PUBLIC

SAP BusinessObjects Business Intelligence platform

Document Version: 4.3 Support Package 4 – 2023-12-07

BI Semantic Layer Java SDK Developer Guide

© 2024 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

Content

1 Document History. 5

2 Introducing the BI Semantic Layer Java SDK Developer Guide. 6

2.1 Audience. 6

2.2 Conventions in This Guide. 6

3 About the Semantic Layer. 8

3.1 About Universes. 8

3.2 About Connections. 9

3.3 About Security Profiles. 9

4 About the BI Semantic Layer Java SDK. 10

4.1 Use Cases. 10

4.2 User Requirements. 11

4.3 Environment Requirements. 11

4.4 Application Purpose. 11

Creating Resources. 12

Publishing Resources. 12

Retrieving Resources. 12

Working with Data Foundations. 13

Working with Business Layers. 16

Working with Security Profiles. 19

Working with Connections. 20

Converting Universes. 20

4.5 SDK Object Models. 20

Resources. 21

Data Foundations. 21

Business Layers. 22

Business Layer Items. 22

Display Formats. 22

Lists of Values. 23

Parameters and Answers. 24

Properties. 24

Security Profiles. 25

Connections. 26

4.6 SDK Packages. 26

4.7 SDK Services. 27

4.8 SDK Methods to Work with Universes 28

5 Installing and Deploying the BI Semantic Layer Java SDK. 31
5.1 To Install the BI Semantic Layer Java SDK with Client Tools. 31
5.2 Installed Content. 32
5.3 To Deploy the BI Semantic Layer Java SDK in Standalone on Windows. 33
5.4 To Deploy the BI Semantic Layer Java SDK in Standalone on UNIX or Linux. 33
5.5 To Deploy the BI Semantic Layer Java SDK in a Non-OSGI Eclipse Configuration. 34
5.6 To Deploy the BI Semantic Layer Java SDK in an Eclipse OSGI Framework Configuration. 35

6 Using the BI Semantic Layer Java SDK 38
6.1 BI Semantic Layer Java SDK Event Workflow. 38
Creating Sessions. 39
Loading Services. 39
Using Factories. 40
Instantiating Resources. 42
Saving Objects. 42
Checking Objects. 43
Releasing Objects. 45
Raising Errors. 45
6.2 Using the java.util.List Methods. 46
Deleting a Table. 47
Deleting a Business Layer Item. 47
Moving a Business Layer Item. 48
Deleting a Join. 48
Adding Extra Tables. 49
Adding a Join to a Context. 50
Removing a Column from a Primary Key. 50
Adding a Value to a Row of a Static List of Values. 50
Adding a Dimension to a Custom Navigation Path. 51
6.3 Working with Default Values. 51
6.4 Working with Data Display Formats. 51
6.5 Working with Data Foundation Views. 53
6.6 Working with Linked Universes. 55
6.7 Working with Query Script Properties. 58
6.8 How Business Filter Expressions and Business Queries are Saved. 60
6.9 Working with the Tables Setting of a Data Security Profile. 65
6.10 Working with the Secured Elements of a Business Security Profile. 65
6.11 Refreshing the Structure of a Data Foundation. 67
6.12 Running Check Integrity. 67
6.13 Working with Multisource-Enabled Universes. 68
Managing the Connection Life Cycle. 69
Creating a Multisource-Enabled Data Foundation. 70
Removing Data Sources from a Multisource-Enabled Data Foundation. 70

Creating Derived Tables for a Multisource-Enabled Data Foundation. 71

7 Developing with the BI Semantic Layer Java SDK Samples. 73

7.1 Sample Packages. 73

7.2 To Develop with the Help of Samples. 74

8 Troubleshooting the BI Semantic Layer Java SDK. 76

8.1 CreateProcess Error. 76

8.2 csEx Unsupported Operation. 76

8.3 Driver is Unknown. 77

8.4 Invalid Database Table. 77

9 Appendix. 78

9.1 Connection Parameter Reference. 78

About User Rights. 78

About Connection Parameters. 79

RDBMS Connections. 82

SAP Connections. 91

OLAP Connections. 95

9.2 Predefined Query Script Property Reference. 96

9.3 Identifiable Object Reference. 98

9.4 ValueFormat String Pattern Reference. 99

Numeric Patterns. 99

Date-Time Patterns. 100

1 Document History

The following table provides an overview of the most important document changes.

Version	Date	Changes
SAP BusinessObjects Business Intelligence platform 4.3	June, 2020	Initial release

2 Introducing the BI Semantic Layer Java SDK Developer Guide

The guide relates to the SAP BusinessObjects Business Intelligence platform 4.2 Support Package 4 release.

The *SAP BusinessObjects BI Semantic Layer Java SDK Developer Guide* provides functional and technical information on the SAP BusinessObjects BI Semantic Layer Java SDK:

- Purpose and functionality
- How to install and deploy the SDK
- API usage with code examples

For information on the interfaces, classes and methods of the BI Semantic Layer Java APIs, see the *SAP BusinessObjects BI Semantic Layer Java API reference* at the SAP Help Portal.

For information on the underlying object model of the API, download the *SAP BusinessObjects BI Semantic Layer Java SDK Object Model Diagrams* from the SAP Help Portal.

Audience [page 6]

Conventions in This Guide [page 6]

2.1 Audience

The *SAP BusinessObjects BI Semantic Layer Java SDK Developer Guide* is intended for Java developers.

Java developers are responsible for writing applications that perform creation, editing, publication, and security tasks on universes and their resources. These applications can be developed for internal use or embedded within Business Intelligence solutions that integrate the BI platform.

2.2 Conventions in This Guide

In this guide, the variable `<slsdk-install-dir>` is the install root path of the BI Semantic Layer Java SDK. On Microsoft Windows, the default `<slsdk-install-dir>` stands for the `C:\Program Files (x86)\SAP Business Objects\SAP BusinessObjects Enterprise XI 4.0\SL SDK` directory.

The variable `<bip-install-dir>` is the installation root path of the BI platform or Client Tools. On MS Windows (64-bit), it stands for the `C:\Program Files (x86)\SAP Business Objects` directory.

The connection and universe root folders of the Central Management Server (CMS) refer to the following directories throughout the guide:

- `/Connections`

- `/Universes`

This refers to the `CONNECTIONS_ROOT` and `UNIVERSES_ROOT` constant values of the API. See the *SAP BusinessObjects BI Semantic Layer Java API reference*.

3 About the Semantic Layer

This section presents some definitions of the Semantic Layer resources that the BI Semantic Layer Java SDK is using.

About Universes [page 8]

About Connections [page 9]

About Security Profiles [page 9]

3.1 About Universes

A universe is an organized collection of metadata objects that enable business users to analyze and report on corporate data in a non-technical language.

The role of the universe is to provide the business user with semantically understandable business objects. The user can analyze data and create reports using the relevant business language, regardless of the underlying data sources and structures.

A relational universe allows users to analyze data from one or several relational databases. It is made of a business layer, a data foundation, and one or more connections to relational databases. A multidimensional universe allows users to analyze data from an OLAP cube and is made of a business layer and its connection to an OLAP cube.

The business layer, data foundation and connection can be created in the information design tool. Once created, they are merged and published as a universe that can be used by SAP BusinessObjects reporting tools.

A universe can be published locally on a file system or in a CMS repository so it can be shared and can benefit from the CMS repository security.

A universe can support aggregate awareness. This is the ability of a relational universe to take advantage of database tables that contain pre-aggregated data (aggregate tables). Setting up aggregate awareness accelerates queries by processing fewer facts and aggregating fewer rows.

A single-source, relational UNX universe published in a CMS repository can serve as a core universe for one or more universes, which are called linked universes. When the end-user creates a linked universe in the information design tool, tables, joins, lists of values, and prompts from the data foundation of the core universe are visible but read-only in the linked data foundation. Folders, objects, lists of values, and prompts from the business layer of the core universe are visible but read-only in the linked business layer. Only the status can be changed.

For more information about universes, business layers, and data foundations, see the *Information Design Tool User Guide*.

3.2 About Connections

A connection is a set of parameters that define how one or more SAP BusinessObjects applications can access relational or OLAP databases. The connection can be a local file, or a remote object in a CMS repository that is referenced by a local shortcut in the information design tool.

Connections can be either local or secured. Local connections are saved as independent objects on the local file system as `.cnx` files. Business users can create a secured connection by publishing a local connection to a CMS repository, or by creating the connection directly in the CMS repository.

When users publish a connection to the CMS repository, either as part of a universe definition, or as a separate resource, it is stored as a secured connection. Users must create a connection shortcut locally to reference the secured connection.

For more information about local and secured connections, see the *Information Design Tool User Guide*.

3.3 About Security Profiles

A security profile is a group of security settings that apply to a universe published in the repository. The settings control the data that is displayed and modify the parameters defined in the data foundation or business layer or both.

There are two types of profiles:

- Data security profiles have security settings defined on the data foundation and on connections.
- Business security profiles have security settings defined on the business layer.

> **Note**
>
> Since 4.2 SP04, you can now download universe security business and data profiles and edit them locally in one operation rather than getting and editing each profile individually. For more information, see Editing Business and Data Security Profiles [page 25].

For more information about security, see the *Information Design Tool User Guide*.

4 About the BI Semantic Layer Java SDK

The BI Semantic Layer Java SDK provides you with an API that you use as a framework for developing some functionality of the information design tool.

The API allows you to work with universes and secured connections to one or more data sources, without using the information design tool. For example, you can implement services for creating a local `.unx` universe from a local business layer and publishing it to a repository, or retrieving a secured connection from the repository to the user workspace.

The SDK provides all the resources and information that you need to use the interfaces, classes, and methods of the API, enabling you to implement the functionality that your business requires.

The SDK provides Java samples with their source code. They allow you to facilitate and bootstrap the development of your application in the IDE.

The SDK cannot be used with any CMS repository delivered before SAP BusinessObjects Business Intelligence platform 4.0 Feature Pack 3.

Use Cases [page 10]

User Requirements [page 11]

Environment Requirements [page 11]

Application Purpose [page 11]

SDK Object Models [page 20]

SDK Packages [page 26]

SDK Services [page 27]

SDK Methods to Work with Universes [page 28]

4.1 Use Cases

You can address the following requirements by using the BI Semantic Layer Java SDK:

- Customizing universe resources programmatically
  For example, your application can change the owner and qualifier of the data foundation tables to adapt universe resources to the technical requirements of your customer database, and publish the resources to the CMS.
- Customizing business objects programmatically
  For example, your application can change the names of some business items contained in the universe and publish the universe resources to the CMS. It can also create a data or business security profile, assign it to a user group.
- Creating universe resources programmatically and publishing them
- Performing bulk conversions of `.unv` universes to `.unx` universes

4.2 User Requirements

To develop an application, you must have the following knowledge and experience:

- Developing and testing Java applications using Eclipse preferably (OSGI mode)
- Understanding the BusinessObjects Enterprise (BOE) Java SDK
- Knowing the information design tool
- Understanding metadata design concepts (data foundation, business layer, connections, and universe)
- Knowing the universe authoring and publishing lifecycle
- Understanding the concept of local and secured connections
- Understanding the concept of data and business security profiles

4.3 Environment Requirements

To develop and build your application, you need the following environment:

- Computer hardware capable of running a Java development environment
- Eclipse 3.8.2 or higher
- 64-bit Java Runtime Environment 1.6 or higher

The BI Semantic Layer Java SDK is supported on Microsoft Windows and UNIX flavors in this release.

Related Information

Installed Content [page 32]

4.4 Application Purpose

The BI Semantic Layer Java SDK helps you to write an application that can offer some of the functionality described in the next sections. The SDK provides only the functionality listed in these sections.

Creating Resources [page 12]

Publishing Resources [page 12]

Retrieving Resources [page 12]

Working with Data Foundations [page 13]

Working with Business Layers [page 16]

Working with Security Profiles [page 19]

Working with Connections [page 20]

Converting Universes [page 20]

4.4.1 Creating Resources

- Creating a relational or OLAP connection locally
- Creating a connection shortcut locally from a published connection
- Creating a single-source data foundation locally
- Creating a multisource-enabled data foundation locally
- Creating a business layer locally
- Creating a linked data foundation or business layer locally

> **Note**
>
> The SDK does not allow you to create a linked universe in the following cases:
>
> - The linked data foundation is multisource-enabled.
> - The linked business layer is multidimensional.
> - The core universe is a linked universe.
> - The core universe is a UNV universe.

Related Information

Working with Multisource-Enabled Universes [page 68]

4.4.2 Publishing Resources

- Publishing a single-source universe locally on the file system
- Publishing a single-source or multisource-enabled universe to the CMS repository
- Publishing a local connection (`.cnx`) to the CMS repository
- Publishing a linked universe to the CMS repository

> **Note**
>
> The SDK does not allow you to publish a linked universe locally.

4.4.3 Retrieving Resources

- Retrieving the local business layer, data foundation and connection from a local single-source universe

- Retrieving a published universe to edit its associated resources
- Creating a connection shortcut to a secured connection published in a CMS repository
- Retrieving the revision number of a `.unx` universe stored in a CMS repository
- Retrieving the path of a `.unx` or `.unv` universe or secured connection from its CUID in a CMS repository
- Retrieving the CUID of a `.unx` or `.unv` universe or secured connection from its path in a CMS repository

4.4.4 Working with Data Foundations

Contexts

- Retrieving the lists of included and excluded joins of a context
- Editing the list of joins of a context (adding or removing joins)

Custom Properties

- Creating custom properties for a data foundation
- Retrieving and editing the list of custom properties of a data foundation (adding or removing custom properties)

Data Foundations

- Editing the name, description, and path of a data foundation
- Setting the SQL options of a data foundation
- Creating tables, columns, joins, contexts, and primary keys in a data foundation
- Creating calculated columns
- Creating derived tables
- Creating static and SQL-based lists of values and attaching them to a data foundation
- Retrieving and editing the list of lists of values of a data foundation (adding or removing lists of values)
- Creating parameters and attaching them to a data foundation
- Retrieving and editing the list of parameters of a data foundation (adding or removing parameters)
- Refreshing the structure of the data foundation
- Running check integrity on a data foundation
- Setting the connection of a data foundation

Data Foundation Views

- Creating data foundation views
- Retrieving the list of views of a data foundation
- Retrieving the list of tables of a data foundation view
- Setting and getting the description of a data foundation view
- Setting and getting the position, width, and display state of a table in a view
- Inserting comments into a data foundation view and editing their text and display (font, colors, and opacity)

Tables and Columns

- Retrieving the list of tables of a data foundation
- Editing the list of tables of a data foundation (adding or removing tables)
- Retrieving the fully qualified name of a table from its name, owner, and qualifier
- Retrieving the name, owner, and qualifier of a table from its fully qualified name
- Editing the name, owner, qualifier, and list of columns of a table
- Setting and retrieving the SQL expression of a calculated column
- Setting column properties (nullable, signed and visible)
- Getting core universes and object identifiers in these core universes (if any)

Joins

- Retrieving the list of joins of a data foundation
- Editing the list of joins of a data foundation (adding or removing joins)
- Getting core universes and object identifiers in these core universes (if any)

Keys

- Retrieving the primary key of a table
- Editing the primary key of a table (adding or removing a column)
- Deleting the primary key of a table

Lists of Values

- Editing the name, description of a list of values

- Setting the options of a SQL-based list of values (query options)
- Creating and editing columns and rows for a static list of values
- Detecting columns of a SQL-based list of values
- Setting the properties of the columns of a static or SQL-based list of values (column name, object name, key column, data type, hidden)
- Getting core universes and object identifiers in these core universes (if any)

Multisource-enabled Data Foundations

- Retrieving the list of data sources of a multisource-enabled data foundation
- Editing the list of data sources of a multisource-enabled data foundation (adding or removing sources)

Parameters

- Editing the name, description of a parameter
- Setting the parameter options (prompt options, list of values, and default values)
- Creating and editing custom properties for a parameter
- Creating and editing answers and values for a parameter
- Getting core universes and object identifiers in these core universes (if any)

Query Script Properties

- Adding predefined or custom query script properties to a data foundation
- Editing the list of query script properties of a data foundation (adding or removing query script properties)
- Setting the values of the predefined or custom query script properties of a data foundation
- Ressetting the list of query script properties of a data foundation
- Retrieving the list of predefined and custom query script properties of a data foundation

Related Information

Using the java.util.List Methods [page 46]
Refreshing the Structure of a Data Foundation [page 67]

4.4.5 Working with Business Layers

Aggregate Awareness Incompatibilities

- Creating aggregate awareness incompatibilities
- Retrieving the list of aggregate awareness incompatibilities

Business Layers

- Editing the name, description, and path of a business layer
- Setting the query options of a business layer
- Editing comments in a business layer (adding or removing)
- Creating folders and business layer items (measures, dimensions, attributes, and filters)
- Creating business filters
- Getting, editing, and validating a business filter expression
- Creating lists of values (static, SQL-based, hierarchical, and based on a query built from business objects) and attaching to them a business layer
- Retrieving and editing the list of lists of values of a business layer (adding or removing lists of values)
- Creating parameters and attaching them to a business layer
- Retrieving and editing the list of parameters of a business layer (adding or removing parameters)
- Moving folders and business layer items
- Browsing business layer folders and reading business layer items contained within
- Deleting folders and business layer items
- Running check integrity on a business layer

> ⓘ Note
>
> - The SDK does not support multidimensional business layer items.
> - The SDK does not allow you to find aggregate awareness incompatibilities automatically.

Business Layer Items

- Retrieving the business layer item ID
- Retrieving and editing the name, description, state, and projection function of a business layer item
- Retrieving a business layer item from its full path
- Retrieving the full path of a business layer item
- Retrieving and editing the SQL definition of a business layer item (SELECT, WHERE, and extra tables)
- Retrieving and editing the advanced properties of a business layer item (access level, object used in, lists of values)

- Retrieving and editing the source information of a business object (technical information, mapping, and lineage)
- Retrieving implicit tables of business layer items
- Retrieving the list of incompatible objects for aggregation awareness
- Creating custom properties for a business layer item
- Retrieving and editing custom properties of a business layer item (adding or removing custom properties)
- Creating predefined and custom display formats for numeric and date-time business objects of a business layer
- Retrieving and editing the predefined and custom display formats
- Getting core universes and object identifiers in these core universes (if any)

Business Layer Views

- Creating and editing business layer views
- Defining the list of objects contained in the business layer views
- Hiding the master view

Custom Properties

- Creating custom properties for a business layer
- Retrieving and editing the list of custom properties of a business layer (adding or removing custom properties)

Linked Universes

- Creating a linked universe
- Editing the list of core universes linked to a business layer (adding or removing)
- Retrieving the list of core universes linked to a business layer
- Including the core universe components into a local, linked universe
- Synchronizing core universe components of a linked universe with their latest versions in the CMS repository
- Updating the paths and names of the core universes used in a linked universe

Lists of values

- Editing the name, description of a list of values
- Setting the options of a list of values

- Creating and editing columns and rows for a static list of values
- Detecting columns of a SQL-based list of values
- Setting the properties of the columns of a static or SQL-based list of values (column name, object name, key column, data type, hidden)
- Detecting columns of a list of values based on a business query
- Getting, editing, and validating the business query of a list of values
- Associating a list of values with a business object (dimension, measure or attribute)
- Associating a list of values with a parameter

> **⚠ Restriction**
>
> Only the lists of values created in the business layer can be associated with a business object.

Navigation Paths

- Creating custom navigation paths for a business layer
- Retrieving and editing custom navigation paths (adding or removing dimensions)
- Setting and getting custom navigation paths to use for a business layer

Parameters

- Editing the name, description of a parameter
- Setting the parameter options (prompt options, list of values, and default values)
- Creating and editing custom properties for a parameter
- Retrieving and editing the custom properties of a parameter (adding or removing custom properties)
- Creating and editing answers and values for a parameter
- Associating a business object (dimension, attribute) with a parameter

Query Script Properties

- Adding predefined or custom query script properties to a business layer
- Editing the list of query script properties of a business layer (adding or removing query script properties)
- Setting the values of the predefined or custom query script properties of a business layer
- Ressetting the list of query script properties of a business layer
- Retrieving the list of predefined and custom query script properties of a business layer

Related Information

Using the java.util.List Methods [page 46]

4.4.6 Working with Security Profiles

Data Security Profiles

- Creating a data security profile with a "Rows" setting and attaching it to a universe
- Creating a data security profile with a "Tables" setting and attaching it to a universe
- Creating a data security profile with a "Connections" setting and attaching it to a universe
- Getting the data security profile(s) attached to a universe
- Detaching a data security profile from a universe
- Assigning or unassigning a data security profile to a user or a group

> **Note**
>
> Since 4.2 SP04, you can get the (data and business) security profiles in one operation and edit all the settings locally before publishing them. This avoids retrieving the settings one by one.

Business Security Profiles

- Creating a business security profile with a "Create Query" setting and attaching it to a universe
- Create a business security profile with a "Display Data" setting and attaching it to a universe
- Getting the business secuity profile(s) attached to a universe
- Detaching a business security profile from a universe
- Assigning or unassigning a business security profile to a user or a group

> **Note**
>
> Since 4.2 SP04, you can get the (data and business) security profiles in one operation and edit all the settings locally before publishing them. This avoids retrieving the settings one by one.

Related Information

Working with the Secured Elements of a Business Security Profile [page 65]
Working with the Tables Setting of a Data Security Profile [page 65]
Editing Business and Data Security Profiles [page 25]

4.4.7 Working with Connections

- Testing a local connection
- Changing connection drivers
- Updating some parameters of a secured connection stored in the CMS repository
- Replacing a secured connection attached to a single-source universe with another connection in the CMS repository
- Replacing several secured connections attached to a multisource-enabled universe with others in the CMS repository
- Running check integrity on a connection

> ⓘ Note
>
> - The SDK supports the same connection types as the information design tool.
> - The SDK does not allow you to replace the connection attached to a linked universe with another connection in the CMS repository.

> → Remember
>
> It is not possible to create any `.cnx` connection in the CMS. A connection must be created and saved locally, and then published to the CMS.

4.4.8 Converting Universes

- Converting a local `.unv` universe into a `.unx` universe
- Converting a secured `.unv` universe into a `.unx` universe in the CMS repository

4.5 SDK Object Models

A series of models defines the objects that are used by the SDK. The following sections give some details about these objects. For more information, see the *SAP BusinessObjects BI Semantic Layer Java SDK Object Model Diagrams* under Development on the BI platform page of the SAP Help Portal.

Resources [page 21]

Data Foundations [page 21]

Business Layers [page 22]

Business Layer Items [page 22]

Display Formats [page 22]

Lists of Values [page 23]

Parameters and Answers [page 24]

Properties [page 24]

Security Profiles [page 25]

Connections [page 26]

4.5.1 Resources

Resources define the objects that a universe is made of. The objects are `DataFoundation`, `BusinessLayer` and `Connection`. The root object of resources is `SlResource`.

4.5.2 Data Foundations

`DataFoundation` objects represent data foundations in the SDK object model. `MonoSourceDataFoundation` objects specify data foundations based on a single data source (one connection), while `MultiSourceDataFoundation` objects specify data foundations based on multiple data sources (several connections).

`Table` objects define data foundation tables and `Column` objects define table columns. A table can be a database table, a derived table, an alias table, or a federated table. `DatabaseTable` objects define data source tables and provide database information (owner and qualifier). `DerivedTable` objects define virtual tables in a data foundation. `AliasTable` objects define references to standard or derived tables in a data foundation.

`CalculatedColumn` and `InputColumn` objects define calculated columns and input columns explicitly. These types of columns are used for specific connections, such as connections to SAP systems.

`Context` objects define data foundation contexts and `SQLJoin` objects define joins between data foundation tables.

`PrimaryKey` objects define `DatabaseTable` primary keys.

`DataFoundationView` objects represent the master view and the custom data foundation views in the SDK object model. A `TableView` object represents the particular view of a data foundation table in a view. A view can reference any table, including a federated table.

`FederatedTable` objects represent federated tables created in the Data Foundation Editor of the information design tool.

Related Information

Working with Data Foundation Views [page 53]

4.5.3 Business Layers

`BusinessLayer` objects represent business layers in the SDK object model. The `RootFolder` object is the root container of the `BusinessLayer` objects. `RelationalBusinessLayer` objects, which inherit from `BusinessLayer`, represent business layers for relational data sources.

`BusinessLayerView` objects represent the business layer views of a business layer in the SDK object model. The business layer views contain a set of business objects of a business layer.

`AggregateIncompatibility` objects represent relationships of incompatibility between business layer items and aggregate tables of the database. You create these objects to define the aggregate awareness in a business layer.

`NavigationPath` objects represent custom navigation paths in a business layer. A navigation path is a list of drillable business objects that allows a user of a reporting tool such as SAP BusinessObjects Web Intelligence to drill down on a business object of type dimension. Navigation paths can contain only visible dimensions. You create `NavigationPath` objects to define custom navigation paths in a business layer. Default navigation paths are defined by the hierarchical organization of the business objects in the business layer. They cannot be created, edited, nor deleted. To retrieve the default navigation paths, you need to drill through the object hierarchy.

4.5.4 Business Layer Items

A business layer is a collection of `BlItem` objects. A `BlItem` represents any object of the business layer (dimension, measure, attribute, folder, business and native filters).

A `BusinessObject` represents any metadata object of the business layer (dimension, measure, and attribute). These objects have a role in queries.

A `BlContainer` object is the container of some `BlItem` objects that can also be seen as containers of `BlItem` objects. Dimensions and measures have attributes. Folders contain measures and dimensions.

`CustomProperty` objects define the custom properties of business layers and business layer items (dimensions, measures, attributes, folders and filters). These properties do not depend on the source information of business layer items (Technical Information, Mapping, and Lineage).

`BusinessFilter` objects defined filters based on business objects in business layers. They benefit from the methods inherited from the `Filter` object.

`NativeRelationalFilter` objects define predefined filters of type native attached to a relational business layer. A native filter can be defined only using an SQL expression.

A `RelationalBinding` object holds the SQL definition parts (SELECT, WHERE, and extra tables) of a `BusinessObject` or a `NativeRelationalFilter`. `NativeRelationalFilter` objects do not use SELECT.

4.5.5 Display Formats

`DisplayFormat` objects represent the display format of business objects in the SDK object model. This release supports the data display format only. `DataFormat` objects represent the data display format.

> → Remember
>
> The object model does not support the other display formats (alignment, border, shading, and font).

The model provides the `DateTimeFormat` and `NumberFormat` interfaces, which inherit from `DataFormat`, to support the display format of DateTime and numeric values of business objects. Similarly, it provides the following interfaces to support custom and predefined formats for DateTime and Numeric values:

DateTimeFormat Objects	NumberFormat Objects
`CustomDateTimeFormat`	`CustomNumberFormat`
`PredefinedDateTimeFormat`	`PredefinedNumberFormat`

The predefined formats for numeric and date-time values are defined in the following enums:

- `PredefinedDateTimeFormatType` - for example `SHORT_DATE` and `LONG_TIME`
- `PredefinedNumberFormatType` (`NUMERIC`, `PERCENT`, `SCIENTIFIC`, and `BOOLEAN`)

`ValueFormat` objects represent the value of the custom format. They consist of the following parts:

- The string that defines the format, which is an empty string by default
- The color that can be a value from the `FormatColor` enum and whose default value is "Automatic", which means that no color is defined

> ⓘ Note
>
> See the *Information Design Tool User Guide* for more information about display formats.

Related Information

Working with Data Display Formats [page 51]

4.5.6 Lists of Values

A `Lov` object represents a list of values in the SDK object model. The following types of list of values are supported:

- `StaticLov`, for a list of which the values are defined explicitly
- `SQLQueryLov`, to represent lists of values based on a custom SQL expression
- `BusinessQueryLov`, to represent lists of values based on a query built from business objects
- `BusinessHierarchicalLov`, to represent lists of values based on a custom hierarchy of dimensions

A `LovColumn` object represents a column of a list of values. The model defines an object for the column of each type of list of values:

- `StaticLovColumn`
- `SQLQueryLovColumn`

- `BusinessQueryLovColumn`
- `BusinessHierarchicalLovColumn`

A `StaticLovRow` object defines a row of a static list of values. A row contains a set of values defined for all the columns of the list of values.

4.5.7 Parameters and Answers

The `Parameter` object represents a universe parameter in data foundations and business layers.

The `Answer` object represents an answer that is given to a parameter. The following types of answers are supported:

- `SingleValueAnswer`, to represent an answer with one value
- `IntervalAnswer`, to represent a set of values defined in an interval
- `MultipleValueAnswer`, to represent a set of values

The `TypedValue` object represents the value of a list of values that can be given as answer to a parameter. A value can be either flat (`FlatValue`) or hierarchical (`HierarchicalValue`). A hierarchical value is a list of flat values. The possible types of flat values are the following:

- `NumericValue` for numeric values
- `StringValue` for string values
- `DateValue` for dates

4.5.8 Properties

The `Property` interface manages custom properties and query script properties in the SDK object model. Custom properties are represented via `CustomProperty` and query script properties via `QueryScriptProperty`.

A property is defined as a {key, value} pair. Keys of predefined query script properties are specified as enums for data foundations and business layers in `DataFoundationQueryScriptPropertyKey` and `BusinessLayerQueryScriptPropertyKey`.

`DataFoundation` and `BusinessLayer` interfaces extend `QueryScriptCustomizable` to support query script properties.

> → Remember
>
> Query script properties are named query script parameters in the information design tool. See the *Information Design Tool User Guide* for more information.

Related Information

Predefined Query Script Property Reference [page 96]

4.5.9 Security Profiles

`SecurityProfile` objects represent security profiles defined in business layers.

The present release addresses both data and business security profiles. The model provides the `DataSecurityProfile` and `BusinessSecurityProfile` objects.

The scope of the data security profile settings is restricted to "Rows", "Tables", and "Connections".

The scope of the business security profile setting is restricted to "Create Query" and "Display Data".

`SecuredElements` objects represent the lists of business layer views and business layer items that are defined as granted or denied in a business security profile.

4.5.9.1 Editing Business and Data Security Profiles

Previously, you had to download security profile settings one by one in order to edit them. Now, you can retrieve all the data and business profile settings with one method and edit the settings locally before publishing the new settings. You can still use the original methods, but this new approach is a big improvement in performance.

Method	Action
`getUniverseSecurityCache`	Use this to retrieve the universe security profile settings in order to edit the settings locally.
`getBusinessSecurityProfiles`	Use this method to get the business security profiles locally.
`getDataSecurityProfiles`	Use this method to get the data security profiles locally.
`getPrincipals`	Use this method to get the principle settings locally.
`getUniversePath`	This is for information only - use his method to get the universe path for the universe security settings file that you are working on locally.
`commit`	Use this method to publish the security profile settings to the universe.
`close`	Closes the `UniverseSecurityCache` once it is no longer in use.

4.5.10 Connections

A `Connection` is the base object for representing local and secured connections (`.cnx`). `DatabaseConnection` objects represent local connections only and provide connection parameter information.

`RelationalConnection`, `OlapConnection` and `DataFederatorConnection` objects, which all inherit from `DatabaseConnection`, represent connections to relational data sources and OLAP data sources, and connections based on data federation services respectively.

A `ConnectionShortcut` represents a local `.cns` shortcut to secured connection.

4.6 SDK Packages

The BI Semantic Layer Java SDK consists of the following packages:

Package	Description
`com.sap.sl.sdk.authoring.businesslayer`	Interfaces used to work with business layers, business layer items, and business layer views
`com.sap.sl.sdk.authoring.checkintegrity`	Interface used to run check integrity
`com.sap.sl.sdk.authoring.cms`	Interfaces used to work with universe resources stored in a CMS repository
`com.sap.sl.sdk.authoring.commons`	Defines the `SlResource` object, which is the root object of all resource objects of the model. Provides interfaces that allow objects to be identified, named, customized, and inherited. Provides an interface to retrieve the SDK version number.
`com.sap.sl.sdk.authoring.connection`	Interfaces used to create and manage connections and connection shortcuts
`com.sap.sl.sdk.authoring.datafoundation`	Interfaces used to work with data foundations and contents (tables, joins, contexts, primary keys, lists of values)
`com.sap.sl.sdk.authoring.local`	An interface used to work with local resources
`com.sap.sl.sdk.authoring.security`	Interfaces to create and work with data and business security profiles
`com.sap.sl.sdk.framework`	Defines classes and interfaces for dealing with Semantic Layer SDK exceptions
`com.sap.sl.sdk.framework.cms`	An interface used to retrieve a CMS session from the BI Platform Java SDK

See the *SAP BusinessObjects BI Semantic Layer Java API reference* for more information about the package contents.

The com.sap.sl.sdk.authoring.commons Package

The name, ID, and custom properties are common to almost all SDK objects. Consequently, new classes have been introduced to:

- Help you to factorize the code and simplify the SDK implementation
- Manage these common properties in a generic way

The classes are the following:

- `Identifiable`, to get the object ID
- `Nameable`, to store, get, and set the object name
- `Customizable`, to store and get the list of custom properties
- `Inheritable`, to allow an object of a core universe to be inherited by a linked universe

The `Customizable` class allows the extension of the support of custom properties to data foundations and parameters.

The `SdkService` service has also been added to this package.

Related Information

BI Semantic Layer Java SDK Event Workflow [page 38]

4.7 SDK Services

The following table describes the services that the BI Semantic Layer Java SDK provides:

Service	Description
`BusinessLayerFactory`	Used to create business layers, business layer items, business layer views, lists of values, parameters, answers, custom and query script properties, aggregate incompatibilities, navigation paths, and data display formats
`BusinessLayerService`	Used to retrieve the implicit tables of a business layer item. Used to detect columns of SQL-based lists of values.
`CmsResourceService`	Used to manage secured universe resources

Service	Description
`CmsSecurityService`	Used to assign security profiles to resources
`CmsSessionService`	Used to retrieve a CMS session with its context
`ConnectionFactory`	Used to create connections
`ConnectionService`	Used to test connections and change connection drivers
`DataFederatorService`	Used to manage connections stored on the Data Federator Query Server. Used to work with multisource-enabled universes.
`DataFoundationFactory`	Used to create single-source and multisource-enabled data foundations, tables, columns, joins, contexts, primary keys, views, lists of values, parameters, answers, custom and query script properties
`DataFoundationService`	Used to build or deconstruct the fully qualified names of the data foundation tables. Used to refresh the structure of a data foundation. Used to detect columns of SQL-based lists of values.
`LocalResourceService`	Used to manage local universe resources
`SdkService`	Used to manage the services common to the BI Semantic Layer Java SDK.
`SecurityFactory`	Used to create data and business security profiles

4.8 SDK Methods to Work with Universes

The following sections provide details on the methods that you can use to work with universes in the BI Semantic Layer Java SDK. See the *SAP BusinessObjects BI Semantic Layer Java API reference* for more information.

Publish and Retrieve Methods

Method	Description
`CmsResourceService.publish`	Publishes or republishes a universe to the repository. The method is using the `.blx` business layer as argument and publishes the `.unx` universe, which contains the full path of the `.blx` business layer, the `.dfx` data foundation and the `.cnx` connection.
`CmsResourceService.retrieveUniverse`	Retrieves a secured universe from the CMS repository to the user workspace. The method is using the `.unx` universe as argument, and creates all of its resources locally. Resources are `.bfx`, `.dfx`, and `.cns` for relational single-source universes and multisource-enabled universes. They are `.bfx` and `.cns` for OLAP universes. An optional encryption level for the created resources is provided.
`CmsResourceService.createShortcut`	Creates a connection shortcut (`.cns`) from a secured connection stored in the CMS (`.cnx`)
`LocalResourceService.publish`	Publishes resources on the user workspace. The method is using the `.blx` business layer as argument and creates the `.unx` universe locally.
`LocalResourceService.retrieve`	Retrieves the universe resources from its local `.unx`

Load and Save Methods

Method	Description
`CmsResourceService.loadConnection`	Loads in memory a connection from its path in the CMS. You use `CmsResourceService.saveConnection(DatabaseConnection)` to update a secured connection to the CMS.
`LocalResourceService.load`	Loads in memory any kind of local resource (`.blx`, `.dfx`, `.cnx`, or `.cns`)
`LocalResourceService.save`	Saves a resource (`.blx`, `.dfx`, or `.cnx`) locally as a file

.unv to .unx Conversion Method

Method	Description
`CmsResourceService.convertUniverse`	Converts a `.unv` universe stored in the CMS into a `.unx` universe in the CMS

Method	Description
`LocalResourceService.convertUniverse`	Converts a local `.unv` universe into a local `.unx` universe. In this release, the conversion of a local universe does not require a CMS session.

5 Installing and Deploying the BI Semantic Layer Java SDK

The BI Semantic Layer Java SDK is delivered as part of the SAP BusinessObjects Business Intelligence platform 4.2 release. It is installed by default when running the SAP BusinessObjects Business Intelligence platform installation either on Windows, UNIX or Linux. You can also install it when running the SAP BusinessObjects Business Intelligence Platform Client Tools installation. See the *SAP BusinessObjects Business Intelligence platform Installation Guide for Windows* and *for Unix* for more information.

The SDK does not create any dependencies with connection drivers. You must install drivers with the platform, independently of the SDK installation.

The BI Semantic Layer Java SDK can be deployed either in an Eclipse environment or as a standalone project.

The supported configurations are the following:

- OSGI with Eclipse
- non-OSGI with Eclipse
- non-OSGI without Eclipse

Perform the following procedures to deploy the BI Semantic Layer Java SDK resources in your development project. Eclipse versions supported are 3.8.2 and higher.

These procedures apply to Eclipse 3.8.2 Classic. Procedure steps may be different if you deploy the SDK with another version of Eclipse.

To Install the BI Semantic Layer Java SDK with Client Tools [page 31]

Installed Content [page 32]

To Deploy the BI Semantic Layer Java SDK in Standalone on Windows [page 33]

To Deploy the BI Semantic Layer Java SDK in Standalone on UNIX or Linux [page 33]

To Deploy the BI Semantic Layer Java SDK in a Non-OSGI Eclipse Configuration [page 34]

To Deploy the BI Semantic Layer Java SDK in an Eclipse OSGI Framework Configuration [page 35]

5.1 To Install the BI Semantic Layer Java SDK with Client Tools

> **Note**
>
> The installation on Microsoft Windows requires that the account being used is a member of the Windows Administrators group, and that the default privileges assigned to the Administrators group have not been modified.

1. Locate the installation program of SAP BusinessObjects Business Intelligence platform Client Tools.

2. Run the `setup.exe` file.
3. When reviewing the list of features to install, in *Developer Components*, check *SAP BusinessObjects Semantic Layer Java SDK*.
4. To install the samples along with the SDK resources for your development project, leave *Samples* checked. Otherwise uncheck *Samples*.
5. Perform the remaining installation steps to install the SDK resources on your machine.

You have installed the SDK resources in the `C:\Program Files (x86)\SAP Business Objects\SAP BusinessObjects Enterprise XI 4.0\SL SDK` directory.

> **Note**
>
> You can also run a silent installation of the SDK by using the BI Client Tools Installer command line. See the *SAP BusinessObjects Business Intelligence platform Installation Guide* for more information.

After installation, you must deploy the BI Semantic Layer Java SDK in your development environment.

5.2 Installed Content

On Windows

The resources installed in the `<slsdk-install-dir>` directory consist of the following:

- `eclipse\plugins` folder, which contains all plug-ins and JAR resources needed for OSGI and non-OSGI deployments in Eclipse. This specifically embeds the BOE Java SDK necessary libraries (`com.businessobjects.boesdk`).
  The folder also contains a series of Eclipse plug-ins that you do not need to select in the Target Platform when you deploy the SDK in an Eclipse OSGI framework.
- `java\sl_sdk.jar`, which contains a manifest file defining the `CLASSPATH` for a standalone deployment.
- `javadoc`, which contains the *SAP BusinessObjects BI Semantic Layer Java API reference* as a ZIP file (`sl_sdk_javadoc.zip`).
- `SDK Samples` folder, which consists of the following archives:
  - `com.sap.sl.sdk.authoring.samples.jar`
    This contains a compiled version of the samples available for testing.
  - `com.sap.sl.sdk.authoring.samples.source.jar`
    This contains the source code of the samples.

On UNIX and Linux

The resources installed in the `<slsdk-install-dir>` directory consist of the following:

- `eclipse/plugins` folder, which contains JAR resources. This specifically embeds the BOE Java SDK necessary libraries (`com.businessobjects.boesdk`).

- `java/sl_sdk.jar`, which contains a manifest file defining the `CLASSPATH` for a standalone deployment.
- `javadoc`
- `SDK Samples` folder

5.3 To Deploy the BI Semantic Layer Java SDK in Standalone on Windows

1. Develop your application, for example `<my_application.java>`, by using the SDK resources.
2. Set the `CLASSPATH` environment variable with the help of the manifest file provided in the `sl_sdk.jar` file.

   Run for example the following command:

   ```
   set CLASSPATH=<bip-install-dir>\SAP BusinessObjects Enterprise XI 4.0\SL
   SDK\java\sl_sdk.jar;
   ```

3. Set the `PATH` environment variable to the Connection Server binaries directory:

   Run for example the following command:

   ```
   set PATH=<bip-install-dir>\SAP BusinessObjects Enterprise XI
   4.0\win64_x86;%path%
   ```

4. Compile your program by running the following command:

   ```
   javac my_application.java
   ```

5. Execute your program by running the following command:

   ```
   java -Dbusinessobjects.connectivity.directory="<bip-install-dir>\SAP
   BusinessObjects
    Enterprise XI 4.0\dataAccess\connectionServer" my_application
   ```

   The VM argument `-Dbusinessobjects.connectivity.directory` specifies the Connection Server installation directory. This is required for using connections established with Connection Server.

You have performed the task implemented in your application.

Related Information

Conventions in This Guide [page 6]

5.4 To Deploy the BI Semantic Layer Java SDK in Standalone on UNIX or Linux

You have installed and configured the required third-party middleware. Check that the environment variables such as `LD_LIBRARY_PATH` and `PATH`are properly set with third-party middleware paths.

1. Develop your application, for example `<my_application.java>`, by using the SDK resources.
2. Set the environment variables used by the SDK with the help of the `env.sh` file.

   This file is located in the `<bip-install-dir>/setup` folder. Run the following command:

   ```
   source <bip-install-dir>/setup/env.sh
   ```

3. Compile your program by running the following command:

   ```
   javac my_application.java
   ```

4. Execute your program by running the following command:

   ```
   java -Dbusinessobjects.connectivity.directory="<bip-install-dir>/
   enterprise_xi40/
   dataAccess/connectionServer" -jar my_application.jar
   ```

   The VM argument `-Dbusinessobjects.connectivity.directory` specifies the Connection Server installation directory. This is required for using connections established with Connection Server.

You have performed the task implemented in your application.

Related Information

Conventions in This Guide [page 6]

5.5 To Deploy the BI Semantic Layer Java SDK in a Non-OSGI Eclipse Configuration

This procedure allows you to create, develop, and run an SDK development project.

1. Set the `PATH` environment variable to the Connection Server binaries directory:

   Run for example the following command:

   ```
   set PATH=<bip-install-dir>\SAP BusinessObjects Enterprise XI
   4.0\win64_x86;%path%
   ```

2. Launch Eclipse by double-clicking the `eclipse.exe` file and select your workspace.
3. Create a Java project:
   a. Choose *File* > *New* > *Java Project*.
      The *New Java Project* dialog box opens.
   b. Enter the project name, select *jre6* as project JRE, and choose *Finish*.
      The project folder appears in the *Package Explorer* view.
4. Add the SDK JAR file to your project:
   a. Right-click the project and select *Build Path* > *Configure Build Path*.
      The *Properties* dialog box opens.

b. Choose the *Libraries* tab then choose *Add External JARs*.
c. Browse to the `sl_sdk.jar` file located in the `<slsdk-install-dir>\java` directory, and choose *Open*.
d. Add the JUnit JAR files if you want to run unit tests:
   1. Choose *Add Library*.
   2. Select JUnit and choose *Next*.
   3. Select the JUnit version and choose *Finish*.

5. Insert the javadoc into your development environment:
   a. Expand the `sl_sdk.jar` node in the libraries list.
   b. Select the *Javadoc location* entry, then choose *Edit*.
   c. Enter the archive path `'C:\Program Files\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\SL SDK\javadoc\sl_sdk_javadoc.zip'` and choose *OK*.
6. Choose *OK* to close the *Properties* dialog.
   The SDK resource packages appear in the *Referenced Libraries* under your project in the *Package Explorer* view.
7. Develop your own application using the SDK packages.
8. Create a Run Configuration:
   a. Choose *Run* > *Run Configurations*.
   b. Select your run configuration and choose the *Arguments* tab.
   c. Add the following line to the *VM arguments* to point to Connection Server:

```
-Dbusinessobjects.connectivity.directory="<bip-install-dir>\SAP
BusinessObjects Enterprise XI 4.0\dataAccess\connectionServer"
```

9. Choose *Run* to run your application.
10. Verify the results of your application in the information design tool.

You have performed the task implemented in your application.

Related Information

Conventions in This Guide [page 6]

5.6 To Deploy the BI Semantic Layer Java SDK in an Eclipse OSGI Framework Configuration

This procedure allows you to create, develop, and run an SDK development project.

1. Set the `PATH` environment variable to the Connection Server binaries directory:

   Run for example the following command:

```
set PATH=<bip-install-dir>\SAP BusinessObjects Enterprise XI
4.0\win64_x86;%path%
```

2. Launch Eclipse by double-clicking the `eclipse.exe` file and select your workspace.
3. Create a target platform for the SDK development:
   a. Choose *Window* > *Preferences*, expand *Plug-in Development* and select *Target Platform*.
   b. Choose *Add*.
      The *New Target Definition* window appears.
   c. Check *Default: Default target for the running platform* and choose *Next*.
   d. Enter the target name, for example **SL SDK**.
   e. Select the *Locations* tab and choose *Add*.
      The *Add Content* window appears.
   f. Select *Directory* and choose *Next*.
   g. Browse to `<slsdk-install-dir>\eclipse\plugins`, choose *OK*, then *Next* and *Finish*.
   h. Select the *Locations* tab and choose *Add*.
      The *Add Content* window appears.
   i. Select *Directory* and choose *Next*.
   j. Browse to the Eclipse distribution plug-ins to your target platform, choose *OK*, then *Next* and *Finish*.
   k. In the *New Target Definition* window, select the *Content* tab, then select *Location* in the *Group by* drop-down menu.
   l. Uncheck all the plug-ins from the Eclipse distribution to avoid conflicts with the SDK plug-ins.
   m. In the Eclipse distribution, select the `org.hamcrest.core` and `org.junit` plug-ins.
   n. Choose *Finish* and select the target platform to make it active.
4. Create a plug-in project:
   a. Choose *File* > *New* > *Project*.
      The *New Project* dialog box appears.
   b. Expand *Plug-in Development*, choose *Plug-in Project*, then choose *Next*.
   c. Enter the project name, and under *Target Platform*, select the Eclipse version 3.4, then choose *Next*.
   d. Choose Java 1.6 as the execution environment, select the appropriate options for your plug-in, and choose *Finish*.

   The project folder displays in the *Package Explorer* view.
5. Open the `manifest.mf` file, click *Dependencies*, and add the following plug-ins to the *Required Plug-ins* area:
   - `com.businessobjects.boesdk`
   - `com.sap.sl.sdk.authoring`

   > → Tip
   >
   > You can also add `org.junit` to run unit tests.
6. Develop your own application using the SDK packages.
7. Create a Run Configuration:
   a. Choose *Run* > *Run Configurations*.
   b. In the left pane, double-click *JUnit Plug-in Test* to create a run configuration for the current test class.
   c. In the *Main* tab, select *Run an application*, and select either *[No Application] - Headless Mode* or your application to run in UI mode.

      Make sure you use the JRE 1.6 for your runtime environment.

d. In the *Arguments* tab, add the following line to the *VM arguments* to point to the Connection Server:

```
-Dbusinessobjects.connectivity.directory="<bip-install-dir>\SAP
BusinessObjects Enterprise XI 4.0\dataAccess\connectionServer"
```

8. Choose *Run* to run your application.
9. Verify the results of your application in the information design tool.

You have performed the task implemented in your application.

Related Information

Conventions in This Guide [page 6]

6 Using the BI Semantic Layer Java SDK

This section presents detailed technical information and good practices with examples for making the best use of the BI Semantic Layer Java APIs.

BI Semantic Layer Java SDK Event Workflow [page 38]

Using the java.util.List Methods [page 46]

Working with Default Values [page 51]

Working with Data Display Formats [page 51]

Working with Data Foundation Views [page 53]

Working with Linked Universes [page 55]

Working with Query Script Properties [page 58]

How Business Filter Expressions and Business Queries are Saved [page 60]

Working with the Tables Setting of a Data Security Profile [page 65]

Working with the Secured Elements of a Business Security Profile [page 65]

Refreshing the Structure of a Data Foundation [page 67]

Running Check Integrity [page 67]

Working with Multisource-Enabled Universes [page 68]

6.1 BI Semantic Layer Java SDK Event Workflow

The following steps represent the event workflow that is common to all BI Semantic Layer Java SDK samples that edit an object of the Semantic Layer. This is also the basic, essential workflow that you should apply in any development project that uses the SDK.

1. An SDK context is created.
2. A session is created in the CMS repository and attached to the context.
3. The context instantiates a service.
4. The necessary resources are loaded in memory using the service.
5. Operations are performed on the objects as on a Java bean to edit it.
6. The resources are saved using the service. Most security and integrity controls are performed during this step.
7. The resources are released using the service.
8. The context is released using the service.
9. The session is logged off.

Creating Sessions [page 39]

Loading Services [page 39]

Using Factories [page 40]

Instantiating Resources [page 42]
Saving Objects [page 42]
Checking Objects [page 43]
Releasing Objects [page 45]
Raising Errors [page 45]

6.1.1 Creating Sessions

Before the BI Semantic Layer Java SDK loads a service that needs an access to the CMS, a session is created using the BI Platform Java SDK. The BI Semantic Layer Java SDK then gets the session on the CMS. The session gives access to the following:

- All service operations on resources linked to the CMS
- CMS objects depending on user connection rights

Example

1. First you create the SDK context:

```
SlContext context = SlContext.create();
```

2. You create the `IEnterpriseSession` object using the BI Platform Java SDK:

```
IEnterpriseSession enterpriseSession =
CrystalEnterprise.getSessionMgr().logon(cmsUserName, cmsPassword, cmsHost,
cmsAuthMode);
```

3. You register the session in the `CmsSessionService` and attach it to the SDK context:

```
context.getService(CmsSessionService.class).setSession(enterpriseSession);
```

4. After all service operations have been performed, you release the `SlContext` object:

```
context.close();
```

5. Log off from the session (it is not logged off automatically when the context is closed):

```
enterpriseSession.logoff();
```

6.1.2 Loading Services

The `getService()` method allows you to load services as follows:

```
<Service Class Name> service = context.getService(<Service Class Name>.class);
```

Example

You can instantiate `LocalResourceService` and retrieve `LocalResourceService.class` with the help of the session:

```
LocalResourceService service = context.getService(LocalResourceService.class);
```

Related Information

SDK Services [page 27]

6.1.3 Using Factories

Most of the services are using objects that can be created by `Factory` interfaces. The following table shows the `Factory` interfaces available in this release:

Interface	Description
`BusinessLayerFactory`	Provides the following methods: • `createRelationalBusinessLayer` • `createBlItem` • `createCustomProperty` • `createQueryScriptProperty` • `createAggregateIncompatibility` • `createNavigationPath` • `createBusinessLayerView` • `createSQLQueryLov` • `createStaticLov`, `createStaticLovColumn`, and `createStaticLovRow` • `createBusinessHierarchicalLov` and `createBusinessHierarchicalLovColumn` • `createBusinessQueryLov` • `createParameter` • `createSingleValueAnswer`, `createMultipleValueAnswer`, and `createIntervalAnswer` • `createHierarchicalValue` • `createNumericValue`, `createStringValue`, and `createDateValue` • `createPredefinedDateTimeFormat` and `createCustomDateTimeFormat` • `createPredefinedNumberFormat` and `createCustomNumberFormat`

Interface	Description
`ConnectionFactory`	Provides the following methods: • `createRelationalConnection` • `createOlapConnection`
`DataFoundationFactory`	Provides the following methods: • `createMonoSourceDataFoundation` • `createMultiSourceDataFoundation` • `createDatabaseTable` • `createDerivedTable` • `createAliasTable` • `createColumn` • `createCalculatedColumn` • `createSqlJoin` • `createContext` • `createDataFoundationView` • `createTableView` • `createDataFederatorSourceInfo` • `createPrimaryKey` • `createNativeRelationalFilter` • `createCustomProperty` • `createQueryScriptProperty` • `createSQLQueryLov` • `createStaticLov`, `createStaticLovColumn`, and `createStaticLovRow` • `createParameter` • `createSingleValueAnswer`, `createMultipleValueAnswer`, and `createIntervalAnswer` • `createHierarchicalValue` • `createNumericValue`, `createStringValue`, and `createDateValue`
`SecurityFactory`	Provides the following methods: • `createBusinessSecurityProfile` • `createDataSecurityProfile` • `createConnectionMapping` • `createRowRestriction` • `createTableMapping`

Example

You can instantiate a `SecurityFactory` object as a service and retrieve `SecurityFactory.class`:

```
final SecurityFactory securityFactory =
context.getService(SecurityFactory.class);
```

Then you can call a method of the interface and perform tasks:

```
DataSecurityProfile dataSecurityProfile =
securityFactory.createDataSecurityProfile();
```

6.1.4 Instantiating Resources

Before calling methods to perform tasks on resources, you must instantiate them. The following methods allow you to instantiate an `SlResource` object:

Method	What You Can Do
`BusinessLayerFactory.createRelationalBusinessLayer(String, String)`	Creating a `RelationalBusinessLayer` object in the memory
`DataFoundationFactory.createMonoSourceDataFoundation(String, String)`	Creating a `MonoSourceDataFoundation` object in the memory
`DataFoundationFactory.createMultiSourceDataFoundation(String)`	Creating a `MultiSourceDataFoundation` object in the memory
`LocalResourceService.load(String)`	Loading in memory a resource that has been saved locally on the user machine. The resource can be a connection (`.cnx`) a data foundation (`.dfx`) or a business layer (`.blx`). It can also be a connection shortcut, for example a `DataFederatorConnectionShortcut` object.
`CmsResourceService.loadConnection(String)`	Loading in the memory of the user machine a connection that has been stored in the CMS repository
`ConnectionFactory.createRelationalConnection(String, String, String)`	Creating a `RelationalConnection` object in the memory
`ConnectionFactory.createOlapConnection(String, String, String)`	Creating an `OlapConnection` object in the memory

6.1.5 Saving Objects

After you edit a connection (`.cnx`), data foundation (`.dfx`) or business layer (`.blx`), you must save the resource explicitly. You do this by using the `LocalResourceService.save(SlResource, String,`

`boolean)` method before the publication of the resource. This maintains the resource as a file on the user machine. There is no automatic save.

If you do not implement the resource save, the publication is performed from the resource file path locally on the machine, but not from the object in memory.

After you edit a connection, you can also use the `CmsResourceService.saveConnection(String)` method to save the connection in the repository.

> → Remember
>
> You do not need to load nor save the data foundation explicitly after a refresh of the structure. This operation is performed by the `refreshStructure(String)` method.

Related Information

Refreshing the Structure of a Data Foundation [page 67]

6.1.6 Checking Objects

In the SDK released before 4.1 SP5, some tests were performed when creating or editing a data foundation or business layer. Since 4.1 SP5 all the validity tests are now performed when the resource is saved.

The following table lists the errors that were raised when you created or edited a resource, and which are now detected at save time.

Resource	Error
Data foundation	• Multiple tables have the same qualifier, owner and table names. • An alias table refers a table that is not a database table, a derived table or does not belong to the data foundation. • The primary key refers to an empty column or a column that does not exist. • An alias table contains some columns. • A derived or database table contains columns with the same name. • Two joins have the same expression. • A table referenced by a join does not exist.
Multisource-enabled data foundation	• Invalid qualifier • A connection is not deployed. • Two connections share the same short name. • A connection has an invalid path in the CMS repository. • A connection used by a multisource-enabled data foundation does not exist in the CMS repository.

Resource	Error
Business layer	• The master view is not visible and there is no other visible view. • A view has an empty name. • Two views have the same name. • A view contains an unsupported object. • A view contains duplicated objects. • In the same folder, two objects of the same type have the same name. • A business layer item has an invalid name. • Two objects have the same identifier. • An object has an unknown type. • A dimension or a measure has an object that is not an attribute as a child. • The WHERE clause of a relational filter is empty. • A custom property has an empty key. • Two custom properties of the same object have the same name.
Static list of values	• Empty column name • Column moved from another list of values • Duplicated column • Auto-reference • Unknown column reference • More columns in a row than in the list of values
List of values based on an SQL query	• Column moved from another list of values • Auto-reference • Unknown column reference • Incorrect timeout • Incorrect max row count • Incorrect or null SQL expression
Hierarchical list of values	• A hierarchy contains the same dimension twice. • Unknown dimension • Column moved from another list of values • Incorrect timeout • Incorrect max row count
List of values based on a business query	• Column moved from another list of values
Prompt	• Two prompts have the same name. • A prompt has an empty name. • A prompt has an empty text. • The associated list of values is not supported. • No column from the associated list of values is visible.
Prompt value	• Unexpected type (flat/hierarchical) • Answer value is null. • Answer value does not have the same type expected by the prompt.

Resource	Error
Query	• Warning: invalid SQL expression in a list of values based on a SQL query • Warning: invalid SQL expression in a derived table • Warning: invalid SQL expression in a calculated column • Warning: invalid SQL expression in a join • Warning: null table name in an AggregateIncompatibility • Warning: unknown table in an AggregateIncompatibility • Warning: folder in an AggregateIncompatibility • Warning: unknown item in an AggregateIncompatibility • Exception: inexisting table referenced by alias table • Exception: join included and excluded from a context or included or excluded several times • Exception: a context contains a join that does not belong to the data foundation • Exception: missing connection • Exception: missing data foundation • Exception: not found or duplicate extra table

6.1.7 Releasing Objects

Besides the `SlContext`, `SlResource` objects must also be released after use to avoid memory leaks.

The following methods allow you to release the corresponding resources:

- `LocalResourceService.close(SlResource)`
- `CmsResourceService.close(SlResource)`
- `ConnectionFactory.close(SlResource)`

6.1.8 Raising Errors

The 4.0 FP3 and SP4 releases of the BI Semantic Layer Java SDK provide a lot of methods to build different categories of exceptions. Starting with the 4.0 SP5 release and in the 4.1 and 4.2 releases, all exceptions that the API methods throw are managed as `SlException` objects.

An `SlException` is raised if an error occurs at any stage of the workflow. Exceptions consist of the following information:

- A code made of three letters and a number, for example `SLS 10000`
- An error message

Each `SlException` object can provide `IStatus` objects that give access to error details such as code, message, severity and cause. For example:

```
exception.getStatus().getCode();
```

6.2 Using the java.util.List Methods

The BI Semantic Layer Java SDK allows you to perform many tasks on objects of the data foundations and business layers, such as the following:

- Creating, adding and removing data foundation tables
- Creating, adding, moving, and removing business layer items (folders and business objects)
- Creating lists of values and parameters
- Adding joins to data foundations, retrieving and deleting joins
- Adding joins to contexts, retrieving and deleting joins
- Adding columns to primary keys, deleting columns
- Adding columns to static lists of values, removing columns
- Adding rows to static lists of values, removing rows
- Adding values to static lists of values, removing values
- Adding a value to an answer with multiple values
- Retrieving, adding, and removing extra tables
- Adding granted or denied objects to business security profiles, removing objects

The SDK provides specific methods to create the SDK objects. For example, you use `DataFoundationFactory.createSqlJoin(String, String)` to create `SQLJoin` objects.

For the other operations, you need to use the `java.util.List` methods. In the object model, a `BlContainer` object is implemented as a list of `BlItem` objects. To add to, move or delete a `BlItem` object, you use the `BlContainer` interface.

- To add a `BlItem` object, you add it to the end of the list or at a specified position in the list.
- To delete a `BlItem` object, you remove it from the list to which it is attached.
- To move a `BlItem` object, you remove it from the list to which it is attached and add it to another list, or you change its position within the list to which it is attached.

Lists adhere to the following rules:

- Strong object relationship: an object cannot belong to two different lists. It must be stored in only one list. For example, a join can belong to only one data foundation (`DataFoundation.getJoins()`).
- Weak object relationship: an object can belong to multiple lists. For example, a join can belong to multiple contexts (`Context.getIncludedJoins()`).
- A list cannot contain the same object twice.
- Lists are ordered.

The following sections illustrate some of the operations you can perform on the objects of the BI Semantic Layer Java SDK. See the descriptions of the standard `List` methods under the link below.

Deleting a Table [page 47]

Deleting a Business Layer Item [page 47]

Moving a Business Layer Item [page 48]

Deleting a Join [page 48]

Adding Extra Tables [page 49]

Adding a Join to a Context [page 50]

Removing a Column from a Primary Key [page 50]

Adding a Value to a Row of a Static List of Values [page 50]

Adding a Dimension to a Custom Navigation Path [page 51]

Related Information

Interface List (Java Platform SE 6)

6.2.1 Deleting a Table

You can use `DataFoundation.getTables()` and one of the `java.util.List.remove()` methods to delete a data foundation table.

Example

```
MonoSourceDataFoundation dataFoundation = ...;
Table table = ...;
dataFoundation.getTables().remove(table);
```

6.2.2 Deleting a Business Layer Item

You can use `BlContainer.getChildren()` and one of the `Java.util.List.remove()` methods to delete a business layer item.

Example

You can get the measure either by determining its index (`0`) and parent:

```
Folder folder = ...;
Measure measure = (Measure) folder.getChildren().get(0);
folder.getChildren().remove(measure);
```

Or by retrieving its parent using `BlItem.getParent()`:

```
measure.getParent().getChildren().remove(measure);
```

6.2.3 Moving a Business Layer Item

You can move a `BlItem` object from one folder to another, or you can move it inside a folder.

An item can only belong to one parent folder. When you add an item to a container, it is automatically removed from the container to which it belongs.

When you move an item inside the same container, you first have to remove it from the container, and then add it again to the container. Use one of the `java.util.List.add()` methods.

Example: Moving a measure from one folder to another

You can add the measure at the end of the list. If the list already contains the measure, it remains unchanged and the method returns `false`.

```
Folder folder1 = ...;
Folder folder2 = ...;
Measure measure = (Measure) folder1.getChildren().get(0);
folder2.getChildren().add(measure);
```

You can add the measure at the 3 position in the list. If the list already contains the measure, an exception is thrown.

```
folder2.getChildren().add(3, measure);
```

Example: Changing places of a measure

You can add the measure at the end of the list:

```
Folder folder = ...;
Measure measure = (Measure) ...
folder.getChildren().remove(0);
folder.getChildren().add(measure);
```

You can add the measure at the 3 position in the list:

```
folder.getChildren().add(3, measure);
```

6.2.4 Deleting a Join

You can use `DataFoundation.getJoins()` and one of the `java.util.List.remove()` methods to delete a join from a data foundation.

Example

```
MonoSourceDataFoundation dataFoundation = ...;
SQLJoin join = (SQLJoin) ...
dataFoundation.getJoins().remove(0);
```

6.2.5 Adding Extra Tables

Extra tables are included in the query using a join in the SQL expression when returning values for the business object. The list of extra tables is a subset of tables displayed in the information design tool.

`RelationalBinding.getExtraTables()` allows you to retrieve the list of extra tables of a business object.

Once you retrieve this list in memory, you can add a table to the end of the list using one of the `java.util.List.add()` methods.

The following formatting rules apply on `qualifer.owner.table` strings that you can pass to the `List` interface methods:

- If the qualifier, owner or table name contains a period (`.`), a double quote (`"`) or a space, it must be quoted.
- A double quote inside a qualifier, owner or table name must be doubled.
- Double quotes must be escaped using a backslash `\`.

Example

The examples illustrate the ways `qualifier.owner.table` strings must be passed as arguments of the `add()` method. The following pieces of code add an extra table called `City` to the end of the list:

Code	Description
`getExtraTables().add("City")`	Qualifier and owner are null.
`getExtraTables().add("Warehouse.dbo.City")`	Qualifier is `Warehouse` and owner is `dbo`.
`getExtraTables().add("Warehouse."".City")`	Qualifier is `Warehouse` and owner is not supported.
`getExtraTables().add("\"Ware house\".dbo.City")`	Qualifier is `"Ware house"` and owner is `dbo`.
`getExtraTables().add("\"Ware\"\"house\".dbo.\"Ci.ty\"")`	Qualifier is `"Ware"house"`, owner is `dbo` and table name is `Ci.ty`.

You can use `DataFoundationService.getTableFullName(String, String, String)` to build the fully qualified name of the table.

6.2.6 Adding a Join to a Context

When you add a join to a data foundation, it is not added to the contexts of the data foundation. You must add it to a specific context explicitely.

You can use the `Context.getIncludedJoins()` or `Context.getExcludedJoins()` method to retrieve the list of included or excluded joins of a context.

Once you retrieve this list in memory, you can add a join to the end of the list using one of the `java.util.List.add()` methods.

Example

```
Context context = ...;
SQLJoin join = ...;
context.getExcludedJoins().add(join);
```

6.2.7 Removing a Column from a Primary Key

You can use `Key.getColumns()` and one of the `java.util.List.remove()` methods to remove a column from the list of columns that make up the data foundation key.

Example

```
DatabaseTable table = ...;
PrimaryKey key = (PrimaryKey) table.getPrimaryKey();
Column column = ...;
key.getColumns().remove(column);
```

6.2.8 Adding a Value to a Row of a Static List of Values

You can use `StaticLovRow.getValues()` and one of the `java.util.List.add()` methods to add a value to a row of a static list of values of a business layer.

Example

```
StaticLovRow staticLovRow = ...;
```

```
staticLovRow.getValues().add(businessLayerFactory.createDateValue(Date.valueOf("2
014-02-03")));
```

6.2.9 Adding a Dimension to a Custom Navigation Path

You can use `NavigationPath.getDimensions()` and one of the `java.util.List.add()` methods to add a dimension to a custom navigation path of a business layer.

Example

```
BusinessLayer businessLayer = ... ;
NavigationPath navigationPath = ...;
Dimension dimension = (Dimension) businessLayerService.getBlItem(businessLayer,
"Dimperiod\\Date", true);
navigationPath.getDimensions().add(dimension);
```

6.3 Working with Default Values

The SDK resources adhere to the following rules:

- When a data foundation is created, the default values of the SQL options are set automatically.
- When a business layer is created, the default values of the query options are set automatically.
- When a business object is created, it is associated with the default list of values automatically. The default list of values is the one computed for this object.

See the *SAP BusinessObjects BI Semantic Layer Java API reference* to know the default values.

6.4 Working with Data Display Formats

Creating a Data Format

Use one of the following "create" methods of the `BusinessLayerFactory` interface to set a data display format and append it to the related dimension, measure, or attribute:

- `createPredefinedDateTimeFormat(BusinessObject)`
- `createCustomDateTimeFormat(BusinessObject)`
- `createPredefinedNumberFormat(BusinessObject)`
- `createCustomNumberFormat(BusinessObject)`

Example

```
Dimension requiredDateDimension = (Dimension)
businessLayerService.getBlItem(businessLayer, "CustOrder\\RequiredDate");
CustomNumberFormat customNumberFormat =
businessLayerFactory.createCustomNumberFormat(requiredDateDimension);
...
```

Editing a Custom Data Format

Use the following methods:

- The `setFormat` and `setColor` methods of the `ValueFormat` interface to define the custom data format of a business object
- The methods of the `CustomDateTimeFormat` and `CustomNumberFormat` interfaces to retrieve the values stored in the `ValueFormat` object

A custom data format is valid if the `ValueFormat` values adhere to the following rules:

- PositiveFormat and DateTimeFormat cannot be null nor empty.
- NegativeFormat, ZeroFormat and UndefinedFormat values can be null or empty.

You cannot do a bulk change of the format of multiple business objects. You must do the change for each object explicitly.

Example

```
...
customNumberFormat.getPositiveFormat().setFormat("E+");
customNumberFormat.getPositiveFormat().setColor(FormatColor.BLUE);
customNumberFormat.getNegativeFormat().setFormat("#");
customNumberFormat.getNegativeFormat().setColor(FormatColor.RED);
customNumberFormat.getZeroFormat().setFormat("'nothing'");
customNumberFormat.getZeroFormat().setColor(FormatColor.AUTOMATIC);
customNumberFormat.getUndefinedFormat().setFormat("'??'");
customNumberFormat.getUndefinedFormat().setColor(FormatColor.BLACK);
```

String Patterns

The BI Semantic Layer Java SDK provides a series of numeric and date-time patterns that help you build the string part of the data display format.

There is no incorrect format. The string pattern is replaced with the approriate values if it is recognized. If it is not, the pattern is considered as a simple string. For example:

- The pattern `hour` is interpreted as "*[Hours with one or two digits from 1 to 12]*]ou*[Japanese Imperial Number]*".

- The pattern `xx/MM/yyyy` is interpreted as "`xx/`*[Month with two digits, from 01 to 12]*`/`*[Year number with four digits from 0000 to 9999]*".

You escape characters of a string pattern by enclosing them with single quotes. Therefore you double quotes to escape single quotes. For example:

- The pattern `'hour'` is interpreted as "`hour`".
- The pattern `d'd'` is interpreted as "*[Day in the month with one or two digits from 1 to 31]*`d`".
- The pattern `''d''` is interpreted as "`'`*[Day in the month with one or two digits from 1 to 31]*`'`".

Format Colors

The `FormatColor` enum defines a series of colors that can be used in custom data formats (`Black`, `White`, `Red`, `Green`, `Blue`, `Yellow`, `Magenta`, `Cyan`, `Dark Red`, `Dark Green`, `Dark Blue`, `Dark Yellow`, `Dark Magenta`, `Dark Cyan`, `Light Gray`, and `Gray`).

Retrieving the List of Custom Formats

The information design tool displays the list of custom formats. In the BI Semantic Layer Java SDK, you need to check that the display format of each object of the business layer is a custom format before you retrieve it.

Related Information

ValueFormat String Pattern Reference [page 99]

6.5 Working with Data Foundation Views

Editing the Master View

The master view is the default data foundation view. It contains views of all the tables of the data foundation (one view for one table). The master view cannot be created or deleted, but it can be edited to modify the existing views.

When a new table is added to the data foundation, the related view is added to the master view automatically, unless it already exists in the master view. When removing a table from the data foundation, the related view is also removed from the master view automatically, unless it has been done already.

> → Remember
>
> The following changes will be reverted when saving the master view:

- Removing views of existing tables from the master view
- Adding a view related to a table that already has a view in the master view

Creating a Custom Data Foundation View

Use the `createDataFoundationView` method to create a custom data foundation view and append it to the related data foundation.

Example

```
DataFoundation dataFoundation = ... ;
DataFoundationView dataFoundationView =
dataFoundationFactory.createDataFoundationView("MyView", dataFoundation);
dataFoundationView.setDescription("My description");
```

Adding a Table to a Custom Data Foundation View

To add a table to a custom data foundation view, you have to:

- Create an instance of the `TableView` object for the table to add and for the target view. `TableView` is used to define how a table is displayed in a view.
- Set `TableView` properties (x-axis and y-axis positions, width, and display state)

You can set different positions, widths, or display states to a table, depending on the data foundation view.

Example

```
Table table = dataFoundation.getTables().get(0);
TableView tableView = dataFoundationFactory.createTableView(table,
dataFoundationView);
tableView.setX(123);
tableView.setY(456);
tableView.setTableState(TableState.JOINS_ONLY);
```

Adding a Federated Table to a Custom Data Foundation View

The `FederatedTable` class has been added to the API to allow views to reference federated tables. This class inherits from `Table` (name, description, and columns). You cannot create, edit or delete a federated table, but you can move a federated table from a position to another within the list of data foundation tables.

In this release, you can add to and position a federated table in a data foundation view by using the `TableView` object.

Example

```
FederatedTable table = ... ;
dataFoundation.getTables().remove(0);
dataFoundation.getTables().add(2, table); // moves the table to position 2
dataFoundation.getTables().get(2);
TableView tableView = dataFoundationFactory.createTableView(table,
dataFoundationView);
```

Related Information

Saving Objects [page 42]

6.6 Working with Linked Universes

Getting the Core Universes

Use the following methods:

- `DataFoundation.getCoreUniverseReferences` to retrieve the list of core universes referenced by a data foundation
- `BusinessLayer.getCoreUniverseReferences` to retrieve the list of core universes referenced by a business layer

Getting the Core Universe Information

CUID, path, and revision number of the core universe are used as reference information in the linked data foundation or business layer. Use the `getCUID`, `getPath`, and `getRevisionNumber` methods of the `UniverseReference` class to retrieve this information.

Getting the Objects Inherited From a Core Universe

Data foundations and business layers can inherit the `BlItem`, `Table`, `Join`, `Lov`, and `Parameter` objects from core universes. Since these objects implement the `Inheritable` interface, you can retrieve their core universe and identifier by using the `Inheritable.getInheritedData` method.

The `InheritedData` class provides the following methods:

- `getCoreReference` to retrieve reference information about the core universe from which the object is inherited
- `getCoreItemIdentifier` to retrieve the identifier of the object within the core universe

Example

```
//BlItem
BlItem item = (BlItem) ...
String coreBlItemIdentifier = item.getInheritedData().getCoreItemIdentifier();
UniverseReference universeReference = item.getInheritedData().getCoreReference();
//Join
SQLJoin sqlJoin = (SQLJoin) ...
String coreJoinIdentifier = sqlJoin.getInheritedData().getCoreItemIdentifier();
UniverseReference universeReference =
sqlJoin.getInheritedData().getCoreReference();
```

Creating and Editing Linked Universes

Use one of the following methods of the `BusinessLayerService` interface to create or edit a linked universe:

Method	What You Can Do
`addCoreUniverses`	Adding core universes to a linked universe. A warning is raised if the core universe connection is different from the data foundation connection.
`removeCoreUniverses`	Removing core universes from a linked universe
`includeCoreUniverses`	Integrating all core universe content into the linked business layer and its underlying data foundation (folders, objects, tables, joins, views, and so on). The links to the core universes are removed and their objects become writable objects of the linked universe. The core universes are removed from the list of core universes.

Method	What You Can Do
`synchronizeCoreUniverses`	Synchronizing the core universe content of a linked universe with their latest versions in the CMS repository (folders, objects, tables, joins, views and so on). Synchronizing means updating, adding, or removing core universe components from the local, linked business layer and data foundation. The core universes are synchronized if their revision numbers have been modified, that is, if new versions of these universes have been published. New versions can be previous or later versions of the core universes. The `BlItem` object state is not updated during synchronization.
`refreshUniverseReferences`	Updating the path or name of the core universe in the linked universe if it has moved or been renamed in the CMS repository. **Note** The information design tool does not provide this function.

You can manage more than one core universe at once, because these methods take the `java.util.List<String>` method as argument to represent the list of core universe identifiers. Universe identifiers accepted are universe paths and CUIDs. Universe paths start with `/Universes/`.

These methods load the business layer and its underlying data foundation before performing their own operation, and save them afterwards.

> **Note**
> You can test the linked universe creation and editing by using the `EditLinkedUniverseTest.java` sample.

Saving a Linked BusinessLayer or Data Foundation

Use the `LocalResourceService.save` method to save the linked business layer or data foundation as you do for any other universe.

Since the content inherited from a core universe cannot be changed in the linked universe, an error is raised at save time if any content update is made.

Only the state of `BlItem` objects can be changed. Use the `isStatusOverloaded` method to know whether the item state has changed compared to its value in the core universe.

When the `BlItem.setState(ItemState)` method is used, the `isStatusOverloaded` return value is set to `true` automatically. The modified `State` value is retained at save time. When the business layer is loaded, the `State` value of the core universe object in the CMS repository does not overwrite the one previously modified.

Publishing Linked Universes

Use the `CmsResourceService.publish` method to publish the linked business layer or data foundation to the CMS repository as you do for any other universe.

> → Remember
>
> You must synchronize the core universes before publishing the linked universe.

Related Information

Developing with the BI Semantic Layer Java SDK Samples [page 73]

6.7 Working with Query Script Properties

> → Remember
>
> Query script properties are named query script parameters in the information design tool.

About Query Script Properties

Query script properties can be as follows:

- Predefined in the data foundation or business layer, they come with the SDK as enum values
- Created by the user and added to the data foundation or business layer dynamically.

Predefined query script properties can be boolean, string or integer (positive and negative). However, the object model defines them as {key, value} pairs whose value is always stored or returned as a string. Therefore you must cast the expected data type to string in your code. See the data types in the Predefined Query Script Property Reference [page 96].

Predefined query script properties can be mandatory or optional in a business layer or data foundation. When a data foundation or business layer is created, all mandatory query script properties are added automatically and cannot be removed.

Working With Query Script Properties

Method	What You Can Do
• `BusinessLayerFactory.createQueryScriptProperty` • `DataFoundationFactory.createQueryScriptProperty`	Adding an optional, predefined query script property or a new one (custom) to the business layer or data foundation. This method adds the query script property to the list of query script properties. The key or name of the new query script property must be unique.
`QueryScriptCustomizable.getQueryScriptProperties`	Getting the list of predefined and custom query script properties of a business layer or data foundation
`QueryScriptCustomizable.resetQueryScriptProperties`	Performing the following tasks at once on the list: 1. Removing the optional, predefined and custom query script properties that have been added to the list 2. Leaving the mandatory, predefined query script properties in the list 3. Setting the default values to these properties
`Java.util.List.remove`	Deleting a query script property from the list. You must not remove a mandatory, predefined query script property from the list.
`Property.setKey`	Modifying the key of an optional, predefined or custom query script property. The key or name of the query script property must be unique.
`Property.getKey`	Getting the key of an optional, predefined or custom query script property
`Property.setValue`	Modifying the value of a predefined (mandatory or optional) or custom query script property
`Property.getValue`	Getting the value of a predefined (mandatory or optional) or custom query script property

Example

```
// Gets the predefined and custom query script properties of the data foundation
List<QueryScriptProperty> dataFoundationProperties =
dataFoundation.getQueryScriptProperties();
// Creates a custom query script property
QueryScriptProperty dataFoundationQueryScriptProperty =
dataFoundationFactory.createQueryScriptProperty("A_KEY", "A_VALUE",
dataFoundation);
// Sets a new value to the custom property
dataFoundationQueryScriptProperty.setValue("A_NEW_VALUE");

// Removes the custom property
dataFoundationProperties.remove(dataFoundationQueryScriptProperty);
```

Saving a Business Layer or Data Foundation

An error occurs at save time if:

- Two query script properties have the same name.
- A mandatory query script property has been removed from the list.

> **Note**
>
> The following two mandatory query script properties have been added to previous 4.x releases and may not be present in a data foundation, for example if this one has been converted from a universe created with universe design tool:
>
> - `REPLACE_COMMA_BY_CONCAT`
> - `USE_ENHANCED_QUERY_STRIPPING`
>
> You can save a data foundation without them if they were not in the data foundation that has been loaded by the BI Semantic Layer Java SDK.

6.8 How Business Filter Expressions and Business Queries are Saved

When you create or edit the expression of a business filter and the business query of a list of values, the XML specification used as filter expression or business query is parsed and changed to the expected format before it is actually saved. The expression has of course the same meaning.

The XML specification is the one from the BI Semantic Layer RESTful Web Services SDK and is described in the *SAP BusinessObjects RESTful Web Services SDK User Guide for Web Intelligence and the BI Semantic Layer*. The following tables describe the updates made to the specification.

Query Options

Element	Behavior	Expression

`<queryOptions>`	• The following query options are added if missing:
	  • `duplicatedRows`
	  • `maxRetrievalTimeInSeconds`
	  • `maxRowsRetrieved`
	  • `samplingResultSetSize`
	  • `samplingResultSetFixed`
	• If there is a query option without activated attribute, then `activated="true"` is added.
	• If the `activated` and `value` attributes of `samplingResultSetFixed` have different values, they are both set to `false`.
		Business Query

Example

```
<querySpecification version="1.0">
    <queryOptions>
        <queryOption name="duplicatedRows" value="true"/>
        <queryOption name="maxRetrievalTimeInSeconds" activated="false"
value="600"/>
        <queryOption name="maxRowsRetrieved" activated="true" value="5000"/>
        <queryOption name="samplingResultSetSize" activated="false" value="200">
        <queryOption name="samplingResultSetFixed" activated="false"
value="false">
    </queryOptions>
    ...
```

Comparison Filter with Zero, One, or Two Right Operands

Element	Behavior	Expression
`<answerValue>`	`<answerValue dataType="xxx">yyy</answerValue>` becomes `<value>   <caption type="xxx">yyy</caption> </value>` • If the `dataType` attribute is not specified, the `caption type` attribute is set to String. • The `Date` type becomes `DateTime`, and `T00:00:00.000Z` is added to the value.	• Business Filter • Business Query

Example

```
<filterPart>
    <comparisonFilter operator="EqualTo" path="Time|folder\Calendar|
folder\CalendarYear|dimension"
        id="_IBo8FLIhEeCk0Ylv-tlF2Q">
        <constantOperand>
            <answerValue dataType="String">2011</answerValue>
        </constantOperand>
    </comparisonFilter>
</filterPart>
```

Comparison Filter with a List of Values for Right Operand

Element	Behavior	Expression
`<caption>` in `<value>` or `<level>`	If not specified, the `type` attribute is set to `string`.	• Business Filter • Business Query

Example

```
<filterPart>
    <comparisonFilter operator="InList" id="_zPwYENK-EeSNS_-8mYpikg" path="Store
Names|folder\Cascading_Associate_LOV_City|dimension">
        <constantOperand>
            <value>
                <caption type="String">Yancheng</caption>
```

```
            </value>
            <value>
                <caption type="String">London</caption>
                    <path>
                        <level>
                            <caption type="String">England</caption>
                        </level>
                        <level>
                            <caption type="String">London</caption>
                        </level>
                    </path>
            </value>
        </constantOperand>
    </comparisonFilter>
</filterPart>
```

Comparison Filter with a Parameter

Element	Behavior	Expression
`<parameter>`	The attributes `constrained="true"`, `keepLastValues="true"`, `optional="false"` and `promptWithLov="false"` are set if they are not specified.	• Business Filter • Business Query

Example

```
<filterPart xmlns="http://www.sap.com/rws/sl/universe">
    <comparisonFilter id="_6zk08A-8EeO1RP--CtxScg" operator="EqualTo"
path="Custorder\Orderid">
        <parameterOperand>
            <parameter constrained="true" keepLastValues="true" optional="false"
promptWithLov="true">
                <question>Enter Orderid</question>
            </parameter>
        </parameterOperand>
    </comparisonFilter>
</filterPart>
```

Ranking Filters

Element	Behavior	Expression
`<rankedByDimension>`	The `path` attribute is removed.	• Business Filter • Business Query

Example

```
<filterPart>
            <rankingFilter level="3" function="Top">
                <prompt>Enter ranking level :</prompt>
                <dimension path="Dimcustomer|folder\Customer Name|dimension"
id="_7Zkd8A-8EeO1RP--CtxScg"/>
                <basedOnMeasure path="Custorderline|folder\Quantity|measure"
id="_60Bg4g-8EeO1RP--CtxScg"/>
                <rankedByDimensions>
                    <rankedByDimension path="Dimcustomer|folder\Regionname|
dimension" id="_60xHwQ-8EeO1RP--CtxScg"/>
                    <rankedByDimension path="Dimcustomer|folder\Countryname|
dimension" id="_60xHwQ-8EeO1RP--CtxScg"/>
                </rankedByDimensions>
            </rankingFilter>
        </filterPart>
```

Subquery Filters

Element	Behavior	Expression
`<subQueryFilter>`	If not specified, the `correlationType` attribute is set to `None`.	• Business Filter • Business Query
`<filterObject>`	The `path` attribute is removed.	

Example

```
<filterPart>
    <subQueryFilter operator="EqualTo" correlationType="Any">
        <filterObjects>
            <filterObject path="Customer|folder\Geography|folder\Continent|
dimension"
            id="_IBo8M7IhEeCk0Ylv-tlF2Q"/>
            <filterObject path="Customer|folder\Geography|folder\Country|
dimension"
            id="_IBo8NrIhEeCk0Ylv-tlF2Q"/>
        </filterObjects>
        <queryData>
        <resultObjects>
            <resultObject path="Customer|folder\Geography|folder\Continent|
dimension"
            id="_IBo8M7IhEeCk0Ylv-tlF2Q"/>
            <resultObject path="Customer|folder\Geography|folder\Country|
dimension"
            id="_IBo8NrIhEeCk0Ylv-tlF2Q"/>
        </resultObjects>
        <filterPart>
        ...
```

6.9 Working with the Tables Setting of a Data Security Profile

The `securityFactory.createDataSecurityProfile()` method creates an empty data security profile. You need to create the profile settings ("Rows", "Tables", or "Connections") by using the `createRowRestriction`, `createTableMapping`, and `createConnectionMapping` methods.

In the case of the "Tables" setting, the original table to set using `setOriginalTable(String)` must be a table that belongs to the data foundation.

The table name can be one of the following names:

- Table name without qualifier and owner, if the table has the default owner and qualifier or if the database does not support qualifiers and owners.
- Fully qualified table name of the format `"qualifier"."owner"."table"`. The string to pass must contain all the information, even if the database does not support qualifiers and owners. For example, if owners are not supported, then the string must be `"qualifier"."".."table"`.

You can use `DataFoundationService.getTableName(String, String, String)` to build the fully qualified name of the table. You can also use this string as argument of `RowRestriction.setTable(String)`.

6.10 Working with the Secured Elements of a Business Security Profile

The `securityFactory.createBusinessSecurityProfile()` method creates an empty business security profile. You need to add business layer views or items explicitly to the profile to create the profile settings.

You create the settings of a business security profile by using `SecuredElements` objects. A `SecuredElements` object defines the list of business layer views or business layer items that are granted or denied for a profile. A `SecuredElement` object is identified by a string that is the path to the object in the business layer.

You build the strings by following the rules below:

- If the object is a business layer view, then it is identified by its name.The master view is identified by an empty string (`""`).
- If the object is a business layer item, then the string represents the full path of the object. The following rules apply:
  - Each node in the path is made of the business layer item name and its type, separated by `|`: "`<name>|<type>`".
  - The escape character of `|` and ~ is ~.
  - The nodes of the path are concatenated with `\`: "`Age Group|folder\Age Max|dimension`".
  - The escape character of `\` and § is §.

Example

The following examples illustrate the naming rules:

Path	Object	String
Root folder	"CustomerName" dimension	"`CustomerName|dimension`"
"Customer" folder	"Name" dimension	"`Customer|folder\Name|dimension`"
"Customer" dimension in "Contact" folder	"Name" attribute	"`Contact|folder\Customer|dimension\Name|attribute`"
"Customer\Large" dimension in "Country|US" folder	"First~Name" attribute	"`Country~|US|folder\Customer§\Large|dimension\First~~Name|attribute`"

The types of business objects allowed in a business security profile are the following:

- `folder`
- `dimension`
- `measure`
- `attribute`
- `filter`

You can use the methods provided by the `BusinessSecurityProfile` and `SecuredElements` interfaces and one of the `java.util.List.add()` methods to add a list of granted or denied objects to a business security profile. To set all the objects to granted or denied, you can use the `SecuredElements.setAllElementsStatus()` method.

Example

```
//Creates an empty profile and adds its name
BusinessSecurityProfile bsp = securityFactory.createBusinessSecurityProfile();
bsp.setName(bspName);
//In the "Create Query" settings, adds all views of the business layer as
granted, except the master view that is set to Denied
bsp.getCreateQuerySecuredViews().setAllElementsStatus( SecurityStatus.Granted );
bsp.getCreateQuerySecuredViews().getDeniedElements().add ("");
//In the "Create Query" settings, adds all objects of the business layer as
granted, except the object CompanyName that is set to Denied
bsp.getCreateQuerySecuredObjects().setAllElementsStatus( SecurityStatus.Granted )
;
bsp.getCreateQuerySecuredObjects().getDeniedElements().add("Customer|
folder\CompanyName|dimension");

//In the "Display Data" settings, adds all objects of the business layer as
granted, except the object CompanyName that is set to Denied
bsp.getDisplayDataSecuredObjects().setAllElementsStatus( SecurityStatus.Granted )
;
```

```
bsp.getDisplayDataSecuredObjects().getDeniedElements().add("Customer|
folder\CompanyName|dimension");
```

6.11 Refreshing the Structure of a Data Foundation

You can use `DataFoundationService.refreshStructure(String)` to synchronize the metadata of the database and the metadata of the data foundation.

Contrary to the information design tool, the method does not allow you to choose the type of object to update. It refreshes in the data foundation all the objects modified in the database.

The `refreshStructure(String)` method performs the following operations:

- Loading the data foundation
- Refreshing the data foundation (joins, tables, and so on.)
- Saving the data foundation

> → Remember
>
> If you load the data foundation before running the method, the data foundation will not be updated.

6.12 Running Check Integrity

You can use `LocalResourceService.createCheckIntegrityRunner(String)` to check if the data foundation or business layer that you save locally contains any issue about contexts, joins, connections, parameters, and lists of values. You can also use this method to check a local connection by passing as argument the path to the connection.

Contrary to the information design tool, the method does not allow you to choose the type of object to check. The rules are run all at once.

The check integrity runs only on a saved resource. It returns an `IStatus` object that contains the error severity, message, and code, and a status for each rule. The result status of a rule is managed through a `RuleStatus` object. So an `IStatus` object can be represented as a tree made of `RuleStatus` and `IStatus` objects, for example:

```
ERROR Check integrity // IStatus
-->OK Rule "Prompts validity rules" execution succeeded. // RuleStatus
-->OK Rule "Check Business Object Name " execution succeeded.
-->OK Rule "Query validity rules" execution succeeded.
-->OK Rule "Broken dependencies" execution succeeded.
-->OK Rule "Check Connection" execution succeeded.
-->OK Rule "LOV validity rules" execution succeeded.
-->OK Rule "Business filter validity" execution succeeded.
-->ERROR Business object validity (SQL/MDX) // RuleStatus
---->ERROR The business object 'Dimperiod\Order Year' return data type is not
correct. (CIM 01404) // IStatus
java.lang.RuntimeException: Wrong data type: String
       at ...
```

```
        at com.sap.checkintegrity.core.AbstractRuleExecutor.execute(Unknown
Source)
        at
com.sap.checkintegrity.core.internal.IntegrityThread.runChecked(Unknown Source)
        at com.sap.checkintegrity.core.internal.IntegrityThread.run(Unknown
Source)
-->OK Rule "Business object validity (Binding)" execution succeeded.
```

You can cancel the process if it takes too much time. Before the cancel operation is performed, the process completes the current task. The `IStatus` object provides only the results for the rules processed before the cancel operation.

The check integrity does not run all the tests that can be performed on an object. Most of the tests are done when the object is saved.

Example

```
public void checkIntegrityOnLoadedObject() throws Exception {
    //...
    LocalResourceService localResourceService =
context.getService(LocalResourceService.class);
    final CheckIntegrityRunner checkIntegrityRunner =
localResourceService.createCheckIntegrityRunner("...");
        Thread thread = new Thread() {
            public void run() {
                IStatus status = checkIntegrityRunner.start();
                //...
            }
        };
        thread.start();
        //...
        checkIntegrityRunner.cancel();
    }
```

Related Information

Checking Objects [page 43]

6.13 Working with Multisource-Enabled Universes

Most of the features that apply to single-source universes also apply to multisource-enabled universes, but some workflows are specific to multisource-enabled universes:

- Managing the data source life cycle for a multisource-enabled universe
- Creating a multisource-enabled data foundation
- Removing a data source from a multisource-enabled data foundation

- Creating derived tables for a multisource-enabled data foundation

Managing the Connection Life Cycle [page 69]

Creating a Multisource-Enabled Data Foundation [page 70]

Removing Data Sources from a Multisource-Enabled Data Foundation [page 70]

Creating Derived Tables for a Multisource-Enabled Data Foundation [page 71]

6.13.1 Managing the Connection Life Cycle

You use the `DataFederatorService` to manage the connection life cycle in the case of multisource-enabled universes. Follow the steps below before you create a multisource-enabled data foundation.

1. Retrieve the catalog of the data source associated with the connection using `getCatalog(ConnectionShortcut)`.
2. Publish the connection to the Data Federation Query Server using `deploy(ConnectionShortcut, String)`.

> → Remember
>
> This is a mandatory step before using a connection in any operation involving multisource-enabled universes.

This service also provides the following methods:

- `undeploy(String catalog)`
  This removes the universe connection from the Data Federation Query Server.
- `closeServerConnection()`
  This closes the connection to Data Federation Query Server.

> ⓘ Note
>
> The server connection is closed automatically when `SlContext.close()` is called.

Example

```
//Instantiates the resource
ConnectionShortcut connectionShortcut = (ConnectionShortcut) ...;
//Loads the service in memory
DataFederatorService dataFederatorService =
context.getService(DataFederatorService.class);
//Retrieves the catalog
String catalog = dataFederatorService.getCatalog(connectionShortcut);
//Deploys the connection to the Query Server if no catalog is returned
if (catalog == null)
 catalog = ...;
 dataFederatorService.deploy(connectionShortcut, catalog);
```

Related Information

Creating Sessions [page 39]

6.13.2 Creating a Multisource-Enabled Data Foundation

You use the `DataFoundationFactory` to create multisource-enabled data foundations on the CMS. Follow the steps below:

1. Create a multisource-enabled data foundation without connections using `createMultiSourceDataFoundation(String)`.
2. Create a `DataFederatorSourceInfo` object and attach it to the given data foundation using `createDataFederatorSourceInfo(String connectionPath, String shortName, MultiSourceDataFoundation dataFoundation)`.
   This object holds the path to the connection shortcut. By creating `DataFederatorSourceInfo`, you have added a data source to the multisource-enabled data foundation. The `shortName` argument is the unique identifier of the associated connection in the data foundation.

> ⚠ Caution
>
> You cannot save the multisource-enabled data foundation if you do not create the `DataFederatorSourceInfo` object first.

Example

```
factory = context.getService(DataFoundationFactory.class);
MultiSourceDataFoundation dataFoundation =
factory.createMultiSourceDataFoundation("new dfo");
factory.createDataFederatorSourceInfo(connectionShortcut, "DFSourceInfo_1",
dataFoundation);
```

6.13.3 Removing Data Sources from a Multisource-Enabled Data Foundation

You can use `MultisourceDataFoundation.getSourceInfo()` and one of the `java.util.List.remove()` methods to remove a connection from the list of connections attached to the multisource-enabled data foundation. The list is not ordered.

Example

```
MultiSourceDataFoundation dataFoundation = ...;
DataFederatorSourceInfo source = ...;
dataFoundation.getSourceInfo().remove(source);
```

You can remove all the connections from the data foundation using `java.util.List.clear()`.

Example

```
MultiSourceDataFoundation dataFoundation = ...;
dataFoundation.getSourceInfo().clear();
```

Removing a data source from the data foundation does not imply that the tables, joins and filters associated with the data source have also been removed. Once you have removed a data source, you must remove them explicitly.

Example

```
MultiSourceDataFoundation dataFoundation = ...;
DataFederatorSourceInfo source = ...;
dataFoundation.getSourceInfo().remove(source);
Table table = ...;
dataFoundation.getTables().remove(table);
SQLJoin join = (SQLJoin) ...
dataFoundation.getJoins().remove(join);
```

6.13.4 Creating Derived Tables for a Multisource-Enabled Data Foundation

You can use `DataFoundationFactory.createDerivedTable(String, String, DataFoundation)` to create a derived table for a single-source or a multisource-enabled data foundation. In the case of a multisource-enabled data foundation, this method creates by default a derived table that supports the ANSI SQL-92 standard syntax.

You can use `DataFoundationFactory.createDerivedTable(String name, String sql, String shortName, MultiSourceDataFoundation dataFoundation)` to create a database-specific derived table for a multisource-enabled data foundation. The `sql` argument complies with the SQL syntax of the specified data source. The name of the specific data source used is passed in the `shortName` argument.

You can use `DerivedTable.getSourceShortName()` to retrieve the short name of the data source used to built the derived table.

If the data foundation is single-source, the method returns an empty string. If it is multisource-enabled, the following cases apply:

- If the derived table is built from tables of multiple data sources, the method returns an empty string.
- If the derived table is built from tables of one data source that uses a database-specific SQL syntax, the method returns the result of `DataFederatorSourceInfo.getShortName()`.

You can use `DerivedTable.setSourceShortName(String)` to change the data source used to build the derived table.

> ⚠ Caution
>
> SAP does not recommend to use this method to transform a database-specific table into an ANSI SQL-92 standard table, but recommends to create an ANSI SQL-92 table directly.

7 Developing with the BI Semantic Layer Java SDK Samples

Samples are code snippets that demonstrate the usage of the BI Semantic Layer Java SDK. They allow you to understand how a Java application can be implemented. As a Java developer, you can use these samples as a basis to facilitate and accelerate your own programming. You can also place breakpoints into the snippet code to explore it.

These samples are supplied in the archive `<slsdk-install-dir>\SDK Samples\com.sap.sl.sdk.authoring.samples.source.jar`. They are self-documented in this release. Find descriptions and explanations of how samples work in the sample code files.

Sample Packages [page 73]

To Develop with the Help of Samples [page 74]

7.1 Sample Packages

The SDK samples are described in the following table.

Class	Description
`AssignProfileTest`	Assigns and unassigns a data security profile to a user or a group
`ChangeUniverseConnectionsTest`	Replaces several connections attached to a multisource-enabled universe with others in a CMS repository
`ChangeUniverseConnectionTest`	Replaces a connection attached to a single-source universe with another in a CMS repository
`CreateBusinessSecurityProfileTest`	Creates a business security profile with "Create Query" and "Display Data" settings and attaches this profile to a universe
`CheckIntegrityTest`	Runs check integrity on a resource saved locally
`CreateUniverseTest` (updated in 4.2 SP3)	Creates the data foundation and business layer of a universe, and publishes it to a CMS repository
`DataSecurityProfileTest`	Creates a data security profile with "Connections", "Rows", and "Tables" settings and attaches this profile to a universe. Replaces `CreateRowRestrictionTest` sample.
`DisplayLinkedUniverseTest`	Retrieves a universe and checks if it has a core universe

Class	Description
`EditLinkedUniverseTest` (new in 4.2 SP3)	Edits a linked universe to demonstrate core universe features (adding, removing, refreshing, synchronizing, and including)
`EditLocalBusinessLayerTest`	Edits fields of a business layer item in a local business layer
`EditLocalUniverseTest`	Edits the name of the connection, the data foundation, and the business layer of a local universe, and publishes the updated universe locally
`GetUniverseConnectionPathsTest`	Retrieves the paths of secured connections attached to a universe in a CMS repository
`LovAndParameterTest` (updated in 4.2 SP3)	Creates lists of values and associates them with parameters
`PublishResourceTest`	Publishes or republishes a universe to a CMS repository
`QueryScriptPropertyTest` (new in 4.2 SP3)	Sets and retrieves query script properties of a data foundation and a business layer
`RefreshStructureTest`	Synchronizes the database metadata and the data foundation
`RetrieveResourceSaveForAllTest`	Retrieves a universe and its associated connections from a CMS repository to a local folder with optional encryption
`UnvConversionTest`	Converts a `.unv` universe into a `.unx` universe
`UpdateConnectionTest`	Edits connection parameters (user, password, etc.) in a CMS repository

Utilities Package

The `com.sap.sl.sdk.samples.util` package provides the following classes:

- `AuthenticationMode` defines the authentication modes used in samples.
- `SamplesUtilities` mainly provides factorized methods to simplify sample code.

7.2 To Develop with the Help of Samples

1. Follow one of the deployment procedures described previously in this guide.
2. Import into your IDE the sample source archive file:
   a. Right-click `src` project folder.
   b. Select *Import* > *General* > *Archive File*.

c. Select `<slsdk-install-dir>\SDK samples\com.sap.sl.sdk.authoring.samples.source.jar` and choose *OK*.

You have imported the archive file into your project source folder.

3. Open the sample you want to use, copy the code snippet that embeds the functionality demonstrated by the sample and paste it into your application source file.

> → Remember
>
> Make sure you edit the parameters as specified in the sample file.

4. Develop your application.

You have developed an application with the use of the SDK samples.

Related Information

To Deploy the BI Semantic Layer Java SDK in a Non-OSGI Eclipse Configuration [page 34]
To Deploy the BI Semantic Layer Java SDK in an Eclipse OSGI Framework Configuration [page 35]

8 Troubleshooting the BI Semantic Layer Java SDK

This section describes some problems you may have with the BI Semantic Layer Java SDK with their solutions.

CreateProcess Error [page 76]

csEx Unsupported Operation [page 76]

Driver is Unknown [page 77]

Invalid Database Table [page 77]

8.1 CreateProcess Error

Syntax

```
Cannot run program "C:\Program Files\Java\jre6\bin\Javaw.exe" (in directory
"C:\Users\Administrator\SL SDK 4.1"): CreateProcess error=206, The filename or
extension is too long
```

This error message may appear if you are running a JUNIT test in Eclipse 3. Use Eclipse 4.2 to avoid this problem.

8.2 csEx Unsupported Operation

Syntax

```
java.lang.UnsupportedOperationException: csEX
```

If this error message appears, verify that you have added the following argument to the Java Virtual Machine:

```
-Dbusinessobjects.connectivity.directory="D:\SAP BusinessObjects\SAP
BusinessObjects Enterprise XI 4.0\dataAccess\connectionServer"
```

8.3 Driver is Unknown

Syntax

```
com.sap.sl.sdk.authoring.internal.AuthoringException: The connection driver
"xxxx" is unknown. (SLS 16000)
```

If this error message appears, verify the following items:

- You have installed a 32-bit JRE on your machine. A 64-bit JRE will cause this error.
- You have set the `Path` environment variable to `<bip-install-dir>\SAP BusinessObjects Enterprise XI 4.0\win64_x86;%Path%`. For example set the following path:

  ```
  set Path=D:\SAP BusinessObjects\SAP BusinessObjects Enterprise XI
  4.0\win64_x86;%Path%
  ```

- The DBMS and network layer values you pass to the `createRelationalConnection` method are valid. See the *Connection Parameter Reference* chapter for valid values of each native connection type.

8.4 Invalid Database Table

Syntax

```
com.sap.sl.sdk.authoring.internal.AuthoringException: The database table is not
valid. (SLS 11004)
```

This error message may appear if you are creating a database table using the `createDatabaseTable(String qualifier, String owner, String name, DataFoundation dataFoundation)` method and if you use lowercase for `name`.

If you create the connection definition with a username in lowercase, the DBMS lets the connection access the database, because this operation disregards the credential case. However, if you pass the username in lowercase when creating a table, a problem occurs because the query that creates the table is case-sensitive.

To solve the problem, use the username in uppercase when you define the connection and create the table.

9 Appendix

Connection Parameter Reference [page 78]

Predefined Query Script Property Reference [page 96]

Identifiable Object Reference [page 98]

ValueFormat String Pattern Reference [page 99]

9.1 Connection Parameter Reference

This section deals with the connectivities supported by the BI Semantic Layer Java SDK and the user rights associated with them.

About User Rights [page 78]

About Connection Parameters [page 79]

RDBMS Connections [page 82]

SAP Connections [page 91]

OLAP Connections [page 95]

9.1.1 About User Rights

The BI Semantic Layer Java SDK allows you to update connections and connection parameters with the help of the following methods:

- `CmsResourceService.loadConnection`
- `DatabaseConnection.getParameter`
- `ConnectionParameter.setValue`
- `ConnectionParameter.getValue`
- `CmsResourceService.saveConnection`

For security reasons, these operations are secured by the following user rights in the CMC:

- "Log on to the Designer and view this object in the CMC"
- "Edit objects"
- "Download connection locally"
- "Create, modify or delete connections"

The user right called "Log to the CMS", which is attached to the information design tool, is checked when the CMS session is created, before any service operation.

"Edit objects" and "Download connection locally" are user rights attached to connection objects of the CMS, whereas "Create, modify or delete connections" is attached to the information design tool. The "Download connection locally" user right is related to relational connections only. Data Federator data sources and OLAP connections do not support it.

User rights are assigned to users or user groups in the CMC. Methods verify the rights before saving the changes that happen to the resources.

9.1.2 About Connection Parameters

Most of the secured connection parameters contain sensitive information and must remain stored in the CMS repository. They are called private parameters. This means that if used in relational connections, they can be retrieved only if the "Download connection locally" right is granted. This is the case for `USER_NAME`, `DATASOURCE`, and so on. The `loadConnection` method returns the full list of connection parameters, but with no private parameter values if this right is denied.

`PASSWORD` parameter is private and write-only. Its value cannot be retrieved from the CMS, even if the "Download connection locally" right is granted. This means the `loadConnection` method returns a connection without the `PASSWORD` value and the `getValue` method cannot be used for this parameter.

Some of the connection parameters stored in the CMS are public. They can be retrieved even if the "Download connection locally" right is denied. This is the case for `AUTHENTICATION_MODE`. `CONNECTIVITY_TYPE`, `DBMS` and `NETWORK_LAYER` are public and read-only parameters. However, you can modify them indirectly with the `changeDriver` method.

To edit connections, users must have the "Create, modify or delete connections" and "Edit objects" rights granted. If one of these rights is denied, the `saveConnection` method cannot be used and an error message is returned.

List of Connection Parameters [page 79]

Connection Server Connection Parameters [page 81]

Data Federator and OLAP Connection Parameters [page 82]

9.1.2.1 List of Connection Parameters

The following table lists the connection parameters defined as constant values in the `DatabaseConnection.java` file and shows for each parameter whether it is public or private, readable and writable. Parameter names are constant values. This list shows most of the login parameters which are common to relational, Data Federator and OLAP connections. Also presented are the SAP connection parameters.

The `DatabaseConnection.java` file does not provide constants for some connection parameters specific to connection types such as Apache Hadoop HIVE, IBM Informix and Oracle EBS.

Parameter Name	Access Type	Readable	Writable
AUTO_RECONNECT	Private	YES	YES
AUTHENTICATION_MODE	Public	YES	YES
CLASS	Private	YES	YES
CONNECTIVITY_TYPE	Public	YES	NO, except with `changeDriver` method
DATABASE	Private	YES	YES
DATASOURCE	Private	YES	YES
DBMS	Public	YES	NO, except with `changeDriver` method
FETCH_SIZE	Private	YES	YES
LANGUAGE	Private	YES	YES
NETWORK_LAYER	Public	YES	NO, except with `changeDriver` method
OLAP_CUBE	Public	YES	YES
OLAP_PATH	Public	YES	YES
OLAP_SERVER_NAME	Public	YES	YES
OLEDBPROVIDER	Private	YES	YES
PASSWORD	Private	NO	YES
PROVIDERSTRING	Private	YES	YES
SAP_APPLICATION_SERVER_NAME	Private	YES	YES
SAP_CLIENT_NUMBER	Private	YES	YES
SAP_GATEWAY_HOST_NAME	Private	YES	YES
SAP_GATEWAY_SERVICE_NAME	Private	YES	YES
SAP_GROUP_NAME	Private	YES	YES
SAP_HANA_HOST_NAME	Private	YES	YES

Parameter Name	Access Type	Readable	Writable
SAP_HANA_INSTANCE_NUMBER	Private	YES	YES
SAP_HANA_SERVER_TYPE	Private	YES	YES
SAP_MESSAGE_SERVER_NAME	Private	YES	YES
SAP_PROGRAM_MAPPING	Private	YES	YES
SAP_SAVE_LANGUAGE	Private	YES	YES
SAP_SERVER_TYPE	Private	YES	YES
SAP_SSO_USE_SNC	Private	YES	YES
SAP_SYSTEM_ID	Private	YES	YES
SAP_SYSTEM_NUMBER	Private	YES	YES
SAP_USE_GATEWAY	Private	YES	YES
SAP_USE_PROGRAM_MAPPING	Private	YES	YES
USER_NAME	Private	YES	YES
USE_SSL	Private	YES	YES

9.1.2.2 Connection Server Connection Parameters

This section deals only with connections managed by Connection Server.

The following table shows the restrictions that apply to the use of the connection update methods with respect to the user rights in the CMC. If a method cannot be used to retrieve or set a parameter, an exception is thrown. Where not specified, the methods can be used with these rights for the specified parameter.

	Public Parameter	Private Parameter
"Edit objects": Granted "Download connection locally": Granted	All methods can be used. If the parameter is read-only: • `saveConnection` can be used but the parameter value is not updated.	All methods can be used.

	• `setValue` cannot be used.	
"Edit objects": Denied "Download connection locally": Granted	`saveConnection` cannot be used, because edit is denied.	`saveConnection` cannot be used, because edit is denied.
"Edit objects": Granted "Download connection locally": Denied	All methods can be used. If the parameter is read-only: • `saveConnection` can be used but the parameter value is not updated. • `setValue` cannot be used.	`loadConnection` returns a connection that contains all parameters. `getValue` returns an empty string.
"Edit objects": Denied "Download connection locally": Denied	`saveConnection` cannot be used.	`loadConnection` returns a connection that contains all parameters. `getValue` returns an empty string. `saveConnection` cannot be used.

9.1.2.3 Data Federator and OLAP Connection Parameters

This section covers Data Federator data sources and OLAP connections. These connections do not support the "Download connection locally" right.

The following restrictions apply to the use of the connection update methods with respect to the user rights in the CMC. If a method cannot be used to retrieve or set a parameter, an exception is thrown. Where not specified, the methods can be used with these rights for the specified parameter.

	Public Parameter	Private Parameter
"Edit objects": Granted	All methods can be used. If the parameter is read-only: • `saveConnection` can be used but the parameter is not updated. • `setValue` cannot be used.	All methods can be used.
"Edit objects": Denied	`saveConnection` and `setValue` cannot be used.	`saveConnection` and `setValue` cannot be used.

9.1.3 RDBMS Connections

The following tables list the parameters supported for each type of RDBMS connectivity where Connection Server is used. Use the parameter values in your implementation to allow connections to these databases.

DB2 Connections [page 83]
JDBC Connections [page 84]
ODBC Connections [page 85]
OLE DB Connections [page 87]
Oracle Client Connections [page 88]
SAP Sybase Connections [page 89]
SAS Connections [page 90]
Text File Connections [page 90]

9.1.3.1 DB2 Connections

The following table lists the parameters that describe a connection to IBM DB2 databases through Connection Server.

Parameter Name	Description
`CONNECTIVITY_TYPE`	Value is `relational`.
`DBMS`	Type is String. Value is one of the following: • `DB2 v9` • `DB2 for z/OS v9` • `DB2 10 for LUW` • `DB2 10.5 for LUW` • `DB2 10 for z/OS` • `DB2 UDB for iSeries v5` • `DB2 for i v6` • `DB2 for i v7`
`NETWORK_LAYER`	Value is `IBM DB2 Client`.
`AUTHENTICATION_MODE`	Type is enum. Possible values are the following: • `FIXED` • `MAPPING`
`USER_NAME`	Type is String.
`PASSWORD`	Type is String.
`DATASOURCE`	Alias name. Type is String.

9.1.3.2 JDBC Connections

The following table lists the parameters that describe a JDBC connection established through Connection Server.

Parameter Name	Description
`CONNECTIVITY_TYPE`	Value is `relational`.
`DBMS`	Type is String. Value depends on the database. See table below.
`NETWORK_LAYER`	Value is `JDBC Drivers`.
`AUTHENTICATION_MODE`	Type is enum. Possible values are the following: • `FIXED` • `MAPPING` • `SSO` (if supported)
`USER_NAME`	Type is String.
`PASSWORD`	Type is String.
`DATABASE`	Default database name. Type is String. May not be required for all data sources. Is used as Net Service for Oracle databases.
`DATASOURCE`	Type is String. Format is `<server_host>:<port>`. Also represents JDBC URL string for `Generic JDBC datasource` only.
`CLASS`	JDBC driver class. Type is String. Required for `Generic JDBC datasource` only.

The following table describes the possible values of the DBMS parameter per database vendor.

Database Vendor	DBMS
Apache	• `Apache Hadoop HIVE` • `Amazon EMR HIVE` • `Derby 10 Embedded`
Generic	`Generic JDBC datasource`
GreenPlum	`GreenPlum 4`
Hewlett Packard	• `HP Neoview` • `HP Vertica 6.1`
HSQLDB	`HSQLDB 1.8 Embedded`

Database Vendor	DBMS
IBM	• DB2 v9 • DB2 10 for z/OS • DB2 10 for LUW • DB2 10.5 for LUW • Informix Dynamic Server 11
Ingres	Ingres Database 9
Microsoft	• MS SQL Server 2008 • MS SQL Server 2012
Netezza	• Netezza Server 4 • Netezza Server 5 • Netezza Server 6 • Netezza Server 7
Oracle	• MySQL 5 • Oracle 10 • Oracle 11 • Oracle Exadata
PostgreSQL	• PostgreSQL 8 • PostgreSQL 9
SAP	• MaxDB 7.7 • SAP HANA database 1.0
SAP Sybase	• Sybase Adaptive Server Enterprise 15.5 • Sybase IQ 15 • Sybase IQ 16 • Sybase SQL Anywhere 11 • Sybase SQL Anywhere 12 • Sybase SQL Anywhere 16
Teradata Corporation	• Teradata 12 • Teradata 13 • Teradata 14 • Teradata 14.1

9.1.3.3 ODBC Connections

The following table lists the parameters that describe an ODBC connection established through Connection Server.

Parameter Name	Description
CONNECTIVITY_TYPE	Value is `relational`.
DBMS	Type is String. Value depends on the database. See table below.
NETWORK_LAYER	Value is `ODBC Drivers`.
AUTHENTICATION_MODE	Type is enum. Possible values are the following: • `FIXED` • `MAPPING` • `SSO` (if supported)
USER_NAME	Type is String.
PASSWORD	Type is String.
DATABASE	Default database name. Type is String. May not be required for all data sources.
DATASOURCE	Type is String. Value is the data source name (DSN).

The following table describes the possible values of the DBMS parameter per database vendor.

Database Vendor	DBMS
Generic	• `Generic ODBC datasource` • `Generic ODBC3 datasource`
GreenPlum	`GreenPlum 4`
Hewlett Packard	• `HP Neoview` • `HP Vertica 6.1`
IBM	• `DB2 UDB for iSeries v5` • `DB2 for i v6` • `DB2 for i v7` • `Informix Dynamic Server 11`
Ingres	`Ingres Database 9`
Microsoft	• `MS Access 2007` • `MS Access 2010` • `MS Access 2013` • `MS Excel 2007` • `MS Excel 2010` • `MS Excel 2013` • `MS SQL Server 2008` • `MS SQL Server 2012` • `Text Files`

Database Vendor	DBMS
Netezza	• `Netezza Server 4` • `Netezza Server 5` • `Netezza Server 6` • `Netezza Server 7`
Oracle	`MySQL 5`
PostgreSQL	• `PostgreSQL 8` • `PostgreSQL 9`
salesforce.com	`OpenAccess for Salesforce`
SAP	• `MaxDB 7.7` • `SAP HANA database 1.0`
SAP Sybase	• `Sybase IQ 15` • `Sybase IQ 16` • `Sybase SQL Anywhere 11` • `Sybase SQL Anywhere 12` • `Sybase SQL Anywhere 16`
Teradata Corporation	• `Teradata 12` • `Teradata 13` • `Teradata 14` • `Teradata 14.1`

9.1.3.4 OLE DB Connections

The following table lists the parameters that describe an OLE DB connection established through Connection Server.

Parameter Name	Description
`CONNECTIVITY_TYPE`	Value is `relational`.
`DBMS`	Type is String. Value depends on the database. See table below.
`NETWORK_LAYER`	Value is `OLE DB Providers`.
`AUTHENTICATION_MODE`	Type is enum. Possible values are the following: • `FIXED` • `MAPPING` • `SSO` (if supported)

Parameter Name	Description
USER_NAME	Type is String.
PASSWORD	Type is String.
DATABASE	Default database name. Type is String.
DATASOURCE	Server name. Type is String.
OLEDBPROVIDER	OLE DB Provider name. Type is String.
PROVIDERSTRING	OLE DB Provider string.

The following table describes the possible values of the DBMS parameter per database vendor.

Database Vendor	DBMS
Generic	`Generic OLEDB Provider`
Microsoft	• `MS SQL Server 2008` • `MS SQL Server 2012`

9.1.3.5 Oracle Client Connections

The following table lists the parameters that describe an Oracle OCI connection established through Connection Server.

Parameter Name	Description
CONNECTIVITY_TYPE	Value is `relational`.
DBMS	Type is String. Values can be the following: • `Oracle 10` • `Oracle 11` • `Oracle EBS` • `Oracle Exadata`
NETWORK_LAYER	Value is `Oracle Client`.
AUTHENTICATION_MODE	Type is enum. Possible values are the following: • `FIXED` • `MAPPING` • `SSO`

Parameter Name	Description
USER_NAME	Type is String.
PASSWORD	Type is String.
DATASOURCE	Type is String. Value could be either the Oracle SID or Oracle connection string.

Example: Data Source

In the following example, Data Source value is either `ORA11` or the expression equal to `ORA11`:

```
ORA11 =
 (DESCRIPTION =
   (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = 1521))
   )
   (CONNECT_DATA =
      (SERVICE_NAME = ORA11)
   )
 )
```

9.1.3.6 SAP Sybase Connections

The following table lists the parameters that describe a connection to SAP Sybase databases through Connection Server.

Parameter Name	Description
CONNECTIVITY_TYPE	Value is `relational`.
DBMS	Type is String. Value is `Sybase Adaptive Server Enterprise 15.5`.
NETWORK_LAYER	Value is `Sybase Open Client`.
AUTHENTICATION_MODE	Type is enum. Possible values are the following: • `FIXED` • `MAPPING`
USER_NAME	Type is String.
PASSWORD	Type is String.
DATABASE	Default database name. Type is String.
DATASOURCE	Type is String. Format is `<server_host>:<port>`.

9.1.3.7 SAS Connections

The following table lists the parameters that describe a connection to SAS database through the Data Federator connector.

Parameter Name	Description
`CONNECTIVITY_TYPE`	Value is `relational`.
`DBMS`	Value is `Data Federator Server XI R4`.
`NETWORK_LAYER`	Value is `JDBC Drivers`.
`AUTHENTICATION_MODE`	Type is enum. Possible values are the following: • `FIXED` • `MAPPING`
`USER_NAME`	Type is String.
`PASSWORD`	Type is String.

9.1.3.8 Text File Connections

The following table lists the parameters that describe a connection to Text files through Connection Server.

Parameter Name	Description
`CONNECTIVITY_TYPE`	Value is `open`.
`DBMS`	Value is `Text Files`.
`NETWORK_LAYER`	Value is `BO OC`.
`AUTHENTICATION_MODE`	Type is enum. Possible values are the following: • `FIXED` • `MAPPING`
`USER_NAME`	Type is String.
`PASSWORD`	Type is String.
`DATASOURCE`	Type is String. Value is UNIX or MS Windows style local file path or URI for HTTP, FTP and SMB data sources.

9.1.4 SAP Connections

The following tables list the parameters that describe a connection to the following SAP data sources:

- SAP ERP systems through Connection Server

Parameter Name	Description
CONNECTIVITY_TYPE	Value is relational.
DBMS	Type is String. Value is one of the following: • mySAP ERP 2004 • SAP ERP 6 • SAP R/3 Release 4
NETWORK_LAYER	Value is SAP Java Connector (SAP JCo).
AUTHENTICATION_MODE	Type is enum. Possible values are the following: • FIXED • MAPPING • SSO
USER_NAME	Type is String.
PASSWORD	Type is String.
LANGUAGE	Locale, for example en_US.
SAP_SSO_USE_SNC	Values are True or False.
SAP_SAVE_LANGUAGE	Values are True or False.
SAP_CLIENT_NUMBER	Consists of three integers. Type is String.
SAP_SYSTEM_ID	Consists of three letters. Type is String.
SAP_SERVER_TYPE	Type is enum. Possible values are: • APPLICATION_SERVER • MESSAGE_SERVER
SAP_SYSTEM_NUMBER	Consists of two letters. Type is String.
SAP_APPLICATION_SERVER_NAME	Type is String. Possible values are the following: • Application server name if server type is APPLICATION_SERVER • An empty string otherwise

Parameter Name	Description
SAP_MESSAGE_SERVER_NAME	Type is String. Possible values are the following: • Message server name if server type is MESSAGE_SERVER • An empty string otherwise
SAP_GROUP_NAME	Type is String. Possible values are the following: • Group name if server type is MESSAGE_SERVER • An empty string otherwise

- SAP HANA data sources

Parameter Name	Description
CONNECTIVITY_TYPE	Value is relational or Olap.
DBMS	Type is String. Value is SAP HANA database 1.0.
NETWORK_LAYER	Possible values are the following: • JDBC Drivers • ODBC Drivers • SAP Hana Client for OLAP
AUTHENTICATION_MODE	Type is enum. Possible values are the following: • FIXED • MAPPING • SSO
USER_NAME	Type is String.
PASSWORD	Type is String.
DATASOURCE	Type is String. Used for the following connections: • ODBC connections • JDBC connections with multiple servers
SAP_HANA_SERVER_TYPE	Type is enum. Used for JDBC connections. Possible values are the following: • SINGLE_SERVER • MULTI_SERVER
SAP_HANA_HOST_NAME	Type is String. Used for JDBC and OLAP connections.
SAP_HANA_INSTANCE_NUMBER	Consists of two numbers. Type is String. Used for JDBC and OLAP connections.
USE_SSL	Type is Boolean. Used for JDBC and OLAP connections.
LANGUAGE	Locale, for example en_US. Used for OLAP connections.

Parameter Name	Description
AUTO_RECONNECT	Type is Boolean. Used for OLAP connections.
FETCH_SIZE	Type is Integer. Used for OLAP connections.

- SAP BW through the Data Federator connector

Parameter Name	Description
CONNECTIVITY_TYPE	Value is `relational`.
DBMS	Value is `Data Federator Server XI R4`.
NETWORK_LAYER	Value is `SAP Java Connector (SAP JCo)`.
AUTHENTICATION_MODE	Type is enum. Possible values are the following: • FIXED • MAPPING • SSO
USER_NAME	Type is String.
PASSWORD	Type is String.
LANGUAGE	Language, for example EN.
SAP_SSO_USE_SNC	Values are `True` or `False`.
SAP_SAVE_LANGUAGE	Values are `True` or `False`.
SAP_CLIENT_NUMBER	Consists of three integers. Type is String. Made of 3 integers.
SAP_SYSTEM_ID	Consists of three letters. Type is String.
SAP_SERVER_TYPE	Type is enum. Possible values are APPLICATION_SERVER or MESSAGE_SERVER.
SAP_SYSTEM_NUMBER	Consists of two integers. Type is String.
SAP_APPLICATION_SERVER_NAME	Type is String. Possible values are the following: • Application server name if server type is APPLICATION_SERVER • An empty string otherwise
SAP_MESSAGE_SERVER_NAME	Type is String. Possible values are the following: • Message server name if server type is MESSAGE_SERVER • An empty string otherwise

Parameter Name	Description
SAP_GROUP_NAME	Type is String. Possible values are the following: • Group name if server type is MESSAGE_SERVER • An empty string otherwise
SAP_USE_GATEWAY	Type is Boolean.
SAP_GATEWAY_HOST_NAME	The name of the machine hosting the SAP BW gateway. Type is String.
SAP_GATEWAY_SERVICE_NAME	The name or port number of the SAP BW gateway service. Type is String.
SAP_USE_PROGRAM_MAPPING	Type is Boolean.
SAP_PROGRAM_MAPPING	The program IDs for the callback that SAP BW uses to contact Data Federator. Type is String.
OLAP_CUBE	The InfoProvider name.

- SAP BW 7.x connection through the SAP BICS Client middleware driver

Parameter Name	Description
CONNECTIVITY_TYPE	Value is `Olap`.
DBMS	Value is `SAP Netweaver BW 7.x`.
NETWORK_LAYER	Value is `SAP BICS Client`.
AUTHENTICATION_MODE	Type is enum. Possible values are the following: • `FIXED` • `MAPPING` • `SSO` • `PROMPTED`
USER_NAME	Type is String.
PASSWORD	Type is String.
SAP_CLIENT_NUMBER	Consists of three integers. Type is String.
LANGUAGE	Locale, for example `en_US`.
SAP_SAVE_LANGUAGE	Values are `True` or `False`.
SAP_SYSTEM_ID	Consists of three letters. Type is String.

Parameter Name	Description
SAP_SERVER_TYPE	Type is enum. Possible values are the following: • APPLICATION_SERVER • MESSAGE_SERVER
SAP_SYSTEM_NUMBER	Consists of two letters. Type is String.
SAP_APPLICATION_SERVER_NAME	Type is String. Possible values are the following: • Application server name if server type is APPLICATION_SERVER • An empty string otherwise
SAP_MESSAGE_SERVER_NAME	Type is String. Possible values are the following: • Message server name if server type is MESSAGE_SERVER • An empty string otherwise
SAP_GROUP_NAME	Type is String. Possible values are the following: • Group name if server type is MESSAGE_SERVER • An empty string otherwise
OLAP_PATH	Value depends on the type of data source: • An empty string if the data source is a server • The infoarea technical name if it is an InfoProvider • The InfoProvider technical name if it is a BEx Query
OLAP_CUBE	Value depends on the type of data source: • An empty string if it is a server • The InfoProvider technical name if it is an InfoProvider • The Info Query technical name if it is a BEx Query

9.1.5 OLAP Connections

The following table lists the parameters that describe an MSAS connection through an XMLA driver.

Parameter Name	Description
CONNECTIVITY_TYPE	Value is Olap.
DBMS	Value is one of the following: • Microsoft Analysis Services 2008 • Microsoft Analysis Services 2012
NETWORK_LAYER	Value is XMLA.

Parameter Name	Description
AUTHENTICATION_MODE	Type is enum. Possible values are the following: • FIXED • MAPPING • SSO • PROMPTED
USER_NAME	Type is String.
PASSWORD	Type is String.
OLAP_SERVER_NAME	Value is a URL.

The following table lists the parameters that describe an Essbase connection through the Essbase middleware.

Parameter Name	Description
CONNECTIVITY_TYPE	Value is `Olap`.
DBMS	Value is `Hyperion Essbase 11.x`.
NETWORK_LAYER	Value is `Essbase API for Analysis`.
AUTHENTICATION_MODE	Type is enum. Possible values are the following: • FIXED • MAPPING • SSO • PROMPTED
USER_NAME	Type is String.
PASSWORD	Type is String.
OLAP_SERVER_NAME	Type is String. Value is a URL.
OLAP_PATH	Type is String.
OLAP_CUBE	Type is String.
LANGUAGE	Locale, for example `en_US`.

9.2 Predefined Query Script Property Reference

The following table lists the predefined query script properties that can apply to a data foundation or business layer. The default value is the one that displays in the information design tool. The table also shows which

property is mandatory in a data foundation or business layer. See the *Information Design Tool User Guide* for comprehensive definitions.

Name	Type	Default Value	In Data Foundation	Mandatory in Data Foundation	In Business Layer	Mandatory in Business Layer
ANSI92	Boolean	No	✓	YES	✓	NO
AUTO_UPDATE_QUERY	Boolean	No	✓	YES	✓	YES
BEGIN_SQL	String	*no value*	✓	YES	✓	NO
BLOB_COMPARISON	Boolean	No	✓	YES	✓	NO
BOUNDARY_WEIGHT_TABLE	Integer	-1	✓	YES	✓	NO
COMPARE_CONTEXTS_WITH_JOINS	Boolean	Yes	✓	YES	✓	NO
CUMULATIVE_OBJECT_WHERE	Boolean	No			✓	YES
DISABLE_ARRAY_FETCH_SIZE_OPTIMIZATION	Boolean	No			✓	YES
DISTINCT_VALUES	String	DISTINCT			✓	YES
END_SQL	String	*no value*	✓	YES	✓	NO
EVAL_WITHOUT_PARENTHESIS	Boolean	No			✓	YES
FILTER_IN_FROM	Boolean	No			✓	NO
FORCE_SORTED_LOV	Boolean	No	✓	YES	✓	YES
GROUPBY_PRIMARY_KEY	Boolean	Yes			✓	NO
INNERJOIN_IN_WHERE	Boolean	No	✓	NO	✓	NO
JOIN_BY_SQL	Boolean	No	✓	YES	✓	NO
MAX_INLIST_VALUES	Integer	-1	✓	YES	✓	NO
NO_CUSTOM_SQL_CHECK	Boolean	No			✓	NO
REPLACE_COMMA_BY_CONCAT	Boolean	No	✓	YES	✓	NO

Name	Type	Default Value	In Data Foundation	Mandatory in Data Foundation	In Business Layer	Mandatory in Business Layer
SELFJOINS_IN_WHERE	Boolean	No	✓	NO	✓	NO
SHORTCUT_BEHAVIOR	String	ShortestPath	✓	YES	✓	NO
SMART_AGGREGATE	Boolean	No			✓	NO
THOROUGH_AGGREGATE_AWARE	Boolean	Yes			✓	NO
THOROUGH_PARSE	Boolean	No	✓	YES	✓	NO
TRUST_CARDINALITIES	Boolean	No	✓	YES	✓	YES
UNICODE_STRINGS	Boolean	No	✓	YES	✓	NO
USE_ENHANCED_QUERY_STRIPPING	Boolean	No	✓	YES	✓	YES

9.3 Identifiable Object Reference

This section lists the interfaces and their subinterfaces that can be identifiable using an ID. All objects represented here inherit from the `Identifiable` interface.

Interface	Subinterface
Column	• CalculatedColumn • DatabaseColumn • InputColumn
Context	N/A
Join	SQLJoin
Lov	• BusinessHierarchicalLov • BusinessQueryLov • SQLQueryLov • StaticLov
Parameter	N/A

Interface	Subinterface
`Table`	• `AliasTable` • `DatabaseTable` • `DerivedTable` • `FederatedTable`
`BlItem`	• `Attribute` • `BlContainer` • `BusinessFilter` • `BusinessObject` • `Dimension` • `Filter` • `Folder` • `Measure` • `NativeRelationalFilter` • `RootFolder`
`BusinessLayerView`	N/A
`NavigationPath`	N/A

9.4 ValueFormat String Pattern Reference

You use the `ValueFormat` object to define a custom data format. The BI Semantic Layer Java SDK provides a series of numeric and date-time patterns to help you build the string part of `ValueFormat`. See the *Information Design Tool User Guide* for the definitions of the numeric and date-time formats.

Numeric Patterns [page 99]
This is the list of numeric patterns used to represent numeric characters in custom data formats.

Date-Time Patterns [page 100]
This is the list of date-time patterns used to represent dates and times in custom data formats.

9.4.1 Numeric Patterns

This is the list of numeric patterns used to represent numeric characters in custom data formats.

Category	Pattern	Description
Signs	-	Negative sign if the value is negative. Nothing if the value is positive or zero.
	`<positive sign>`	Negative sign if the value is negative. Positive sign if the value is positive or zero.

Category	Pattern	Description
	`(`	Open parenthesis if the value is negative. Nothing if the value is positive or zero.
	`)`	Closing parenthesis if the value is negative. Nothing if the value is positive or zero.
Digits	`#`	Optional digit. Displays digit only if significant.
	`0`	Mandatory digit. Displays digit if significant, otherwise displays zero.
Separators	`.`	The symbol used to separate the integer and decimal parts of the number. The symbol used is determined by the locale. The decimal separator can be used only once in an expression.
	`,`	By default, digits are grouped using the rule and the separator defined by the locale. The grouping symbol can be used only once in an expression. It must appear before the decimal separator.
Exponents	`E+`	Exponent sign in upper case, always signed. Can be used only once in an expression.
	`E-`	Exponent sign in upper case, signed only if the value is negative. Can be used only once in an expression.
	`e+`	Exponent sign in lower case, always signed. Can be used only once in an expression.
	`e-`	Exponent sign in lower case, signed only if the value is negative. Can be used only once in an expression.
Percent	`[%]`	The value multiplied by 100.
	`<times 100 and sign>`	The value multiplied by 100 followed by the percent sign (%). Can be used only once in an expression.
Boolean	`<bool>`	Localized value of true if the numerical value is not zero; localized value of false if the numerical value is zero.
	`<true>`	Always displays the localized value of true.
	`<false>`	Always displays the localized value of false.

9.4.2 Date-Time Patterns

This is the list of date-time patterns used to represent dates and times in custom data formats.

Category	Pattern	Description
Day	`dd`	Day in the month with two digits from 01 to 31
	`d`	Day in the month with one or two digits from 1 to 31
	`dddd`	Day name according to the locale, for example, Monday

Category	Pattern	Description
	`ddd`	Short day name with capitalization according to the locale, for example, Mon
	`eee`	Day in the year with three digits from 001 to 366
	`ee`	Day in the year with two or three digits from 01 to 366
	`e`	Day in the year with one, two, or three digits from 1 to 366
	`F`	Day of the week in the month according to the locale, for example, 3 for the 3rd Monday of June
	DDDD	Day name in upper case, for example, MONDAY
	`<lower day name>`	Day name in lower case, for example, monday
	`Dddd`	Capitalized day name, for example, Monday
	`DDD`	Short day name in upper case, for example, MON
	`<lower abb day name>`	Short day name in lower case, for example, mon
	`Ddd`	Capitalized short day name, for example, Mon
Month	`MM`	Month in the year with two digits from 01 to 12
	`M`	Month in the year with one or two digits from 1 to 12
	`mmmm`	Month name with capitalization according to the locale, for example, June
	`mmm`	Short month name with capitalization according to the locale, for example, Jun
	`MMMM`	Month name in upper case, for example, JUNE
	`<lower month name>`	Month name in lower case, for example, june
	`Mmmm`	Capitalized month name, for example, June
	`MMM`	Short month name in upper case, for example JUN
	`<lower abb month name>`	Short month name in lower case, for example, jun
	`Mmm`	Capitalized short month name, for example, Jun
Year and Era	`yy`	Year with two digits from 00 to 99
	`yyyy`	Year with four digits from 0000 to 9999
	`rr`	Japanese Imperial period and year number
	`g`	Japanese Imperial period (English abbreviated) and year number, for example, H20
	`jyy`	Japanese Imperial year number with two digits
	`jy`	Japanese Imperial year number with one or two digits
	`gg`	Japanese Imperial period
	`r`	Deprecated. Returns the same result as `jyy` pattern

Category	Pattern	Description
	`G`	Era abbreviation, for example, AD or BC
Week	`W`	Week in the month with one digit from 1 to 6
	`ww`	Week in the year (ISO week) with two digits from 01 to 53
	`w`	Week in the year (ISO week) with one or two digits from 1 to 53
	`iyyy`	ISO year number (consistent with ISO week) with four digits from 0000 to 9999
	`iy`	ISO year number (consistent with ISO week) with two digits from 00 to 99
Quarter and Semester	`q`	Quarter number with one digit from 1 to 4
	`qq`	Quarter short name from Q1 to Q4
	`qqq`	Quarter name from 1st quarter to 4th quarter
	`p`	Semester number from 1 to 2
Hour	`H`	Hour in 24-hour format with two digits from 00 to 23
	`HH`	Hour in 24-hour format with one or two digits from 0 to 23
	`h`	Hour in 12-hour format with two digits from 01 to 12
	`hh`	Hour in 12-hour format with one or two digits from 1 to 12
	`k`	Hour in 24-hour format with two digits from 01 to 24
	`kk`	Hour in 24-hour format with one or two digits from 1 to 24
	`K`	Hour in 12-hour format with two digits from 00 to 11
	`KK`	Hour in 12-hour format with one or two digits from 0 to 11
Minute	`mm`	Minutes with two digits from 00 to 59
	`m`	Minutes with one or two digits from 0 to 59
Second and sub-second	`ss`	Seconds with two digits from 00 to 59
	`s`	Seconds with one or two digits from 0 to 59
	`SSS`	Milliseconds with three digits from 000 to 999
	`SS`	Hundredths of a second with two digits from 00 to 99
	`S`	Tenths of a second with one digit from 1 to 9
Time Zone	`z`	The offset from Coordinated Universal Time, for example, GMT+00:00
AM/PM	`a`	Morning/afternoon abbreviation, capitalized according to locale, for example, AM or PM. Recommended.
	`A`	Morning/afternoon abbreviation in upper-case, for example, AM or PM
	`<lower am/pm sign>`	Morning/afternoon abbreviation in lower-case, for example, am or pm

Category	Pattern	Description
	`<cap am/pm sign>`	Capitalized morning/afternoon abbreviation, for example, Am or Pm. Not recommended.
Separator	`<date separator>`	Deprecated. This pattern was used as a date separator in Desktop Intelligence and is not recommended. Type the character you wish to use as a date separator directly into the format description, or use a default format.
	`<time separator>`	Deprecated. This pattern was used as a time separator in Desktop Intelligence and is not recommended. Type the character you wish to use as a time separator directly into the format description, or use a default format.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

© 2024 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.

THE BEST RUN SAP