControl JSON Example

```
[
  {
    "filterID": "8e24a0b0-04d3-48d3-9a0e-eb8fba77739e",
    "columnID": "P4Brand",
    "type": "saw:columnFilterControl",
    "filterControlConfig": {
      "_version": "1.0.8",
      "settings": {
        "filterModelClassName": "obitech-listfilter/
listfilter.ListFilterModel",
        "location": "filter_bar"
      }
    },
    "address": 0.0001,
    "formula": {
      "expr": {
        "children": [
        ],
        "type": "sawx:sqlExpression",
        "expression": "\"A - Sample Sales\".\"Products\".\"P4 Brand\""
      }
    },
    "filterOperator": {
      "op": "in"
    },
    "filterControlDefaultValues": {
      "children": [
        {
          "text": "FunPod"
        }
      ]
    }
  ],
]
```

```

        "type": "specificValue"
    },
    "filterControlSource": {
        "type": "saw:fcSpecificChoices",
        "filterControlChoices": {
            "children": [
                {
                    "value": {
                        "text": "FunPod"
                    }
                }
            ]
        }
    }
},
{
    "filterID": "ef6c6429-9d64-481e-9bf8-da0b334c2fb8",
    "columnID": "T05PerNameYear",
    "type": "saw:columnFilterControl",
    "filterControlConfig": {
        "_version": "1.0.8",
        "settings": {
            "filterModelClassName": "obitech-listfilter/
listfilter.ListFilterModel",
            "location": "filter_bar"
        }
    },
    "address": 0.0001,
    "formula": {
        "expr": {
            "children": [
                ],
            "type": "sawx:sqlExpression",
            "expression": "\"A - Sample Sales\".\"Time\".\"T05 Per Name
Year\""
        }
    },
    "filterOperator": {
        "op": "in"
    },
    "filterControlDefaultValues": {
        "children": [
            {
                "text": "2009"
            },
            {
                "text": "2010"
            }
        ],
        "type": "specificValue"
    },
    "filterControlSource": {
        "type": "saw:fcSpecificChoices",
        "filterControlChoices": {
            "children": [
                {

```

```

        "value": {
          "text": "2009"
        }
      },
      {
        "value": {
          "text": "2010"
        }
      }
    ]
  }
}
]

```

Properties — Each filter object within the filters payload must contain the following properties:

- **filterID**: Specifies a unique filter ID. For example, "8e24a0b0-04d3-48d3-9a0e-eb8fba77739e".
- **address**: Don't change this value. Use the value exactly as exported from the JSON you copied from the workbook.
- **filterOperator**: Specifies if the filter definition includes or excludes values. Use `in` to include values. Use `notIn` to exclude values.
- **filterControlDefaultValues**: Specifies the values used in the filter. These are the values the embedded content is filtered by.
- **filterControlSource**: Lists which of the `filterControlDefaultValues` are enabled, so all values that *can be* in the filter selection list. Note that a workbook's filter control can contain selected values that the workbook author disabled in the filter's value Selections list.

For example, suppose you open the workbook in the Workbook editor and the column you're filtering is State and the selected filter values are AK, AL, and AR, where the AK value is disabled. The `filterControlDefaultValues` contains AL and AR (but not AK because it is disabled in the filter selection list), and `filterControlSource` contains AK, AL, and AR.

This example shows how to create an expression filters on a measure value, so Limit Sales to less than 470000.

```

{
  "filterID": "filter_sales",
  "type": "saw:expressionFilterControl",
  "label": {
    "caption": {
      "text": "Sales < 470000"
    }
  },
  "expr": {
    "type": "sawx:sql",
    "expression": "XSA('<name>'. 'Order Activity').\"Columns\".\"Sales\" < 470000"
  },
  "filterControlConfig": {
    "_version": "1.0.8",

```

```

    "settings": {
      "filterModelClassName": "obitech-expressionfilter/
expressionfilter.ExpressionFilterModel",
      "location": "filter_bar"
    }
  }
}

```

Find the JSON Objects for the `filters-v2` Property

The `filters-v2` property requires an array of `FilterControl` JSON objects. Use the JSON tab in the workbook editor to get the initial filter syntax.

If the **Developer** option isn't displayed in the workbook's **Menu**, then you need to enable it. See [Enable Oracle Analytics Developer Options](#).

Make sure to copy each required `FilterControl` JSON object exactly as it is exported, then use the array as the value of the V2 model property bound to `filters-v2`. Don't change generated identifiers, formulas, or `address` values unless you understand the effect on the filter definition.

For the V2 `<oracle-dv>` binding and initialization example, see [Use the `filters-v2` Property to Pass Filters](#).

1. Go to your Oracle Analytics home page and open the workbook containing the filters that you want to embed.
2. Click **Edit** to enter the workbook in author mode.
3. Click the workbook's **Menu** and then click **Developer**.
4. In the Developer window, click the JSON tab.
5. Copy the contents of the JSON tab and paste them into a text editor.
6. In the text editor, search for the `filterControlCollections` section.
7. Within the `filterControlCollections` section, locate `filterControls`. Within the `children` subsection locate the filter definition. Note that each definition starts with a unique `filterID`.

For example:

```

{
  "filterControlCollections": {
    "children": [
      {
        "name": "canvas 1",
        "filterControls": {
          "children": [
            {
              "filterID": "a18ae41b-5a32-409b-a4e7-61df959dcbcd",
              "columnID": "OrderPriority",
              "type": "saw:columnFilterControl",
              "filterControlConfig": {
                "_version": "1.0.9",
                "settings": {
                  "filterModelClassName": "obitech-listfilter/
listfilter.ListFilterModel",

```

```

        "location": "filter_bar"
      }
    },
    "address": 0.0001,
    "formula": {
      "expr": {
        "children": [],
        "type": "sawx:sqlExpression",
        "expression": "XSA('90e7ee56-5b6c-4106-bc9d-a23a9b38ba94'. 'sr_sample_order_lines_2023').\"sample_order_lines2023\".\"Order Priority\""
      }
    },
    "filterOperator": {
      "op": "in"
    },
    "filterControlDefaultValues": {
      "children": [
        {
          "text": "High"
        }
      ],
      "type": "specificValue"
    }
  },

```

Pass Parameters to the HTML Page for Embedded Oracle Analytics Content

You can pass parameter values to the HTML page where you're embedding Oracle Analytics content. The parameter values that you pass can be utilized within query expressions and various parts of the product.

The parameters payload is a Javascript Object containing paired attributes of parameter names and values.

In this example, all instances of *project* refer to *workbook*. Rendering a project while applying parameters look like this:

```

<oracle-dv
  project-path="{{projectPath}}"
  active-page="insight"
  active-tab-id="3"
  parameters="{{parameters}}"
  project-options='{"bDisableMobileLayout": false, "bShowFilterBar": false}'>
</oracle-dv>

```

```

<script>
const model = {
  projectPath: "/users/weblogic/EmbeddingStory",
  parameters: {
    "p1n": "Office",
    "p1v": "Bristol Office",
    "p2n": "Year",
    "p2v": [2023, 2022]
  }
}

```

```

    }
  };

  oracle.oa.embedding.ready().then((application) => {
    application.applyBindings(model);
  });
</script>

```

Supported attributes — Each parameter object within the parameters payload must contain the following attributes:

- `p <number> n`: (Required) Specifies the parameter's name as defined in the workbook. For example, "Office" or "Year".
- `p <number> v`: (Required) Specifies the parameter value that you want to pass. For example "Bluebell Office" or "10" or [2023, 2022].
- `p <number> d`: (Optional) Use with parameters with double columns. Specifies the parameter's display value corresponding to `p <number> v`. For example, "My Office".

Refresh Data in the HTML Page for Embedded Oracle Analytics Content

In the HTML page where you're embedding Oracle Analytics content, you can specify how to refresh the embedded workbook's data.

To refresh data without reloading a workbook, the `refreshData` function is available on each `<oracle-dv>` element. Calling this function refreshes all visualizations under that element.

In this example, all instances of *project* refer to *workbook*.

```

<oracle-dv
  id="project1"
  project-path="{{projectPath}}">
</oracle-dv>

<script>
  function refreshProject() {
    const embeddedWorkbook = document.querySelector("#project1");

    if (embeddedWorkbook) {
      embeddedWorkbook.refreshData();
    }
  }
</script>

```

This is the code to refresh data for multiple embedded workbooks. In this code, all instances of *project* refer to *workbook*.

```

<script>
  function refreshAllProjects() {
    document.querySelectorAll("oracle-dv").forEach((embeddedWorkbook) => {
      embeddedWorkbook.refreshData();
    });
  }

```

```
}
</script>
```

Embed Oracle Analytics Content into a Custom Application that Uses Oracle JET

If the custom application uses Oracle JET, then the embedded Oracle Analytics content extends the application with the component it needs.

Before you begin to embed analytics content, confirm that the custom application uses the same major version of JET that Oracle Analytics uses.

To find the version of JET that Oracle Analytics uses, log into Oracle Analytics, open the browser console, and run this command:

```
requirejs('ojs/ojcore').version
```

This procedure uses an example embedding application named OAJETAPP.

1. Create or open an Oracle JET application. If you use an Oracle JET quickstart application, you can use create an application named OAJETAPP using the navdrawer template.
2. Add the JavaScript Embedding Framework V2 script to the application HTML page, for example, OAJETAPP/src/index.html.

```
<script src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/jet/embedding.js" type="text/javascript">
</script>
```

3. Optionally append lang=<IETF-language-tag> to the script URL to specify the language used for embedded user interface elements and messages.

```
<script
  src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/jet/embedding.js?lang=en-US"
  type="text/javascript"></script>
```

4. Include <oracle-dv> in the appropriate application view, for example OACJETAPP/src/js/views/dashboard.html. The project-path attribute specifies the path of the workbook to render.

```
<div class="oj-hybrid-padding"
  style="position: absolute; width: calc(100% - 40px); height:
calc(100% - 120px)">
  <h3Dashboard Content Area</h3>

  <oracle-dv
    id="oracle-dv"
    project-path="/Shared Folders/embed/test-embed">
  </oracle-dv>
</div>
```

5. Initialize JavaScript Embedding Framework V2 after the <oracle-dv> element is present.

```
<script>
  oracle.oa.embedding.ready().then((application) => {
    application.applyBindings();
  });
</script>
```

oracle.oa.embedding.ready() returns a promise that resolves when the V2 framework is ready. Calling application.applyBindings() processes the <oracle-dv> elements in the page.

6. If your Oracle JET application calls ko.applyBindings() itself, wait for oracle.oa.embedding.ready() before calling the host application's binding function.

```
oracle.oa.embedding.ready().then(() => {
  ko.applyBindings(viewModel, document.getElementById('globalBody'));
});
```

7. Configure authentication as required.

- For V2, configure token authentication or 3-legged OAuth through application.setEmbeddingConfig() after oracle.oa.embedding.ready() resolves and before you call applyBindings().
- Don't use the deprecated TOKEN=true or IDCS_OAUTH3LEGGED=true script query parameters in V2 examples.

8. Build and run the Oracle Jet application.

```
ojet build
ojet serve
```

Complete Example

```
<!-- OAJETAPP/src/index.html -->
<script
  src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/jet/embedding.js"
  type="text/javascript"></script>
...

```html
<!-- OAJETAPP/src/js/views/dashboard.html -->
<div class="oj-hybrid-padding"
 style="position: absolute; width: calc(100% - 40px); height: calc(100% -
120px)">
 <h3>Dashboard Content Area</h3>

 <oracle-dv
 id="oracle-dv"
 project-path="/Shared Folders/embed/test-embed">
 </oracle-dv>
</div>
...

```javascript
```

```
// Initialize after the oracle-dv element is present.
oracle.oa.embedding.ready().then((application) => {
  application.applyBindings();
});
```

Embed Oracle Analytics Content into a Custom Application That Doesn't Use Oracle JET

Use JavaScript Embedding Framework V2 to embed Oracle Analytics content in a custom application that doesn't use Oracle JET.

If the custom application uses a technology other than Oracle JET, use the standalone V2 embedding mode. V2 first loads a lightweight entry script and then loads the embedding implementation required by the page. This approach avoids adding Oracle Analytics dependencies, such as RequireJS, Knockout, and jQuery, to the host application namespace.

If the **Developer** option isn't displayed in the workbook's **Menu**, then you need to enable it. See [Enable Oracle Analytics Developer Options](#).

1. Include the standalone V2 version of `embedding.js`. Note that if the host framework varies between Oracle JET and non-JET implementations, you can use the auto embedding mode instead.

```
<script
  src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/standalone/embedding.js"
  type="text/javascript"></script>
```

2. Optionally, append `lang=<IETF-language-tag>` to the script URL to specify the language used for embedded user interface elements and messages.

```
<script
  src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/standalone/embedding.js?lang=en-US"
  type="text/javascript"></script>
```

3. Find and include an `<oracle-dv>` element inside an appropriately sized container. To find this tag:
 - a. Go to Oracle Analytics and open the workbook containing the analytics content you want to embed.
 - b. Click **Edit** to enter the workbook in author mode.
 - c. Click the workbook's **Menu** and then click **Developer**.
 - d. Click the **Embed** tab.
 - e. Locate the item you want to embed and click **Copy** to copy it.

Example

```
<div style="width: calc(100% - 40px); height: calc(100vh - 120px)">
  <oracle-dv project-path="/@Catalog/users/admin/workbook_name">
    </oracle-dv>
  </div>
```

Here `project-path` specifies the workbook's path.

4. Initialize JavaScript Embedding Framework V2 after the `<oracle-dv>` element is present.

```
<script>
  oracle.oa.embedding.ready().then((application) => {
    application.applyBindings();
  });
</script>
```

`oracle.oa.embedding.ready()` returns a promise that resolves when the V2 framework is ready. Calling `application.applyBindings()` processes the `<oracle-dv>` elements in the page.

You can optionally provide a binding model or a root element to `applyBindings()` when the application uses bindings or must process only a specific section of the page.

5. Configure authentication as required. For V2, configure 3-legged OAuth or token authentication through `application.setEmbeddingConfig()` after `oracle.oa.embedding.ready()` resolves and before calling `applyBindings()`.

Don't use the deprecated `TOKEN=true` or `IDCS_OAUTH3LEGGED=true` query parameters in V2 script URLs.

Complete Example

Here `project-path` specifies the workbook's path.

```
<!DOCTYPE html>
<html dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Standalone Oracle Analytics V2 Embed Demo</title>

    <script
      src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/standalone/embedding.js"
      type="text/javascript"></script>
    </head>

    <body>
      <h1>Embedded Oracle Analytics Workbook</h1>

      <div style="width: calc(100% - 40px); height: calc(100vh - 120px)">
        <oracle-dv project-path="/shared/embed/test-embed">
        </oracle-dv>
      </div>

      <script>
        oracle.oa.embedding.ready().then((application) => {
          application.applyBindings();
        });
      </script>
    </body>
  </html>
```

Add Authentication to an Application or Web Page Containing Embedded Oracle Analytics Content

Use the topics in this section to add an authentication method to your web application or portal web page that contains embedded Oracle Analytics content.

Topics:

- [Use Login Prompt Authentication With Embedded Oracle Analytics Content](#)
- [Use 3-Legged OAuth Authentication With Embedded Oracle Analytics Cloud Content](#)
- [Use Token Authentication With Embedded Oracle Analytics Cloud Content](#)

Use Login Prompt Authentication With Embedded Oracle Analytics Content

Login prompt authentication is the default authentication method for Oracle Analytics content embedded in a web application or portal web page.

When users access embedded analytics content, they're presented with a login screen where they enter login name and password before they can view data. If there is no common identity management between Oracle Analytics and the web application or portal web page, then users are presented with this login screen, even if they've already logged in to the web application or portal web page containing the embedded analytics content

Customize the Login Prompt Authentication Message

Add attributes to the <oracle-dv> tag to customize the login prompt authentication messages. The following attributes are supported:

- `auth-message-prefix`: Specifies the prefix text for the login message. The default value is "Oracle Analytics".
- `auth-message-link`: Specifies the text for the login link. The default value is "Login".
- `auth-message-suffix`: Specifies the suffix text for the login message. The default value is "Required".
- `auth-needed-message`: Specifies the text for the authentication needed message. The default value is "Requires Authentication".
- `auth-message-prefix-small`: Specifies the prefix text for the login message. The default value is "Oracle Analytics". Applicable only if the embedded container size is smaller than 215 pixels.
- `auth-message-link-small`: Specifies the text for the login link. The default value is "Login". Applicable only if the embedded container size is smaller than 215 pixels.
- `auth-message-suffix-small` - Specifies the suffix text for the login message. The default value is the empty string. Applicable only if the embedded container size is smaller than 215 pixels.
- `auth-needed-message-small`: Specifies the text for the authentication needed message. The default value is "Requires Authentication". Applicable only if the embedded container size is smaller than 160 pixels.

Use 3-Legged OAuth Authentication With Embedded Oracle Analytics Cloud Content

Use the 3-Legged OAuth authentication method when embedding Oracle Analytics Cloud content in a portal or web application that already uses its own authentication method.

For a seamless user experience, the custom web page and Oracle Analytics Cloud must use the same authentication provider. For example, a third-party application can use the same Oracle Cloud identity provider as Oracle Analytics Cloud, or Oracle Analytics Cloud can be federated with the third-party application's identity provider.

With V2, do not add the deprecated `IDCS_OAUTH3LEGGED=true` parameter to the `embedding.js` script URL. After the V2 embedding framework is ready, configure the application to use `oauth_3legged` authentication, then apply bindings.

1. Load the V2 embedding script. Select the embedding mode that matches the host application. For example, use `standalone` for an application that does not use Oracle JET.
2. When `oracle.oa.embedding.ready()` resolves, call `setEmbeddingConfig()` and specify `oauth_3legged` as the security configuration type.
3. Apply bindings after the security configuration is set.
4. Configure 3-legged OAuth after JavaScript Embedding Framework V2 is ready.

```
<script>
  oracle.oa.embedding.ready().then((application) => {
    application.setEmbeddingConfig({
      oSecurityConfig: {
        eSecurityType: "oauth_3legged"
      } });

    application.applyBindings();
  });
</script>
```

Example

Here `project-path` specifies the workbook's repository path.

```
<!DOCTYPE html>
<html dir="ltr">
<head>
  <script
    src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/auto/embedding.js"
    type="application/javascript">
  </script>
</head>
<body>
  <div style="position: absolute; width: calc(100% - 40px); height: calc(100%
- 120px)">
    <oracle-dv project-path="/Shared Folders/Embed/Embed Samples">
    </oracle-dv>
  </div>
```

```
<script>
  oracle.oa.embedding.ready().then(function (application) {
    application.setEmbeddingConfig({
      oSecurityConfig: {
        sType: "oauth_3legged"
      }
    });

    application.applyBindings();
  });
</script>
</body>
</html>
```

Use Token Authentication With Embedded Oracle Analytics Cloud Content

Use token authentication when you want to authenticate users to Oracle Analytics Cloud in the background, but don't want to use 3-Legged OAuth.

When you authenticate embedded Oracle Analytics content with tokens, client-side session cookies aren't created. This can prevent third-party-cookie restrictions from blocking embedded content. Your content can be displayed without reconfiguring browsers or configuring Oracle Analytics to use a vanity URL in the same domain as the host embedding site. This benefit applies only to JavaScript framework embedding, not to embedding by workbook URL.

Note

Export to Excel doesn't work correctly when you use token authentication with embedded Oracle Analytics Cloud content.

OAuth 2.0 Token Authentication

This authentication type requires a bearer token. Go to the Access Tokens tab on your **Profile** page to find the required token. See [Obtain OAuth 2.0 Access Tokens for Oracle Analytics Cloud](#).

Note

You must obtain the token with a user context. This means you can't use some grant types, such as the Client Credentials grant type, with embedded content.

Update the HTML page to allow for proper token authentication. For information about how to generate tokens, see [Securing Authorizations in Oracle Cloud](#).

JavaScript Embedding Framework V2

With V2, don't add the deprecated `TOKEN=true` parameter to the `embedding.js` script URL. After the V2 embedding framework is ready, use `setEmbeddingConfig()` to configure token authentication. Specify a `tokenAuthFunction` that returns either a token string or a Promise that resolves to a token string.

1. Load the V2 embedding script. Select the embedding mode that matches the host application. For example, use `standalone` for an application that doesn't use Oracle JET.

2. When `oracle.oa.embedding.ready()` resolves, call `setEmbeddingConfig()` and specify token as the security configuration type.
3. Apply bindings after the security configuration is set.

Example

This example uses an API to obtain the token. If your HTML page uses an API to obtain the token, then you must make the required API available.

Here `project-path` specifies the workbook's repository path.

```
<!DOCTYPE html>
<html dir="ltr">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Standalone DV Embed Demo Using Token</title>

  <script
    src="https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/
embedding/auto/embedding.js"
    type="application/javascript">
  </script>
</head>
<body>
  <b>Standalone embedded workbook test</b>

  <div style="width: calc(50% - 40px); height: 50%; border: 1px solid black;
padding: 10px;">
    <oracle-dv
      project-path="/@Catalog/Shared Folders/Embed/Embed Samples"
      active-page="insight"
      active-tab-id="1">
    </oracle-dv>
  </div>

  <script>
    function getToken() {
      return fetch("/api/analytics-token")
        .then(function (response) {
          if (!response.ok) {
            throw new Error("Unable to obtain an Oracle Analytics access
token.");
          }
          return response.text();
        });
    }

    oracle.oa.embedding.ready().then(function (application) {
      application.setEmbeddingConfig({
        oSecurityConfig: {
          sType: "token",
          tokenAuthFunction: getToken
        }
      });

      application.applyBindings();
    });
  </script>
</body>
</html>
```

```
});  
</script>  
</body>  
</html>
```

6

Embed Oracle Analytics AI Assistant With the JavaScript V2 Embedding Framework

This chapter explains how to use the JavaScript V2 embedding framework to embed the Oracle Analytics AI Assistant into applications and web pages.

Topics:

- [Prepare the HTML Page for Embedding the Oracle Analytics AI Assistant](#)
- [Permission Required to Use Embedded AI Assistant](#)

Prepare the HTML Page for Embedding the Oracle Analytics AI Assistant

To embed the Oracle Analytics AI Assistant, you must create or update the HTML page to use the embedding V2 method and include the required `<oracle-analytics-assistant>` embedding tag.

This topic contains the following information:

- [Doctype Declaration](#)
- [Dir Global Attribute](#)
- [<script> Tag and JavaScript Source Reference](#)
- [<oracle-analytics-assistant> Element](#)
- [Basic Example](#)
- [Example With Tokens](#)

Doctype Declaration

Set the doctype declaration to `<!DOCTYPE html>`. Unpredictable behavior such as the page not rendering properly results if you use a doctype declaration other than `<!DOCTYPE html>`, or if you forget to include a doctype declaration.

Dir Global Attribute

Set the `dir` global attribute as required by the web page's locale. The `dir` global attribute indicates the embedded analytics content's layout direction.

Note

If you need to support multiple locales, then use JavaScript to set the attribute.

The attribute's value options are:

- `rtl` - Use for right to left layout direction.
- `ltr` - Use for left to right layout direction.
- `auto` - Don't use. This value isn't supported by Oracle Analytics.

<script> Tag and JavaScript Source Reference

Add a `<script>` tag that references `embedding.js` to your HTML page.

The JavaScript source's URL structure is:

- `"https://<instance>.analytics.ocp.oraclecloud.com/public/dv/v2/embedding/auto/embedding.js"`

<oracle-analytics-assistant> Element

To embed the assistant, you must add the following HTML snippet with attribute values inside an appropriately sized element. Oracle Analytics generates the HTML that you need to include. The `src-data` parameter is the dataset that the AI assistant uses.

```
<oracle-analytics-assistant
  src-data="[&quot;XSA('<user-name>'. '<data source name>')&quot;]">
</oracle-analytics-assistant>
```

Basic Example

```
<!DOCTYPE html>
<html dir="ltr">
<head>
  <meta charset="utf-8">
  <script src="https://<your-OAC-instance>.analytics.ocp.oraclecloud.com/
public/dv/v2/embedding/auto/embedding.js"></script>
  <style>
    html, body {
      margin: 0;
      height: 100%;
    }
    oracle-analytics-assistant {
      width: 100%;
      height: 100%;
      display: block;
    }
  </style>
</head>

<body>
  <oracle-analytics-assistant
    id="oaAssistant"
    src-data="[&quot;XSA('<user-name>'. '<data source name>')&quot;]">
  </oracle-analytics-assistant>

  <script>
    oracle.oa.embedding.ready().then(oApp => oApp.applyBindings());
  </script>
</body>
</html>
```

Example With Tokens

```
<!DOCTYPE html>
<html dir="ltr">
<head>
  <meta charset="utf-8">
  <script src="https://<your-OAC-Instance>.analytics.ocp.oraclecloud.com/
public/dv/v2/embedding/auto/embedding.js"></script>
  <style>
    html, body {
      margin: 0;
      height: 100%;
    }
    oracle-analytics-assistant {
      width: 100%;
      height: 100%;
      display: block;
    }
  </style>
</head>
<body>
  <oracle-analytics-assistant
    id="oaAssistant"
    <!-- Reference to datasource to use with AI Assistant with " escaped to
    &quot; -->
    src-data="['&quot;XSA('<user-name>'. '<data source name>')&quot;]'">
  </oracle-analytics-assistant>

  <script>
    const token = '<access token>';

    oracle.oa.embedding.ready().then((oApp) => {
      oApp.setEmbeddingConfig({
        oSecurityConfig: {
          eSecurityType: "token",
          oOptions: {
            credentials: "",
            tokenAuthFunction: new Function("return '" + token + '"")
          }
        }
      });
      oApp.applyBindings();
    });
  </script>
</body>
</html>
```

Permission Required to Use Embedded AI Assistant

Confirm with your administrator that the User Assistant in Workbooks permission is assigned to the appropriate application roles. This permission enables users assigned to those roles access to the embedded AI Assistant.

By default, the DV Content Author application role contains the Use Assistant in Workbooks permission.

In some cases your Administrator might want to create a custom application role and assign the Use Assistant in Workbooks permission to it.

For more information about application roles and permissions, see [About Permissions](#) and [About Application Roles](#).

Part IV

Connect Oracle Analytics to AI

This part explains how to connect Oracle Analytics to AI, such as AI agentic workflows, applications, agents, MCP hosts/clients, and so on.

Chapters:

- [Get Started with Oracle Analytics Cloud MCP Tools \(Preview\)](#)
- [Oracle Analytics Cloud MCP Tools Developer's Reference \(Preview\)](#)

Get Started with Oracle Analytics Cloud MCP Tools (Preview)

The September 2026 Oracle Analytics Cloud release is based on MCP Protocol 2025-11-25.

This chapter explains the Oracle Analytics Cloud MCP tools and helps you start using them. It describes how to connect MCP clients to Oracle Analytics Cloud, introduces the available tools, explains the MCP request and session model, and provides recommended workflows, best practices, and troubleshooting information for working safely and effectively with OAC MCP tools.

The information in this guide is intended for users who want to discover catalog assets, inspect data models, run ANSI SQL or Logical SQL, create or update catalog content, export workbooks, and manage catalog items through JSON-RPC MCP calls.

Caution

Oracle Analytics provides a Model Context Protocol (MCP) server that allows AI clients direct access and interaction with Oracle Analytics data and functionality. This introduces security risks since you might inadvertently expose unintended data.

The Oracle Analytics MCP tools are authenticated with an OAuth 2.0 access token and run with the same permissions as the user who obtains the access token. Ensure that any users who use the MCP server only have access to data that can be shared with LLMs.

MCP is a new protocol and a dynamic area that is constantly improving. While the Oracle Analytics Cloud endpoints are supported and available for use in your projects, there may be some changes as Oracle improves this area. When using the MCP server in your agents and AI applications, Oracle recommends implementing evaluations and observability to ensure reliability.

Topics:

- [About oac-mcp-connect \(Preview\)](#)
- [Available Oracle Analytics Cloud MCP Tools \(Preview\)](#)
- [Visible MCP Tool Metadata \(Preview\)](#)
- [JSON-RPC Shape \(Preview\)](#)
- [HTTP Session Initialization \(Preview\)](#)
- [Common Workflows \(Preview\)](#)
- [Safety Rules \(Preview\)](#)
- [Troubleshooting \(Preview\)](#)
- [Quick Reference \(Preview\)](#)
- [Deprecated MCP Server Tools \(Preview\)](#)

About oac-mcp-connect (Preview)

Oracle Analytics Cloud can provide an oac-mcp-connect utility from the user's profile when developer options are enabled. The utility is a local stdio bridge for Oracle Analytics Cloud MCP servers and can handle browser-based or token-file authentication on behalf of the MCP client.

Use oac-mcp-connect when:

- You want to connect OAC MCP tools to desktop MCP clients such as Claude Desktop.
- You're using a client that doesn't directly manage MCP HTTP session initialization and Mcp-Session-Id headers.

The built bundle runs with Node.js 18 or newer. The detailed README is in the downloaded utility bundle.

High-Level Codex Configuration Shape

```
[mcp_servers.oac]
enabled = true
command = "node"
args = [
  "/absolute/path/to/oac-mcp-connect.js",
  "https://my-oac.example.com",
  "--sync"
]
```

High-Level Claude Desktop Configuration Shape

```
{
  "mcpServers": {
    "oac": {
      "command": "node",
      "args": [
        "/absolute/path/to/oac-mcp-connect.js",
        "https://my-oac.example.com",
        "--sync"
      ]
    }
  }
}
```

For browser authentication, omit --headless and omit token-file arguments; the connector can open a browser when it needs a new login. For headless or non-interactive runs, use a token file downloaded from the OAC profile and pass it before --headless:

```
[mcp_servers.oac_token]
enabled = true
command = "node"
args = [
  "/absolute/path/to/oac-mcp-connect.js",
  "https://my-oac.example.com",
```

```
"/absolute/path/to/token.json",
"--headless",
"--sync"
]
```

The connector can also read a JSON config file for multiple OAC environments. Use the README in the utility bundle for the full set of options, including token paths, browser/headless mode, logging, custom headers, tool filters, and experimental/deprecated tool flags.

Available Oracle Analytics Cloud MCP Tools (Preview)

The raw MCP tool names use a hyphen after the `oracle_analytics` prefix. Codex normalizes tool identifiers to underscores in its generated function namespace, so `oracle_analytics-describe_data` appears in Codex as `mcp__oac_mcp.oracle_analytics_describe_data`. Use the raw MCP name in JSON-RPC tools/call payloads.

MCP Tool Name	Purpose	Modifies Oracle Analytics Cloud? *	Approval Required
oracle_analytics-search_catalog	Search catalog objects such as workbooks, folders, datasets, connections, subject areas, analyses, reports, dashboards, and scripts. **	No	No
oracle_analytics-find_matching_datasources	Match a natural-language question to relevant OAC data sources.	No	No
oracle_analytics-describe_data	Retrieve table and column metadata for subject areas or datasets.	No	No
oracle_analytics-execute_oac_ansi_sql	Execute OAC ANSI SQL. Default for standard analytical queries.	No	No
oracle_analytics-execute_logical_sql	Execute Oracle Analytics Logical SQL and return query results.	No	No
oracle_analytics-save_catalog_content	Create or replace workbook or connection catalog content. **	Yes	Yes
oracle_analytics-export_workbook	Export a workbook to PNG or PDF. **	No, but starts an export job	No
oracle_analytics-create_catalog_folder	Create a catalog folder. **	Yes	Yes
oracle_analytics-copy_catalog_item	Copy a catalog item. **	Yes	Yes
oracle_analytics-move_catalog_item	Move a catalog item. **	Yes	Yes
oracle_analytics-delete_catalog_item	Delete a catalog item. **	Yes	Yes

MCP Tool Name	Purpose	Modifies Oracle Analytics Cloud? *	Approval Required
oracle_analytics-update_catalog_acl	Update catalog permissions. **	Yes	Yes

* Modifying Oracle Analytics Cloud means that a change is made to the Oracle Analytics Cloud catalog.

Modification tools require userApproved: true or an explicit approval flow. Treat create, replace, move, copy, delete, and ACL operations as production-impacting actions

** Identifies an experimental MCP tool. Experimental tools are marked in the visible MCP metadata with:

⚠ Experimental: This capability is experimental and may be modified or removed without notice.

Visible MCP Tool Metadata (Preview)

This topic describes the metadata for each exposed Oracle Analytics Cloud MCP tool.

The longer reference sections later in this guide add examples, usage guidance, and safety notes.

MCP Tool Name	Visible Description	Key Inputs	Output/Behavior
oracle_analytics-search_catalog	Search Oracle Analytics Cloud catalog objects across one or more types. Supports cursor-based pagination and basic sorting.	search type types rootFolder limit sortBy sortOrder cursor	Returns catalog items, effective sorting, pagination state, nextCursor, and hasMore. Items may include objectId, dataset xsaExpr, metadataResource, contentResource, and viewUrl.

MCP Tool Name	Visible Description	Key Inputs	Output/Behavior
oracle_analytics-find_matching_datasources	Finds matching data sources for a natural-language analytics question.	nl_question	Returns data sources matching the query intent, identified by name and described by displayName, description, and score when available. The datasets must be indexed for use with AI features. Datasets are indexed using the same indexing process that Oracle Analytics Cloud uses for its AI features.
oracle_analytics-describe_data	Retrieves comprehensive column-level metadata for Oracle Analytics Cloud subject areas and datasets.	datamodelName tablesOnly tableName tableNames	Returns table and column metadata including names, display names, fully qualified names, column/measure classification, data type, aggregation metadata, nullable, precision, and scale.
oracle_analytics-execute_oac_ansi_sql	Executes an ANSI grammar SQL query against Oracle Analytics Cloud and streams results	query maxRows	Returns row data or error details.
oracle_analytics-execute_logical_sql	Executes an Oracle Analytics Cloud Logical SQL query against the configured Oracle Analytics Cloud server and streams results.	query maxRows	Returns JSON query results, row data, or error details. Metadata includes extensive Logical SQL generation guidance.
oracle_analytics-save_catalog_content	Create or replace catalog content for workbooks or connections.	type name description content parentId id userApproved	Create mode creates a new item under parentId; replace mode updates an existing item by id. Workbook validation failures return VALIDATION_ERROR; upstream/blob failures return UPSTREAM_ERROR.

MCP Tool Name	Visible Description	Key Inputs	Output/Behavior
oracle_analytics-export_workbook	Export a workbook as a file in one long-running operation: create export, poll work request, fetch artifact.	workbookId format name canvasIds screenWidth screenHeight	Returns image content for PNG exports and embedded/blob PDF resource content for PDF exports.
oracle_analytics-create_catalog_folder	Create a catalog folder.	path parentId name createIntermediateFolders userApproved	Uses path-first resolution when path is provided. Returns standardized status, upstream status code, and created item metadata.
oracle_analytics-copy_catalog_item	Copy a catalog item.	type id destId destParentId newItemName overwrite createIntermediateFolders userApproved	Copies the source item to the requested destination and returns a standardized envelope with copied item payload.
oracle_analytics-move_catalog_item	Move a catalog item.	type id destId destParentId newItemName overwrite createIntermediateFolders userApproved	Moves the source item to the requested destination and returns a standardized envelope with moved item payload.
oracle_analytics-delete_catalog_item	Delete a catalog item by type.	type id recursive userApproved	Deletes folders, workbooks, or connections. Uses the shared catalog modification approval flow and returns a standardized delete outcome.
oracle_analytics-update_catalog_acl	Update ACL for a catalog item.	type id aclList updateMode recursive userApproved	Applies ACL updates using ReplaceAll, ReplaceMatchingAccounts, or DeleteMatchingAccounts; returns a standardized envelope with updated ACL payload.

JSON-RPC Shape (Preview)

All tools are called through the MCP tools/call method.

Note

Most users of MCP-enabled clients, such as Claude Desktop or Codex, don't need to construct JSON-RPC requests manually. These clients translate natural language prompts into the appropriate MCP tool calls automatically. This topic is for developers who are building custom AI applications or MCP clients and need to understand the underlying JSON-RPC request and response format.

An example of the general request shape is:

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-describe_data",
    "arguments": {
      "datamodelName": "Sales History Subject Area",
      "tablesOnly": true
    }
  }
}
```

In MCP JSON-RPC, the name value must be the raw MCP tool name, including the hyphen.

HTTP Session Initialization (Preview)

When calling the OAC MCP server over HTTP directly, the endpoint may require an MCP session before it accepts tools/call requests.

If you call a tool before initializing the session, the server can return an error like:

```
{
  "jsonrpc": "2.0",
  "error": {
    "code": -32000,
    "message": "Unknown error",
    "data": "Mcp-Session-Id header required for tools/call"
  },
  "id": 10
}
```

Note

Future MCP protocols may not use the MCP-Session-ID.

Use this sequence

1. Send initialize to the MCP endpoint.
2. Capture the Mcp-Session-Id response header.
3. Send notifications/initialized with that same session ID.
4. Send all later tools/call requests with Mcp-Session-Id: <session-id>.

Initialize the session

```
curl -i --location 'https://<oac-instance>.analytics.ocp.oraclecloud.com/api/mcp' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json, text/event-stream' \
--header 'Authorization: Bearer <token>' \
--data '{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "initialize",
  "params": {
    "protocolVersion": "2025-06-18",
    "capabilities": {},
    "clientInfo": {
      "name": "curl-test",
      "version": "1.0"
    }
  }
}'
```

Look for the session header in the response

Mcp-Session-Id: <session-id>

Notify the server that initialization is complete

```
curl --location 'https://<oac-instance>.analytics.ocp.oraclecloud.com/api/mcp' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json, text/event-stream' \
--header 'Authorization: Bearer <token>' \
--header 'Mcp-Session-Id: <session-id>' \
--data '{
  "jsonrpc": "2.0",
  "method": "notifications/initialized"
}'
```

Then call tools using the same session ID

```
curl --location 'https://<oac-instance>.analytics.ocp.oraclecloud.com/api/mcp' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json, text/event-stream' \
--header 'Authorization: Bearer <token>' \
--header 'Mcp-Session-Id: <session-id>' \
--data '{
  "jsonrpc": "2.0",
  "id": 10,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-search_catalog",
    "arguments": {
      "types": ["subjectAreas"],
      "limit": 20,
      "sortBy": "name",
      "sortOrder": "ASC"
    }
  }
}'
```

The `Mcp-Session-Id` is session state for the MCP HTTP transport. It is separate from the bearer token. The bearer token authenticates the request; the MCP session ID binds later JSON-RPC messages to the initialized MCP session. Reuse the same session ID for all requests in that MCP session.

Common Workflows (Preview)

This topic provides common Oracle Analytics Cloud MCP server workflows for discovering data, executing queries, and safely replacing or cloning workbooks.

Topics:

- [Discover Data and Run a Query Workflow \(Preview\)](#)

Discover Data and Run a Query Workflow (Preview)

Use this workflow when you're building a custom MCP client or AI application that calls Oracle Analytics MCP tools directly. This workflow guides you through identifying the appropriate data source, exploring its available tables, and executing a bounded query.

1. Find likely data sources.

```
{
  "jsonrpc": "2.0",
  "id": 200,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-find_matching_datasources",
    "arguments": {
      "nl_question": "Show revenue by sales channel"
    }
  }
}
```

```
    }
  }
```

2. Describe the chosen subject area tables.

```
{
  "jsonrpc": "2.0",
  "id": 201,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-describe_data",
    "arguments": {
      "datamodelName": "Sales History Subject Area",
      "tableNames": ["SALES", "CHANNELS"]
    }
  }
}
```

3. Execute a bounded ANSI SQL query. Use ANSI SQL for standard analytical queries. Use Logical SQL only when the query requires advanced semantic functions or aggregation overrides.

```
{
  "jsonrpc": "2.0",
  "id": 202,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-execute_oac_ansi_sql",
    "arguments": {
      "query": "SELECT \"Sales History Subject  
Area\".\"CHANNELS\".\"CHANNEL_DESC\" AS channel, SUM(\"Sales History  
Subject Area\".\"SALES\".\"AMOUNT_SOLD\") AS revenue FROM \"Sales History  
Subject Area\" GROUP BY \"Sales History Subject  
Area\".\"CHANNELS\".\"CHANNEL_DESC\" ORDER BY revenue DESC FETCH FIRST 100  
ROWS ONLY",
      "maxRows": 100
    }
  }
}
```

Safety Rules (Preview)

Follow these safety rules when using the Oracle Analytics Cloud MCP tools to help prevent unintended changes, protect catalog content, and ensure queries are executed safely.

- Never delete, replace, move, copy, create folders, or update ACLs without explicit approval.
- Always prefer IDs from `oracle_analytics-search_catalog` over hand-entered IDs.
- For destructive actions, confirm type, id, and user-facing target details.
- Use `overwrite: false` unless the user explicitly wants overwrite behavior.
- Use `recursive: true` only after confirming folder scope.
- For ACL changes, show the exact permissions before applying them.

- Keep ANSI SQL and Logical SQL queries bounded. Use selective filters, a deterministic ORDER BY, FETCH FIRST, and maxRows.
- Validate data model metadata before query execution.

Troubleshooting (Preview)

This topic describes common Oracle Analytics Cloud MCP tools issues, their likely causes, and the recommended steps to resolve them.

Symptom	Likely Cause	Next Step
No catalog results	Wrong type, search term, permissions, or root folder ID.	Retry with search: "*", broader types, or no rootFolder.
describe_data fails for a table	Table name mismatch or unsupported table filter.	Call with tablesOnly: true, then use exact tableName.
Logical SQL column error	Column not fully qualified or not present in model.	Use describe_data and copy fullyQualifiedName.
Logical SQL returns too much data	Missing filters or row limit.	Add selective WHERE, deterministic ORDER BY, and FETCH FIRST.
XSA query fails	Incomplete XSA expression or missing table name in column references.	Use xsaExpr from search_catalog and full XSA(...)."Table"."Column" references.
ANSI aggregate looks wrong	A predefined measure's default semantic aggregation might override AVG, MIN, or MAX.	If MIN, AVG, MAX, and SUM return the same value, don't rely on the requested statistic. Use Logical SQL for a true aggregation override, or retrieve lowest-grain rows and calculate the result outside Oracle Analytics Cloud.
Repeated identical grouped XSA rows	Grouped XSA results can repeat the same dimension and aggregate values.	Treat the result as suspect. Retry the query once with Logical SQL, using the exact XSA fullyQualifiedName expressions.
Workbook save validation error	Workbook JSON structure is invalid for OAC runtime.	Inspect validation metadata, fix structure, and retry once.
Export fails	Invalid workbook ID, canvas ID, render issue, or permissions.	Export all canvases first, then narrow to canvas IDs.
ACL update has unexpected effect	Wrong update mode or broad recursive target.	Prefer ReplaceMatchingAccounts; avoid recursive changes unless required.

Quick Reference (Preview)

Use this quick reference to identify the recommended Oracle Analytics Cloud MCP tool sequences for common tasks, including data discovery, catalog management, and query generation.

Read-Only Discovery

```
search_catalog -> describe_data -> execute_oac_ansi_sql  
find_matching_datasources -> describe_data -> execute_oac_ansi_sql  
search_catalog -> export_workbook
```

Catalog Management

```
search_catalog -> create_catalog_folder  
search_catalog -> copy_catalog_item  
search_catalog -> move_catalog_item  
search_catalog -> delete_catalog_item  
search_catalog -> update_catalog_acl
```

Minimum Metadata Loop Before Writing a Query

1. search_catalog for subject areas or datasets
2. describe_data with tablesOnly
3. describe_data with tableName or tableNames
4. generate ANSI SQL with fully qualified columns
5. execute_oac_ansi_sql with maxRows and FETCH FIRST

Use execute_logical_sql only when the query requires Oracle Analytics Cloud semantic or time-series functions, advanced analytics, or a true aggregation override.

Deprecated MCP Server Tools (Preview)

This topic describes the deprecated Oracle Analytics Cloud MCP tools, their recommended replacements, and guidance for maintaining compatibility with existing workflows.

Deprecated Tool: oracle_analytics-discover_data

The oracle_analytics-discover_data tool is deprecated and should no longer be used for new workflows. It has been replaced by oracle_analytics-search_catalog, which provides a more consistent and flexible way to find catalog objects such as subject areas, datasets, workbooks, folders, and other supported content types.

Existing prompts or integrations that use oracle_analytics-discover_data should be updated to use oracle_analytics-search_catalog instead.

Recommended replacement:

```
{  
  "name": "oracle_analytics-search_catalog",  
  "arguments": {  
    "query": "sales",
```

```
    "types": ["datasets", "subjectAreas"],  
    "limit": 20  
  }  
}
```

The deprecated `oracle_analytics-discover_data` tool can still be exposed for compatibility when using the `oac-mcp-connect` utility. To enable deprecated tools, add the `--include-deprecated` option to the MCP server configuration:

```
[mcp_servers.oac]  
command = "node"  
args = [  
  "/absolute/path/to/oac-mcp-connect.js",  
  "https://your-oac-host.example.com",  
  "--sync",  
  "--include-deprecated"  
]
```

Use `--include-deprecated` only when you need to run legacy prompts, test backward compatibility, or compare old and new tool behavior. For normal use, prefer `oracle_analytics-search_catalog`.

8

Oracle Analytics Cloud MCP Tools Developer's Reference (Preview)

This chapter provides detailed information about each Oracle Analytics Cloud MCP tool, including its purpose, parameters, request and response formats, usage guidance, and examples.

Topics:

Note

It's critical that you understand the potential security risks before you use the Oracle Analytics Cloud MCP server. To learn about the risks, see [Get Started with Oracle Analytics Cloud MCP Tools \(Preview\)](#).

- [ANSI SQL Execution \(Preview\)](#)
- [Catalog Discovery \(Preview\)](#)
- [Copy Catalog Item \(Preview\)](#)
- [Data Model Metadata \(Preview\)](#)
- [Delete Catalog Item \(Preview\)](#)
- [Folder Creation \(Preview\)](#)
- [Logical SQL Execution \(Preview\)](#)
- [Move Catalog Item \(Preview\)](#)
- [Natural Language Data Source Matching \(Preview\)](#)
- [Save Catalog Content \(Preview\)](#)
- [Update Catalog ACL \(Preview\)](#)
- [Workbook Export \(Preview\)](#)

ANSI SQL Execution (Preview)

oracle_analytics-execute_oac_ansi_sql

Input Schema

```
{
  "type": "object",
  "properties": {
    "maxRows": {
      "type": "number"
    },
    "query": {
```

```
    "type": "string"
  }
}
```

Executes an ANSI grammar SQL query against Oracle Analytics Cloud and streams the results. Use this as the default query tool for standard analytical queries.

When to Use ANSI SQL

Use ANSI SQL for standard analytical queries, including grouping, filtering, multiple measures, calculated fields, and Top-N queries that use ORDER BY and FETCH FIRST.

Use oracle_analytics-execute_logical_sql instead when the query requires Oracle Analytics Cloud semantic time functions such as AGO, TODATE, or PERIODROLLING; advanced ranking or analytics; running calculations; forecasting; or a reliable non-default aggregation.

Important Parameters

Parameter	Type	Notes
query	string	ANSI SQL query. Required. Don't add the OBIS_ENABLE_ANSI_SQL prefix because the tool adds it automatically.
maxRows	number	Maximum number of rows to stream. Keep this less than or equal to the query FETCH FIRST limit.

ANSI SQL Rules to Follow

- Use fully qualified column names and explicit aliases.
- Use an explicit GROUP BY for projected dimensions when the query includes aggregate expressions.
- Each leaf subquery must include exactly one subject area in its FROM clause.
- Use projected expressions in ORDER BY.
- Keep row limits bounded with FETCH FIRST n ROWS ONLY.
- For XSA datasets, use an unquoted source expression, such as FROM XSA('namespace'. 'dataset').
- For XSA column references, copy the fullyQualifiedName value from oracle_analytics-describe_data exactly.
- Avoid QUALIFY, quantified SOME or EVERY, OVERLAPS, direct subject-area/CTE comma combinations, and direct CROSS JOIN with a subject area or XSA source.

Example

```
{
  "jsonrpc": "2.0",
```

```

    "id": 50,
    "method": "tools/call",
    "params": {
      "name": "oracle_analytics-execute_oac_ansi_sql",
      "arguments": {
        "query": "SELECT \"Sales History Subject
Area\".\"CHANNELS\".\"CHANNEL_DESC\" AS channel, SUM(\"Sales History Subject
Area\".\"SALES\".\"AMOUNT_SOLD\") AS revenue FROM \"Sales History Subject
Area\" GROUP BY \"Sales History Subject Area\".\"CHANNELS\".\"CHANNEL_DESC\"
ORDER BY revenue DESC FETCH FIRST 100 ROWS ONLY",
        "maxRows": 100
      }
    }
  }
}

```

Output

Returns row data. If the query fails, the response includes error details.

Troubleshooting

If the service returns REQUEST_ERROR: Connection reset, retry the same query once.

If grouped XSA results contain repeated identical dimension and aggregate rows, treat the result as suspect. Retry the query once with Logical SQL, using the exact XSA fullyQualifiedName expressions.

For a predefined measure, AVG, MAX, or MIN might resolve to the model's default aggregation. If MIN, AVG, MAX, and SUM return the same value, don't rely on the requested statistic. Use Logical SQL or retrieve lowest-grain rows and calculate the result outside Oracle Analytics Cloud.

Safety

This tool is read-only and doesn't require approval. However, queries can be expensive or expose data. Use selective filters, deterministic ordering, FETCH FIRST, and maxRows to keep results bounded.

Catalog Discovery (Preview)

oracle_analytics-search_catalog

Input Schema

```

{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "cursor": {
      "type": "string"
    },
    "limit": {
      "type": "integer"
    },
    "rootFolder": {
      "type": "string"
    }
  }
}

```

```
    },
    "search": {
      "type": "string"
    },
    "sortBy": {
      "type": "string"
    },
    "sortOrder": {
      "type": "string"
    },
    "type": {
      "type": "string"
    },
    "types": {
      "type": "array",
      "items": {
        "type": "string"
      }
    }
  }
}
```

Searches OAC catalog objects across one or more catalog types. Use this before workbook operations, catalog modifications, or dataset Logical SQL work because it returns catalog IDs, resource URIs, and deep links where available.

Common Uses

- Find a workbook by name.
- List folders under a known root folder ID.
- Find datasets and retrieve their xsaExpr.
- Find subject areas before describe_data.
- Locate connections before replacing or copying them.

Supported Catalog Types Include

- workbooks
- folders
- datasets
- connections
- dataflows
- models
- sequences
- subjectAreas
- analysis
- reports
- dashboardgroupfolders

- dashboardfolders
- dashboardpages
- dashboards
- scripts

subjectareas is accepted as an alias for subjectAreas.

Important Parameters

Parameter	Type	Notes
search	string	Query string. Defaults to *.
type	string	Single catalog type.
types	string[]	Multiple catalog types. If type and types are both supplied, the tool uses the union.
rootFolder	string	Base64url folder ID. Folder paths are not accepted.
limit	integer	Page size. Default 50, maximum 200.
sortBy	string	One of name, lastModified, type, owner, description.
sortOrder	string	ASC or DESC.
cursor	string	Pass nextCursor from a previous response to fetch the next page. Ignore other args when using a cursor.

Example - Find Subject Areas

```
{
  "jsonrpc": "2.0",
  "id": 10,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-search_catalog",
    "arguments": {
      "types": ["subjectAreas"],
      "limit": 20,
      "sortBy": "name",
      "sortOrder": "ASC"
    }
  }
}
```

Example - Search for Workbooks by Name

```
{
  "jsonrpc": "2.0",
  "id": 11,
  "method": "tools/call",
```

```

    "params": {
      "name": "oracle_analytics-search_catalog",
      "arguments": {
        "search": "Sales",
        "type": "workbooks",
        "limit": 25
      }
    }
  }
}

```

Typical Response Fields

Field	Notes
items	Matching catalog objects.
items[].id	Base64url catalog ID. Use this for save, export, copy, move, delete, and ACL calls.
items[].objectId	Upstream object ID when available. Useful for datasets.
items[].xsaExpr	Dataset Extended Subject Area expression, when available. Use this in Logical SQL FROM.
items[].metadataResource	MCP resource URI for raw metadata, if exposed by the client.
items[].contentResource	MCP resource URI for workbook or connection content, if exposed by the client.
items[].viewUrl	Browser deep link where available.
nextCursor	Cursor for the next page.
hasMore	Whether more pages are available.

Copy Catalog Item (Preview)

oracle_analytics-copy_catalog_item

Input Schema

```

{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "createIntermediateFolders": {
      "type": "boolean"
    },
    "destId": {
      "type": "string"
    },
    "destParentId": {
      "type": "string"
    },
    "id": {
      "type": "string"
    },
    "newItemName": {
      "type": "string"
    }
  }
}

```

```

    },
    "overwrite": {
      "type": "boolean"
    },
    "type": {
      "type": "string"
    },
    "userApproved": {
      "type": "boolean"
    }
  }
}

```

Copies a catalog item to a destination item or parent folder.
This is a modification tool and requires approval.

Important Parameters

Parameter	Type	Notes
type	string	Catalog type of the source item.
id	string	Source item ID.
destId	string	Destination item ID.
destParentId	string	Destination parent folder ID.
newItemName	string	Optional copied item name.
overwrite	boolean	Whether to overwrite an existing item.
createIntermediateFolders	boolean	Create missing destination folders where supported.
userApproved	boolean	Must be true or confirmed through approval.

Example

```

{
  "jsonrpc": "2.0",
  "id": 80,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-copy_catalog_item",
    "arguments": {
      "type": "workbooks",
      "id": "base64url-source-workbook-id",
      "destParentId": "base64url-destination-folder-id",
      "newItemName": "Revenue Overview Copy",
      "overwrite": false,
      "userApproved": true
    }
  }
}

```

Use Cases

- Make a backup before replacing a workbook.
- Promote a workbook from a development folder to a shared folder.
- Clone a template workbook before editing.

Data Model Metadata (Preview)

oracle_analytics-describe_data

Input Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "datamodelName": {
      "type": "string"
    },
    "tableName": {
      "type": "string"
    },
    "tableNames": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "tablesOnly": {
      "type": "boolean"
    }
  }
}
```

Retrieves table and column metadata for OAC subject areas and datasets. This is the primary tool for validating column names, data types, measure classifications, and aggregation behavior before writing Logical SQL or generating workbooks.

Import Parameters

Parameter	Type	Notes
datamodelName	string	Subject area name, such as Sales History Subject Area, or an XSA expression such as XSA('owner' . 'dataset').
tablesOnly	boolean	Returns only available table names. Use this before requesting specific tables.
tableName	string	Returns columns for one table. Use the table name only, not the fully qualified name.

Parameter	Type	Notes
tableNames	string[]	Returns columns for multiple tables in one call, when supported.

Example - List Tables Only

```
{
  "jsonrpc": "2.0",
  "id": 30,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-describe_data",
    "arguments": {
      "datamodelName": "Sales History Subject Area",
      "tablesOnly": true
    }
  }
}
```

Example - Describe One Table

```
{
  "jsonrpc": "2.0",
  "id": 31,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-describe_data",
    "arguments": {
      "datamodelName": "Sales History Subject Area",
      "tableName": "SALES"
    }
  }
}
```

Example - Describe Multiple Tables in One Request

```
{
  "jsonrpc": "2.0",
  "id": 32,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-describe_data",
    "arguments": {
      "datamodelName": "Sales History Subject Area",
      "tableNames": ["SALES", "TIMES", "CHANNELS"]
    }
  }
}
```

Typical Response Fields

Field	Notes
subjectArea	Data model name.
tables[].tableName	Table name.
tables[].fullQualifiedName	Fully quoted subject-area/table reference.
tables[].columns[]	Column metadata.
columns[].name	Column name.
columns[].displayName	Display name.
columns[].fullyQualifiedName	Fully quoted column reference for Logical SQL.
columns[].columnType	Usually attribute or measure.
columns[].dataType	Data type such as VARCHAR, NUMERIC, TIMESTAMP, DATE.
columns[].aggregation	Default aggregation where provided.
columns[].nullable	Nullability flag.
columns[].precision, columns[].scale, columns[].length	Numeric/string metadata where available.

Practical Guidance

- Start with `tablesOnly: true` to avoid very large responses.
- Prefer `tableName` or `tableNames` once the relevant tables are known.
- Use `fullyQualifiedName` directly in Logical SQL.
- Check `columnType` before aggregating. Measures often already have model-defined aggregation.
- Don't assume measures should be wrapped in SUM; inspect metadata first.

Delete Catalog Item (Preview)

oracle_analytics-delete_catalog_item

Input Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "id": {
      "type": "string"
    },
    "recursive": {
      "type": "boolean"
    },
    "type": {
      "type": "string"
    },
    "userApproved": {
      "type": "boolean"
    }
  }
}
```

```
}  
}
```

Deletes a catalog item by type and ID.

This is a destructive modification tool and requires explicit approval. Confirm the item name, type, path, and ID before calling it.

Important Parameters

Parameter	Type	Notes
type	string	folders, workbooks, or connections.
id	string	Base64url catalog item ID.
recursive	boolean	Applies only to folders.
userApproved	boolean	Must be true or confirmed through approval.

Example

```
{  
  "jsonrpc": "2.0",  
  "id": 100,  
  "method": "tools/call",  
  "params": {  
    "name": "oracle_analytics-delete_catalog_item",  
    "arguments": {  
      "type": "workbooks",  
      "id": "base64url-workbook-id",  
      "userApproved": true  
    }  
  }  
}
```

Safer Deletion Workflow

1. Search for the item with `oracle_analytics-search_catalog`.
2. Verify name, type, id, and path or owner metadata where available.
3. Ask the user to approve the exact deletion target.
4. Call `oracle_analytics-delete_catalog_item`.
5. Search again to confirm the item was deleted.

Folder Creation (Preview)

oracle_analytics-create_catalog_folder

Input Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "createIntermediateFolders": {
      "type": "boolean"
    },
    "name": {
      "type": "string"
    },
    "parentId": {
      "type": "string"
    },
    "path": {
      "type": "string"
    },
    "userApproved": {
      "type": "boolean"
    }
  }
}
```

Creates a catalog folder. Prefer path when creating by absolute catalog path. Use parentId and name when the parent folder ID is already known.

This is a modification tool and requires approval.

Import Parameters

Parameter	Type	Notes
path	string	Preferred absolute catalog path, for example /shared/Revenue.
parentId	string	Parent folder ID. Alternative to path.
name	string	Folder name when using parentId.
createIntermediateFolders	boolean	Create missing parent folders.
userApproved	boolean	Must be true or confirmed through approval.

Example

```
{
  "jsonrpc": "2.0",
  "id": 70,
```

```
"method": "tools/call",
"params": {
  "name": "oracle_analytics-create_catalog_folder",
  "arguments": {
    "path": "/shared/MCP Generated",
    "createIntermediateFolders": true,
    "userApproved": true
  }
}
```

Notes

- If both path and parentId/name are provided, path takes precedence.
- The response includes standardized status, upstream status code, and created item metadata.

Logical SQL Execution (Preview)

```
oracle_analytics-execute_logical_sql
```

Input Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "maxRows": {
      "type": "integer"
    },
    "query": {
      "type": "string"
    }
  }
}
```

Executes Oracle Analytics Logical SQL against the configured OAC server and streams results.

When to Use Logical SQL

Use the Logical SQL Execution tool when the ANSI SQL tool can't provide the required semantic behavior. For example, use Logical SQL for OAC time-series functions such as AGO, TODATE, and PERIODROLLING; advanced ranking and analytics; moving or running calculations; forecasting; trendline, cluster, or regression functions; model-aware default aggregation; or a true aggregation override.

Important Parameters

Parameter	Type	Notes
query	string	Logical SQL query. Required.

Parameter	Type	Notes
maxRows	integer	Maximum rows to stream. Keep this less than or equal to the query FETCH FIRST limit.

Example

```
{
  "jsonrpc": "2.0",
  "id": 40,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-execute_logical_sql",
    "arguments": {
      "query": "SELECT \"Sales History Subject
Area\".\"CHANNELS\".\"CHANNEL_DESC\" AS llm_0, \"Sales History Subject
Area\".\"SALES\".\"AMOUNT_SOLD\" AS llm_1 FROM \"Sales History Subject Area\"
ORDER BY llm_1 DESC FETCH FIRST 100 ROWS ONLY",
      "maxRows": 100
    }
  }
}
```

Logical SQL Rules to Follow

For standard analytical queries such as filtering, grouping, multiple measures, calculated fields, and Top-N results, use `oracle_analytics-execute_oac_ansi_sql` instead.

- Use fully qualified column names: "Subject Area"."Table"."Column".
- Use the quoted subject area name in the FROM clause.
- Do not write manual joins. OAC resolves joins through the semantic model.
- Do not write explicit GROUP BY for normal presentation-layer aggregation. OAC infers grouping from selected dimensions.
- Include ORDER BY when using FETCH FIRST.
- Keep row limits bounded with FETCH FIRST n ROWS ONLY.
- Alias selected columns as llm_0, llm_1, and so on for deterministic downstream handling.

Example - Subject Area

```
SELECT
  "Sales History Subject Area"."CHANNELS"."CHANNEL_DESC" AS llm_0,
  "Sales History Subject Area"."SALES"."AMOUNT_SOLD" AS llm_1
FROM "Sales History Subject Area"
ORDER BY llm_1 DESC
FETCH FIRST 100 ROWS ONLY
```

Example - Dataset/XSA

```

SELECT
  XSA('admin'. 'sales_data'). "Orders". "Customer" AS llm_0,
  SUM(OVERRIDEAGGR(XSA('admin'. 'sales_data'). "Orders". "Amount")) AS llm_1
FROM XSA('admin'. 'sales_data')
ORDER BY llm_1 DESC
FETCH FIRST 100 ROWS ONLY

```

Aggregation Guidance

- If a measure's default aggregation is what you want, use the measure directly.
- Use OVERRIDEAGGR only when intentionally overriding the default aggregation.
- For XSA datasets, full column references must include the XSA expression, table name, and column name.
- Do not mix columns from different subject areas or XSA datasets in the same query.
- Use IN for subqueries. Avoid = with subqueries.

Time-Series Guidance

- AGO, TODATE, and PERIODROLLING require valid time dimensions.
- In time-series functions, the time level parameter may require two-part naming such as "Time". "Year" instead of the full subject-area prefix, depending on the model.
- Validate time columns with oracle_analytics-describe_data before writing time-series expressions.

Move Catalog Item (Preview)

oracle_analytics-move_catalog_item

Input Schema

```

{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "createIntermediateFolders": {
      "type": "boolean"
    },
    "destId": {
      "type": "string"
    },
    "destParentId": {
      "type": "string"
    },
    "id": {
      "type": "string"
    },
    "newItemName": {
      "type": "string"
    },
    "overwrite": {

```

```
        "type": "boolean"
      },
      "type": {
        "type": "string"
      },
      "userApproved": {
        "type": "boolean"
      }
    }
  }
}
```

Moves a catalog item to another destination.

This is a modification tool and requires approval.

Import Parameters

Parameter	Type	Notes
type	string	Catalog type of the source item.
id	string	Source item ID.
destId	string	Destination item ID.
destParentId	string	Destination parent folder ID.
newItemName	string	Optional new item name.
overwrite	boolean	Whether to overwrite an existing item.
createIntermediateFolders	boolean	Create missing destination folders where supported.
userApproved	boolean	Must be true or confirmed through approval.

Example

```
{
  "jsonrpc": "2.0",
  "id": 90,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-move_catalog_item",
    "arguments": {
      "type": "workbooks",
      "id": "base64url-workbook-id",
      "destParentId": "base64url-destination-folder-id",
      "newItemName": "Revenue Overview",
      "overwrite": false,
      "userApproved": true
    }
  }
}
```

Use Cases

- Move generated content out of a temporary folder.
- Rename a workbook while moving it.
- Reorganize catalog folders after creation.

Natural Language Data Source Matching (Preview)

oracle_analytics-find_matching_datasources

This tool requires datasets to be indexed before they can be matched to natural language questions. Datasets are indexed using the same indexing process that Oracle Analytics Cloud uses for its AI features.

Input Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "nl_question": {
      "type": "string"
    }
  }
}
```

Finds candidate subject areas or datasets for a natural-language analytics question.

Use this when the user starts with a business question and does not know which data source to use. If it can't find a good match, fall back to oracle_analytics-search_catalog.

Import Parameters

Parameter	Type	Notes
nl_question	string	Natural-language analytics question.

Example

```
{
  "jsonrpc": "2.0",
  "id": 20,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-find_matching_datasources",
    "arguments": {
      "nl_question": "Which products had the highest revenue last quarter by channel?"
    }
  }
}
```

Typical Response Fields

Field	Notes
name	Data source name.
displayName	User-facing name.
description	Business description where available.
score	Intent-match score.

Recommended Flow

1. Call `oracle_analytics-find_matching_datasources`.
2. Re-rank candidates using descriptions and business context.
3. Call `oracle_analytics-describe_data` for the best candidate.
4. Build and validate Logical SQL.
5. Execute with `oracle_analytics-execute_logical_sql`.

Save Catalog Content (Preview)

`oracle_analytics-save_catalog_content`

Input Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "content": {
      "type": [
        "object",
        "string"
      ],
      "description": "Canonical workbook payload is content.json. Connector shortcut for workbooks: content.jsonPath (or content.path) may be an absolute local JSON file path; the parsed object is converted to content.json before save.",
      "properties": {
        "json": {
          "description": "Canonical workbook JSON object. Takes precedence over jsonPath/path when present."
        },
        "jsonPath": {
          "type": "string",
          "description": "Connector-side absolute local UTF-8 JSON file path for workbook JSON. Allowed roots apply."
        },
        "path": {
          "type": "string",
          "description": "Alias for content.jsonPath for workbook JSON file input. Allowed roots apply."
        }
      }
    },
    "description": {
```

```

        "type": "string"
      },
      "id": {
        "type": "string"
      },
      "name": {
        "type": "string"
      },
      "parentId": {
        "type": "string"
      },
      "type": {
        "type": "string"
      },
      "userApproved": {
        "type": "boolean"
      }
    }
  }
}

```

Creates or replaces catalog content for workbooks or connections.

This is a modification tool. Use `userApproved: true` only after the user has approved the exact target and intended operation.

Modes

- Create mode: supply type, name, parentId, and content.
- Replace mode: supply type, id, and optionally content, name, or description.

Important Parameters

Parameter	Type	Notes
type	string	workbooks or connections.
name	string	Required for create. Optional for replace.
description	string	Optional metadata.
content	object or string	Content JSON. For workbooks, prefer canonical { "json": { ... }, "blobs": [] }.
parentId	string	Destination folder ID for create mode.
id	string	Existing catalog item ID for replace mode.
userApproved	boolean	Must be true or confirmed through approval.

Example - Canonical Workbook Create

This example shows the canonical save envelope for workbook content. The criteria, layouts, and views fields are shown in abbreviated form. A real workbook must populate these

sections with the data model, visual layout, and visualization definitions generated from discovered metadata.

```
{
  "jsonrpc": "2.0",
  "id": 50,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-save_catalog_content",
    "arguments": {
      "type": "workbooks",
      "name": "Revenue Overview",
      "description": "Revenue overview generated from MCP metadata.",
      "parentId": "base64url-folder-id",
      "content": {
        "json": {
          "projectVersion": "26.03",
          "criteria": {
            "...": "data source, columns, calculations, and filters"
          },
          "layouts": [
            {
              "...": "canvas layout and visualization placement"
            }
          ],
          "views": [
            {
              "...": "visualization definitions and data bindings"
            }
          ]
        },
        "blobs": []
      },
      "userApproved": true
    }
  }
}
```

Workbook Content Fields

Field	Purpose	Typical Contents
projectVersion	Workbook format version expected by OAC.	Version string such as 26.03; usually copied from a known-good exported workbook template or generated by workbook authoring logic.
criteria	Defines the workbook data model usage.	Subject area or dataset references, selected columns, calculated expressions, column IDs, display names, aggregation metadata, filters, and Logical SQL expressions used by visualizations.

Field	Purpose	Typical Contents
layouts	Defines canvas/page structure and placement.	Canvases, sections, visualization containers, coordinates, sizing, layout IDs, filter bar placement, and the relationship between canvases and views.
views	Defines the actual visualizations.	Chart type, table, pivot, map, or KPI configuration; bound data columns; grammar roles such as category, value, and color; titles; formatting; sorting; and visualization-specific properties.
blobs	Optional binary or embedded resources.	Usually empty for simple generated workbooks. May contain referenced assets or workbook resources when required by the workbook format.



Warning

Don't hand-author these fields from scratch unless you're following a known good workbook template. The safest pattern is to export an existing workbook, reuse its structural conventions, and replace the data model references, criteria columns, layout objects, and view bindings using metadata returned by `oracle_analytics-describe_data`.

Example - Workbook Replace

```
{
  "jsonrpc": "2.0",
  "id": 51,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-save_catalog_content",
    "arguments": {
      "type": "workbooks",
      "id": "base64url-workbook-id",
      "content": {
        "json": {
          "projectVersion": "26.03",
          "criteria": {
            "...": "updated data source, columns, calculations, and filters"
          },
          "layouts": [
            {
              "...": "updated canvas layout and visualization placement"
            }
          ],
          "views": [
            {
              "...": "updated visualization definitions and data bindings"
            }
          ]
        }
      }
    }
  }
}
```

```

        ]
      },
      "blobs": []
    },
    "userApproved": true
  }
}

```

Local Workbook JSON File Shortcut

```

{
  "jsonrpc": "2.0",
  "id": 52,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-save_catalog_content",
    "arguments": {
      "type": "workbooks",
      "id": "base64url-workbook-id",
      "content": {
        "jsonPath": "/absolute/path/to/workbook.json"
      }
    },
    "userApproved": true
  }
}

```

Notes

- `content.json` is canonical and takes precedence over `jsonPath`.
- `jsonPath` must be readable by the connector process and under an allowed file root.
- Workbook validation failures are returned as `VALIDATION_ERROR`.
- Non-validation upstream failures are returned as `UPSTREAM_ERROR`.
- Direct workbook root objects may be auto-wrapped into `{ "json": ... }`, but explicit canonical shape is preferred.

Update Catalog ACL (Preview)

oracle_analytics-update_catalog_acl

Input Schema

```

{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "aclList": {
      "type": "array",
      "items": {
        "type": "object",

```

```

    "properties": {
      "accountDisplayName": {
        "type": "string"
      },
      "accountGuid": {
        "type": "string"
      },
      "accountType": {
        "type": "string"
      },
      "permissions": {
        "type": "object",
        "properties": {
          "changePermission": {
            "type": "boolean"
          },
          "delete": {
            "type": "boolean"
          },
          "execute": {
            "type": "boolean"
          },
          "list": {
            "type": "boolean"
          },
          "read": {
            "type": "boolean"
          },
          "takeOwnership": {
            "type": "boolean"
          },
          "write": {
            "type": "boolean"
          }
        }
      }
    },
    "id": {
      "type": "string"
    },
    "recursive": {
      "type": "boolean"
    },
    "type": {
      "type": "string"
    },
    "updateMode": {
      "type": "string"
    },
    "userApproved": {
      "type": "boolean"
    }
  }
}

```

Updates catalog item permissions.

This is a modification tool and requires approval. ACL changes can expose sensitive data or remove access, so confirm the target item and all permission changes before calling it.

Important Parameters

Parameter	Type	Notes
type	string	Catalog item type.
id	string	Catalog item ID.
aclList	object[]	ACL entries to apply.
updateMode	string	ReplaceAll, ReplaceMatchingAccounts, or DeleteMatchingAccounts.
recursive	boolean	Applies to folders.
userApproved	boolean	Must be true or confirmed through approval.

ACL Entry Shape

```
{
  "accountDisplayName": "Analytics Authors",
  "accountGuid": "optional-guid",
  "accountType": "group",
  "permissions": {
    "read": true,
    "write": true,
    "delete": false,
    "list": true,
    "execute": true,
    "changePermission": false,
    "takeOwnership": false
  }
}
```

Example

```
{
  "jsonrpc": "2.0",
  "id": 110,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-update_catalog_acl",
    "arguments": {
      "type": "workbooks",
      "id": "base64url-workbook-id",
      "updateMode": "ReplaceMatchingAccounts",
      "aclList": [
        {
          "accountDisplayName": "Analytics Authors",
          "accountType": "group",

```

```

        "permissions": {
            "read": true,
            "write": true,
            "list": true,
            "execute": true,
            "delete": false,
            "changePermission": false,
            "takeOwnership": false
        }
    },
    "userApproved": true
}
}
}

```

Use Cases

- Grant a group read access to a generated workbook.
- Replace permissions for known accounts.
- Remove matching ACL entries by account.
- Apply permissions recursively to a folder tree.

Workbook Export (Preview)

oracle_analytics-export_workbook

Input Schema

```

{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "properties": {
    "canvasIds": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "format": {
      "type": "string"
    },
    "name": {
      "type": "string"
    },
    "screenHeight": {
      "type": "integer"
    },
    "screenWidth": {
      "type": "integer"
    },
    "workbookId": {
      "type": "string"
    }
  }
}

```

```
}
}
}
```

Exports a workbook to PNG or PDF. The tool creates an export request, polls the work request, and fetches the resulting artifact.

Import Parameters

Parameter	Type	Notes
workbookId	string	Base64url workbook catalog ID. Required.
format	string	png or pdf. Required.
name	string	Optional export name.
canvasIds	string[]	Optional list of canvas IDs. If omitted, canvases are auto-resolved
screenWidth	integer	Optional viewport width. Default is typically 1920.
screenHeight	integer	Optional viewport height. Default is typically 1080.

Example - Export All Canvases to PDF

```
{
  "jsonrpc": "2.0",
  "id": 60,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-export_workbook",
    "arguments": {
      "workbookId": "base64url-workbook-id",
      "format": "pdf",
      "name": "Revenue Overview Export",
      "screenWidth": 1920,
      "screenHeight": 1080
    }
  }
}
```

Example - Export Selected Canvases to PNG

```
{
  "jsonrpc": "2.0",
  "id": 61,
  "method": "tools/call",
  "params": {
    "name": "oracle_analytics-export_workbook",
    "arguments": {
      "workbookId": "base64url-workbook-id",
      "format": "png",

```

```
        "canvasIds": ["canvas-1", "canvas-2"],  
        "screenWidth": 1440,  
        "screenHeight": 900  
    }  
}
```

Typical Outputs

- PNG exports return image content.
- PDF exports return an embedded or blob resource.
- Long-running export failures usually indicate workbook rendering issues, invalid canvas IDs, permissions, or server-side export limits.

Part V

Use APIs

Oracle Analytics offers REST APIs and session-based web services (SOAP APIs) to integrate with your applications.

Topics:

- [REST APIs](#)
- [SOAP APIs](#)

9

REST APIs

Use the REST APIs to automate processes and programmatically access features and functionality in Oracle Analytics.

API Version

The base path of the endpoint includes the API version (for example, 20210901). Here's an example GET request to get analysis details:

```
GET
https://<myOACinstance>.analytics.ocp.oraclecloud.com/api/20210901/catalog/
analysis/<analysis_id>
```

API Breaking Changes Policy

Oracle will provide 12 months advance notice prior to the date of removing or changing an existing API that you have deployed which would require you to update your code.

REST API Reference

See [REST API for Oracle Analytics Cloud](#).

SOAP APIs

Use the Oracle Analytics session-based web services (SOAP APIs) to enable external applications to communicate with Oracle Analytics.

Topics:

- [Introduction to Oracle Analytics Web Services](#)
- [Description of Services and Methods in Oracle Analytics Web Services](#)
- [Description of Structures in Oracle Analytics Web Services](#)

Introduction to Oracle Analytics Web Services

This topic describes the Oracle Analytics session-based web services (SOAP APIs).

Topics:

- [About Oracle Analytics Web Services](#)
- [What are the Oracle Analytics Session-Based Web Services?](#)

About Oracle Analytics Web Services

You can use Oracle Analytics session-based web service to call Oracle Analytics programmatically and to perform various analytics tasks.

The Oracle Analytics session-based web services require a valid Oracle Analytics session ID to be passed as a parameter. The calling application must first make a call to get the session ID before calling the web service.

You can obtain the session ID by using an access token in the authorization header of the `logon()` method of the `SAWSessionService` service.

This document includes descriptions of the web services supported by Oracle Analytics. Note that the WSDL document might include additional services not documented here.

What are the Oracle Analytics Session-Based Web Services?

The Oracle Analytics session-based web services are an application programming interface (API) that implements SOAP (Simple Object Access Protocol). These web services are designed for programmatically invoking analysis and interactive reporting objects. These web services also provide functionality to perform a wide range of operations.

The Oracle Analytics session-based web services allow you to perform three types of functions:

- Extract results from analysis and interactive reporting objects and deliver them to external applications and web application environments.
- Perform management functions.
- Execute Intelligent Agents.

Oracle Analytics session-based web services allow external applications such as J2EE and .NET to use Oracle Analytics as an analytical calculation and data integration engine. It provides a set of services that allow external applications to communicate with Oracle Analytics.

Oracle Analytics session-based web services require a valid Oracle Analytics session ID to be passed as a parameter. This means that the calling application first needs to make a call to get the session ID before calling the web service. A final call is made to log out.

The formal definition of services and methods in Oracle Analytics web services can be retrieved in WSDL format. You can use a proxy generation tool to create proxy/stub code to access the services. Depending upon the client version you are using, you can access the WSDL document at the following Oracle Analytics web services URL:

`https://host:port/analytics-ws/saw.dll/wsd1/v13`

Description of Services and Methods in Oracle Analytics Web Services

This topic describes the services and methods used by the Oracle Analytics session-based web services. This document uses JavaScript-like syntax to describe structures. The exact syntax and implementation depend on the SOAP code generation tool and the target language used by your application development environment.

Topics:

- [AdministrationService Service](#)
- [AnalysisExportViewsService Service](#)
- [ConditionService Service](#)
- [HtmlViewService Service](#)
- [iBotService Service](#)
- [MetadataService Service](#)
- [ReportEditingService Service](#)
- [SAWSessionService Service](#)
- [SchedulerService Service](#)
- [SecurityService Service](#)
- [UserPersonalizationService Service](#)
- [WebCatalogService Service](#)
- [XMLViewService Service](#)

AdministrationService Service

Use the AdministrationService service to manage the Oracle Analytics safe domains and configure log level.

Method Name	Description
deleteCSPWhitelist() Method	Deletes all safe domains entries. Use with caution.

Method Name	Description
getCSPDefaultAllowList() Method	This method is used internally by Oracle Analytics to return the default entries for the content security policy required by the product functionality.
getCSPWhitelist() Method	Retrieves the list of domains and options specified in the safe domains configuration.
updateCSPWhitelist() Method	Updates the current set of domains and allowed options in the content security policy that's sent to the web browsers.

deleteCSPWhitelist() Method

Use the deleteCSPWhitelist() method to delete all safe domains entries. Use with caution.

Signature

```
deleteCSPWhitelistResult deleteCSPWhitelist(String sessionId);
```

Argument	Description
String sessionId	Specifies the session ID.

Returns

Returns deleteCSPWhitelistResult.

getCSPDefaultAllowList() Method

The getCSPDefaultAllowList() method is used internally by Oracle Analytics to return the default entries for the content security policy required by product functionality.

You can use REST APIs to create, update, and delete safe domains. See [Safe Domains REST Endpoints](#).

Signature

```
getCSPDefaultAllowList(String sessionId);
```

Argument	Description
String sessionId	Specifies the session ID.

Returns

Returns an output as defined in the getCSPWhitelist structure. See [CSPWhitelist Structure](#).

getCSPWhitelist() Method

Use the getCSPWhitelist() method to obtain the current set of domains and allowed options in the content security policy that is sent to web browsers.

Signature

```
getCSPWhitelist(String sessionId);
```

Argument	Description
String sessionId	Specifies the session ID.

Returns

Returns an output as defined in the `getCSPWhitelist` structure. See [CSPWhitelist Structure](#).

updateCSPWhitelist() Method

Use the `updateCSPWhitelist()` method to update the current set of domains and allowed options in the content security policy that is sent to web browsers.

Signature

`updateCSPWhitelist(String cspWhitelistXml, String sessionId);`

Argument	Description
String cspWhitelistXml	Specifies the content security policy in the <code>CSPWhitelist</code> structure. See CSPWhitelistXml Structure .
String sessionId	Specifies the session ID.

Returns

Returns confirmation of the updated CSP whitelist.

AnalysisExportViewsService Service

Use the `AnalysisExportViewsService` service to initiate export of analysis reports and retrieve exported files in PDF, MHTML, Excel, and CSV formats.

Method Name	Description
completeAnalysisExport() Method	Retrieves the exported files.
initiateAnalysisExport() Method	Initiates and retrieves the exported files of analysis.

completeAnalysisExport() Method

Use the `completeAnalysisExport()` method to retrieve exported files in PDF, MHTML, Excel, or CSV formats.

Signature

`completeAnalysisExport(String queryID, String sessionId);`

Arguments	Description
String queryID	Specifies query ID returned by the <code>initiateAnalysisExport()</code> method. See initiateAnalysisExport() Method .
String SessionID	Specifies the unique session ID.

Returns

Returns an output as defined in the AnalysisExportResult structure.

For information on the AnalysisExportResult structure, see [AnalysisExportResult Structure](#).

initiateAnalysisExport() Method

Use initiateAnalysisExport() method to initiate and retrieve exported files of analysis in the following formats - PDF, MHTML, Excel, CSV.

Signature

```
initiateAnalysisExport(ReportRef reportRef, AnalysisExportOutputFormat outputFormat,  
AnalysisExportExecutionOptions executionOptions, ReportParams reportParams, String  
reportViewName, String sessionId);
```

Arguments	Description
ReportRef reportRef	Specifies the path to the analysis definition, supplied in the ReportRef common structure.
AnalysisExportOutputFormat outputFormat	Specifies one of the following output formats that you can select: - String PDF - Specifies the PDF format - String MHTML - Specifies the MHTML format - String Excel 2007 - Specifies the Excel 2007 format - String CSV - Specifies the CSV format
AnalysisExportExecutionOptions executionOptions	Specifies the execution options in the AnalysisExportExecutionOptions structure.
ReportParams reportParams	Specifies the filters or variables to apply to the analysis before execution, supplied in the ReportParams common structure.
String reportViewName	Specifies the view to export. If this parameter is null, the analysis default view is used. The view name should match the one used to identify the view in the analysis XML definition.
String sessionId	Specifies the session ID.

Returns

Returns an output as defined in the AnalysisExportResult structure.

For information on the AnalysisExportResult structure, see [AnalysisExportResult Structure](#).

ConditionService Service

Use the ConditionService service to evaluate Oracle Analytics conditions programatically. This service also allows users to obtain the customizable filters available in a condition.

Method Name	Description
evaluateCondition() Method	Evaluates a condition saved to the catalog.
evaluateInlineCondition() Method	Evaluate a condition supplied as a parameter.
getConditionCustomizableReportElements() Method	Obtains the customizable filters of a condition saved to the catalog.

evaluateCondition() Method

Use the `evaluateCondition()` method to evaluate a Condition that is stored in the catalog. This method returns an XML string containing the result of the condition (true or false).

Signature

```
boolean evaluateCondition(String path, String[] reportCustomizationParameters, String sessionID);
```

Arguments	Description
String path	Specifies the full path and name of the Condition in the catalog. For example, <code>/users/jchan/Conditions/IsRegionUnderBudget</code> .
String [] reportCustomizationParameters	Specifies the customization parameters XML, which is only used if the Condition has customizable filters. This XML is validated against the <code>analysis_customization.xsd</code> customization schema. See analysis_customization.xsd.
String sessionID	Specifies the session ID, which is usually returned by the <code>logon</code> method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

evaluateInlineCondition() Method

Use the `evaluateInlineCondition()` method to evaluate a condition defined outside of Oracle Analytics. The Condition XML is supplied in the `conditionXML` parameter. This method returns an XML string with the result of the condition evaluation, true or false.

Signature

```
boolean evaluateInlineCondition(String conditionXML, String[] reportCustomizationParameters, String sessionID);
```

Arguments	Description
String conditionXML	Specifies the Condition XML. This XML is validated against the <code>condition.xsd</code> condition schema. See condition.xsd.
String [] ListreportCustomizationParameters	Specifies the customization parameters XML, which is only used if the Condition has customizable filters. This XML is validated against the <code>analysis_customization.xsd</code> customization schema. See analysis_customization.xsd.
String sessionID	Specifies the session ID, which is usually returned by the <code>logon</code> method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getConditionCustomizableReportElements() Method

Use the `getConditionCustomizableReportElements()` method to determine the customizable filters available in a condition that is stored in the catalog.

This method returns an XML string containing the definition of the customizable filters available in the condition.

The XML is in the format defined in the `analysis_customization.xsd` customization schema. See [analysis_customization.xsd](#).

Signature

```
String[] getConditionCustomizableReportElements(String path, String sessionId);
```

Arguments	Description
String path	Specifies the full path and name of the condition in the catalog. For example, /users/jchan/Conditions/IsRegionUnderBudget.
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

HtmlViewService Service

Use the HtmlViewService service to embed Oracle Analytics HTML results in third-party dynamic web pages, such as Active Server Pages (ASP) or JavaServer Pages (JSP), and portal frameworks.

The embed process merges Oracle Analytics web services content with the content of third-party web pages.

Method Name	Description
addReportToPage() Method	Adds results to an HTML page.
endPage() Method	Destroys a server page object and all data associated with it.
getCommonBodyHTML() Method	Retrieves HTML to include in the <BODY> section.
getHeadersHTML() Method	Retrieves HTML to include in the <HEAD> section.
getHtmlforPageWithOneReport() Method	Retrieves HTML for a page that contains only one analysis.
getHTMLForReport() Method	Retrieves HTML to display a particular set of results.
setBridge() Method	Specifies a bridge URL to receive communications. Can be useful when the Oracle Analytics web services server and the Presentation Services that the user is accessing reside on different machines or when you want to modify the results in your application development environment.
startPage() Method	Creates a new page object and returns its ID.

The methods in HtmlViewService extract fragments of HTML code that can be inserted in third-party web pages.

HTML Code Fragment	Desired Page Location
Header	Should be inserted in the <HEAD> section of an HTML page. The code contains links to common JavaScript files and style sheets.
Report Objects	Can be inserted anywhere in the <BODY> section.
Common Body	Should be inserted in the <BODY> tag after all analysis links. The code contains hidden HTML elements that are used to implement drilldown links.

For each returned analysis object, the HTML code fragment contains a callback link that is followed automatically when the web page is loaded by the browser. The code fragment does not contain the full user interface definition of the analysis. While the analysis is being

constructed by Oracle Analytics Presentation Services, the interface displays the Oracle Analytics web services "Searching..." image embedded on the third-party web page.

For smooth analysis transitioning, Oracle Analytics Presentation Services tracks the Oracle Analytics analyses that have been added to third-party web pages by maintaining information in an internal logical page object during the construction of the third-party web page. The `HtmlViewService` service methods explicitly refer to the internal logical page by its ID.

About HtmlViewService Bridging and Callback URLs

To embed an analysis with active drilldown links, the `HtmlViewService` service allows the web browser to issue callback requests from embedded analyses to the Oracle Analytics Presentation Services server.

Although it is possible to route requests directly to the Oracle Analytics Presentation Services server, in many cases it is preferable to route requests through the Oracle Analytics instance that originally serviced the third-party page. Also, in situations where Oracle Analytics Presentation Services and the third-party web server don't belong to the same Domain Name Service (DNS) domain, users may get JavaScript errors related to browser security constraints for cross-domain scripting.

To avoid these issues, use the `setBridge()` method to modify callback URLs to point to the third-party web server. Be aware that a web component executed by the third-party web server to re-route requests to Oracle Analytics Presentation Services isn't provided. This function would need to be fulfilled by the third-party application. For more information about the `setBridge()` method, see [setBridge\(\) Method](#).

addReportToPage() Method

Use the `addReportToPage()` method to add results to an HTML page.

Signature

```
void addReportToPage(String pageID, String reportID, ReportRef report,  
String reportViewName, ReportParams reportParams, ReportHTMLOptions options,  
String sessionID);
```

Arguments	Description
String pageID	Specifies a character string page ID returned by the <code>startPage()</code> method. See startPage() Method .
String reportID	Specifies a character string that identifies the analysis containing the results to add to the page. It should be used to reference this analysis in subsequent method invocations; for example, corresponding user interface elements generated by the Oracle Analytics Presentation Services server would reference the same ID.
ReportRef report	Specifies the analysis definition, supplied in the <code>ReportRef</code> structure.
String reportViewName	Specifies the view to display. If this parameter is null, the analysis' default view is used. The view name should match the one used to identify the view in the analysis XML definition.
ReportParams reportParams	Optional. Specifies the filters or variables to apply to the analysis before execution, supplied in the <code>ReportParams</code> common structure.
ReportHTMLOptions options	Optional. Specifies the display options to apply to the analysis after execution, supplied in the <code>ReportHTMLOptions</code> structure. See QueryResults Structure .

Arguments	Description
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

endPage() Method

Use the endPage() method to destroy the Oracle Analytics Presentation Services server page object and all data associated with it.

Signature

```
void endpage(String pageID, String sessionId);
```

Arguments	Description
String pageID	Specifies the ID of the page object, which is returned by the startPage() method. See startPage() Method .
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getCommonBodyHTML() Method

Use the getCommonBodyHTML() method to retrieve HTML to include in the <BODY> section.

Signature

```
String getCommonBodyHTML(String pageID, String sessionId);
```

Arguments	Description
String pageID	Specifies the ID of the page object, which is returned by the startPage() method. See startPage() Method .
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a string containing the HTML to include in the <BODY> section.

getHeadersHTML() Method

Use the getHeadersHTML() method to retrieve HTML to include in the <HEAD> section.

Signature

```
String getHeadersHTML(String pageID, String sessionId);
```

Arguments	Description
String pageID	Specifies the ID of the page object, which is returned by the <code>startPage()</code> method. See startPage() Method .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a string containing the HTML to include in the <HEAD> section.

getHtmlforPageWithOneReport() Method

Use the `getHtmlforPageWithOneReport()` method to retrieve the HTML for a page that contains only one analysis. A page that contains only one analysis doesn't have <BODY> and <HEAD> sections.

Signature

```
String getHtmlforPageWithOneReport(String reportID, ReportRef report, String  
reportViewName, ReportParams reportParams, ReportHTMLOptions reportOptions,  
StartPageParams pageParams, String sessionID);
```

Arguments	Description
String pageReportID	Specifies the analysis ID returned by the <code>getHtmlforPageWithOneReport()</code> method. See addReportToPage() Method .
ReportRef report	Specifies the analysis definition, supplied in the ReportRef structure.
String reportViewName	Specifies the view to display. If this parameter is null, the analysis' default view is used. The view name should match the one used to identify the view in the analysis XML definition.
ReportParams reportParams	Optional. Specifies the filters or variables to apply to the analysis before execution, supplied in the ReportParams common structure.
ReportHTMLOptions reportOptions	Optional. Specifies the display options to apply to the analysis after execution, supplied in the ReportHTMLOptions structure. See QueryResults Structure .
StartPageParams pageParams	Specifies the options to use when starting the page, supplied in the StartPageParams structure. See StartPageParams Structure .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getHTMLForReport() Method

Use the `getHTMLForReport()` method to retrieve an HTML excerpt to display the results for a particular analysis. Before invoking this method, use the `addReportToPage` method to add the results to an HTML page.

Signature

```
String getHTMLForReport(String pageID, String pageReportID, String sessionID);
```

Arguments	Description
String pageID	Specifies the ID of the page object, which is returned by the startPage() method. See startPage() Method .
String pageReportID	Specifies the analysis ID returned by the addReportToPage() method. See addReportToPage() Method .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a string containing the HTML excerpt that displays the specified analysis.

setBridge() Method

Use the setBridge() method to specify a bridge URL to receive communications. Specifying a bridge URL can be useful when the Oracle Analytics web services server and the user's web server reside on different machines, or when you want to modify the results in your application development environment.

After the setBridge() method is called, all requests from the client browser to the Oracle Analytics Presentation Services server are sent to the bridge URL, which then forwards requests to the Oracle Analytics Presentation Services server.

Signature

setBridge(String bridge, String sessionID);

Arguments	Description
String bridge	Specifies the bridge URL. For example, <code>http://myserver/myapplication/sawbridge</code>
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Usage

You must make sure that the client browser provides a handler to the bridge URL in the form of a Java servlet, an Active Server Pages (ASP) page, a Common Gateway Interface (CGI), an Internet Server application programming interface (ISAPI), or an equivalent application.

You must also perform the following tasks:

- Decode the path of the requested Oracle Analytics web services resource in the RedirectURL argument of the request character string. See [How Callback URLs Are Replaced](#).
- Forward all other request arguments, together with all headers and the request body, to the bridge URL.
- Copy the response from the Oracle Analytics Presentation Services server to the response stream.

How Callback URLs Are Replaced

The new callback URL is based on the bridge URL, with the addition of a RedirectURL argument. The value of the RedirectURL argument should be the original value of the URL, encoded using standard URL encoding rules.

Internally, Oracle Analytics web services usually uses relative URLs for callback links. For example, if the original callback link is saw.dll?Go and the bridge URL is

```
http://myserver/myapplication/sawbridge
```

then the new callback URL is

```
http://myserver/myapplication/sawbridge?RedirectURL=saw.dll%3fGo
```

startPage() Method

Use the startPage() method to create a page object and returns its ID.

Signature

```
String startPage(StartPageParams options, String sessionID);
```

Arguments	Description
StartPageParams options	Specifies the options to use when starting the page, supplied in the StartPageParams structure. See StartPageParams Structure .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a string containing the Oracle Analytics Presentation Services page ID.

iBotService Service

Use the iBotService service to save, edit, delete, subscribe, unsubscribe, customize, and execute Oracle Analytics agents. Note that as of the Oracle Business Intelligence 11g (11.1.1) release, "iBots" have been renamed to "agents."

Method Names	Description
deleteIBot() Method	Deletes an agent from the catalog and deregisters it from the Oracle Analytics Scheduler.
enableIBot() Method	Enables an Oracle Analytics agent.
executeIBotNow() Method	Executes an agent saved in the catalog.
getAgentPaths() Method	Gets the paths of the agents of a specified user.
getAgents() Method	Gets the agents from a specified path.
moveIBot() Method	Moves an agent from one catalog folder to another.
purgeAlerts() Method	Specifies the alerts to be deleted.

Method Names	Description
getBotStatus() Method	Gets the status of the specified agent.
sendMessage() Method	Sends a message to an Oracle Analytics user, group, or user and group.
subscribe() Method	Subscribes to a published agent. Also customizes your subscription.
unsubscribe() Method	Unsubscribes from an agent.
writeBot() Method	Writes a new agent into the catalog and registers it with Oracle Analytics Scheduler.

deleteBot() Method

Use the `deleteBot()` method to delete a saved agent. Deleting an agent not only removes it (the object) from the catalog, but it also deregisters the agent from the Oracle Analytics Scheduler. Note that this method is different from the `deleteitem()` method of `WebCatalogService` because the `deleteitem()` method doesn't deregister the agent from the Oracle Analytics Scheduler.

Signature

```
void deleteBot(String path, String sessionID);
```

Arguments	Description
String path	Specifies the full path and name of the agent in the catalog. For example, <code>/users/jchan/iBots/BrandDollars</code> .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

enableBot() Method

Use the `enableBot()` method to enable an Oracle Analytics agent.

Signature

```
enableBotResult enableBot(String path, boolean enable, String sessionID);
```

Argument	Description
String path	Specifies the full path and name of the agent in the catalog. For example, <code>/users/jchan/iBots/BrandDollars</code> .
boolean enable	Specifies whether to enable the specified agent.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns `enableBotResult`.

executeIBotNow() Method

Use the `executeIBotNow()` method to execute an agent that's stored in the catalog. Note that this method doesn't change the agent's original schedule.

Signature

```
void executeIBotNow(String path, String sessionID);
```

Arguments	Description
String path	Specifies the full path and name of the agent in the catalog. For example, <code>/users/jchan/iBots/BrandDollars</code> .
String sessionID	Specifies the session ID, which is usually returned by the <code>logon</code> method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getAgentPaths() Method

Use the `getAgentPaths()` method to get the paths of the Oracle Analytics agents of a specified user.

Signature

```
getAgentPathsResult getAgentPaths(String userID, String sessionID);
```

Argument	Description
String userID	Specifies the user ID of the user to get the paths of the agents.
String sessionID	Specifies the session ID, which is usually returned by the <code>logon</code> method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns `getAgentPathsResult`.

getAgents() Method

Use the `getAgents()` method to get the Oracle Analytics agents in a specified path.

Signature

```
getAgentsResult getAgents(String agentPath, String sessionID);
```

Argument	Description
String agentPath	Specifies the full path in the catalog.
String sessionID	Specifies the session ID, which is usually returned by the <code>logon</code> method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns `getAgentsResult`.

getIBotStatus() Method

Use the getIBotStatus() method to get the status of the specified Oracle Analytics agent.

Signature

```
getIBotStatusResult getIBotStatus(String path, String sessionID);
```

Argument	Description
String path	Specifies the full path of the agent in the catalog.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns getIBotStatusResult.

moveIBot() Method

Use the moveIBot() method to move an agent from one catalog folder to another. Note that this method is different from the moveItem() method of WebCatalogService Service because the moveItem() method moves the catalog object and informs the Oracle Analytics Scheduler that the object was moved.

Signature

```
void moveIBot(String fromPath, String toPath, boolean resolveLinks, boolean allowOverwrite, String sessionID);
```

Arguments	Description
String fromPath	Specifies the full catalog path of the agent to be moved.
String toPath	Specifies the full catalog path where the agent is moved to.
boolean resolveLinks	Specifies if you want to move the child objects. If this argument is set to TRUE and the path specified in the "fromPath" argument is a link, then the child object pointed to by that link is moved.
boolean allowOverwrite	Specifies if you want to overwrite an existing object. If this argument is set to TRUE and another catalog object existed in the path specified by "toPath", it is overwritten.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

purgeAlerts() Method

Use the purgeAlerts() method to delete the alerts that are older than the specified time.

Signature

```
purgeAlertsResult purgeAlerts(Unsigned Integer maxAgeSecs, String userGlob, String sessionID);
```

Argument	Description
Unsigned Integer maxAgeSecs	Specifies the age of the alert in seconds. Alerts older than this will be deleted. Use 0 to delete the alert irrespective of the age.
String userGlob	Specifies the user whose alerts has to be deleted. Specify the wildcard * to delete alerts for all users.
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns the number of alerts purged, how many user's alerts were checked, alerts remaining, and if any errors occurred.

sendMessage() Method

Use the sendMessage() method to send a message to an Oracle Analytics user, group, or user and group. The message is delivered according to the corresponding recipient's delivery profile, which was set up in the My Account dialog in Oracle Analytics Presentation Services.

Signature

```
String sendMessage(String[] recipient, String[] group, String subject, String body, String priority, String sessionId);
```

Arguments	Description
String[] recipient	Specifies the GUID of the Oracle Analytics user to whom you want to send the message. You can include more than one user in this argument.
String[] group	Specifies the GUID of the Oracle Analytics group to whom you want to send the message. You can include more than one group in this argument.
String subject	Specifies the subject line of the message.
String body	Specifies the text to be included in the body of the message.
String priority	Specifies the message's priority. You can specify "High," "Normal," or "Low."
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

subscribe() Method

Use the subscribe() method to subscribe to a published agent. If the agent allows customization, then you can also specify the customization XML.

Signature

```
void subscribe(String path, String customizationXml, String sessionId);
```

Arguments	Description
String path	Specifies the full path and name of the agent in the catalog. For example, /users/jchan/iBots/BrandDollars.
String customizationXml	Specifies the customization XML (only if the agent allows customizations). This XML is validated against the analysis_customization customization schema. See analysis_customization.xsd .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

unsubscribe() Method

Use the unsubscribe() method to unsubscribe from an agent. This method also deletes any user customizations.

Signature

```
void unsubscribe(String path, String sessionID);
```

Arguments	Description
String path	Specifies the full path and name of the agent in the catalog. For example, /users/jchan/iBots/BrandDollars.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

writeIBot() Method

Use the writeIBot() method to write a new agent to the catalog and to register it with Oracle Analytics Scheduler. Note that this method is different from the writeObjects() method of WebCatalogService. The writeObjects() method only writes to the catalog.

Signature

```
int writeIBot(CatalogObject obj, String path, boolean resolveLinks, boolean allowOverwrite, String sessionID);
```

Arguments	Description
CatalogObject obj	Specifies the object to be written to the catalog. The object's XML is validated against the analysis_ibot.xsd schema. See analysis_ibot.xsd .
String path	Specifies the full path and name of the agent in the catalog. For example, /users/jchan/iBots/BrandDollars.
boolean resolveLinks	If set to TRUE and the path in the catalog refers to a link, then the object is written to the location pointed to by the link.
boolean allowOverwrite	Specifies whether to overwrite an existing object. Set to TRUE to overwrite any object already present in the location specified by "path."

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

MetadataService Service

Use the MetadataService() service to retrieve descriptions of Oracle Analytics Presentation Services schema objects, such as columns, tables, and subject areas.

Method Names	Description
clearQueryCache() Method	Clears the query cache.
describeColumn() Method	Retrieves column information for a specified column in a specified subject area and table.
describeSubjectArea() Method	Retrieves subject area information for a specified subject area.
describeTable() Method	Retrieves table information for a specified table in a specified subject area.
getSubjectAreas() Method	Retrieves the list of subject areas available.
reloadLogConfiguration() Method	Forces changes to logging configuration to take effect without manually restarting Oracle Analytics Presentation Services.
reloadMetadata() Method	Reloads XML message files, refresh server metadata, and clear caches.

clearQueryCache() Method

Use the clearQueryCache() method to clear the query cache.

Signature

```
boolean clearQueryCache(String sessionID);
```

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

describeColumn() Method

Use the describeColumn() method to retrieve column information for a specified column in a specified subject area and table.

Signature

```
SAColumn describeColumn(String subjectAreaName, String tableName, String columnName, String sessionID);
```

Arguments	Description
String subjectAreaName	Specifies the subject area to be queried.

Arguments	Description
String tableName	Specifies the table to be queried.
String columnName	Specifies the name of the column to be queried.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns an SAColumn Object. For information on the SAColumn structure.

See [SAColumn Structure](#).

describeSubjectArea() Method

Use the describeSubjectArea() method to retrieve subject area information about the specified subject area.

Signature

```
SASubjectArea describeSubjectArea(String subjectAreaName,  
SASubjectAreaDetails detailsLevel, String sessionID);
```

Arguments	Description
String subjectAreaName	Specifies the subject area to be queried.
SASubjectAreaDetails details Level	Specifies the information to be retrieved about the subject area. For information on the SASubjectAreaDetails structure, see SASubjectAreaDetails Values .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

SASubjectAreaDetails Values

Use the SASubjectAreaDetails values to specify what information should be retrieved about the subject area.

Values	Description
IncludeTables	Include table list with minimum information about each table.
IncludeTablesAndColumns	Include full table and column information.
Minimum	Don't include table and column information.

Returns

Returns an SASubjectArea Object.

See [SASubjectArea Structure](#).

Usage

The detailsLevel parameter values for the describeSubjectArea() method.

Depending on the value of the detailsLevel parameter, the returned object contains the information specified in this table.

Value of detailsLevel	Description
IncludeTables	Specifies that the tables field is not null and contains the collection of tables for this subject area. Each table object has the columns field set to null.
IncludeTablesAndColumns	Specifies that the tables field is not null and contains the collection of tables for this subject area. For each table object the columns field contains the corresponding collection of columns.
Minimum	Specifies that the table list is not available. The tables field in the resulting subject area object is null.

describeTable() Method

Use the describeTable() method to retrieve table information for a specified table in a specified subject area.

Signature

```
SATable describeTable(String subjectAreaName, String tableName,  
SATableDetails detailsLevel, String sessionID);
```

Arguments	Description
String subjectAreaName	Specifies the subject area to be queried.
String tableName	Specifies the table to be queried.
SATableDetails detailsLevel	Specifies the information to retrieve about the table. For information on the SATableDetails structure, see SATablesDetails Values .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

SATablesDetails Values

The SATablesDetails values specify the information to retrieve about the subject area table.

Values	Description
IncludeColumns	Populate the columns field in the SATable Object.
Minimum	Do not include column information. The columns field in the SATable Object is set to null.

Returns

Returns an SATable Object. For information on the SATable structure.

See [SATable Structure](#).

getSubjectAreas() Method

Use the `getSubjectAreas()` method to retrieve the list of subject areas that are available.

Signature

```
List[] getSubjectAreas(String sessionID);
```

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns an array of `SASubjectArea` objects. For information on the `SASubjectArea` structure.

See [SASubjectArea Structure](#).

Usage

`SASubjectArea` objects returned by this method do not have table information available.

The `tables` field is null. The approach to querying at all levels is to use `getSubjectAreas()` to retrieve the list of subject areas and then use `describeSubjectArea()` to retrieve the list of tables. Next, use `describeTable()` to retrieve the list of columns in a specified table, and finally, use `describeColumn()` to retrieve information on a specified column.

reloadLogConfiguration() Method

Use the `reloadLogConfiguration()` method to force changes to logging configuration to take effect without manually restarting Oracle Analytics Presentation Services.

Signature

```
boolean reloadLogConfiguration(String sessionID);
```

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

reloadMetadata() Method

Use the `reloadMetadata()` method to reload XML message files, refresh server metadata, and clear caches.

Signature

```
boolean reloadMetadata(String sessionID);
```

Arguments	Description
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a boolean to indicate if the operation is successful.

ReportEditingService Service

Use the ReportEditingService service to merge arguments and Oracle Analytics Presentation Services data to create and return the results.

Method Names	Description
applyReportDefaults() Method	Applies analysis default arguments to the analysis and returns the results.
applyReportParams() Method	Applies report arguments to the analysis object and returns the results.
getPromptElements() Method	Retrieves a list of criteria prompt column definitions in a given analysis and current runtime state.
generateReportSQL() Method	Retrieves the SQL query for a given analysis.
getReportColumns() Method	Retrieves a list of criteria columns in an analysis.
getReportElements() Method	Retrieves a list of prompts, unprotected filters, and referenced presentation variables in the given report.

applyReportDefaults() Method

Use the applyReportDefaults() method to apply analysis default arguments to the analysis and returns the results.

Signature

String applyReportDefaults(ReportRef reportRefs, String sessionId);

Arguments	Description
ReportRef object	Specifies the path to the analysis definition, supplied in the ReportRef common structure.
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns the result of applying the default analysis arguments to the specified analysis object.

applyReportParams() Method

Use the applyReportParams() method to apply analysis arguments to the analysis and return the results.

Signature

Object applyReportParams(ReportRef reportRef, ReportParams reportParams, boolean encodeInString, String sessionId);

Arguments	Description
ReportRef reportRef	Specifies the path to the analysis definition, supplied in the ReportRef common structure.
ReportParams reportParams	Optional. Specifies the filters or variables to apply to the analysis before execution, supplied in the ReportParams common structure.
boolean encodeInString	If set to TRUE, then the returned analysis object is encoded as a character string.
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns the result of applying analysis arguments to the specified analysis object. If you set encodeInString to true, then the result is encoded as a character string.

getPromptElements() Method

Use the getPromptElements() method to retrieve a list of criteria prompt column definitions in a given analysis and current runtime state.

Signature

getPromptElements(ReportRef promptRef, String viewState, String viewID, String portalPath, String page, NameValuePair optionalParams, String sessionId);

Arguments	Description
ReportRef promptRef	Required. Specifies the catalog path to a saved report or a report xml document.
String viewState	Optional. Specifies the runtime viewState. Runtime prompt definition may be changed based on other prompts setting view viewState.
String viewID	Optional: Specifies the view ID
String portalPath	Optional: Specifies the dashboard catalog path where the report resides.
String page	Optional: Specifies the dashboard page name where the report resides.
NameValuePair optionalParams	Optional: Specifies the name of additional parameters which are usually used for debugging purposes.
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

A set of report prompt columns.

See [PromptsObjectModel Structure](#).

generateReportSQL() Method

Use the generateReportSQL() method to retrieve the logical SQL query for a given analysis.

Signature

```
String generateReportSQL(ReportRef reportRef, ReportParams reportParams,  
String sessionID);
```

Arguments	Description
ReportRef reportRef	Specifies the path to the analysis definition supplied in the ReportRef common structure.
ReportParams reportParams	Optional. Specifies the path to the filters or variables to apply to the analysis before execution, supplied in the ReportParams common structure.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

A string containing the SQL query for the specified analysis.

getReportColumns() Method

Use the getReportColumns() method to retrieve a list of criteria columns in a given analysis.

Signature

```
ReportColumn[] getReportElements(String reportPath, String sessionID);
```

Arguments	Description
ReportRef reportRef	Specifies the path to the analysis definition supplied in the ReportRef common structure.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

ReportColumn[] - array of columns in the report.

See [ReportHierarchicalColumn Structure](#) and [ReportRegularColumn Structure](#).

getReportElements() Method

Use the getReportElements() method to retrieve a list of all report prompts, unprotected filters, and referenced presentation variables in the given report.

Signature

```
String getReportElements(String reportPath, String sessionID);
```

Arguments	Description
String reportPath	Required. Specifies the report catalog path for the report.
String sessionID	Optional. Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Report ADF parameters.

If present, report ADF parameters for the following objects, in the order listed:

- Prompts
- Unprotected filters
- Presentation variables referenced in the report
- Context variables referenced by the Oracle Analytics Presentation Services

See [ReportADFParameters Structure](#).

SAWSessionService Service

Use the SAWSessionService service to provide authentication methods such as logon and logoff, and other session-related methods.

Method Name	Description
getCurUser() Method	Retrieves the current user ID for the session.
GetSessionEnvironment() Method	Retrieves the environment object for the current session.
getSessionVariables() Method	Retrieves a list of session variables.
impersonate() Method	Logs on and then impersonates the user.
impersonateex() Method	Logs on and then impersonates the user. Similar to the impersonate method, but impersonateex can specify optional session parameters.
keepAlive() Method	Instructs Oracle Analytics Presentation Services not to end particular sessions due to inactivity.
logoff() Method	Logs the user off Oracle Analytics Presentation Services.
logon() Method	Authenticates the user.
logonex() Method	Authenticates the user. Similar to the logon method, but logonex can specify optional session parameters.

getCurUser() Method

Use the getCurUser() method to retrieve the current user name for the session.

Signature

```
String getCurUser(String sessionID);
```

Argument	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a string indicating the current user name for the session.

GetSessionEnvironment() Method

Use the `GetSessionEnvironment()` method to retrieve the environment object for the current session.

Signature

`SessionEnvironment getSessionEnvironment(String sessionID);`

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

This method returns a session environment object.

See [SessionEnvironment Structure](#).

getSessionVariables() Method

Use the `getSessionVariables()` method to retrieve a list of session variables.

Signature

`List[] getSessionVariables(List[] names, String sessionID);`

Arguments	Description
List[] names	Specifies the names of the session variables.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

This method returns values of the variables associated with the current session.

impersonate() Method

Use the `impersonate()` method to log on and impersonate the user during the `SAWSessionService` service.

This method is useful when you need to create sessions for multiple users and have only the administrator's name and password. You do not need to use the (logon) method if you use the impersonate() method.

If user authentication or impersonation fails, an exception is thrown.

Signature

String impersonate(String name, String password, String impersonateID);

Arguments	Description
String name	Specifies the user name to log on and authenticate.
String password	Specifies the password for the user. If there is no password for the user, leave this field empty (void).
String impersonateID	Specifies the user name to impersonate the authenticated user.

Returns

This method returns the session ID and sets an HTTP session cookie.

The session ID is used in other methods to identify the Oracle Analytics web services session.

impersonateex() Method

Use the impersonateex() method to log on and impersonate the user in the SAWSessionService service.

Similar to the impersonate method, but impersonateex can specify optional session parameters. This method is useful when you need to create sessions for multiple users and have only the administrator's name and password. You do not need to use the (logon) method if you use the impersonateex() method.

If user authentication or impersonation fails, then an exception is thrown.

Signature

AuthResults impersonateex(String name, String password, String impersonateID, SAWSessionParameters sessionparams);

Arguments	Description
String name	Specifies the user name to log on and authenticate.
String password	Specifies the password for the user. If there is no password for the user, leave this field empty (void).
String impersonateID	Specifies the user name to impersonate the authenticated user.
SAWSessionParameters sessionparams	Optional. Specifies the session parameters to use, supplied in the SAWSessionParameters structure. For information about the SAWSessionParameters structure, see SAWSessionParameters Structure .

Returns

This method returns the AuthResult structure containing the session ID, and also sets an HTTP session cookie.

The session ID is used in other methods to identify the Oracle Analytics Presentation Services session. For more information, see [AuthResult Structure](#).

keepAlive() Method

Use the keepAlive() method to instruct Oracle Analytics Presentation Services not to end particular web user sessions due to inactivity.

The effect of this method on session lifetime is the same as if those users performed an activity in the browser such as clicking an analysis, or invoking a method.

Signature

```
void keepAlive(String[] sessionID);
```

Argument	Description
String[] sessionID	Specifies the session IDs to remain logged on.

logoff() Method

Use the logoff() method to log off the user from Oracle Analytics.

Signature

```
void logoff(String sessionID);
```

Argument	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

logon() Method

Use the logon() method to authenticate the user. If authentication fails, an exception is thrown.

Signature

```
String logon(String name, String password);
```

Arguments	Description
String name	Specifies the user name to authenticate.
String password	Specifies the password for the user. If there is no password, leave this field empty (void).

fjvbvr

Returns

This method returns the session ID and sets an HTTP session cookie.

The session ID is used in other methods to identify the Oracle Analytics session.

logonex() Method

Use the `logonex()` method to authenticate the user. The `logonex()` method is similar to the `logon` method, but `logonex` can specify optional session parameters.

If authentication fails, an exception is thrown.

Signature

```
AuthResult logonex(String name, String password, SAWSessionParameters sessionparams);
```

Arguments	Description
String name	Specifies the user name to authenticate.
String password	Specifies the password for the user. If there is no password, leave this field empty (void).
SAWSessionParameters sessionparams	Optional. Specifies the sessionparams to use, supplied in the SAWSessionParameters structure. For information about the SAWSessionParameters structure, see SAWSessionParameters Structure .

Returns

This method returns the `AuthResult` structure containing the session ID, and also sets an HTTP session cookie.

The session ID is used in other methods to identify the Oracle Analytics Presentation Services session.

SchedulerService Service

The following terms are associated with the SchedulerService service:

- Agents (or iBots) - These are Presentation Services catalog objects.
- Jobs - These are scheduler objects, and every agent has at least one job stored in the BI_PLATFORM schema.
- JobInstances - These are scheduler objects representing active or completed jobs and are stored in the BI_PLATFORM schema.

Use this service to list, and detail scheduler jobs and job instances, and to purge and remove job instances (removing a job automatically purges job instances). Deleting agents automatically removes jobs.

To use this service, you must obtain a user session ID to return a list of Job or Job Instance Reference ID's, which you use to get details or remove the object. You obtain a user session ID by using the `logon()` method of the SAW session service (for more information, see [logon\(\) Method](#)). You must then apply filters to specify particular Job Reference IDs and Job Instance IDs.

For examples of using the SchedulerService service methods, see [Examples of Using the SchedulerService API](#)

The client must be granted the privileges `Access SOAP`, and `Access SchedulerService` Service to call methods in the SchedulerService service API. These privileges are granted by

default to the BIConsumer application role. You manage these privileges using the Manage Privileges Page in Presentation Services Administration.

The methods described in this section are synchronous unless stated otherwise.

[iBotService Service](#) shows the supported methods.

Method Names	Description
getJobReferences() Method	Returns a list of Jobs that match the selection criteria in the specified filter argument.
getJobInstanceReferences() Method	Returns a list of Job Instances that match the selection criteria in the specified filter argument.
getJob() Method	Returns a Job definition for a specified Job reference.
getJobInstance() Method	Returns Job Instance details for a specified Job Reference.
cancelJobInstance() Method	Requests to cancel an executing Job Instance for a specified Job Reference.
removeJobs() Method	Requests to remove a Job definition for a specified Job Reference.
purgeJobInstances() Method	Requests to purge an existing Job Instance for a specified Job Reference.

getJobReferences() Method

Use the `getJobReferences()` method to get a list of job references based on a filter.

Returned references are not live and are invalidated when jobs or instances are deleted.

Signature

```
JobReference[] getJobReferences(List[] JobFilter, String SessionID);
```

Arguments	Description
List[] JobFilter	Selection criteria that returns matching job references. An empty JobFilter results in all job references being returned. Note that filters are applied in the context of the calling user, which means that a user is restricted to listing jobs within their tenancy but does not require any Presentation Services catalog permissions on the agent corresponding to the job. For more information, see JobFilter Structure , and JobReferenceAndInstanceReferences Structure .
String SessionID	Specifies the session ID, which is usually returned by the login method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null. If no Session IDs are specified then all JobReferences are returned.

getJobInstanceReferences() Method

Use the `getJobInstanceReferences()` method to get a list of job instances that correspond to running, cancelled or completed jobs.

The JobInstanceFilter must include one or more Job References (unknown Job References are ignored).

A structure is returned in the list for each Job which has associated Job Instances that satisfy the filter. For example, if the method is invoked with a JobInstanceFilter that specifies three Job

references, and two of these Jobs are found to have associated Job Instances, then there will be two `JobReferenceAndInstanceReferences` structures returned in the list.

There is no limit to the number of structures and job instances that can be returned. Performance of this method is critical and must be optimized to allow efficient polling of job status.

References are not live and are invalidated if jobs or instances are deleted or cancelled.

The following properties can be applied when listing job instance references:

- List of `JobReference` [Mandatory]. For more information, see [JobReferenceAndInstanceReferences Structure](#)
- `JobInstanceStatus` [Optional] For more information, see [JobInstanceStatus Enumeration](#).
- `JobFilter`, `JobInstanceFilter`, and `PurgeJobInstancesFilter` properties are selection criteria that are used as follows:
 - A filter is always applied in the context of the current user.
 - Properties in the filter are selection criteria.
 - Filter and its properties must be populated according to the schema.
 - If the selection criteria are valid, but would return Jobs that are not visible to the current user these jobs is excluded without error.
 - All properties in a filter is used as selection criteria (union).
 - Filter criteria are applied by the server at the time the request is serviced.

For more information, see [JobFilter Structure](#), [JobInstanceFilter Structure](#), and [PurgeJobInstancesFilter Structure](#).

Signature

```
JobReferenceAndInstanceReferences[] getJobInstanceReferences(Array JobInstanceFilter,  
String SessionID);
```

Arguments	Description
<code>JobInstanceFilter filter</code>	Specifies the selection criteria that determines which <code>JobInstanceReferences</code> to return. For more information, see JobInstanceFilter Structure and JobReferenceAndInstanceReferences Structure .
<code>String SessionID</code>	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getJob() Method

Use the `getJob()` method to get the Job definition given a Job Reference.

If you try to get a job that doesn't exist, you get `Job not found` error.

Signature

```
Job[] getJob(String JobReference, String SessionID);
```

Arguments	Description
String JobReference	Specifies a unique job identifier. For more information, see Job Structure .
String SessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getJobInstance() Method

Use the getJobInstance() method to get the Job Instance details, when given a Job Instance Reference.

If you try to get a Job Instance that doesn't exist, you get JobInstance not found error.

Signature

```
JobInstance[] getJobInstance(String JobReference, String JobInstanceReference, String SessionID);
```

Arguments	Description
String JobReference	Specifies a unique job identifier. For more information, see JobInstance Structure .
String JobInstanceReference	Specifies a unique job instance identifier.
String SessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

cancelJobInstance() Method

Use the cancelJobInstance() method to request to cancel an executing Job Instance, when given a reference to an existing Job Instance.

Successful return indicates that the cancellation request has been accepted. Callers must use the getJobInstanceReferences operation (with an appropriate filter - such as Job Reference and State) or using the getJobInstance operation (and checking the JobInstanceStatus property), to determine if the cancellation has completed.

Cancelling a Job Instance that is not running has no effect. Cancelling a Job Instance that doesn't exist raises a JobInstance not found error.

Signature

```
Boolean cancelJobInstance(String JobReference, String JobInstanceReference, String SessionID);
```

Arguments	Description
String JobReference	Specifies a unique job identifier.
String JobInstanceReference	Specifies a unique job instance identifier. For more information, see JobReferenceAndInstanceReferences Structure .

Arguments	Description
String SessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

removeJobs() Method

Use the removeJobs() method to remove associated jobs, given a list of Job References.

If you try to remove a job that does not exist, it will be ignored and logged. The number of Jobs actually removed is returned.

Signature

Unsigned Integer removeJobs(List JobReference[], String SessionID);

Arguments	Description
List JobReference[]	Specifies the job identifiers of the removed jobs. For more information, see JobReferenceAndInstanceReferences Structure .
String SessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

purgeJobInstances() Method

Use the purgeJobInstances() method to purge Job Instances based on a filter.

The purgeJobInstancesFilter must contain either a populated list of Job References or UserIDs, and if this is not done correctly then it will result in an Invalid Choice SOAP fault being raised. Attempting to purge with a Job Reference or UserID that does not exist will be ignored and logged.

The following properties can be applied when purging job instances:

- Choice of only one of the following: [Mandatory]:
 - List of Job Reference [Optional].
 - List of User IDs [Optional].

Signature

Void purgeJobinstances(List[] PurgeJobInstancesFilter, String SessionID);

Arguments	Description
List[] PurgeJobInstancesFilter	Specifies filter properties for the Job Instances to purge. For more information, see PurgeJobInstancesFilter Structure .
String SessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Examples of Using the SchedulerService API

This section assumes that the reader is familiar with Java and SOAP based web services. The Java code that follows is intended to illustrate API usage.

- [Example - Creating a Session and Scheduler Service](#)
- [Example - Finding Job References for a User and Displaying the Job Names](#)
- [Example - Finding Job Instances for Two Given Job References](#)
- [Example - Cancelling All Job Instances Associated with a Job Reference](#)
- [Example - Finding and Displaying all Agents in the WebCatalogService](#)

Example - Creating a Session and Scheduler Service

As with all services detailed here, to use the scheduler service you must first establish a `SAWSession`, then instantiate a handle to the service endpoint:

```
SAWSessionService sessionService = new SAWSessionService();

SAWSessionServiceSoap sessionServiceSoap =
    sessionService.getSAWSessionServiceSoap();

String sessionId = sessionServiceSoap.logon("<USER_NAME>", "<PASSWORD>");

SchedulerService schedulerService = new SchedulerService();

SchedulerServiceSoap schedulerServiceSoap =
    schedulerService.getSchedulerServiceSoap();
```

Example - Finding Job References for a User and Displaying the Job Names

Example code for finding job references for a user and displaying the job names.

This example depends on the code from [Example - Creating a Session and Scheduler Service](#) to create the session and service.

```
JobFilter jobFilter = new JobFilter();

jobFilter.getUserID().add("<USER_ID>");

List<Long> jobRefs = schedulerServiceSoap.getJobReferences(jobFilter,
    sessionId);

for (long jobRef : jobRefs)
{
    Job job = schedulerServiceSoap.getJob(jobRef, sessionId);
    System.out.println("Name = " + job.getName());
}
```

Example - Finding Job Instances for Two Given Job References

Example code for finding job instances for two given job references.

This example depends on the code from [Example - Creating a Session and Scheduler Service](#) to create the session and service.

```
// Create Job Instance Filter
JobInstanceFilter jobInstanceFilter = new JobInstanceFilter();
jobInstanceFilter.getJobReference().add(<JOB_REFERENCE_1>);
jobInstanceFilter.getJobReference().add(<JOB_REFERENCE_2>);
jobInstanceFilter.setJobInstanceStatus(null);

List<JobReferenceAndInstanceReferences> listJobRefAndInstanceRefs =
schedulerServiceSoap.getJobInstanceReferences(jobInstanceFilter, sessionId);

System.out.println("Number of Jobs with Instances found: " +
listJobRefAndInstanceRefs.size());
for (JobReferenceAndInstanceReferences jobRefAndInstanceRefs :
listJobRefAndInstanceRefs)
{
    System.out.println("Job Ref = " +
        jobRefAndInstanceRefs.getJobReference() + ", Job Instance Refs = " +
        jobRefAndInstanceRefs.getJobInstanceReference());
}
```

Example - Cancelling All Job Instances Associated with a Job Reference

This example cancels all Job Instances for the Job with the given Job Reference.

This example also depends on the code from [Example - Creating a Session and Scheduler Service](#) to create the session and service.

```
// Create Job Instance Filter
JobInstanceFilter jobInstanceFilter = new JobInstanceFilter();
jobInstanceFilter.getJobReference().add(<JOB_REFERENCE>);

// Get Job Instance References
List<JobReferenceAndInstanceReferences> listJobRefAndInstanceRefs =
schedulerServiceSoap.getJobInstanceReferences(jobInstanceFilter, sessionId);

// Cancel all the obtained Job Instances
for (JobReferenceAndInstanceReferences jobRefAndInstanceRefs :
listJobRefAndInstanceRefs)
{
    for (BigInteger jobInstanceRef :
        jobRefAndInstanceRefs.getJobInstanceReference())
    {
        System.out.println("Cancelling Job = " +
            jobRefAndInstanceRefs.getJobReference() + ", Job Instance = " +
            jobInstanceRef);
        boolean result = schedulerServiceSoap.cancelJobInstance(jobRef,
            jobInstanceRef, sessionId);
        System.out.println("Result = " + result);
    }
}
```

Example - Finding and Displaying all Agents in the WebCatalogService

This example shows how to iterate through the WebCatalogService and display the path of all the agents found.

This example also depends on the code from [Example - Creating a Session and Scheduler Service](#) to create the session and service.

You could extend this example, so that once an agent has been found, it can be used to call iBot web service methods (for example, Enable the Agent using the Agent path found).
[iBotService Service](#)

```
WebCatalogService webCatalogService = new WebCatalogService();
WebCatalogServiceSoap webCatalogServiceSoap =
webCatalogService.getWebCatalogServiceSoap();

void outputAgentsInFolder(String path) {    List<ItemInfo> items = null;    try
{        // Get folder items        items =
webCatalogServiceSoap.getSubItems(path, "*", false, null, sessionId);
for (ItemInfo itemInfo : items) {            // If item is Agent then display
Agent path            if (itemInfo.getSignature().equals("coibot1"))
{                System.out.println("Agent Path = " +
itemInfo.getPath());            }            // If item is a folder then recurse
if (itemInfo.getType() == ItemInfoType.FOLDER)
{                outputAgentsInFolder (itemInfo.getPath());            }        } catch
(SOAPFaultException soapFault){            System.out.println("SOAP Fault for
path: " + path);} catch (Exception e) {            e.printStackTrace();        }}
```

SecurityService Service

Use the SecurityService service to provide methods for identifying accounts and privileges.

Method Names	Description
forgetAccountsEx() Method	Removes accounts from the catalog.
getAccounts() Method	Searches for Oracle Analytics user accounts.
getGlobalPrivilegeACL() Method	Gets the Access Control List for global privileges.
getGlobalPrivileges() Method	Gets the list of all global privileges.
getPermissions() Method	Get the list of permissions for the specified user.
getPermissionsEx() Method	Get the list of permissions for the specified user.
getPrivilegesStatus() Method	Lists all privileges and their statuses.
renameAccountsEx() Method	Changes the name of an user account.
updateGlobalPrivilegeACL() Method	Updates the Access Control List for global privileges.

forgetAccountsEx() Method

Use the forgetAccountsEx() method to remove accounts from the catalog.

Signature

`forgetAccountsStatus forgetAccountsEx(ForgetAccount forgetAccountsList, String sessionID);`

Argument	Description
ForgetAccount forgetAccountsList	Specifies the accounts to forget, supplied in the ForgetAccount structure. For information about the ForgetAccount structure, see ForgetAccount Structure .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a list of account names with status of forgetAccounts operation in the ForgetAccountsStatus Structure.

See [ForgetAccountsStatus Structure](#).

getAccounts() Method

Use the getAccounts() method to search for Oracle Analytics user accounts (for example, LDAP users, catalog groups, or application roles).

Signature

`List[] getAccounts(List[], String sessionID);`

Argument	Description
List[]	Specifies user names, catalog group names, and application role names. A flag indicates if the name is a user, group, or application role.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getGlobalPrivilegeACL() Method

Use the getGlobalPrivilegeACL() method to retrieve the Access Control List for global privileges.

Signature

`ACL getGlobalPrivilegeACL(String privilegeName, String sessionID);`

Argument	Description
String privilegeName	Specifies the name of the privilege to retrieve.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getGlobalPrivileges() Method

Use the `getGlobalPrivileges()` method to retrieve the list of global privileges.

Signature

```
List[] getGlobalPrivileges(String sessionID);
```

Argument	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getPermissions() Method

Use the `getPermissions()` method to retrieve a list of permissions for the specified user, based on the specified access control list.

This method also returns any permissions that are inherited by a user's security group, even if the access control list doesn't specify the group's permissions.

Signature

```
List[] getPermissions(List[], Account account, String sessionID);
```

Argument	Description
List[]	Specifies the access control list for the user specified by Account account.
Account account	Specifies the name of the user for whom to find permission for the ACLs. Can be the user's name or a GUID.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns permissions information in the `permissionMask` field in the `AccessControlToken` structure.

See [AccessControlToken Structure](#).

getPermissionsEx() Method

Use the `getPermissionsEx()` method to retrieve a list of permissions for the specified user, owner, or creator, based on the specified access control list.

Signature

```
List[] getPermissionsEx(List[], Account account, Owner owner, Creator creator, String sessionID);
```

Argument	Description
List[]	Specifies the access control list for the user specified by Account account.
Account account	Specifies the name of the user for whom to find permission for the ACLs. Can be the user's name or a GUID.
Owner owner	Specifies the name of the owner for whom to find permission for the ACLs. Can be the owner's user name or a GUID.
Creator creator	Specifies the name of the creator for whom to find permission for the ACLs. Can be the creator's user name or a GUID.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns permissions information in the permissionMask field in the AccessTokenToken structure.

See [AccessControlToken Structure](#)).

getPrivilegesStatus() Method

Use the getPrivilegesStatus() method to list all privileges and their statuses.

Signature

```
List[] getPrivilegesStatus(List[] privileges, String sessionID);
```

Argument	Description
List[] privileges	Specifies a list of privileges.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

renameAccountsEx() Method

Use the renameAccountsEx() method to rename accounts.

Signature

```
RenameAccountsStatus renameAccountsStatusEx(RenameAccount renameAccountsList,  
String sessionID);
```

Argument	Description
RenameAccount renameAccountsList	Specifies a list of old names and new names with their account types in the RenameAccount structure. For information about the RenameAccount structure, see RenameAccountsStatus Structure
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a list of account names with status of renameAccounts operation in the RenameAccountsStatus Structure.

See [RenameAccountsStatus Structure](#).

updateGlobalPrivilegeACL() Method

Use the updateGlobalPrivilegeACL() method to update the Access Control List for global privileges.

Signature

```
void updateGlobalPrivilegeACL(String privilegeName, ACL acl, UpdateACLParams updateACLParams, String sessionID);
```

Arguments	Description
String privilegeName	Specifies the name of privilege to update.
ACL acl	Specifies the Access Control List to update, supplied in the ACL structure. For information about the ACL structure, see ACL Structure .
UpdateACLParams updateACLParams	Specifies the Access Control List parameters to update, supplied in the UpdateACLParams structure. For information about the UpdateACLParams structure, see UpdateACLParams Structure .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

UserPersonalizationService Service

Use the UserPersonalizationService service to manage favorites and favorite categories.

Method Names	Description
addFavorite() Method	Adds a catalog object as a favorite in favorite list.
deleteFavorite() Method	Deletes an existing favorite item from favorite manager.
getFavorites() Method	Returns all existing favorite items and category.
updateFavorites() Method	Removes all existing favorite items and category and regenerates all favorite manager items using the supplied list of favorite items.
getMostRecents() Method	Returns all most recent used items.

addFavorite() Method

Use the addFavorite() method to add a catalog object as favorite in the favorite list.

Signature

```
void addFavorite(String catalogObjectPath, String categoryPath, String sessionID);
```

Arguments	Description
String catalogObjectPath	Specifies the catalog object path to add as a favorite.
String categoryPath	Specifies the category location (in favorite manager) to create the favorite item. If blank, the favorite item is added as a root element in favorite manager.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

deleteFavorite() Method

Use the deleteFavorite() method to delete an existing favorite item from favorite manager.

Signature

```
void deleteFavorite(String catalogObjectPath, String categoryPath, String sessionID);
```

Arguments	Description
String catalogObjectPath	Specifies the catalog object path for which to delete the favorite item.
String categoryPath	Specifies the category location (in favorite manager) from where favorite item needs to be deleted. If empty, catalog object will be removed from all categories, that is, it will be marked as non favorite object.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getFavorites() Method

Use the getFavorites() method to return all existing favorite items and category.

Signature

```
List[] getFavorites(String categoryPath, boolean recursive, boolean categoryOnly, String sessionID);
```

Arguments	Description
String categoryPath	Specifies the category location (in favorite manager) to create favorite item. If blank, favorite item is added as root element in favorite manager.
Boolean recursive	Specifies whether the child-level categories were included in the result.
Boolean categoryOnly	Specifies if only categories were included in the result and excluding all favorite items.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

All existing favorite items and category.

updateFavorites() Method

Use the `updateFavorites()` method to remove all existing favorite items and category, and regenerate all favorite manager items using supplied list of favorite items.

Signature

```
void updateFavorites(List [] favoriteItems, String sessionId);
```

Arguments	Description
List [] favoriteItems	Specifies a list of FavoriteItem structure, for more information, see ForgetAccount Structure . In this FavoriteItem structure, there is no need to fill ItemInfo structure (see ItemInfo Structure), only name, path and type parameters are mandatory. The catalogPath argument is required to add a catalog object in favorite manager. Additionally for optional subItems, you can include a nested FavoriteItem list.
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getMostRecents() Method

Use the `getMostRecents()` method to return all most recent used items.

Signature

```
List[] getMostRecents(UnsignedShort listType, String sessionId);
```

Arguments	Description
UnsignedShort listType	listType value is as following flags: 1 = Recently Updated 2 = Recently Viewed 3 = Frequently Viewed 4 = Recent 5 = Suggestions
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

A list of MRUItem.

See [MRUItem Structure](#).

WebCatalogService Service

Use the WebCatalogService service to provide methods for navigating and managing the catalog, and to read and write catalog objects in XML format.

Enumeration and Method Names	Description
copyItem() Method	Copies an object from one location to another in the catalog.
copyItem2() Method	Generates an archive file from the catalog.
createFolder() Method	Creates a new folder in the catalog.
createLink() Method	Creates a link to the catalog.
deleteItem() Method	Deletes an object from the catalog.
getItemInfo() Method	Retrieves catalog information for an object.
getMaintenanceMode() Method	Retrieves the maintenance mode status.
getObjectCategories() Method	Retrieves all supported categories.
getObjectCreateList() Method	Retrieves all creatable objects.
getObjectTypes() Method	Retrieves all supported catalog object types.
getSubItems() Method	Retrieves the collection of child subitems for an object in the catalog.
getUnarchiveJobState() Method	Retrieves the state of the unarchive job.
getUserHomeDirPath() Method	Retrieves the home directory path of a user.
maintenanceMode() Method	Locks the catalog during maintenance.
moveItem() Method	Moves an object in the catalog to a different location in the catalog.
pasteItem2() Method	Pastes the copied items. The pasteItem2() method runs synchronously. The pasteItem2() method unarchives the archive file generated by the copyItem2() method. For large catalog archives, it's recommended to use the asynchronous unarchive() method instead of the pasteItem2() method.
readObjects() Method	Reads an object from the catalog.
removeFolder() Method	Deletes a folder from the catalog.
setItemAttributes() Method	Sets attribute flags for the specified catalog item.
setItemProperty() Method	Sets a property for an object in the catalog.
setOwnership() Method	Sets ownership of the specified item.
unarchive() Method	Unarchives an archived object.
updateCatalogItemACL() Method	Update the Access Control List for an item in the catalog.
writeObjects() Method	Writes a list of objects to the catalog.

copyItem() Method

Use the copyItem() method to copy an object from one location in the catalog to another location in the catalog.

Signature

```
void copyItem(String pathSrc, String pathDest, int flagACL, String sessionID);
```

Arguments	Description
String pathSrc	Specifies the current path to the object in the catalog.

Arguments	Description
String pathDest	Specifies the location in the catalog where the object should be copied.
int flagACL	Specified whether the item is copied with security. 0 indicates that the item is copied without security. 1 indicates that the item is copied with security.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

copyItem2() Method

Use the copyItem2() method to generate an archive file from the catalog.

Signature

DataHandler copyItem2(List[] path, boolean recursive, boolean permissions, boolean timestamps, boolean useMtom, String skipPath, CopyItem2Params options, String sessionID);

Arguments	Description
List[] path	Specifies the location in the catalog from which the archive was created.
boolean recursive	Specifies whether the child-level folders were included in the archive.
boolean permissions	Specified whether the items are copied with security.
boolean timestamps	Specifies whether to preserve the items' time stamps were preserved.
boolean useMtom	Specifies whether MTOM was used to encode the content returned by the SOAP message.
String skipPath	Specifies the location of the folder to skip while archiving.
CopyItem2Params options	Specifies the source application.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

createFolder() Method

Use the createFolder() method to create a folder in the catalog.

Signature

void createFolder(String path, boolean createlfNotExists, boolean createIntermediateDirs, String sessionID);

Arguments	Description
String path	Specifies the location in the catalog where the folder should be created, including the name of the new folder.
boolean createlfNotExists	If set to TRUE, then the folder object is created in the catalog if it does not already exist. If set to FALSE, then the folder object is not recreated if it already exists.
boolean createIntermediateDirs	If set to TRUE, then an intermediate directory is created. If set to FALSE, the an intermediate directory is not created.

Arguments	Description
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

createLink() Method

Use the createLink() method to create a link to the catalog.

Signature

```
void createLink(String path, String pathTarget, boolean overwriteIfExists, String sessionId);
```

Arguments	Description
String Path	Specifies the path to the parent object in the catalog.
String TargetPath	Specifies the location in the catalog to which the link being created should refer.
boolean overwriteIfExists	If set to TRUE, then the link is overwritten if it already exists in the catalog. If set to FALSE, then the link is not overwritten if it already exists in the catalog.
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

deleteItem() Method

Use the deleteItem() method to delete an object from the catalog.

To delete a folder, see [removeFolder\(\) Method](#).

Signature

```
void deleteItem(String path, String sessionId);
```

Arguments	Description
String path	Specifies the path to the object in the catalog.
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

getItemInfo() Method

Use the getItemInfo() method to retrieve catalog information for an object.

Signature

```
ItemInfo getItemInfo(String path, boolean resolveLinks, String sessionId);
```

Arguments	Description
String path	Specifies the path to the object in the catalog.
boolean resolveLinks	If set to TRUE and the path in the catalog refers to a link, then Oracle Analytics retrieves information for the object pointed to by the link.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns catalog information for an object in an `ItemInfo` structure.

See [ItemInfo Structure](#).

getMaintenanceMode() Method

Use the `getMaintenanceMode()` method to retrieve the maintenance mode status.

Signature

Boolean `getMaintenanceMode(String sessionID)`;

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns true or false value indicating current maintenance mode status.

See [maintenanceMode\(\) Method](#).

getObjectCategories() Method

Use the `getObjectCategories()` method to retrieve the supported categories.

Signature

Boolean `getObjectCategories(String sessionID)`;

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a list of supported categories.

getObjectCreateList() Method

Use the getObjectCreateList() method to retrieve a list of all creatable objects.

Signature

```
Boolean getObjectCreateList(String sessionID);
```

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a list of creatable objects.

getObjectTypes() Method

Use the getObjectTypes() method to retrieve a list of all supported catalog object types.

Signature

```
Boolean getObjectTypes(String sessionID);
```

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a list of supported object types.

getSubItems() Method

Use the getSubItems() method to retrieve the collection of child sub-items for an object in the catalog.

Signature

```
List[] getSubItems(String path, String mask, boolean resolveLinks, GetSubItemsParams options, String sessionID);
```

Arguments	Description
String path	Specifies the path to the parent object in the catalog.
String mask	Specifies a mask that indicates the child subitems to retrieve. The mask character is an asterisk (*). To retrieve all child subitems, use a single asterisk.

Arguments	Description
boolean resolveLinks	If set to TRUE and the path in the catalog refers to a link, then information is retrieved for the child subitems of the object pointed to by the link.
GetSubItemsParams options	Optional. Specifies parameters to supply to the GetSubItemsParams structure. For information about the GetSubItemsParams structure, see GetSubItemsParams Structure .
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns a collection of child subitems in an ItemInfo structure.

See [ItemInfo Structure](#).

getUnarchiveJobState() Method

Use the getUnarchiveJobState() method to verify the state of an unarchive job.

Signature

getUnarchiveJobStateResult getUnarchiveJobState(String jobID, String sessionID);

Argument	Description
String jobID	Specifies the ID of the unarchived job to verify.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns getUnarchiveJobStateResult.

getUserHomeDirPath() Method

Use the getUserHomeDirPath() method to retrieve the home directory path of a user.

Signature

String getUserHomeDirPath(String user name, String sessionID);

Arguments	Description
String user name	Specifies the name of the user
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns the path of the specified user's home directory.

maintenanceMode() Method

Use the maintenanceMode() method to lock the catalog during maintenance.

Signature

```
void maintenanceMode(boolean flag, String sessionID);
```

Arguments	Description
boolean flag	Set to TRUE if the catalog is locked.
String sessionID)	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

moveItem() Method

Use the moveItem() method to move an object in the catalog to a different location in the catalog.

Signature

```
void moveItem(String pathSrc, String pathDest, int flagACL, String sessionID);
```

Arguments	Description
String pathSrc	Specifies the current path to the object in the catalog.
String pathDest	Specifies the location in the catalog where the object should be moved.
int flagACL	Specified whether the item is moved with security. 0 indicates that the item is moved without security. 1 indicates that the item is moved with security.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

pasteItem2() Method

Use the pasteItem2() method to paste the copied items.

Signature

```
pasteItem2(Binary stream DataHandler archive, String replacePath, int flagACL, int flagOverwrite, String sessionID);
```

Arguments	Description
Binary stream DataHandler archive	Specifies the returned content of the item as string or bytes. What you specify in this field is determined by the readObjects method.
String replacePath	Specifies the location to paste the copied item.
int flagACL	Specifies whether the item is pasted with security. 0 indicates that the item is pasted without security. 1 indicates that the item is pasted with security. However, whether the ACLs are updated on the target path depends on the value flagOverwrite.

Arguments	Description
int flagOverwrite	Specifies whether the pasted item overwrites existing item. 0 indicates replace all, 1 indicates replace old, 2 indicates replace none, and 3 indicates force replace.
int flagReplaceReferences	Specifies whether to replace the path references in the xml objects.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

readObjects() Method

Use the readObjects() method to read an object from the catalog and return a CatalogObject structure.

Signature

```
List[] readObjects(List[] paths, boolean resolveLinks, ErrorDetailsLevel errorMode,
ReadObjectsReturnOptions returnOptions, String sessionID);
```

Arguments	Description
List[] paths	Specifies the location of the object in the catalog.
boolean resolveLinks	If set to TRUE and the path in the catalog refers to a link, then the object is written to the location pointed to by the link.
ErrorDetailsLevel errorMode	Specifies the amount of error information in the errorInfo field in the CatalogObjects structure. For more information, see CatalogObject Structure .
ReadObjectsReturnOptions returnOptions	Specifies a list of valid values.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns an array of CatalogObject Structure objects.

See [CatalogObject Structure](#).

If a read operation fails for a catalog object (for example, due to an invalid path or insufficient privileges), the errorInfo field for that object contains a description of the error.

removeFolder() Method

Use the removeFolder() method to delete a folder and its contents from the catalog.

To delete an object other than a folder and its contents, see [deleteItem\(\) Method](#).

Signature

```
void removeFolder(String path, boolean recursive, String sessionID);
```

Arguments	Description
String path	Specifies the path to the folder in the catalog.
boolean recursive	If set to TRUE, then remove the specified folder and its contents. If set to FALSE, then only remove the specified folder if it is empty, otherwise display an exception message.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

setItemAttributes() Method

Use the setItemAttributes() method to set attribute flags for a specified catalog item.

Signature

```
void setItemAttributes (List[] path, int value, int valueOff, boolean recursive, String sessionID);
```

Arguments	Description
List[] path	Specifies the path to the folder in the catalog.
int value	Specifies which attributes is added. Specifies a combination of the following flags: 1 = read only 2 = archive 4 = hidden 8 = system
int valueOff	Specifies which attributes is removed. See the above int value cell for flags.
boolean recursive	Specifies whether to set the properties of items in sub-directories.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

setItemProperty() Method

Use the setItemProperty() method to set a property for an object in the catalog.

Signature

```
void setItemProperty(List[] path, List[] name, List[] value, boolean recursive, String sessionID);
```

Arguments	Description
List[] path	Specifies the path to the object in the catalog.
List[] name	Specifies the name of the property to set.
List[] value	Specifies the new setting for the property.
boolean recursive	Specifies whether to set the properties of items in sub-directories.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

setOwnership() Method

Use the setOwnership() method to take ownership of the specified item.

Signature

```
void setOwnership(List[]path, Account owner, boolean recursive, String sessionID);
```

Arguments	Description
List[] path	Specifies the location in the catalog of the object to take ownership.
Account owner	Specifies the account to assign as owner.
boolean recursive	If set to TRUE, then apply this action to the specified folder and its contents. If set to FALSE, then only apply this action to the specified folder if it is empty, otherwise display an exception message.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

unarchive() Method

Use the unarchive() method to unarchive an archived object.

Signature

```
unarchiveResult unarchive(Base64Binary archive, String replacePath, Integer flagACL, Integer flagOverwrite, Integer flagReplaceReferences, UnarchiveParams options, String sessionID);
```

Argument	Description
Base64Binary archive	Specifies the object to unarchive.
String replacePath	Specifies the location to unarchive the object.
Integer flagACL	Specifies whether the item is unarchived with security. 0 indicates that the item is unarchived without security. 1 indicates that the item is unarchived with security.
Integer flagOverwrite	Specifies whether the unarchived item overwrites existing item. 0 indicates replace all, 1 indicates replace old, 2 indicates replace none, and 3 indicates force replace.
Integer flagReplaceReferences	Specifies whether to replace the path references in the XML objects.
UnarchiveParams options	Specifies the options for unarchiving.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns getUnarchiveJobStateResult.

updateCatalogItemACL() Method

Use the updateCatalogItemACL() method to update the Access Control List for an item in the catalog.

Signature

void updateCatalogItemACL(List[] path, ACL acl, UpdateCatalogItemACLParams options, String sessionId);

Fields	Description
List[] path	Specifies the path to the object in the catalog.
ACL acl	Specifies the Access Control List. For more information, see ACL Structure .
UpdateCatalogItemACLParams options	Specifies additional parameters. For more information, see UpdateCatalogItemACLParams Structure .
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

writeObjects() Method

Use the writeObjects() method to write an array of objects to the catalog.

Signature

List[] writeObjects(List[] catalogObjects, boolean allowOverwrite, boolean createIntermediateDirs, ErrorDetailsLevel errorMode, String sessionId);

Argument	Description
List [] catalogObjects	Specifies the objects to write to the catalog, supplied in the CatalogObject structure. For information about the CatalogObject structure, see CatalogObject Structure . All fields of object.itemInfo are ignored, except for the array of item properties, which are applied to the object. The signature of the resulting document is always COXMLDocument1.
boolean allowOverwrite	If set to TRUE, then if the object already exists in the catalog, it is overwritten. If set to FALSE, then if the object already exists in the catalog, it is not overwritten.
boolean createIntermediateDirs	If set to TRUE and the path in the catalog refers to a link, then the object is written to the location pointed to by the link.
ErrorDetailsLevel errorMode	Specifies the amount of error information in the errorInfo field in the CatalogObject Structure .
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Array of ErrorInfo Structure objects.

See [ErrorInfo Structure](#).

XMLViewService Service

Use the XMLViewService service to retrieve results from Oracle Analytics Presentation Services in XML format.

Enumeration and Method Name	Description
cancelQuery() Method	Cancels the current query.
executeSQLQuery() Method	Runs a SQL query.
executeXMLQuery() Method	Runs an XML query.
fetchNext() Method	Returns the next page of data rows.
getPromptedFilters() Method	Returns a filter XML structure containing only the analysis' columns with a prompted filter.
upgradeXML() Method	Returns the upgraded XML file.

XMLQueryOutputFormat Enumeration

The XMLQueryOutputFormat enumeration specifies a list of values for the [executeSQLQuery\(\) Method](#) and [executeXMLQuery\(\) Method](#).

This enumeration specifies a list of valid values for the [executeSQLQuery\(\) Method](#) and [executeXMLQuery\(\) Method](#). For example, you might want to return data rows and metadata, or data rows only.

Note

Only one of the values in XMLQueryOutputFormat can be selected.

Values	Description
String SAWRowsetData	Specifies that the query returns only data rows.
String SAWRowsetSchema	Specifies that the query returns only metadata.
String SAWRowsetSchemaAndData	Specifies that the query returns both metadata and data rows.

cancelQuery() Method

Use the `cancelQuery()` method to cancel a query and clean up resources associated with the query.

This method should only be used if the query row set is not scrolled to the last row in the data set returned.

Note

If you use this method when the query row set is scrolled to the last row in the data set returned, query data is cleaned up during the last `fetchNext` method invocation.

Signature

```
void cancelQuery(String queryID, String sessionID);
```

Argument	Description
String queryID	Specifies the unique ID of the query.
String sessionID	Specifies the unique ID of the session.

executeSQLQuery() Method

Use the `executeSQLQuery()` method to execute a SQL query and return the results of the query.

If the results returned exceed one page, you need to use the [fetchNext\(\) Method](#) to return the next page of rows.

Signature

```
QueryResults executeSQLQuery(String sql, XMLQueryOutputFormat outputFormat,  
XMLQueryExecutionOptions executionOptions, String sessionID);
```

Argument	Description
String sql	Specifies the string of SQL code to execute.
XMLQueryOutputFormat outputFormat	Specifies the output format (for more information, see XMLQueryExecutionOptions Structure).
XMLQueryExecutionOptions executionOptions	Specifies the query execution options (for more information, see XMLQueryExecutionOptions Structure).
String sessionID	Specifies the unique ID of the session.

Returns

Returns the results of the query as one or more rows of data in a `QueryResults` structure.

See [QueryResults Structure](#).

executeXMLQuery() Method

Use the `executeXMLQuery()` method to execute an XML query and return the results of the query.

If the results returned exceed one page, you need to use the [fetchNext\(\) Method](#) to return the next page of rows.

Signature

```
QueryResults executeXMLQuery(ReportRef report, XMLQueryOutputFormat outputFormat,  
XMLQueryExecutionOptions executionOptions, ReportParams reportParams,  
String sessionID);
```

Argument	Description
ReportRef reportRef	Specifies the analysis definition, supplied in the ReportRef common structure.
XMLQueryOutputFormat outputFormat	Specifies the output format (for more information, see XMLQueryExecutionOptions Structure).
XMLQueryExecutionOptions executionOptions	Specifies the query execution options (for more information, see XMLQueryExecutionOptions Structure).
ReportParams reportParams	Optional. Specifies the filters or variables to apply to the analysis before execution, supplied in the ReportParams common structure. For information about the ReportParams structure, see ReportParams Structure . When you use filterExpressions in executeXMLQuery(), make sure the sawx:sqlExpression specifies the column of table where you want to apply the filter.
String sessionID	Specifies the unique ID of the session.

Returns

Returns the results of the query as one or more rows of data in a QueryResults structure.

See [QueryResults Structure](#).

fetchNext() Method

Use the fetchNext() method to return the next page of rows retrieved by a query.

The page returned might contain zero rows. If the finished flag is not set, the remaining rows might not be available immediately.

Signature

QueryResults fetchNext(String queryID, String sessionID);

Argument	Description
String queryID	Specifies the unique ID of the query, which is returned in the QueryResults object.
String sessionID	Specifies the unique ID of the session.

Returns

Returns the next page of query results as one or more rows of data in a QueryResults structure.

See [QueryResults Structure](#).

getPromptedFilters() Method

Use the getPromptedFilters() method to retrieve a saved analysis' prompted columns or the prompted columns from an analysis' XML definition.

Note that to create an analysis with a prompted column, you must assign the isPrompted operator to it.

Signature

```
List[] getPromptedFilters(ReportRef report, String sessionID);
```

Argument	Description
ReportRef report	Specifies the analysis' reportPath or a reportXml (report definition).
String sessionID	Specifies the unique ID of the session.

upgradeXML() Method

Use the upgradeXML() method to upgrade the specified XML file.

Signature

```
upgradeXMLResult upgradeXML(String xml, String sessionID);
```

Argument	Description
String xml	Specifies the XML file to upgrade.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Returns

Returns upgradeXMLResult.

Description of Structures in Oracle Analytics Web Services

This topic describes structures used by the Oracle Analytics session-based web services. This document uses JavaScript-like syntax to describe structures. The exact syntax and implementation depend on the SOAP code generation tool and the target language used by your application development environment.

Topics:

- [AccessControlToken Structure](#)
- [Account Structure](#)
- [ACL Structure](#)
- [AnalysisExportExecutionOptions Structure](#)
- [AnalysisExportResult Structure](#)
- [AuthResult Structure](#)
- [CatalogObject Structure](#)
- [CSPWhitelist Structure](#)
- [ErrorInfo Structure](#)
- [FavoriteItem Structure](#)
- [ForgetAccount Structure](#)
- [ForgetAccountResult Structure](#)

- [ForgetAccountsStatus Structure](#)
- [GetSubItemsParams Structure](#)
- [ItemInfo Structure](#)
- [Job Structure](#)
- [JobFilter Structure](#)
- [JobInstance Structure](#)
- [JobInstanceFilter Structure](#)
- [JobInstanceStatus Enumeration](#)
- [JobReferenceAndInstanceReferences Structure](#)
- [MRUIItem Structure](#)
- [NameValuePair Structure](#)
- [Prompt Structures](#)
- [Privilege Structure](#)
- [PurgeJobInstancesFilter Structure](#)
- [QueryResults Structure](#)
- [RenameAccount Structure](#)
- [RenameAccountResults Structure](#)
- [RenameAccountsStatus Structure](#)
- [ReportADFPParameters Structure](#)
- [ReportHTMLOptions Structure](#)
- [ReportParams Structure](#)
- [ReportHierarchicalColumn Structure](#)
- [ReportRegularColumn Structure](#)
- [ReportRef Structure](#)
- [SAColumn Structure](#)
- [SASubjectArea Structure](#)
- [SATable Structure](#)
- [SAWLocale Structure](#)
- [SAWSessionParameters Structure](#)
- [SegmentationOptions Structure](#)
- [SessionEnvironment Structure](#)
- [StartPageParams Structure](#)
- [UpdateACLParams Structure](#)
- [UpdateCatalogItemACLParams Structure](#)
- [Variable Structure](#)
- [XMLQueryExecutionOptions Structure](#)

AccessControlToken Structure

Use the AccessControlToken structure to describe the permissions granted to a specific account in the access control list. The AccessControlToken structure is used in the [SecurityService Service](#).

AccessControlToken Structure Fields

Fields	Description
Account account	Specifies a reference to the Account structure.
int permissionMask	Specifies a combination of the following flags: 1 = Permission to read item content 2 = Permission to traverse directory 4 = Permission to change item content 8 = Permission to delete an item 16 = Permission to assign permissions to other accounts 32 = Permission to take ownership of the item 2048 = Permission to run an Oracle Analytics Publisher report live 4096 = Permission to schedule an Oracle Analytics Publisher report 8192 = Permission to view output from an Oracle Analytics Publisher report 65535 = Permission to grant full control of the item.

Account Structure

Use the Account structure to hold user names or group names. It has a flag to indicate whether the name is a user or a group.

The Account structure is used in the [SecurityService Service](#).

Account Structure Fields

Fields	Description
String accountName	Specifies an account name or group name.
int accountType	Specifies whether the account is a user or a group or both. Use accountType when it is: - An input to a "non-query" SOAP function (for example, a parameter in calling <code>updateCatalogItemACL()</code>). Or - An output from any SOAP function (for example, as returned data from <code>getAccounts()</code>). 0 = user 1 = catalog group 2 = initblock user 3 = invalid or deleted account

Fields	Description
int accountFindType	Specifies whether the account is a user or a group or both. Use accountFindType when it is being used as an input to a "query" SOAP function (for example, as a parameter in calling getAccounts()): 0 = find a user using name or GUID exact match 1 = find a catalog group using name or GUID exact match 2 = find a application role using name or GUID exact match 3 = find a user OR Catalog group OR application roles using name or GUID exact match Note the following information for advanced use of this field. If accountFindType is greater than or equal to 4, the system treats the Name or GUID as a pattern. 4 = find all users using name or GUID pattern match 5 = find all catalog groups using name or GUID pattern match 6 = find all application roles using name or GUID pattern match 7 = find all users AND webcat groups AND application roles using name or GUID pattern match Using this field in this way can be slow, and result in the system returning many records. When receiving an Account, both Name and GUID are set.
String GUID	Specifies the unique ID which identifies the account.

ACL Structure

Use the ACL structure to hold the access control list (ACL).

The ACL structure is used in the [SecurityService Service](#).

ACL Structure Fields

Fields	Description
AccessControlToken[] accessControlTokens	Specifies the full list of permissions. For more information, see AccessControlToken Structure .
String dummy	For internal purposes.

AnalysisExportExecutionOptions Structure

Use the AnalysisExportExecutionOptions structure to specify the execution options when exporting an analysis.

The AnalysisExportExecutionOptions structure is used in the [AnalysisExportViewsService Service](#).

AnalysisExportExecutionOptions Structure Fields

Fields	Description
boolean async	If set to TRUE, then asynchronous analysis export is enabled. If set to FALSE, asynchronous analysis export is disabled.
boolean useMtom	If set to TRUE, then MTOM is used to encode the content returned by the SOAP message.

Fields	Description
boolean refresh	If set to TRUE, then the server re-submits the query to refresh the data. If set to FALSE, then Oracle Analytics uses data in the cache.

AnalysisExportResult Structure

Use the AnalysisExportResult structure to specify the execution options when exporting the result.

The AnalysisExportResult structure is used in the [AnalysisExportViewsService Service](#).

AnalysisExportResult Structure Fields

Fields	Description
viewData	Specifies the returned content of the view data in the format specified in the AnalysisExportOutputFormat.
String mimeType	Specifies mime type of view data returned.
String queryID	Specifies query ID of the request. You use this as a parameter in completeAnalysisExport method.
completeAnalysisExport	Specifies the status of the download request. The values are: - String InProgress - Specifies that download is in progress - String Error - Specifies that download request resulted in Error. - String Done - Specifies that download is done.

AuthResult Structure

Use the AuthResult structure to specify authorization details during an authentication.

The AuthResult structure is used in the [SecurityService Service](#) (in the [impersonateex\(\) Method](#) and [logonex\(\) Method](#)).

AuthResult Structure Fields

Fields	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.
boolean authCompleted	If set to TRUE, then the authorization is complete. If set to FALSE, then the authorization process is in progress and the logonex or impersonatex process should be called again.

CatalogObject Structure

Use the CatalogObject structure to retrieve or specify all information for a particular catalog object in a single method.

The CatalogObject structure is used in the [WebCatalogService Service](#).

CatalogObject Structure Fields

Fields	Description
String catalogObject	Specifies an XML representation of the object.
catalogObjectBytes	Specifies the returned content of the catalog object as string or bytes. What you specify in this field is determined by the readObjects method.
ItemInfo itemInfo	Specifies catalog information about the object, supplied in the ItemInfo common structure. For information about the ItemInfo structure, see ItemInfo Structure .
ErrorInfo errorInfo	Specifies the level of error information to be supplied as specified by the ErrorDetails argument in the readObjects method.

CSPWhitelist Structure

Use the CSPWhitelist structure to specify the content security policy for the Oracle Analytics instance.

The CSPWhitelist structure is used in the [getCSPDefaultAllowList\(\) Method](#) and [getCSPWhitelist\(\) Method](#).

CSPWhitelist Structure Fields

Fields	Description
String cspWhitelistXml	Specifies a character string that contains a representation of the content security policy.
String sessionId	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

CSPWhitelistXml Structure

Use the CSPWhitelistXml structure to specify the content security policy for the Oracle Analytics instance.

The CSPWhitelistXml structure is used in the [updateCSPWhitelist\(\) Method](#).

CSPWhitelist Structure Fields

Fields	Description
String Directive type	Specifies the content security policy type. Values: • "all" • "img-src" • "frame-src" • "script-src" • "font-src" • "style-src" • "media-src" • "connect-src" • "frame-ancestors" • "form-action"
String value	Specifies the domain to be added to the specified content security policy.

ErrorInfo Structure

Use the ErrorInfo structure to retrieve error information during Presentation Catalog Service method invocations.

The ErrorInfo structure is used in the [WebCatalogService Service](#).

ErrorInfo Structure Fields

Fields	Description
String code	Specifies the error code to display.
String context	Specifies the service and method in which the error occurred.
String details	Specifies detailed information about the error.
String message	Specifies a human-readable description of the error.

Favoriteltem Structure

Use the Favoriteltem structure to retrieve favorite item information during Presentation Catalog Service method invocations.

The Favoriteltem structure is used in the [UserPersonalizationService Service](#).

Favoriteltem Structure Fields

Fields	Description
String name	Specifies the favorite or category name.
String path	Specifies the path of catalog object in case of favorite item or category path in favorite manager in case category object.
UnsignedShort type	Specifies the type stored in Favoriteltem object: Favoriteltem = 0 Category Object = 1

Fields	Description
ItemInfo itemInfo	Specifies catalog information about the object, supplied in the ItemInfo common structure. Valid and stores the actual catalog object if type is 0. Not valid if type is 1. For information about the ItemInfo structure, see ItemInfo Structure .
FavoritelItem favoritelItem	A list of sub items in case type is 1 otherwise not a valid value.

ForgetAccount Structure

Use the ForgetAccount structure to hold the name and type of an account that will be removed from the catalog.

The ForgetAccount structure is used in the [SecurityService Service](#).

ForgetAccount Structure Fields

Fields	Description
String accountName	Specifies account name.
int accountType	Specifies the account type.

ForgetAccountResult Structure

Use the ForgetAccountResult structure to hold the status of the delete accounts operation for each account.

The ForgetAccountResult structure is used in the [SecurityService Service](#).

ForgetAccountResult Structure Fields

Fields	Description
String accountName	Specifies the name of the account.
int accountType	Specifies the type of the account
int Status	Specifies the status of the overall delete accounts operation. The status values are: • 0- Success • 1- Error

ForgetAccountsStatus Structure

Use the ForgetAccountsStatus structure to hold the overall status of the delete accounts operation.

The ForgetAccountsStatus structure is used in the [SecurityService Service](#).

ForgetAccountsStatus Structure Fields

Fields	Description
int Status	Specifies the status of the overall delete accounts operation. The status values are: • 0- Success • 1- Error
ForgetAccountResult[] accountsResult	Specifies the account name and account type with Success or Fail message for each account.

GetSubItemsParams Structure

Use the GetSubItemsParams structure to contain optional parameters used in a getSubItems method.

The GetSubItemsParams structure is used in the [WebCatalogService Service](#).

GetSubItemsParams Structure Fields

Fields	Descriptions
GetSubItemsFilter filter	For internal use only.
boolean includeACL	If set to TRUE, then ACL information is included in the resulting ItemInfo structures.
int withPermission and int withPermissionMask	Specifies that you want to filter the resulting items collection by access level. The only items included in the result are those for which the following expression is true: $(\text{itemPermission} \& \text{withPermissionMask}) = (\text{withPermission} \& \text{withPermissionMask})$ where itemPermission is a combination of permission flags for the current catalog item.
int withAttributes and int withAttributesMask	Specifies that you want to filter the resulting items collection by attribute flags. The only items included in the result are those for which the following expression is true: $(\text{itemAttributes} \& \text{withAttributesMask}) = (\text{withAttributes} \& \text{withAttributesMask})$ Where itemAttributes is a combination of attribute flags for the current catalog item.

ItemInfo Structure

Use the ItemInfo structure to contain catalog information about an object.

The ItemInfo structure is used in the [WebCatalogService Service](#).

ItemInfo Structure Fields

Fields	Description
String path	Specifies the path to the object in the catalog. For example, /users/jchan/analyses/.
ItemInfoType type	Specifies a character string that indicates the type. Valid values are: - Folder - Link - Missing - NoAccess - Object
String caption	Specifies the localized name of the object in the catalog. For example, in French, 'My Folders' is displayed as 'Mes Dossiers'.
int attributes	Specifies a combination of the following flags: 1 = read only 2 = archive 4 = hidden 8 = system
Calendar lastModified	Specifies the date and time that the object was last modified, in Calendar format.
Calendar created	Specifies the date and time that the object was created (saved) in the catalog, in Calendar format.
Calendar accessed	Not used.
String signature	Specifies the signature of the catalog object.
NameValuePair[] itemProperties	Specifies an array of object properties.
ACL aclXX	Specifies the Access Control List for this catalog item.
Account owner	Specifies the owner of the object.
String targetPath	If the ItemInfoType field is set to "Link," this field specifies the target path for the object.

Job Structure

Use the Job structure to contain information about jobs.

The Job structure is used in the [SchedulerService Service](#).

Some Job properties are optional, and may not be present if not relevant to the Job (for example, depending on the Job Trigger type).

Job Structure Fields

Fields	Description
Job Reference	A unique reference for the job associated with an instance.
Name	A short descriptive name for the job.
Description	The text description of the job that describes its actions to end users.
User ID (author ID)	The user ID that created the job.
Script Type	The type of script used to run the job (VBScript, JScript, Java, or NQCmd).

Fields	Description
Script ID (Path of Agent)	The path to the script that runs the job. Use to call the WebCatalogService API to return the agent definition. For more information, see Example - Finding and Displaying all Agents in the WebCatalogService .
Max Run Time (in ms)	The maximum time in milliseconds that the job can run.
Running Instances Count	The total number of currently running instances of this job.
Max Concurrent Instances	The maximum number of concurrent running instances. For an unlimited number of concurrent instances, set this value to zero.
Time Zone	The time zone that is used to execute the job. If missing the timezone is assumed to be the scheduler local timezone.
Last Run Date Time	The last date and time the job started to execute.
Next Run Date Time	The next date and time the job will execute.
Begin Date	The date when the first recurrent interval runs.
Start Time	The time the job starts.
End Date	The date when the first recurrent interval ends.
End Time	The time the job completes.
Interval Minutes	The number of minutes between subsequent executions of a job during the recurrent interval.
Disabled	Specifies that a job script does not execute when the trigger expires.
Delete Job When Done	Specifies whether to delete a job after it completes.
Execute When Missed	Specifies whether to execute a job if running it has failed to occur at the scheduled time.
Job Trigger (with details depending on type)	Specifies what triggers a job including details that depend on job type. A Job Trigger can be one of the following values: RunNever, RunOnce, RunDaily, RunWeekly, RunMonthlyByDate, RunMonthlyByDayOfWeek.

JobFilter Structure

Use the JobFilter structure to filter job lists.

The JobFilter structure is used in the [SchedulerService Service](#).

JobFilter Structure Fields

Fields	Description
List of User IDs (Authors)	(Optional) If no User IDs are specified then all JobReferences are returned.

JobInstance Structure

Use the JobInstance structure to contain information about a job instance corresponding to a running, completed, or cancelled job.

The JobInstance structure is used in the [SchedulerService Service](#).

JobInstance Structure Fields

Fields	Description
Job Reference	A unique reference for the job associated with the job instance.
Job Instance Reference	A unique reference for the job instance.
Job Instance Status	The current status of the job instance. Valid values are: • Completed • Running • Failed • Cancelled • TimedOut • Warning
Begin Date Time	The day and time that the scheduler initiated the job instance.
End Date Time	The day and time that the job scheduler completed the job instance.
Successful Deliveries	The number of successful deliveries for this job instance.
Error Message	The error message, warning, or general message about the job instance execution.

JobInstanceFilter Structure

Use the JobInstanceFilter structure to filter job instance lists.

The JobInstanceFilter structure is used in the [SchedulerService Service](#).

JobInstanceFilter Structure Fields

Fields	Description
List of JobReference (Authors)	(Mandatory) If no Job References are specified then all JobReferences are returned.
Job Instance Status	(Optional) The current status of the Job instance. Valid values are: • Completed • Running • Failed • Cancelled • TimedOut • Warning

JobInstanceStatus Enumeration

Use the JobInstanceStatus enumeration to define job instance state.

The JobInstanceStatus enumeration is used in the [SchedulerService Service](#).

JobInstanceStatus Enumeration Values

Values	Description
Job Instance State	Job instance state is represented using this enumeration. Valid states are: • Completed • Running • Failed • Cancelled • TimedOut • Warning

JobReferenceAndInstanceReferences Structure

Use the JobReferenceAndInstanceReferences structure to group a job and its associated instances.

The JobReferenceAndInstanceReferences structure is important when a list of job references are specified as the selection criteria for job instance listing. This structure is used in the [SchedulerService Service](#).

JobReferenceAndInstanceReferences Structure Fields

Fields	Description
Job Reference	(Mandatory) A unique reference for the job associated with an instance.
List of Job Instance References	(Mandatory) A list of job instances.

MRUItem Structure

Use the MRUItem structure to retrieve most recently used (MRU) item information during Presentation Catalog Service method invocations.

The MRUItem structure is used in the [UserPersonalizationService Service](#).

MRUItem Structure Fields

Fields	Description
String catalogPath	Specifies the catalog path for the recent catalog object.
Boolean isFavorite	Specifies if the MRU item is a favorite item as well.
ItemInfo itemInfo	Specifies catalog information about the object, supplied in the ItemInfo common structure.

NameValuePair Structure

Use the NameValuePair structure to denote named properties, such as COLOR=RED.

The NameValuePair structure is used in the [WebCatalogService Service](#).

NameValuePair Structure Fields

Fields	Description
String name	Specifies a character string that contains the name of the property, such as COLOR.
String value	Specifies a character string that contains the value, such as RED.

ReportHierarchicalColumn Structure

Use the ReportHierarchicalColumn structure to return column properties for a hierarchical column used in an analysis.

The ReportHierarchicalColumn structure is used in [ReportEditingService Service](#).

Note

System wide default column properties that apply to any of the report columns are returned in the report column properties.

ReportHierarchicalColumn Structure Fields

Fields	Description
String ID	Specifies the column identifier of the report column
String tableHeading	Specifies the table heading of the report column
String columnHeading	Specifies the column heading of the report column
Boolean hidden	If set to TRUE, the column is hidden. If set to FALSE, the column is displayed.
String subjectArea	Specifies the subject area of the report column
String tableName	Specifies the table name of the report column
String hierarchyID	Specifies the hierarchy of the report column
String dimensionID	Specifies the dimension of the report column

Prompt Structures

Use the Prompt structures to specify the prompts in the getPromptElements Methods of the ReportEditingService Service.

Prompt structures:

- [PromptsObjectModel Structure](#)
- [PromptCollectionRunTimeInfo Structure](#)
- [PromptStepObjectModel Structure](#)
- [PromptStepRunTimeInfo Structure](#)
- [IndividualPromptObjectModel Structure](#)
- [IndividualPromptRunTimeInfoLimitedByInfo Structure](#)

- [IndividualPromptRunTimeInfo Structure](#)
- [IndividualPromptRunTimeInfoDataTypeHierarchyLevelSQLInfo Structure](#)
- [IndividualPromptRunTimeInfoDataTypeHierarchyLevelInfo Structure](#)
- [IndividualPromptRunTimeInfoDataTypeHierarchyLevels Structure](#)
- [IndividualPromptRunTimeInfoDataTypeHierarchyFormulaLevels Structure](#)
- [IndividualPromptRunTimeInfoDataTypeHierarchyInfo Structure](#)
- [IndividualPromptRunTimeInfoDataTypeDoubleColumnInfo Structure](#)
- [IndividualPromptRunTimeInfoDataType Structure](#)
- [IndividualPromptRunTimeInfoSingleValueType Structure](#)
- [IndividualPromptRunTimeInfoValuesType Structure](#)
- [IndividualPromptRunTimeInfoCurrentValues Structure](#)
- [IndividualPromptRunTimeInfoAvailableOptions Structure](#)
- [IndividualPromptRunTimeInfoDataTypeHierarchyInfo Structure](#)
- [IndividualPromptRunTimeInfoLimitedByPromptReference Structure](#)
- [IndividualPromptRunTimeInfoLimitedByPromptRefGroups Structure](#)

PromptsObjectModel Structure

Use the PromptsObjectModel structure to specify the object model of the prompt.

The PromptsObjectModel structure is used in the [getPromptElements\(\) Method](#).

PromptsObjectModel Structure Fields

Fields	Description
String name	Specifies the name of the prompt.
String description	Specifies the description of the prompt.
String scope	Specifies the scope of the prompt.
String subjectArea	Specifies the subject area of the prompt.
String layout	Specifies the layout of the prompt.
PromptCollectionRunTimeInfo runTimeInfo	Specifies the run time information of the prompt collection.
PromptStepObjectModel promptStepObj	Specifies the object model of the prompt state.

PromptCollectionRunTimeInfo Structure

Use the PromptCollectionRunTimeInfo structure to specify the run time information of the prompt collection.

The PromptCollectionRunTimeInfo structure is used in the [getPromptElements\(\) Method](#).

PromptCollectionRunTimeInfo Structure Fields

Fields	Description
String collectionID	Specifies the collection ID of the prompt.
String viewStatePath	Specifies the view state path of the prompt.
Int currentStep	Specifies the current step of the prompt.
Boolean reloadInline	Specifies whether to reload the prompt inline (true) or not (false).
Boolean supportAutoComplete	Specifies whether the prompt auto completes (true) or not (false).
Boolean showReturnLink	Specifies whether the prompt displays a return link (true) or not (false).
String currentAction	Specifies the current action of the prompt.

PromptStepObjectModel Structure

Use the PromptStepObjectModel structure to specify the step object model of the prompt.

The PromptStepObjectModel structure is used in the [getPromptElements\(\) Method](#).

PromptStepObjectModel Structure Fields

Fields	Description
String title	Specifies the title of the prompt.
String instruction	Specifies the instruction of the prompt.
String buttonsPosition	Specifies the position of the button in the prompt.
String labelPosition	Specifies the position of the label in the prompt.
String wrapLabelText	Specifies the wrap label text in the prompt.
String customWidthUsage	Specifies custom width usage of the prompt.
String customWidthWidth	Specifies custom width of the prompt.
String setWidthToAllPrompts	Specifies whether width applies to all prompts.
Boolean autoApplyPrompt	Specifies whether the prompt is applied automatically (true) or not (false).
Boolean showResetButton	Specifies whether the prompt displays a Reset button (true) or not (false).
PromptStepRunTimeInfo runTimeInfo (nillable)	Specifies the run time information for the prompt step. This can be null.
IndividualPromptObjectModel[] promptObj	Species an array of prompt object models.

PromptStepRunTimeInfo Structure

Use the PromptStepRunTimeInfo structure to specify the step run time information of the prompt.

The PromptStepRunTimeInfo structure is used in the [getPromptElements\(\) Method](#).

PromptStepRunTimeInfo Structure Fields

Fields	Description
Boolean applyToAllSteps	Specifies whether the prompt applies to all steps (true) or not (false).
Boolean autoApplyPrompt	Specifies whether the prompt is applied automatically (true) or not (false).
Boolean showResetButton	Specifies whether the prompt displays the Reset button(true) or not (false).
Int remainingRequiredPromptsOnSubsequentSteps	Specifies the required prompts on subsequent steps.
Int firstPromptStartIndex	Specifies the index of the first prompt.

IndividualPromptObjectModel Structure

Use the IndividualPromptObjectModel structure to specify the prompt object model of the prompt.

The IndividualPromptObjectModel structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptObjectModel Structure Fields

Fields	Description
String label	Specifies the label of the prompt.
String description	Specifies the description of the prompt.
String type	Specifies the type of prompt.
String subjectArea	Specifies the subject area of the prompt.
Boolean placedOnNewColumn	Specifies whether the prompt is applied on a new column (true) or not (false).
Boolean required	Specifies whether the prompt is required (true) or not (false).
String formulaExprString	Specifies the formula expression of the prompt.
String promptUIControlType	Specifies the user interface control type of the prompt.
String promptOperator	Specifies the prompt operator.
String customWidthUsage	Specifies the custom width usage of the prompt.
String customWidthWidth	Specifies the custom width of the prompt.
String setPromptVariableType	Specifies the prompt variable type of the prompt.
String setVariableName	Specifies the variable name of the prompt.
IndividualPromptRunTimeInfo LimitedByInfo limitedByInfo	Specifies the limited by information of a prompt.
IndividualPromptRunTimeInfo runTimeInfo	Specifies the run time information of a prompt.

IndividualPromptRunTimeInfoLimitedByInfo Structure

Use the IndividualPromptRunTimeInfoLimitedByInfo structure to specify the run time limited information of the prompt.

The `IndividualPromptRunTimeInfoLimitedByInfo` structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoLimitedByInfo Structure Fields

Fields	Description
String <code>limitedByType</code>	Specifies whether the prompt is limited by other prompt types, such as none, allPrompts, and specificPrompts.
Boolean <code>isLimitedByNotApplied</code>	Specifies whether constraint is applied to the prompt (true) or not (false).
IndividualPromptRunTimeInfo <code>LimitedByPromptRefGroups</code> specificPrompts	Specifies prompts that limit this prompt.

IndividualPromptRunTimeInfo Structure

Use the `IndividualPromptRunTimeInfo` structure to specify the run time information of the prompt.

The `IndividualPromptRunTimeInfo` structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfo Structure Fields

Fields	Description
Int <code>promptID</code>	Specifies the prompt ID.
Boolean <code>allowUserTypeValues</code>	Specifies whether the user is allowed to type the values of the prompt (true) or not (false).
Boolean <code>allowAutoComplete</code>	Specifies whether the prompt auto completes (true) or not (false).
Boolean <code>multiSelect</code>	Specifies whether the prompt allows multiple selections (true) or not (false).
String <code>showSearch</code>	Specifies to show the search of the prompt.
IndividualPromptRunTimeInfo <code>DataType</code> <code>dataType</code>	Specifies the data type information of a prompt. For more information, see IndividualPromptRunTimeInfoDataType Structure .
IndividualPromptRunTimeInfo <code>CurrentValues</code> <code>currentValues</code>	Specifies the current values of a prompt. For more information, see IndividualPromptRunTimeInfoCurrentValues Structure .
IndividualPromptRunTimeInfo <code>AvailableOptions</code> <code>availableOptions</code>	Specifies the available options for a prompt. For more information, see IndividualPromptRunTimeInfoAvailableOptions Structure .
IndividualPromptRunTimeInfo <code>AdditionalAttributes</code> <code>attributes</code>	Specifies the additional attributes of a prompt. For more information, see IndividualPromptRunTimeInfoDataTypeHierarchyInfo Structure .

IndividualPromptRunTimeInfoDataTypeHierarchyLevelSQLInfo Structure

Use the `IndividualPromptRunTimeInfoDataTypeHierarchyLevelSQLInfo` structure to specify the SQL information of the hierarchy level of the display column of the prompt.

The `IndividualPromptRunTimeInfoDataTypeHierarchyLevelSQLInfo` structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoDataTypeHierarchyLevelSQLInfo Structure Fields

Fields	Description
String displayFormula	Specifies the display formula of the prompt.
String sqlFormula	Specifies the SQL formula of the prompt.
String dataType	Specifies the data type of the prompt.
String category	Specifies the category of the prompt.
String primaryType	Specifies the primary type of the prompt.
Boolean nullable	Specifies whether the prompt displays a nullable value (true) or not (false).
Boolean isMeasure	If set to TRUE, the prompt displays a measure. If set to FALSE, the prompt displays an attribute.
String aggType	Specifies the aggregate type of the prompt.
String aggRule	Species the aggregate rule of the prompt.

IndividualPromptRunTimeInfoDataTypeHierarchyLevelInfo Structure

Use the IndividualPromptRunTimeInfoDataTypeHierarchyLevelInfo structure to specify the level information of a prompt based on a hierarchy column.

The IndividualPromptRunTimeInfoDataTypeHierarchyLevelInfo structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoDataTypeHierarchyLevelInfo Structure Fields

Fields	Description
String levelID	Specifies the level ID of the prompt.
String displayName	Specifies the display name of the prompt.
String displayFormula	Specifies the display formula of the prompt.
String sqlFormula	Specifies the SQL formula of the prompt.
Boolean isDoubleColumn	Specifies whether the prompt displays a double column (true) or not (false).
IndividualPromptRunTimeInfoDataTypeHierarchyLevelSQLInfo keyColumnInfo	Specifies the SQL information of the hierarchy level of the key column of the prompt if the prompt column is a hierarchy level.
IndividualPromptRunTimeInfoDataTypeHierarchyLevelSQLInfo displayColumnInfo	Specifies the SQL information of the hierarchy level of the display column of the prompt if the prompt column is a hierarchy level.

IndividualPromptRunTimeInfoDataTypeHierarchyLevels Structure

Use the IndividualPromptRunTimeInfoDataTypeHierarchyLevels structure to specify the levels of a prompt based on a hierarchy column.

The IndividualPromptRunTimeInfoDataTypeHierarchyLevels structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoDataTypeHierarchyLevels Structure Fields

Fields	Description
IndividualPromptRunTimeInfoDataTypeHierarchyLevelInfo [] levelInfo	Specifies an array of level information of a prompt based on a hierarchy column.

IndividualPromptRunTimeInfoDataTypeHierarchyFormulaLevels Structure

Use the IndividualPromptRunTimeInfoDataTypeHierarchyFormulaLevels structure to specify the metadata/formula information of a prompt based on a hierarchy column.

The IndividualPromptRunTimeInfoDataTypeHierarchyFormulaLevels structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoDataTypeHierarchyFormulaLevels Structure Fields

Fields	Description
String subjectArea	Specifies the subject area of the prompt.
String dimensionID	Specifies the dimension of the prompt.
String tableName	Specifies the table name of the prompt.
String hierarchyID	Specifies the hierarchy ID of the prompt.
String displayName	Specifies the display name of the prompt.
String tableDisplayName	Specifies the display name of a table in the prompt.
String hierarchyDisplayName	Specifies the hierarchy display name of the prompt.
String sqlFormulaIn2Parts	Specifies the two part SQL formula of the prompt.
String sqlFormulaDisplaySubjectAreaPart	Specifies the subject area part of the SQL formula.
IndividualPromptRunTimeInfoDataTypeHierarchyLevels levels	Specifies the levels of a prompt based on a hierarchy column.

IndividualPromptRunTimeInfoDataTypeHierarchyInfo Structure

Use the IndividualPromptRunTimeInfoDataTypeHierarchyInfo structure to specify the hierarchy information of a prompt based on a hierarchy column.

The IndividualPromptRunTimeInfoDataTypeHierarchyInfo structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoDataTypeHierarchyInfo Structure Fields

Fields	Description
String hierarchyID	Specifies the hierarchy ID of the prompt.
String dimensionID	Specifies the dimension ID of the prompt.
String tableName	Specifies the table name of the prompt.

Fields	Description
IndividualPromptRunTimeInfoDataTypeHierarchyFormulaLevels formulaLevels	Specifies the metadata/formula information for a prompt based on a hierarchy column.

IndividualPromptRunTimeInfoDataTypeDoubleColumnInfo Structure

Use the IndividualPromptRunTimeInfoDataTypeDoubleColumnInfo structure to specify the information of a prompt based on a double column.

The IndividualPromptRunTimeInfoDataTypeDoubleColumnInfo structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoDataTypeDoubleColumnInfo Structure Fields

Fields	Description
String codeColumnFormula	Specifies the code column formula of the prompt.
String codeColumnCategory	Specifies the code column category of the prompt.
String codeColumnPrimaryType	Specifies the code of the column primary type of the prompt.
String codeColumnDBPrimaryType	Specifies the code of the database primary type of the prompt.
Boolean enableDoubleColumnInput	Specifies whether the prompt enables input in a double column (true) or not (false).
String codeColumnLabel (nillable)	Specifies the label of the code column for the prompt.
Boolean selectedByCodeValue	Specifies whether the prompt input is in code value (true) or not (false).

IndividualPromptRunTimeInfoDataType Structure

Use the IndividualPromptRunTimeInfoDataType structure to specify the data type information of a prompt.

The IndividualPromptRunTimeInfoDataType structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoDataType Structure Fields

Fields	Description
String subjectArea	Specifies the subject area for the prompt.
String displayColumnFormula	Specifies the column formula for the prompt.
String displayColumnCategory	Specifies the column category for the prompt.
String displayColumnPrimaryType	Specifies the column primary type for the prompt.
String displayColumnDBPrimaryType	Specifies the column data base primary type for the prompt.

Fields	Description
Boolean isMeasureColumn	If set to TRUE, the report column is a measure. If set to FALSE, the report column is an attribute.
String displayTimeZone	Specifies the time zone for the prompt.
Int dataTimeZoneOffset	Specifies the data time zone offset for the prompt.
Int displayToDataOffset	Specifies the data offset for the prompt.
String promptSourceDataType	Specifies the source data type for the prompt display column.
Boolean isHierarchy	Specifies whether the prompt column is a hierarchy column (true) or not (false).
IndividualPromptRunTimeInfo DataTypeHierarchyInfo hierarchyInfo (nillable)	Specifies the hierarchy information of a prompt based on a hierarchy column. This argument can be null.
Boolean isDoubleColumnInput	Specifies whether the prompt input is in a double column (true) or not (false).
IndividualPromptRunTimeInfo DataTypeDoubleColumnInfo codeColumnInfo (nillable)	Specifies the code column information of a prompt based on a double column. This argument can be null.

IndividualPromptRunTimeInfoSingleValueType Structure

Use the IndividualPromptRunTimeInfoSingleValueType structure to specify a single value for the prompt.

The IndividualPromptRunTimeInfoSingleValueType structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoSingleValueType Structure Fields

Fields	Description
String eType	Specifies the type of value in the structure. For example, SQL, customGroup, hierarchyLevels and so on.
String caption	Specifies the caption of the prompt.
String codeValue	Specifies the code value of the prompt.

IndividualPromptRunTimeInfoValuesType Structure

Use the IndividualPromptRunTimeInfoValuesType structure to specify all values that are used by the prompt.

The IndividualPromptRunTimeInfoValuesType structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoValuesType Structure Fields

Fields	Description
IndividualPromptRunTimeInfo SingleValueType[] value	Specifies an array of single values for the prompt.

IndividualPromptRunTimeInfoCurrentValues Structure

Use the IndividualPromptRunTimeInfoCurrentValues structure to specify the current values of a prompt.

The IndividualPromptRunTimeInfoCurrentValues structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoCurrentValues Structure Fields

Fields	Description
String currentOperator	Specifies the current operator of the prompt.
Boolean emptyAsAllChoices	Specifies whether an empty prompt defaults to all choices (true) or not (false).
IndividualPromptRunTimeInfoValuesType values	Specifies the values for the prompt.

IndividualPromptRunTimeInfoAvailableOptions Structure

Use the IndividualPromptRunTimeInfoAvailableOptions structure to specify the available options of a prompt.

The IndividualPromptRunTimeInfoAvailableOptions structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoAvailableOptions Structure Fields

Fields	Description
Int numOptionsPerPage	Specifies the number of options per page.
Int currentPageInOptions	Specifies current page number.
Boolean moreOptions	Specifies whether the prompt consists of more options (true) or not (false).
Boolean includeAllChoices	Specifies whether the prompt include all choices (true) or not (false).
Boolean needToPopulateDropDown	Specifies whether the prompt requires you to populate the drop down (true) or not (false).
String valueTablePromptSourceType	Specifies the source type for the prompt.
String sql	Specifies the SQL for the prompt.
String runTimeCodeAndDisplayValueFormatStr	Specifies the format string for the values of the prompt.
String filterXmlString (nillable)	Specifies the xml filter of the prompt.
IndividualPromptRunTimeInfoValuesType groupPaths (nillable)	Specifies the values of the groups that are used by the prompt. This argument can be null.
IndividualPromptRunTimeInfoValuesType populatedOptions (nillable)	Specifies the populated values that are used by the prompt. This argument can be null.

IndividualPromptRunTimeInfoDataTypeHierarchyInfo Structure

Use the IndividualPromptRunTimeInfoDataTypeHierarchyInfo structure to specify the additional attributes of a prompt.

The IndividualPromptRunTimeInfoDataTypeHierarchyInfo structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoDataTypeHierarchyInfo Structure Fields

Fields	Description
Int opMinNumValues	Specifies the minimum number of values in the prompt.
Int opMaxNumValues	Specifies the maximum number of values in the prompt.

IndividualPromptRunTimeInfoLimitedByPromptReference Structure

Use the IndividualPromptRunTimeInfoLimitedByPromptReference structure to specify prompts that limit this prompt.

The IndividualPromptRunTimeInfoLimitedByPromptReference structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoLimitedByPromptReference Structure Fields

Fields	Description
Int promptID	Specifies the identifier of the prompt.
String formulaExprString	Specifies the formula expression of the prompt.

IndividualPromptRunTimeInfoLimitedByPromptRefGroups Structure

Use the IndividualPromptRunTimeInfoLimitedByPromptRefGroups structure to specify prompts that limit the groups of this prompt.

The IndividualPromptRunTimeInfoLimitedByPromptRefGroups structure is used in the [getPromptElements\(\) Method](#).

IndividualPromptRunTimeInfoLimitedByPromptRefGroups Structure Fields

Fields	Description
IndividualPromptRunTimeInfoLimitedByPromptReference[] limitByPrompt	Specifies an array of prompts that limit this prompt.

Privilege Structure

Use the Privilege structure to represent global privileges.

You configure these privileges using the Manage Privileges screen. The Privilege structure is used in the [SecurityService Service](#).

Privilege Structure Fields

Fields	Description
String name	Specifies the name of a privilege.
String description	Specifies the description of a privilege.

PurgeJobInstancesFilter Structure

Use the PurgeJobInstancesFilter structure when purging job instances.

The PurgeJobInstancesFilter structure is used in the [SchedulerService Service](#).

PurgeJobInstancesFilter Structure Fields

You must choose only one of the two properties.

Fields	Description
List of JobReference	(Optional) If no Job References are specified then all JobReferences are returned.
JobInstanceStatus	(Optional) The current status of the Job instance. Valid values are: • Completed • Running • Failed • Cancelled • TimedOut • Warning

QueryResults Structure

Use the QueryResults structure to specify query details during query execution.

The QueryResults structure is used in the [XMLViewService Service](#) (in the executeXMLQuery method).

QueryResults Structure Fields

Fields	Description
String rowset	Specifies the rowset XML encoded in the string.
String queryID	Specifies the unique ID of the query, which can be used in fetchNext calls.
boolean finished	If set to TRUE, then there are no more rows to return. If set to FALSE, then another fetchNext call is needed to return more rows.

RenameAccount Structure

Use the RenameAccount structure to hold the old name, new name and type of an account which will be renamed.

The RenameAccount structure is used in the [SecurityService Service](#).

RenameAccount Structure Fields

Fields	Description
String oldAccountName	Specifies the old name of the account.
String newAccountName	Specifies the new name of the account.
int accountType	Specifies the account type.

RenameAccountResults Structure

Use the RenameAccountResults structure to hold the status of the rename accounts operation for each account.

The RenameAccountResults structure is used in the [SecurityService Service](#).

RenameAccountResults Structure Fields

Fields	Description
String oldAccountName	Specifies the old name of the account.
String newAccountName	Specifies the new name of the account.
int accountType	Specifies the type of account.
int status	Specifies the status of the overall rename accounts operation. The values are: • 0-Success • 1-Error

RenameAccountsStatus Structure

Use the RenameAccountsStatus structure to hold the overall status of the rename accounts operation.

The RenameAccountsStatus structure is used in the [SecurityService Service](#).

RenameAccountsStatus Structure Fields

Fields	Description
int status	Specifies the status of the overall rename accounts operation. The values are: • 0- Success • 1- Error
RenameAccountResult[] accountsResult	Specifies the old name, new name and account type with Success or Fail message for each account.

ReportADFPParameters Structure

Use the ReportADFPParameters structure to define report parameters (prompt, filter, or variable).

The ReportADFPParameters structure is used in the [ReportEditingService Service](#).

ReportADFPParameters Structure Fields

Field	Description
String name	Specifies the formula of the prompt, filter, or variable
String operator	Specifies the prompt operator
String type	Specifies the type of parameter (filter, prompt, or variable).
String datatype	Specifies the SQL datatype of the parameter
String value	Specifies the field used for an operator needing two operands. Other operators have values specified in the ADFParameterValues field.
ADFPParameterValues values	Specifies the values for this parameter
ADFPParameterValues String []	Specifies an array of string values

ReportHTMLOptions Structure

Use the ReportHTMLOptions structure to define options for displaying results on an HTML page.

The ReportHTMLOptions structure is used in the [HtmlViewService Service](#).

ReportHTMLOptions Structure Field

Field	Description
boolean enableDelayLoading	Internal use only. This field is always set to 1, which means that Oracle Analytics web services is never required to provide results immediately, and displays a message indicating that it is waiting for results.
String linkMode	Specifies whether to display drills or links in the current browser window or a new browser window.

ReportHTMLLinksMode Enumeration

The ReportHTMLLinksMode enumeration specifies a list of valid values for the ReportHTMLLinksMode field.

The ReportHTMLLinksMode enumeration is used in the [ReportHTMLOptions Structure](#).

ReportHTMLLinksMode Enumeration Values

Values	Description
String InPlace	Specifies that drills or links should replace only the content of the current analysis without changing the rest of the page.
String NewPage	Specifies that drills or links should be displayed in a new browser window.
String SamePage	Specifies that drills or links should replace the current browser window.

ReportParams Structure

Use the ReportParams structure to replace existing filters and variables in an analysis.

The ReportParams structure is common to all web services.

ReportParams Structure Fields

Fields	Description
String[] filterExpressions	Specifies an array of Oracle Analytics web services filter expressions in the form Object[] filter_expression, filter_expression ...
Variable[] variables	Specifies an array of variable values to be set before method execution. This structure is used in executeXMLQuery() method and generateReportSQL() method.
NameValuePair[] nameValues	Should be set to NULL. This field is for internal use only.
TemplateInfo[] templateInfos	Should be set to NULL. This field is for internal use only.
String viewName	Specifies which view to use when generating XML data for the analysis.

How Filter Expressions Are Applied to an Analysis in Web Services

Step	Internal Processing
1	Obtains XML representations of the analysis and each filter expression.
2	For each expression element, locates the child node of the type sqlExpression (the type is determined by the value of the xsi:type attribute), and references its inner text.
3	In the analysis XML, locates all nodes that also have a child node of type sqlExpression where the inner text matches that located in the preceding step.
4	Replaces all nodes found in Step 3 with the expression from Step 2.

How Variables Are Applied to an Analysis in Web Services

Step	Internal Processing
1	Obtains XML representations of the analysis.
2	For each variable, locates all nodes in the analysis XML that have a type of variable, attribute scope equal to analysis, and inner text that matches the variable name.
3	Replaces each node located in Step 2 with the new variable value.

ReportRegularColumn Structure

Use the ReportRegularColumn structure to return column properties for a regular column used in an analysis.

The ReportRegularColumn structure is used in [ReportEditingService Service](#).

Note

System wide default column properties that apply to any of the report columns are returned in the report column properties.

ReportRegularColumn Structure Fields

Fields	Description
String ID	Specifies the column identifier of the report column
String tableHeading	Specifies the table heading of the report column
String columnHeading	Specifies the column heading of the report column
Boolean hidden	If set to TRUE, the column is hidden. If set to FALSE, the column is displayed.
String sqlFormula	Specifies the column formula of the report column
Boolean measure	If set to TRUE, the report column is a measure. If set to FALSE, the report column is an attribute column.
aggrRule ReportColumnAggrRule	If the column contains aggregated data, this value specifies the type of aggregation used in the column.

ColumnAggregationRule Values

This structure specifies the default aggregation rule for the column.

The following list shows the aggregation functions available:

- Default
- Server
- Sum
- Average
- Count
- CountDistinct
- Max
- Min
- None
- ServerAggregate
- Unknown

ReportRef Structure

Use the ReportRef structure to reference an analysis.

Use the ReportRef structure in one of the following ways to reference an analysis:

- The location of the analysis in the catalog.
- The ReportDef object that defines the analysis. This field should always be null.
- The XML that defines the analysis.

Note

Only one of the fields in ReportRef should be populated.
The ReportRef structure is common to all web services.

ReportRef Structure Fields

Fields	Description
String reportPath	Specifies a string value that provides the path to the analysis in the catalog. For example, /users/jchan/analyses/.
String reportXML	Specifies a string value that contains the XML that defines the analysis.

SAColumn Structure

Use the SAColumn structure to represent the logical column in the Subject Area.

The SAColumn structure is used in the [MetadataService Service](#).

SAColumn Structure Fields

Fields	Description
String name	Specifies a column name used in SQL statements.
String displayName	Specifies a localized name, used in an Oracle Analytics analysis.
String description	Specifies a string to contain the description of the column name.
boolean nullable	If set to TRUE, then the column can be null.
String dataType	Specifies the type of data that a column contains.
boolean aggregateable	If set to TRUE, then the column can be aggregated.
String aggrRule	If the column contains aggregated data, this value specifies the type of aggregation used.

SADataType Values

The SADataType indicates the type of data that a column contains.

The following list shows the data types available:

- BigInt
- Binary
- Bit
- Char
- Coordinate
- Date
- Decimal
- Double
- Float

- Integer
- Invalid
- LongVarBinary
- LongVarChar
- Numeric
- Real
- SmallInt
- Time
- TimeStamp
- TinyInt
- Unknown
- VarBinary
- VarChar

AggregationRule Values

The AggregationRule specifies the default aggregation rule for the column.

The following list shows the aggregation functions available:

- Avg
- BottomN
- Complex
- Count
- CountDistinct
- CountStar
- DimensionAggr
- First
- Last
- Max
- Min
- None
- Percentile
- Rank
- ServerDefault
- SubTotal
- Sum
- TopN

SASubjectArea Structure

Use the SASubjectArea structure to represent Subject Area attributes.

The SASubjectArea structure is used in the [MetadataService Service](#).

SASubjectArea Structure Fields

Fields	Description
String name	Specifies the table name that is used in SQL statements.
String displayName	Specifies the localized name, used in Oracle Analytics Classic.
String description	Specifies the description of the subject area.
SATable[] tables	Specifies a collection of tables for this subject area.

SATable Structure

Use the SATable structure to represent the logical table in the Subject Area.

The SATable structure is used in the [MetadataService Service](#).

SATable Structure Fields

Fields	Description
String name	Specifies the table name that is used in SQL statements.
String displayName	Specifies the localized name, used in Oracle Analytics Classic.
String description	Specifies the description of the table name.
SAColumn[] columns	Specifies an array of the table's columns.

SAWLocale Structure

Use the SAWLocale structure to define the locale for the current session.

The SAWLocale structure is used in the [SAWSessionService Service](#).

SAWLocale Structure Fields

Fields	Description
String language	Specifies the language code. Values for language should conform to the ones used in Java, in the java.util.Locale class (ISO-639, ISO-3166).
String country	Specifies the country code. Values for country should conform to the ones used in Java, in the java.util.Locale class (ISO-639, ISO-3166).

SAWSessionParameters Structure

Use the SAWSessionParameters structure to define optional parameters for the current session.

The SAWSessionParameters structure is used in the [SAWSessionService Service](#).

SAWSessionParameters Structure Fields

Fields	Description
SAWLocale locale	Specifies the locale to be used, supplied in the SAWLocale structure.

Fields	Description
String userAgent	Specifies whether the HTMLView service is used with current session. It specifies the userAgent string of the browser, where Oracle Analytics Presentation Services HTML content is displayed. Oracle Analytics Presentation Services uses this information to produce browser-specific HTML.
String syndicate	Internal use only.
LogonParameter logonParams	Specifies the parameters used for authentication.
boolean asyncLogon	If set to TRUE, then asynchronous login is enabled. If set to FALSE (default), then asynchronous login is not enabled.
String sessionID	Specifies the unique ID of the session. This field is used in logonex() method and impersonateex() method.

SegmentationOptions Structure

Use the SegmentationOptions structure to define the segment or segment tree to override the defaults specified in the Oracle Marketing Analytics user interface.

The SegmentationOptions structure is used in the [MetadataService Service](#).

SegmentationOptions Structure Fields

Fields	Description
OverrideType cacheOverride	Specifies how you want to override the Oracle Marketing Analytics' "Cache the block for future update counts requests" user interface option. If set to Default, then the cache override is not specified in the structure or the structure is not specified. The Default value specifies to use what is defined in the user interface option for each criteria block. If set to None, the system overrides the user interface-defined values and sets all criteria blocks to disable the "Cache the block for future update counts requests" user interface option. If set to All, the system overrides the user interface-defined values and sets all criteria blocks to enable the "Cache the block for future update counts requests" user interface option.
OverrideType countOverride	Specifies if the system should use the getCounts method to generate the count numbers. If set to Default, then the count override is not specified in the structure or the structure is not specified. If set to All, the system executes the getCounts method. When set to All, the system calculates count numbers for all criteria blocks.
NameValuePair govRules	Specifies a value to enforce the corresponding contract planning rules for the segment or segment tree.
NameValuePair prompts	Specifies the prompt values to apply to the columns in the segment or segment tree. This process filters data when generating counts. If you do not provide a value in this field, then the system does not apply filter criteria to columns in segments.

Fields	Description
Boolean removeCacheHits	Specifies that you want to clear cache entries that contain count information. If set to True, the system queries against the most current data. To do this, the system removes all existing cache entries that contain count information for the target segment or segment tree. The system then repopulates the cache with new count number entries calculated by the getCounts method.
BigDecimal samplingFactor	Specifies the size of the data set for calculating counts. The getCounts method calculates the count number of all criteria blocks against a subset of the data determined by this value. The default value is 100. The default value determines that the count number is calculated against the whole data set.

SessionEnvironment Structure

Use the SessionEnvironment structure to return environment information for the current session.

The SessionEnvironment structure is used in the [SAWSessionService Service](#).

SessionEnvironment Structure Fields

Fields	Description
String userName	Specifies the name of the current user.
ItemInfo homeDirectory	Specifies the full path to the user's home directory in the catalog. For example, /users/<user login ID>.
ItemInfo[] SharedDirectories	Specifies the full paths to shared directories to which the current user has at least read access.

Note

By default, only administrators are allowed to list direct descendents of the "/shared" directory. Retrieving the SessionEnvironment object is the only way to enable users to navigate its shared area.

StartPageParams Structure

Use the StartPageParams structure to define options in startPage method invocations.

The StartPageParams structure is used in the [HtmlViewService Service](#).

StartPageParams Structure Fields

Fields	Description
String idsPrefix	Specifies a prefix to be used with IDs and names of all HTML elements to avoid name conflicts on an HTML page.

Fields	Description
boolean dontUseHttpCookies	If set to TRUE, then Oracle Analytics Presentation Services cannot rely on cookies for passing the sessionID. Instead, the sessionID is included as a parameter in callback URLs.

UpdateACLParams Structure

Use the UpdateACLParams structure to set options in updateACL method invocations.

The UpdateACLParams structure is used in the [SecurityService Service](#).

UpdateACLParams Structure Fields

Fields	Description
UpdateACLMode updateFlag	Specifies how to update the ACL mode.

UpdateACLMode Enumeration

Use the UpdateACLMode enumeration to update the ACL mode.

The UpdateACLMode enumeration specifies a list of valid values for the update flag in the [UpdateACLParams Structure](#).

UpdateACLMode Enumeration Values

Values	Description
String ReplaceACL	Specifies the ACL value to update.
String ReplaceForSpecifiedAccounts	Specifies a list of accounts to update in the ACL.
String DeleteAccountsFromACL	Specifies a list of accounts to remove from the ACL.
String AddPermission	Specifies a list permissions to update for a list of ACL entries
String DeletePermission	Specifies a list permissions to be removed from a list of ACL entries

UpdateCatalogItemACLParams Structure

Use the UpdateCatalogItemACLParams structure to provide additional parameters in the updateCatalogItemACL() method.

The UpdateCatalogItemACLParams structure is used in the [WebCatalogService Service](#).

UpdateCatalogItemACLParams Structure Fields

Fields	Description
UpdateACLMode updateFlag	Specifies how to update the ACL mode.
boolean recursive	If set to TRUE, then the method is applied to the catalog item and all descendents, which are identified by the path. If set to FALSE, then the method is only applied to the catalog item.

Variable Structure

Use the Variable structure to reference a variable in the analysis and replace it with another variable.

The Variable structure is common to all web services.

Variable Structure Fields

Fields	Description
String name	Specifies a character string that contains the name of the variable to replace.
Object value	Specifies the value of the variable.

XMLQueryExecutionOptions Structure

Use the XMLQueryExecutionOptions structure to specify optional parameters during a query.

The XMLQueryExecutionOptions structure is used in the [XMLViewService Service](#) (in the executeXMLQuery method).

XMLQueryExecutionOptions Structure Fields

Fields	Description
boolean async	If set to TRUE, then asynchronous query execution is enabled. If set to FALSE, then asynchronous query execution is disabled.
int maxRowsPerPage	Specifies the maximum number of rows to be returned by a executeXMLQuery or fetchNext method.
boolean refresh	If set to TRUE, then the server re-submits the query to refresh the data. If set to FALSE, then the server uses data in the cache.
boolean presentationInfo	If set to TRUE, then store localized presentation information in the metadata section of the record set XML. Presentation information consists of the following: • Column heading information (stored in the columnHeading field). • Table heading information (stored in the tableHeading field).
String type	Specifies the query ID, which can be used in logs to diagnose errors.

Part VI

Reference

This part provides reference information.

Topics:

- [Schemas for Validating XML Documents](#)

Schemas for Validating XML Documents

This topic lists the schemas used for validating the XML documents.

Topics:

- [analysis_customization.xsd](#)
- [analysis_ibot.xsd](#)
- [condition.xsd](#)

analysis_customization.xsd

This topic lists the content of the analysis_customization.xsd schema file.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="com.siebel.analytics.web/report/v1.1"
xmlns:sawx="com.siebel.analytics.web/expression/v1.1"
xmlns:saw="com.siebel.analytics.web/report/v1.1" xmlns:xs="http://www.w3.org/
2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="reportRef">
    <xs:complexType>
      <xs:sequence minOccurs="0">
        <xs:element ref="saw:filterOverrides"/>
      </xs:sequence>
      <xs:attribute name="path" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="valueString">
    <xs:restriction base="xs:string">
      <xs:maxLength value="200"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="typeString">
    <xs:restriction base="xs:string">
      <xs:maxLength value="50"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="filterOverrides">
    <xs:complexType>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="subjectArea" type="saw:subjectAreaType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="subjectAreaType">
    <xs:sequence>
      <xs:element name="filterOverride" type="saw:filterOverrideType"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

```

        </xs:sequence>
        <xs:attribute name="subjectArea" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:complexType name="filterOverrideType">
        <xs:sequence>
            <xs:element name="value" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="saw:valueString">
                            <xs:attribute name="type" type="saw:typeString"
use="required"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="session" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="saw:valueString">
                            <xs:attribute name="type" type="saw:typeString"
use="required"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="repository" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="saw:valueString">
                            <xs:attribute name="type" type="saw:typeString"
use="required"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="fixed" type="xs:boolean"/>
        <xs:attribute name="hidden" type="xs:boolean"/>
        <xs:attribute name="parentFixed" type="xs:boolean"/>
        <xs:attribute name="parentHidden" type="xs:boolean"/>
        <xs:attribute name="op" use="required">
            <xs:simpleType>
                <!-- equal and notEqual are not used since in and notIn a single
value set are equivlanet -->
                <xs:restriction base="xs:string">
                    <xs:enumeration value="in"/>
                    <xs:enumeration value="notIn"/>
                    <xs:enumeration value="containsAny"/>
                    <xs:enumeration value="containsAll"/>
                    <xs:enumeration value="notContains"/>
                    <xs:enumeration value="like"/>
                    <xs:enumeration value="notLike"/>
                    <xs:enumeration value="beginsWith"/>
                    <xs:enumeration value="endsWith"/>
                    <xs:enumeration value="between"/>
                    <xs:enumeration value="null"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>

```

```

        <xs:enumeration value="notNull"/>
        <xs:enumeration value="less"/>
        <xs:enumeration value="greater"/>
        <xs:enumeration value="lessOrEqual"/>
        <xs:enumeration value="greaterOrEqual"/>
        <xs:enumeration value="top"/>
        <xs:enumeration value="bottom"/>
        <xs:enumeration value="prompted"/>
    </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="column" type="xs:string" use="required"/>
<xs:attribute name="datatype" type="xs:string" use="required"/>
</xs:complexType>
</xs:schema>

```

analysis_ibot.xsd

This topic lists the content of the analysis_ibot.xsd schema file.

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="com.siebel.analytics.web/report/v1.1"
xmlns:sawx="com.siebel.analytics.web/expression/v1.1"
xmlns="com.siebel.analytics.web/report/v1.1"
xmlns:saw="com.siebel.analytics.web/report/v1.1" xmlns:del="com.oracle.bi/
delivers/v1" xmlns:cond="com.oracle.bi/conditions/v1" xmlns:xs="http://
www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:import namespace="com.oracle.bi/conditions/v1"
schemaLocation="condition.xsd"/>
    <xs:include schemaLocation="analysis_base.xsd" />
    <!--<xs:include schemaLocation="analysis.xsd"/>-->
    <!-- An agent is a scheduled job which has a controlling condition,
content, and optional post actions.
        ibot is old name for agent. This xsd is not self contained, since
it uses other schema elements. To check an
        instance of saw:ibot use the top level analysis.xsd which includes
analysis_ibot.xsd.
    -->
    <xs:element name="ibot">
        <xs:complexType>
        <xs:complexContent>
            <xs:extension base="saw:appUpgradeTargetType">
                <xs:sequence>
                    <xs:element name="schedule" type="saw:scheduleType"/>
                    <xs:element name="dataVisibility"
type="saw:dataVisibilityType"/>
                    <xs:element ref="cond:condition"/>
                    <xs:element name="choose" type="chooseType"/>
                    <xs:element name="deliveryDestinations"
type="saw:deliveryDestinationsType">
                        <!-- destinations are all different categories, and
non-null. -->
                        <xs:key name="destinationCategory">
                            <xs:selector xpath="saw:destination"/>

```

```

        <xs:field xpath="@category"/>
    </xs:key>
</xs:element>
<!-- saw:permittedSubscribers. The ibot save xml is
super set of saw:ibot.
The permittedSubscribers is NOT included in
saw:ibot. Instead the
agent's access control list implements the permitted
subscribers.
-->
<!-- The recipients define BI EE users -->
<xs:element name="recipients" type="recipientsType"/>
<xs:element name="emailRecipients"
type="saw:emailRecipientsType" minOccurs="0"/>
</xs:sequence>
<xs:attribute name="version" use="required">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="priority" default="normal">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="high"/>
            <xs:enumeration value="normal"/>
            <xs:enumeration value="low"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<!-- Unique key into the scheduler schema -->
<xs:attribute name="jobID" type="xs:integer" use="optional"/>
<!-- The deleteIBotAfterRun attribute will result in the iBot
being deleted immediately after the first run.
NOTE: Even if the ibot has been scheduled to run
repeatedly till end date, if this attribute is set, the ibot will be deleted
after it's first run.
-->
<xs:attribute name="deleteIBotAfterRun" type="xs:boolean"
use="optional"/>

</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>
<!-- users and groups -->
<xs:complexType name="recipientType">
    <xs:attribute name="name" use="required"/>
    <xs:attribute name="guid" use="required"/>
</xs:complexType>
<!-- Supports ticking zero or more months from the possible twelve months
-->
<xs:complexType name="monthsType">
    <xs:attribute name="jan" type="xs:boolean" use="optional"
default="false"/>

```

```

        <xs:attribute name="feb" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="mar" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="apr" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="may" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="jun" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="jul" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="aug" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="sep" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="oct" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="nov" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="dec" type="xs:boolean" use="optional"
default="false"/>
    </xs:complexType>
    <!-- Define a month schedule by selecting day of week and weeks -->
    <xs:complexType name="monthlyDayOfWeekType">
        <xs:attribute name="day" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="mon"/>
                    <xs:enumeration value="tue"/>
                    <xs:enumeration value="wed"/>
                    <xs:enumeration value="thu"/>
                    <xs:enumeration value="fri"/>
                    <xs:enumeration value="sat"/>
                    <xs:enumeration value="sun"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="first" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="second" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="third" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="fourth" type="xs:boolean" use="optional"
default="false"/>
        <xs:attribute name="last" type="xs:boolean" use="optional"
default="false"/>
    </xs:complexType>
    <!-- Define a month schedule by selecting dates -->
    <xs:complexType name="monthlyDateType">
        <xs:sequence maxOccurs="31">
            <xs:element name="dayOfMonth">
                <xs:complexType>
                    <xs:attribute name="value" use="required">
                        <xs:simpleType>

```

```

        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="31"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<!-- Define the whole schedule of an agent -->
<xs:complexType name="scheduleType">
    <xs:sequence>
        <xs:element name="start" minOccurs="0">
            <xs:complexType>
                <xs:attribute name="startImmediately" type="xs:boolean"
default="false"/>
                <xs:attribute name="date" type="xs:date" use="optional"/>
                <xs:attribute name="time" type="xs:time" use="optional"/>
                <xs:attribute name="repeatMinuteInterval"
type="xs:integer" use="optional"/>
                <xs:attribute name="endTime" type="xs:time"
use="optional"/>
                <!-- UI folks please uncomment after the bug is fixed at
the UI end .. and delete the one below .. search for UI again
                <xs:attribute name="timeZoneId" type="xs:string"
use="optional"/> -->
            </xs:complexType>
        </xs:element>
        <xs:element name="recurrence" minOccurs="0">
            <xs:complexType>
                <xs:choice minOccurs="0">
                    <xs:element name="daily">
                        <xs:complexType>
                            <xs:attribute name="dayInterval"
type="xs:integer" use="required"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="weekly">
                        <xs:complexType>
                            <xs:attribute name="weekInterval"
type="xs:integer" use="required"/>
                            <xs:attribute name="mon" type="xs:boolean"
use="optional" default="false"/>
                            <xs:attribute name="tue" type="xs:boolean"
use="optional" default="false"/>
                            <xs:attribute name="wed" type="xs:boolean"
use="optional" default="false"/>
                            <xs:attribute name="thu" type="xs:boolean"
use="optional" default="false"/>
                            <xs:attribute name="fri" type="xs:boolean"
use="optional" default="false"/>
                            <xs:attribute name="sat" type="xs:boolean"
use="optional" default="false"/>
                            <xs:attribute name="sun" type="xs:boolean"
use="optional" default="false"/>

```

```

        </xs:complexType>
    </xs:element>
    <xs:element name="monthly">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="months"
type="saw:monthsType"/>
                <xs:choice>
                    <xs:element name="monthlyDayOfWeek"
type="saw:monthlyDayOfWeekType"/>
                    <xs:element name="monthlyDate"
type="saw:monthlyDateType"/>
                </xs:choice>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:choice>
<xs:attribute name="runOnce" type="xs:boolean"
use="optional"/>
<xs:attribute name="endDate" type="xs:date"
use="optional"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="timeZoneId" type="xs:string" use="optional"/>
<xs:attribute name="disabled" type="xs:boolean" use="optional"
default="false"/>
<!-- UI folks please delete this after the bug is fixed -->
</xs:complexType>
<!-- RUN AS recipient or RUN as specific user? -->
<xs:complexType name="dataVisibilityType">
    <xs:attribute name="type" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="recipient"/>
                <xs:enumeration value="runAsUser"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="runAs" type="xs:string" use="optional"/>
    <xs:attribute name="runAsGuid" type="xs:string" use="optional"/>
</xs:complexType>
<!-- Defines what needs to be done when the condition is satisfied versus
when the condition is not satisfied-->
<xs:complexType name="chooseType">
    <xs:sequence>
        <xs:element name="when">
            <xs:complexType>
                <!-- Need 'all' not 'sequence' since order may vary on
10g upgrade -->
                <xs:all>
                    <xs:element name="deliveryContent"
type="saw:deliveryContentType"/>
                    <xs:element name="postActions"
type="saw:postActionsType"/>
                </xs:all>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

```

```

        <xs:attribute name="condition" type="xs:boolean"
use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="otherwise">
    <xs:complexType>
        <!-- Need 'all' not 'sequence' since order may vary on
10g upgrade -->
        <xs:all minOccurs="0">
            <xs:element name="deliveryContent" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="message" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element
ref="saw:caption"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="postActions"
type="saw:postActionsType" minOccurs="0"/>
        </xs:all>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<!-- Defines what data is included in the agent delivery. We need to make
this schema better ... ex if disposition attribute is "attachment" then
attachmentMessage is allowed. If
Disposition is inline then message is allowed. Right now the schema seems
to indicate that any one is allowed -->
<xs:complexType name="deliveryContentType">
    <xs:all>
        <xs:element name="headline" minOccurs="0">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="saw:caption"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element ref="saw:reportRef" minOccurs="0"/>
        <xs:element name="dashboardPageRef" minOccurs="0">
            <xs:complexType>
                <xs:attribute name="dashboard" type="xs:string"
use="required"/>
                <xs:attribute name="page" type="xs:string"
use="required"/>
                <xs:attribute name="entireDashboard" type="xs:string"/>
            </xs:complexType>
        </xs:element>
    </xs:all>
</xs:complexType>

```

```

        <!-- Use the underlying conditional request as the content.
             Can have an optional narrative format, and/or
filterOverrides -->
        <xs:complexType>
            <xs:sequence minOccurs="0">
                <xs:element name="narrative" type="saw:narrativeType"
minOccurs="0"/>
                <xs:element ref="saw:filterOverrides" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="briefingBook" minOccurs="0">
        <xs:complexType>
            <xs:attribute name="path" type="xs:string"
use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="disconnectedAppCube" minOccurs="0">
        <xs:complexType>
            <xs:attribute name="name" type="xs:string"
use="required"/>
            <xs:attribute name="dataset" type="xs:string"
use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="attachmentMessage" minOccurs="0">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="saw:caption" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="message" minOccurs="0">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="saw:caption" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:all>
<xs:attribute name="format" use="optional">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="pdf"/>
            <xs:enumeration value="excel"/>
            <xs:enumeration value="excel2007"/>
            <xs:enumeration value="csv"/>
            <xs:enumeration value="html"/>
            <xs:enumeration value="text"/>
            <xs:enumeration value="powerpoint"/>
            <xs:enumeration value="powerpoint2007"/>
            <xs:enumeration value="comma"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="disposition" use="optional">

```

```

        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="deviceDefault"/>
                <xs:enumeration value="attachment"/>
                <xs:enumeration value="inline"/>
                <xs:enumeration value="link"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <!-- Need 'all' not 'sequence' since order varies. Should have the
report, conditional report, briefing book, and page ref in their own element
(eg dataContent)
        They could then be a choice. We cannot have a choice embedded
in an all.
    -->
</xs:complexType>
<!-- Post actions include java actions etc, as well as ibot chaining -->
<xs:complexType name="postActionsType">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
            <xs:element name="action" type="saw:actionType" />
            <xs:element name="reference">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="assignments"
type="saw:assignmentsType" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attribute name="path" type="xs:string" use="required"/>
                    <xs:attribute name="executePerRow" type="xs:boolean"
use="optional"/>
                </xs:complexType>
            </xs:element>
        </xs:choice>
    </xs:sequence>
</xs:complexType>
<!-- Defines the delivery narrative content -->
<xs:complexType name="narrativeType">
    <xs:sequence>
        <xs:element name="caption">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="text"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<!-- Defines the delivery destination for the iBot-->
<xs:complexType name="deliveryDestinationsType">
    <xs:sequence>
        <xs:element name="destination" minOccurs="0" maxOccurs="8">
            <xs:complexType>
                <xs:attribute name="category" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="pcemail"/>

```

```

        <xs:enumeration value="dashboard"/>
        <xs:enumeration value="pager"/>
        <xs:enumeration value="mobilePhone"/>
        <xs:enumeration
value="activeDeliveryProfile"/>
        <xs:enumeration value="sasCache"/>
        <xs:enumeration value="dacCache"/>
        <xs:enumeration value="pda"/>
        <xs:enumeration value="serverFile"/>
    </xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
<!-- uniqueness defined by destinationCategory unique key
constraint.
    0 destinations are allowed since a user may only want the
actions (eg chaining)
-->
</xs:sequence>
</xs:complexType>
<!-- Specifies the recipients/subscribers of the ibot -->
<xs:complexType name="recipientsType">
    <xs:all>
        <xs:element name="specificRecipients" minOccurs="0">
            <xs:complexType>
                <xs:sequence minOccurs="0" maxOccurs="unbounded">
                    <xs:choice>
                        <xs:element name="user" type="saw:recipientType"/>
                        <xs:element name="group"
type="saw:recipientType"/>
                    </xs:choice>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="subscribers" minOccurs="0">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="user" minOccurs="0"
maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:attribute name="name" use="required"/>
                            <xs:attribute name="guid" use="required"/>
                            <xs:attribute name="subscriptionid"
use="optional"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="dynamicRecipients" minOccurs="0">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="recipientColumn" maxOccurs="2">
                        <xs:complexType>
                            <xs:attribute name="columnID"

```

```

type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:sequence>
<xs:attribute name="relevantRows" type="xs:boolean"
use="required"/>
    </xs:complexType>
</xs:element>
</xs:all>
<xs:attribute name="specificRecipients" type="xs:boolean"/>
<xs:attribute name="dynamicRecipients" type="xs:boolean"/>
<xs:attribute name="subscribers" type="xs:boolean"/>
<xs:attribute name="customize" type="xs:boolean"/>
<!-- use all not sequence since can only have 0 or 1 of each, and in
some old ibots
        order of the elements varies. -->
<!-- For consistency, all are optional -->
</xs:complexType>
<!-- Specifies the direct email recipients of the ibot. This allows an
ibot to be delivered to
        people who are not BI EE users.
-->
<xs:complexType name="emailRecipientsType">
    <xs:sequence>
        <xs:element name="emailRecipient" minOccurs="0"
maxOccurs="unbounded">
            <xs:complexType>
                <xs:attribute name="address" use="required"/>
                <xs:attribute name="type" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:schema>

```

condition.xsd

This topic lists the content of the condition.xsd schema file.

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="com.oracle.bi/conditions/v1"
xmlns:cond="com.oracle.bi/conditions/v1"
xmlns:sawkpi="com.siebel.analytics.web/kpi/v1"
xmlns:sawx="com.siebel.analytics.web/expression/v1.1"
xmlns:saw="com.siebel.analytics.web/report/v1.1" xmlns:xs="http://www.w3.org/
2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:import namespace="com.siebel.analytics.web/report/v1.1"
schemaLocation="analysis.xsd"/>
    <xs:import namespace="com.siebel.analytics.web/expression/v1.1"
schemaLocation="expressions.xsd"/>
    <xs:import namespace="com.siebel.analytics.web/kpi/v1"
schemaLocation="kpi.xsd"/>
    <xs:element name="condition">
        <xs:complexType>

```

```

    <xs:complexContent>
      <xs:extension base="saw:appUpgradeTargetType">
        <xs:sequence minOccurs="0">
          <xs:choice>
            <xs:element name="comparison"
type="cond:comparisonType"/>
            <xs:element name="kpiComparison"
type="cond:kpiComparisonType"/>
            <xs:element name="conditionRef"
type="cond:conditionRefType"/>
          </xs:choice>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="autoGenerateName" type="xs:boolean"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:complexType name="comparisonType">
  <xs:sequence minOccurs="0">
    <xs:element name="rowcount" type="cond:rowcountType"/>
    <xs:element ref="saw:expr" maxOccurs="2"/>
  </xs:sequence>
  <xs:attribute name="op" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="equal"/>
        <xs:enumeration value="notEqual"/>
        <xs:enumeration value="less"/>
        <xs:enumeration value="lessOrEqual"/>
        <xs:enumeration value="greater"/>
        <xs:enumeration value="greaterOrEqual"/>
        <xs:enumeration value="between"/>
        <xs:enumeration value="notBetween"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
<xs:complexType name="kpiComparisonType">
  <xs:sequence>
    <xs:element name="kpi" type="cond:kpiType"/>
    <xs:element name="kpiRange" type="cond:kpiRangeType"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="rowcountType">
  <xs:sequence>
    <xs:element ref="saw:reportRef"/>
  </xs:sequence>
  <xs:attribute name="op" type="xs:string" fixed="ROWCOUNT"/>
</xs:complexType>
<xs:complexType name="kpiType">
  <xs:sequence>
    <xs:element name="kpiRef" type="cond:kpiRefType"/>
  </xs:sequence>
  <xs:attribute name="xmlVersion" use="required"
type="xs:string" />

```

```

</xs:complexType>
<xs:complexType name="kpiRefType">
  <xs:sequence>
    <xs:element name="dimensions" type="sawkpi:dimensions"/>
  </xs:sequence>
  <xs:attribute name="path" type="xs:string" use="required"/>
  <xs:attribute name="subjectArea" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="kpiRangeType">
  <xs:attribute name="assessmentStateKey" type="xs:string"
use="required"/>
</xs:complexType>
<xs:complexType name="conditionRefType">
  <xs:sequence minOccurs="0">
    <!-- define by ref so that filterOverrides element is in saw
namespace. If we
        defined locally with name="filerOverrides"
type="saw:filterOverridesType" the namespace would
        incorrectly be cond. -->
    <xs:element ref="saw:filterOverrides" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="path" type="xs:string"/>
</xs:complexType>
</xs:schema>

```