PUBLIC

SAP BusinessObjects Business Intelligence platform

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

Business Intelligence Platform Java SDK Developer Guide

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

THE BEST RUN SAP

Content

1 Document History. 6

2 Start here. 7

2.1 What's new in the SAP BusinessObjects Business Intelligence platform Java SDK. 7

JAR file changes. 9

2.2 Migrating your application. 9

Deprecated APIs. 9

Changes to auditing. 12

3 SDK fundamentals. 15

3.1 Class overview. 15

3.2 Common workflows. 19

To create an InfoObject. 19

To delete an InfoObject. 21

To modify an InfoObject. 22

3.3 Using the query language to retrieve objects. 24

To run a query. 25

Object types. 26

Accessing properties in query results. 29

The SELECT clause. 32

The FROM clause. 34

The WHERE clause. 34

The ORDER BY clause. 42

Query string best practices. 43

3.4 Using URI queries to retrieve objects. 47

Prior knowledge 48

To run a URI path query 48

Object types. 50

Protocols. 52

Operators 56

Relationships 60

URI query best practices. 64

3.5 Working with multilingual content. 66

3.6 Best practices for this SDK. 67

Cleanup unused sessions and resources. 68

Store the IEnterpriseSession object. 68

Determine supported interfaces. 68

Catch and handle SDK exceptions. . . . 69
Keep the CMS in a consistent state. . . . 70
3.7 Serializing objects in horizontal or vertical clusters. . . . 71
3.8 Use the same CMS to read and write an object. . . . 73

4 Setting up the development environment. . . . 75
4.1 Setting your target JVM for compiling. . . . 75
4.2 Web application setup. . . . 75
To set up your web application. . . . 75
Configuring your web.xml file. . . . 76
JAR files needed for deployment of SAP BusinessObjects software. . . . 83
4.3 Setting up your SAP BusinessObjects Business Intelligence platform environment. . . . 89
Required SAP BusinessObjects Business Intelligence platform components. . . . 89
Configuring SAP BusinessObjects Business Intelligence platform settings. . . . 89
4.4 Setting up your Report Application Server environment . . . 90
Required Report Application Server components. . . . 90
Specifying Report Application Server locations. . . . 92
Report location. . . . 92
Horizontal and vertical clustering. . . . 94
4.5 Setting up your SAP Crystal Reports viewers. . . . 95
Required viewers components. . . . 95
Side-by-side installation of the viewers SDK. . . . 95
4.6 Directories. . . . 96

5 Using the SDK. . . . 97
5.1 Authentication. . . . 97
Authentication basics . . . 97
Logging on to the CMS. . . . 99
Logging off and freeing up licenses. . . . 106
5.2 Enhanced Credential Mapping. . . . 107
5.3 Security. . . . 108
Users and Groups. . . . 109
Setting Rights. . . . 113
5.4 Scheduling. . . . 141
Scheduling workflow. . . . 142
Output destinations. . . . 142
To schedule a report. . . . 150
To schedule a report with custom values. . . . 152
To schedule a report containing specific parameter values. . . . 154
To schedule a report using calendars. . . . 157
To schedule a report using events. . . . 160
5.5 Events and alerting. . . . 162

To create a schedule event. . . . 162
To create a file event. . . . 164
To create a custom event. . . . 165
To enable alerting for an event. . . . 166
To subscribe to an event. . . . 167
To trigger a custom event. . . . 169
To view alert notifications. . . . 170
5.6 Publications. . . . 171
Publishing workflow. . . . 172
To create a publication. . . . 172
Adding documents to a publication. . . . 174
Adding recipients to a publication. . . . 179
Personalizing publication documents. . . . 185
Configuring destinations and output formats. . . . 193
Troubleshooting publication exceptions. . . . 196
Publication extensions. . . . 198
5.7 Server administration. . . . 209
Server Intelligence architecture. . . . 210
Server Intelligence components. . . . 210
Managing servers. . . . 212
Adding servers. . . . 219
Configuring services and service containers. . . . 225
Managing server groups. . . . 233
5.8 Federation. . . . 235
Configuring federation. . . . 236
Classes used for federation. . . . 236
To create a replication list. . . . 237
To create a remote connection. . . . 239
To create and schedule a replication job. . . . 241
Remote scheduling. . . . 243
Dependency types. . . . 245
5.9 Monitoring. . . . 246
Adding a new probe. . . . 246
5.10 Platform Search. . . . 252
Classes used for Platform Search. . . . 252
To retrieve search results . . . 252
5.11 Auditing . . . 257
To retrieve an auditing event. . . . 258
To retrieve an auditing event detail. . . . 260
To change the current auditing level. . . . 261
To enable an auditing event. . . . 263

To enable an auditing event detail. . . . 265
To restore custom auditing events. . . . 266
To retrieve auditing categories. . . . 269
To retrieve a server auditing metric. . . . 270
To retrieve auditing data store connection information. . . . 271
To change the duration for which auditing events are stored. . . . 274
To enhance traceability for auditing through custom applications. . . . 275
Custom application CUID values and names. . . . 276
5.12 Localization and language packs. . . . 278
To set the preferred viewing locale. . . . 278
To display the current preferred viewing locale. . . . 279
To display installed language packs. . . . 280
To display all supported languages. . . . 281
5.13 Cryptographic key management. . . . 282
Cryptographic key states. . . . 282
To create a new cryptographic key. . . . 283
To re-encrypt data with a new key. . . . 284
To delete a cryptographic key. . . . 286
5.14 Business Intelligence Archive (BIAR) 287
Import considerations. . . . 289
To export InfoObjects to a BIAR file. . . . 289
To import InfoObjects from a BIAR file. . . . 292
To perform a live-to-live transfer. . . . 294
To perform callback. . . . 296

6 References. . . . 298
6.1 Processing Extension API Reference. . . . 298
Creating an extension. . . . 298
Loading an extension. . . . 298
Processing Extension API. . . . 300
Examples. . . . 333

1 Document History

This table provides an overview of the most important document changes.

Version	Date	Description
SAP BusinessObjects Business Intelligence platform 4.3 SP2	December, 2021	Updated JAR files needed for deployment of SAP BusinessObjects software [page 83].
SAP BusinessObjects Business Intelligence platform 4.3	June, 2020	Initial Release

2 Start here

The SAP BusinessObjects Business Intelligence platform 4.1 Java SDK allows you to build applications that interface directly with the BI platform to perform tasks such as user authentication, scheduling, publications, and server management.

Start using the SDK

- Using the SDK [page 97]
  Understand the common use cases of this SDK with tasks and code examples.
- Setting up the development environment [page 75]
  Review which JAR files to deploy to your development environment and how to configure your web application to work with the SDK.

Learn about the SDK

- What's new in the SAP BusinessObjects Business Intelligence platform Java SDK [page 7]
  Learn about new features of the SDK in this release.
- Migrating your application [page 9]
  Learn what APIs have been deprecated in this release and how changes may affect your existing applications.
- SDK fundamentals [page 15]
  Explore the fundamentals of the SDK, including an overview of the key classes and interfaces.

2.1 What's new in the SAP BusinessObjects Business Intelligence platform Java SDK

This section provides a brief summary of the new features and enhancements that are offered with the SAP BusinessObjects Business Intelligence platform 4.1 Java SDK.

Server auditing

There is a completely new framework for both server and client auditing, which have been amalgamated. Auditing event objects are now organized and manipulated by a central interface in the repository. Auditing

events record data based on the auditing event details which are associated with them. These auditing details are configured separately and independently from the auditing events.

For more information, see Auditing [page 257].

Enabled auditing events are now managed through auditing levels, ranging from *OFF* to *COMPLETE*. There is also a *CUSTOM* auditing level which offers greater flexibility and customizability.

> **Note**
>
> To ensure your existing applications continue to work properly, see Changes to auditing [page 12].

Data security

In the BI platform, symmetric encryption keys known as cryptographic keys are used to encrypt and decrypt sensitive data.

In order to manage cryptographic keys, you must be logged on to the BI platform using an account that is a member of the Cryptographic Officers group. You can then use this SDK to programmatically create, revoke, and delete cryptographic keys.

For more information, see Cryptographic key management [page 282].

Alerting

The alerting feature enables you to send notifications to users and groups when an event occurs. To use the alerting feature, you enable alerting for an event and then subscribe a user or group to the event. When the event is triggered, the subscribed users and groups are automatically sent an alert notification message. By subscribing to alerting-enabled events, users are automatically sent an alert notification when the event occurs, and do not need to manually check that an event has occurred.

For example, you can use the alerting feature to notify users that a scheduled job has completed successfully, a document alert has been triggered, or a performance threshold has been crossed.

For more information, see Events and alerting [page 162]

Business Intelligence Archive (BIAR)

New APIs have been added that allow you to directly transfer objects from one Central Management Server (CMS) to another. You can programmatically export objects from a source from a source environment and then import those objects into a destination environment, without having to use a BIAR file. The `com.businessobjects.sdk.biar.ILiveToLivePipe` interface facilitates this transfer.

For more information, see Business Intelligence Archive (BIAR) [page 287] and To perform a live-to-live transfer [page 294].

Access to `CeProgID` and `CeKind` constants

Each plugin interface now contains constants for their respective `ProgID` and `CeKind` values. For example, instead of calling `CeKind.SERVER`, call `IServer.KIND`. Similarly, `CeProgID.SERVER` will be replaced by `IServer.PROGID`. The `CeProgID` and `CeKind` interfaces are deprecated.

For more information, see Deprecated APIs [page 9].

2.1.1 JAR file changes

This section lists deprecated and obsolete JAR files in this SDK as of this release compared with BI4.0. While deprecated items are supported for backwards compatibility, they are not included or recommended for use in the current version of the SDK.

Jar file changes from BI4.0 to BI4.1

New JAR files	Deprecated JAR files
`ebus405.jar`	`backport-util-concurrent-2.2.jar`
	`stax-api-1.0.1.jar`
	`wxts-asl-3.2.1.jar`

2.2 Migrating your application

This section contains information about changes to be aware of when you upgrade from an earlier version of this SDK.

2.2.1 Deprecated APIs

This section lists deprecated and obsolete APIs in the SDK as of this release. While deprecated items are supported for backwards compatibility, they are not recommended for use in the current version of the SDK.

> **Note**
>
> For a complete list of deprecated API members and their replacements, see the *SAP BusinessObjects Business Intelligence Platform Java API Reference*.

Deprecated Interfaces

Interface Name	Details
`com.businessobjects.sdk.plugin.desktop.auditeventinfo.IAuditDetail`	Replaced `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetail`.
`com.businessobjects.sdk.plugin.desktop.auditeventinfo.IAuditDetails`	Replaced `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetails`
`com.businessobjects.sdk.plugin.desktop.auditeventinfo.IAuditEvent`	Replaced `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvent`
`com.businessobjects.sdk.plugin.desktop.auditeventinfo.IAuditEventInfo`	Replaced `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
`com.businessobjects.sdk.plugin.desktop.auditeventinfo.IAuditEvents`	Replaced `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvents`
`com.crystaldecisions.sdk.plugin.authentication.secwinnt.IsecWinNT`	Windows NT authentication is no longer supported.
`com.crystaldecisions.sdk.plugin.CeKind`	Each plugin interface now contains a field that specifies the SI_KIND property for the appropriate object type. For example: • `IFolder.KIND` replaces `CeKind.FOLDER` • `IServer.KIND` replaces `CeKind.SERVER` • `IUser.KIND` replaces `CeKind.USER`
`com.crystaldecisions.sdk.plugin.CeProgID`	Each plugin interface now contains a field that specifies the SI_PROGID property for the appropriate object type. For example: • `IFolder.PROGID` replaces `CeProgID.FOLDER` • `IServer.PROGID` replaces `CeProgID.SERVER` • `IUser.PROGID` replaces `ProgID.USER`

Deprecated Methods

Method Name	Details
`com.businessobjects.publisher.distribution.IDistributionCompleteContext.getAdminLog`	Use `IDistributionCompleteContext.getAdminLogAdapter`.
`com.businessobjects.publisher.postprocessing.IPublicationPostProcessingContext.getSession`	
`com.businessobjects.sdk.plugin.desktop.common.IConfiguredService.getServiceStatus`	Use `IConfiguredService.getServiceStatusEnum` instead.

Method Name	Details
`com.businessobjects.sdk.plugin.desktop.common.IConfiguredService.setServiceStatus(int)`	Use `IConfiguredService.setServiceStatusEnum` instead.
`com.businessobjects.sdk.plugin.desktop.profile.IProfileValue.getFormula()`	Use `IProfileValue.getFormula(String)` instead.
`com.businessobjects.sdk.plugin.desktop.profile.IProfileValue.setFormula(String)`	Use `IProfileValue.setFormula(String, String)` instead.
`com.businessobjects.sdk.plugin.desktop.publication.IPublicationProfileTarget.setTargetID`	Use `setSourceDocumentID` instead.
`com.businessobjects.sdk.plugin.desktop.publication.IPublicationProfileTarget.setVariable`	Use `getVariableMappings` instead.
`com.crystaldecisions.sdk.plugin.desktop.common.IFormatInfo.getSourceDocumentKind`	It is no longer required to set source document kind.
`com.crystaldecisions.sdk.plugin.desktop.common.IFormatInfo.setSourceDocumentKind`	It is no longer required to set source document kind.
`com.crystaldecisions.sdk.plugin.desktop.event.IEventBase.getEventInterface`	Cast to the specific event type interface instead.
`com.crystaldecisions.sdk.plugin.desktop.event.IEventBase.getEventName`	Use `IInfoObject.getTitle` instead.
`com.crystaldecisions.sdk.plugin.desktop.event.IEventBase.setEventName`	Use `IInfoObject.setTitle` instead.
`com.crystaldecisions.sdk.plugin.desktop.event.IEventBase.getEventType`	Use `IInfoObject.getSpecificKind` or `IInfoObject.getSpecificProgID` instead.
`com.crystaldecisions.sdk.plugin.desktop.event.IEventBase.setEventType`	Create events according to their specific type instead, for example a `FileEvent` object.
`com.crystaldecisions.sdk.uri.PagingQueryOptions.isUriIsUnique`	This option is not used.
`com.crystaldecisions.sdk.uri.PagingQueryOptions.setUriIsUnique`	This option is not used.

Deprecated Fields

Field Name	Details
`com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo.GroupChoice.PREFERED`	Replaced by `ISchedulingInfo.GroupChoice.PREFERRED`.
`com.crystaldecisions.sdk.uri.PagingQueryOptions.SKIP_QUERY_URI_DECODE`	Query URI automatically avoids decoding inside the query section of the URI.

Obsolete Fields

Field Name	Details
`com.crystaldecisions.sdk.plugin.desktop.report.CeReportRightID.DOWNLOAD`	No replacement. The right no longer exists.

2.2.2 Changes to auditing

> **Note**
>
> Many of the interfaces used for server auditing prior to this release have been deprecated. Also, server and client auditing are now both handled by the same packages and need to be considered as a single interface. Be sure to carefully review the auditing sections in this guide and the *SAP BusinessObjects Business Intelligence Platform Java API Reference* to see how the changes will affect existing applications.

Several improvements in this release affect how server auditing is performed. Auditing events are no longer managed on a per-service, per-server basis. All events are controlled and configured centrally.

Auditing events and details

As a result of these changes, the classes and interfaces in the `com.crystaldecisions.sdk.plugin.desktop.auditeventinfo` package have been deprecated and replaced with those in the `com.crystaldecisions.sdk.plugin.desktop.auditeventinfo2` package. The new package now controls all auditing events and their associated details, as well as their organization. It is no longer necessary to visit individual services on each server and toggle auditing events through them. Events and details are now set globally.

Auditing levels

There are now pre-configured auditing levels which can be set to mirror popular custom sets of events. A new custom auditing level is available, which must be set if you want to enable or disable individual events.

The old packages remain bundled with the system. Applications created under the previous version's framework will still compile, but will not work as expected. It is recommended not to use applications developed under BusinessObjects Enterprise XI 3.x framework. Auditing applications will have to be re-written to employ the new `com.crystaldecisions.sdk.plugin.desktop.auditeventinfo2` package.

Enhanced traceability for custom applications

The Central Management Server (CMS) now reserves twenty-five CUID values to identify custom applications. You can pass one of these CUID value to the `IEnterpriseLogonInformation.setClientType` method,

and use one of the overloaded logon methods that accept an `IEnterpriseLogonInformation` object to authenticate users in your application. All auditing events collected by the system will also record the name and CUID of the custom application that triggered the event. Some administrators will find this method useful for enhanced traceability. A special string is used if no application name is supplied.

Example

To enable an event in BusinessObjects Enterprise XI 3.x , it was necessary to isolate a specific service on a specific server, and then make changes to the service's property bag. The first example shows how to enable the *Job Scheduling Succeeded* auditing event (ID 327681) for the Crystal Reports Scheduling service on the Crystal Reports Job Server.

> **Note**
>
> This operation is shown under the BusinessObjects Enterprise XI 3.x framework, and will not work on SAP BusinessObjects Business Intelligence platform 4.1 installations. It is provided for comparative purposes only.

```
void enableEvent31() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseSession enterpriseSession = sessionManager.logon("username",
"password", "MyCMS:6400", "secEnterprise");
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID, SI_HOSTED_SERVICES FROM CI_SYSTEMOBJECTS
WHERE SI_KIND = '" +
    CeKind.SERVER + "' AND SI_DESCRIPTION = 'Crystal Reports Job Server'";
  IServer server = (IServer) infoStore.query(serverQuery).get(0);
  String serviceQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '" +
          CeKind.SERVICE + "' AND SI_CUID = '" +
CeSecurityCUID.ServerIntelligence.CR_SCHEDULING_SERVICE + "'";
  IService service = (IService) infoStore.query(serviceQuery).get(0);

  IConfiguredServices cfgServices = server.getHostedServices();
  IConfiguredService cfgService = cfgServices.get(service.getID());
  Set enabledAuditEvents = cfgService.getEnabledAuditEvents();

  enabledAuditEvents.add(327681);
  server.save();
}
```

To enable an event in SAP BusinessObjects Business Intelligence platform 4.1, simply retrieve the `IAuditEventInfo` object, and adjust the set of enabled auditing events as needed. The following example adds the *Retrieve* event (ID 1013) to the set of enabled auditing events on the entire system.

> **Note**
>
> This example also demonstrates use of the new logon method which identifies the custom application as "Custom Application 1" (`CUID = AZ4HLbS0loRMpE2twk442RU`). All SAP BusinessObjects Business Intelligence platform 4.1 server events that are triggered as a result of this session will record this name and CUID along with the event details. This allows you to identify events that are triggered by user actions in this custom application. This task also assumes that you have already set your current auditing level to custom.

```
void enableEvent40() throws SDKException
```

```
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseLogonInformation logonInfo = sessionManager.createLogonInfo();
  logonInfo.setClientType("AZ4HLbS0loRMpE2twk442RU");

  IEnterpriseSession enterpriseSession = sessionManager.logonEx("username",
"password", "MyCMS:6400", "secEnterprise", logonInfo);

  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_ENABLED_AUDIT_EVENTS FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);

  Set enabledAuditEvents = auditEventInfo.getEnabledEventTypeIDs();
  enabledAuditEvents.add(1013);
  infoStore.commit(infoObjects);
}
```

Both examples use the `ISessionMgr` interface to get an `IInfoStore` object, and then query the object for an item to manipulate. In SAP BusinessObjects Business Intelligence platform 4.1, it is only necessary to make one query, as the auditing interface is now more exposed. This results in fewer method calls to perform the same operations.

Related Information

Auditing [page 257]
To change the current auditing level [page 261]
To enhance traceability for auditing through custom applications [page 275]
Server administration [page 209]

3 SDK fundamentals

In this section, you explore the fundamentals of this SDK, including an overview of the key classes and interfaces, common workflows you will encounter, and the basics of how to interact with objects in the Central Management Server (CMS).

3.1 Class overview

The SAP BusinessObjects Business Intelligence platform 4.1 SDK manages objects stored in the Central Management Server (CMS). There are three fundamental interfaces that communicate and interact with the CMS:

Interface	Package
`IInfoObject`	`com.crystaldecisions.sdk.occa.infostore`
`IInfoObjects`	`com.crystaldecisions.sdk.occa.infostore`
`IInfoStore`	`com.crystaldecisions.sdk.occa.infostore`

The `IInfoObject` and `IInfoObjects` interfaces allow you to manipulate CMS objects, referred to as InfoObjects. The `IInfoStore` interface is used to communicate InfoObject requests and changes to the CMS. Understanding how these interfaces interact with the CMS is key to using the SDK.

InfoObjects

The InfoObject is a generic model that represents an information entity in the CMS. Everything that can be managed in the system by the SDK is represented by an InfoObject, including Crystal reports, Web Intelligence documents, folders, users, servers, and so on.

Each InfoObject is a record in the CMS database that contains many fields, referred to as properties. Examples of properties include an InfoObject's identifier, name, and type. The state and behavior of the system is controlled by accessing and modifying the properties of InfoObjects with the SDK.

Individual InfoObjects can be added, retrieved, updated, or scheduled by the `IInfostore` interface only through an `IInfoObjects` collection. `IInfoObject` instances cannot be retrieved from or committed to the CMS directly.

The InfoStore service

The `IInfoStore` interface acts as a controller, or gateway, to InfoObjects in the CMS. The `IInfoStore` interface is used to communicate messages between your application and the CMS to create, retrieve, and commit (through an `IInfoObjects` collection) all instances of `IInfoObject` that are stored in the CMS repository.

To create a connection to the InfoStore service, you must log on to a valid session (`com.crystaldecisions.sdk.framework.IEnterpriseSession`). The following example assumes that the variable `enterpriseSession` represents a valid session:

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

Once you have established a connection to the InfoStore service, you can use the methods of the `IInfoStore` interface, such as `query`, `commit`, and `schedule`, to communicate with the CMS.

Plugins

To differentiate and effectively interact with the different types of InfoObjects, the system uses plugins. There is one plugin for every type of InfoObject in the CMS, and these plugins define the common properties of a particular type. There are many different types of plugins that define the behavior for different types of InfoObjects.

The SDK exposes methods that allow you to interact with InfoObjects of a particular type through an appropriate plugin interface. For example, the interface `com.crystaldecisions.sdk.plugin.desktop.user.IServer` exposes specific methods that allow you to interact with a deployed server in a cluster. You cast individual objects from an `IInfoObjects` collection to the correct plugin interface. This example assumes that the variable `infoobjects` is a collection of servers, and it uses the `get` method to return the first server:

```
IServer server = (IServer) infoobjects.get(0);
```

As with all plugin interfaces, `IServer` also extends the generic `IInfoObject` interface, where common methods of InfoObjects are exposed.

The following tables highlight typical plugins exposed by the SDK:

Authentication Plugins

Package	Description
`com.crystaldecisions.sdk.plugin.authentication.enterprise`	Contains the plugin interface `IsecEnterprise` that defines security options for native BI platform users.
`com.crystaldecisions.sdk.plugin.authentication.ldap`	Contains the plugin interface `IsecLDAP` that allows you to configure security options and map users from an LDAP directory server to BI platform users.

Package	Description
`com.crystaldecisions.sdk.plugin.authentication.secwinad`	Contains the plugin interface `IsecWinAD` that allows you to configure security options and map users from a Windows Active Directory server to BI platform users.

Destination Plugins

Package	Description
`com.crystaldecisions.sdk.plugin.destination.diskunmanaged`	Contains the plugin interface `IDiskUnmanaged` that allows you to define destination options for objects that are scheduled to a local or network file system.
`com.crystaldecisions.sdk.plugin.destination.ftp`	Contains the plugin interface `IFTP` that allows you to define destination options for objects that are scheduled to an FTP server.
`com.crystaldecisions.sdk.plugin.authentication.managed`	Contains the plugin interface `IManaged` that allows you to define destination options for objects that are scheduled to BI platform inboxes.
`com.crystaldecisions.sdk.plugin.destination.smtp`	Contains the plugin interface `ISMTP` that allows you to define destination options for objects that are scheduled to an email address relayed through an SMTP server.

Desktop Plugins

Desktop plugins represent the largest variety of InfoObjects. This table shows a sample of the many types of desktop plugins available in the SDK. For the complete list of packages, see the *SAP BusinessObjects Business Intelligence Platform 4.1 Java API Reference*.

Type	Package Examples	Description

Document InfoObjects	`com.businessobjects.sdk.plugin.desktop.flash`
	`com.businessobjects.sdk.plugin.desktop.fullclient`
	`com.businessobjects.sdk.plugin.desktop.webi`
	`com.businessobjects.sdk.plugin.desktop.xcelsius`
	`com.crystaldecisions.sdk.plugin.desktop.excel`
	`com.crystaldecisions.sdk.plugin.desktop.pdf`
	`com.crystaldecisions.sdk.plugin.desktop.powerpoint`
	`com.crystaldecisions.sdk.plugin.desktop.program`
	`com.crystaldecisions.sdk.plugin.desktop.report`
	`com.crystaldecisions.sdk.plugin.desktop.rtf`
	`com.crystaldecisions.sdk.plugin.desktop.txt`
	`com.crystaldecisions.sdk.plugin.desktop.word`
		These packages contain plugin interfaces that allow you to interact with documents such as Crystal reports, Web Intelligence documents, and PDFs. Managed documents physically reside on the File Repository Server (FRS) while being represented as an InfoObject in the CMS.

System InfoObjects	`com.businessobjects.sdk.plugin.desktop.publication`
	`com.businessobjects.sdk.plugin.desktop.profile`
	`com.crystaldecisions.sdk.plugin.desktop.server`
	`com.crystaldecisions.sdk.plugin.desktop.user`
		These packages contain plugin interfaces that allow you to interact with system objects such as users, servers, and publications. Unlike InfoObjects for documents, which represent files on the FRS, system objects are self-contained in the CMS.

Type	Package Examples	Description

Organizational InfoObjects	`com.businessobjects.sdk.plugin.desktop.category`
	`com.businessobjects.sdk.plugin.desktop.inbox`
	`com.crystaldecisions.sdk.plugin.desktop.folder`
	`com.crystaldecisions.sdk.plugin.desktop.objectpackage`
	`com.crystaldecisions.sdk.plugin.desktop.servergroup`
	`com.crystaldecisions.sdk.plugin.desktop.usergroup`
		These packages contain plugin interfaces that allow you to logically group and organize individual system and document InfoObjects. For example, folders to group documents or user groups to organize users.

3.2 Common workflows

To modify the BI platform with this SDK, you interact with InfoObjects and their properties in the Central Management System (CMS). There are a few workflows common to all major tasks:

- Creating a new InfoObject
- Deleting an InfoObject
- Modifying an InfoObject
- Scheduling an InfoObject

You will use these common workflows when restarting a server, delivering instances of reports to an inbox, applying security rights on a folder to a user, and so on.

This section shows examples of how to create, delete, and modify an InfoObject. For information about scheduling InfoObjects, see the *Scheduling* section in *Using the SDK*.

Related Information

Scheduling [page 141]

3.2.1 To create an InfoObject

Certain tasks, such as creating a new user or folder, require that you create new InfoObject records in the Central Management Server (CMS) database. You can use the `IInfoStore.newInfoObjectCollection`

factory method to request a new `IInfoObjects` collection that will contain your new objects. During this step, the CMS reserves a unique ID for each new object in the collection to be committed.

This example creates a folder InfoObject in the CMS.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Call the `newInfoObjectCollection` method of the `IInfoStore` interface to create an `IInfoObjects` collection.

   ```
   IInfoObjects newFolders = infostore.newInfoObjectCollection();
   ```

3. Call the `add` method of the collection to create an InfoObject, and cast the result to the appropriate plugin interface.

   The `add` method takes an `IFolder.FOLDER_KIND` field value representing the type of InfoObject to create. This example creates a folder and casts the result to the `IFolder` plugin interface.

   ```
   IFolder newFolder = (IFolder) newFolders.add(IFolder.FOLDER_KIND);
   ```

4. Set various properties of the new InfoObject, including its name, description, and parent folder information.

   In this example, the parent ID is set to zero, indicating a top-level folder under the root folder of the CMS.

   ```
   newFolder.setTitle(newInfoObjectName);
   newFolder.setDescription("This is a new folder InfoObject.");
   newFolder.setParentID(0);
   ```

5. Commit the new `IInfoObjects` collection containing the new InfoObject back to the CMS to save the changes.

   ```
   infostore.commit(newFolders);
   ```

Example

This example creates a folder InfoObject in the CMS:

```
void createInfoObject(IEnterpriseSession enterpriseSession, String
newInfoObjectName) throws SDKException
{
    IInfoStore infostore = (IInfoStore)
enterpriseSession.getService("InfoStore");
    IInfoObjects newFolders = infostore.newInfoObjectCollection();
    IFolder newFolder = (IFolder) newFolders.add(IFolder.FOLDER_KIND);

    newFolder.setTitle(newInfoObjectName);
    newFolder.setDescription("This is a new folder InfoObject.");
    newFolder.setParentID(0);

    infostore.commit(newFolders);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`

Related Information

To delete an InfoObject [page 21]
To modify an InfoObject [page 22]
Class overview [page 15]

3.2.2 To delete an InfoObject

This example deletes an existing folder from the CMS.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Query the CMS for an `IInfoObjects` collection containing the InfoObject you want to delete.

```
String query = "SELECT SI_ID FROM CI_INFOOBJECTS "
    + "WHERE SI_KIND='" + IFolder.FOLDER_KIND + "' "
    + "And SI_NAME='" + infoObjectName + "' And SI_INSTANCE=0";
IInfoObjects folders = infostore.query(query);
```

   This example uses the query language, which is a subset of the commonly used Structured Query Language (SQL), to request an InfoObject from the CMS. Querying for the correct objects and retrieving them from the CMS is fundamental to using the SDK successfully. For more information about the query language, see the section *Using the query language to retrieve objects* in *SDK fundamentals*.

3. Retrieve the InfoObject you want to delete from the collection.

```
IInfoObject folder = (IInfoObject) folders.get(0);
```

   > **Note**
   >
   > Your query may return multiple InfoObjects. This example gets the first InfoObject at the zeroth index of the collection.

4. Call the `delete` method of the `IInfoObjects` collection.

   Pass in the InfoObject you want to remove as the argument to the `delete` method.

```
folders.delete(folder);
```

   The `delete` method does not remove the object from the collection, but flags the object for deletion upon commitment to the CMS.

5. Commit the modified `IInfoObjects` collection back to the CMS to save the changes.

```
infostore.commit(folders);
```

Example

This example deletes an existing folder from the CMS.

```
void deleteInfoObject(IEnterpriseSession enterpriseSession, String
infoObjectName) throws SDKException
{
    IInfoStore infostore = (IInfoStore)
enterpriseSession.getService("InfoStore");
    String query = "SELECT SI_ID FROM CI_INFOOBJECTS "
        + "WHERE SI_KIND='" + IFolder.FOLDER_KIND + "' "
        + "And SI_NAME='" + infoObjectName + "' And SI_INSTANCE=0";
    IInfoObjects folders = infostore.query(query);
    IInfoObject folder = (IInfoObject) folders.get(0);
    folders.delete(folder);
    infostore.commit(folders);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`

Related Information

To create an InfoObject [page 19]
To modify an InfoObject [page 22]
Class overview [page 15]
Using the query language to retrieve objects [page 24]

3.2.3 To modify an InfoObject

Most tasks involve modifying properties of an InfoObject. Committing InfoObjects with modified property values allows you to affect the state of your BI platform installation.

This example modifies the name of a folder in the Central Management Server (CMS).

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Query the CMS for an `IInfoObjects` collection containing the InfoObject you want to modify.

```
String query = "SELECT SI_ID FROM CI_INFOOBJECTS "
        + "WHERE SI_KIND='" + IFolder.FOLDER_KIND + "' "
        + "And SI_NAME='" + oldName + "' And SI_INSTANCE=0";
    IInfoObjects folders = infostore.query(query);
```

This example uses the query language, which is a subset of the commonly used Structured Query Language (SQL), to request an InfoObject from the CMS. Querying for the correct objects and retrieving them from the CMS is fundamental to using the SDK successfully. For more information about the query language, see the section *Using the query language to retrieve objects* in *SDK fundamentals*.

3. Retrieve the InfoObject you want to modify from the collection.

```
IInfoObject folder = (IInfoObject) folders.get(0);
```

> **Note**
>
> Your query may return multiple InfoObjects. This example gets the first InfoObject at the zeroth index of the collection.

4. Call the `setTitle` method to modify the name of the InfoObject.

> **Note**
>
> The `setTitle` method is exposed through the `IInfoObject` base interface. To modify properties specific to a folder object type, you can cast the result to the `IFolder` plugin interface instead.

```
folder.setTitle(newName);
```

> **Note**
>
> You must change at least one property value before you commit the InfoObject to the CMS. The CMS does not allow you to commit an unchanged InfoObject.

5. Commit the `IInfoObjects` collection containing the modified InfoObject back to the CMS to save the changes.

```
infostore.commit(folders);
```

Example

This example modifies the name of a folder in the CMS.

```
void modifyInfoObject(IEnterpriseSession enterpriseSession, String oldName,
String newName) throws SDKException
{
    IInfoStore infostore = (IInfoStore)
enterpriseSession.getService("InfoStore");
    String query = "SELECT SI_ID FROM CI_INFOOBJECTS "
```

```
        + "WHERE SI_KIND='" + IFolder.FOLDER_KIND + "' "
        + "And SI_NAME='" + oldName + "' And SI_INSTANCE=0";
    IInfoObjects folders = infostore.query(query);

    IInfoObject folder = (IInfoObject) folders.get(0);
    folder.setTitle(newName);
    infostore.commit(folders);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`

Related Information

To create an InfoObject [page 19]
To delete an InfoObject [page 21]
Class overview [page 15]
Using the query language to retrieve objects [page 24]

3.3 Using the query language to retrieve objects

Use the `IInfoStore.query` method with a query language string to search for and retrieve a collection of objects stored in the Central Management Server (CMS) . For example, to retrieve the root-level folders of your BI platform installation, you can get an instance of the `IInfoStore` class and pass the following query string to its `query` method:

```
SELECT
    SI_ID, SI_NAME, SI_DESCRIPTION
FROM
    CI_SYSTEMOBJECTS
WHERE
    SI_KIND = 'Folder' AND
    SI_PARENTID = 4
```

The `query` method retrieves the objects from the repository and stores them in objects that you can access from your code. You can then modify the properties of these objects programmatically and save the changes back to the repository.

The query language is a subset of SQL, which is commonly used for retrieving data from relational databases. It has many of the same capabilities as SQL, but it does not support the more sophisticated features of SQL such as table joins and nested `SELECT` statements.

In this section, you learn how to use the query language to retrieve objects from the CMS repository. You learn about the most commonly used elements in a query and how SDK classes interact with the query language.

Related Information

To run a query [page 25]
Using URI queries to retrieve objects [page 47]

3.3.1 To run a query

To run a query against the Central Management Server (CMS), you must have a valid `IEnterpriseSession` object.

The following interfaces are used when querying the CMS:

- `IInfoStore`, which provides the methods for querying and retrieving objects from the repository.
- `IInfoObjects`, which contains the query results.
- `IInfoObject`, which contains an individual item in the query result set.

These classes are part of the `com.crystaldecisions.sdk.occa.infostore` package.

1. Call the `getService` factory method of the `IEntepriseSession` object, and set the argument to **`InfoStore`**. Cast the return object to `IInfoStore`.

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Create the query string.

   The query string consists of the following parts:

   - The `SELECT` clause, which specifies the object properties that are included in each object in the result set.
   - The `FROM` clause, which specifies the tables in the repository that contain the objects to retrieve.
   - The (optional) `WHERE` clause, which allows you to apply filters for more precise results.
   - The (optional) `ORDER BY` clause, which allows you to specify the order in which the results are returned.

   The following code creates a query string that can be used to retrieve all the Crystal Report objects in the repository.

   ```
   String queryString = "SELECT SI_NAME, SI_DESCRIPTION "
           + "FROM CI_INFOOBJECTS WHERE SI_KIND = 'CrystalReport'";
   ```

   > **Note**
   >
   > Query language keywords, property names, and table names are not case-sensitive. For example, `SELECT` is the same as `select`.

3. Pass the query string to the `query` method of the `IInfoStore` object.

   ```
   IInfoObjects infoObjects = infoStore.query(queryString);
   ```

The `query` method returns an `IInfoObjects` collection that contains the results of the query.

4. Use the `iterator` method of the `IInfoObjects` collection to get an iterator object.

   You can use the iterator to retrieve individual objects from the collection. Cast the objects returned by the iterator to `IInfoObject`.

```
Iterator infoObjectsIter = infoObjects.iterator();
while(infoObjectsIter.hasNext()) {
    IInfoObject folder = (IInfoObject) infoObjectsIter.next();
    ...
}
```

Example: Example

The following example retrieves all the Crystal Report objects from the repository and prints their names. It is assumed that `<enterpriseSession>` is a valid `IEnterpriseSession` object.

```
import com.crystaldecisions.sdk.framework.IEnterpriseSession;
import com.crystaldecisions.sdk.occa.infostore.IInfoStore;
import com.crystaldecisions.sdk.occa.infostore.IInfoObjects;
import com.crystaldecisions.sdk.occa.infostore.IInfoObject;
import java.util.Iterator;
...
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
String queryString = "SELECT SI_NAME, SI_DESCRIPTION "
        + "FROM CI_INFOOBJECTS WHERE SI_KIND = 'CrystalReport'";
IInfoObjects infoObjects = infoStore.query(queryString);
Iterator infoObjectsIter = results.iterator();
while(infoObjectsIter.hasNext()) {
    IInfoObject folder = (IInfoObject) infoObjectsIter.next();
    System.out.println(folder.getTitle());
}
```

Related Information

Object types [page 26]
Accessing properties in query results [page 29]

3.3.2 Object types

The following table lists common object types in the CMS repository. Use this table to identify the Java interface name and package for each object type and to determine which `SI_KIND` values and repository table names to use in your queries.

> **Note**
>
> Refer to each plugin interface in the *SAP BusinessObjects Business Intelligence Platform Java API Reference* for a comprehensive list of all `SI_KIND` constant field values. For example, the

> `com.crystaldecisions.sdk.plugin.desktop.server.IServer` interface has an `IServer.KIND` field.

Class	SI_KIND identifier	Repository table	Package
ICalendar	Calendar	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.calendar
IConnection	Connection	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.connection
ICustomRole	CustomRole	CI_SYSTEMOBJECTS	com.businessobjects.sdk.plugin.desktop.customrole
IDiskUnmanaged	DiskUnmanaged	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.destination.diskunmanaged
IEvent	Event	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.event
IExcel	Excel	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.excel
IFolder	Folder	CI_INFOOBJECTS CI_SYSTEMOBJECTS CI_APPOBJECTS	com.crystaldecisions.sdk.plugin.desktop.folder
IFTP	Ftp	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.destination.ftp
IFullClient	FullClient	CI_INFOOBJECTS	com.businessobjects.sdk.plugin.desktop.fullclient
IHyperlink	Hyperlink	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.hyperlink
ILicenseKey	LicenseKey	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.licensekey
IManaged	Managed	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.destination.managed

Class	SI_KIND identifier	Repository table	Package
IObjectPackage	ObjectPackage	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.objectpackage
IPDF	Pdf	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.pdf
IPowerPoint	Powerpoint	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.powerpoint
IProfile	Profile	CI_SYSTEMOBJECTS	com.businessobjects.sdk.plugin.desktop.profile
IProgram	Program	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.program
IPublication	Publication	CI_INFOOBJECTS	com.businessobjects.sdk.plugin.desktop.publication
IReport	CrystalReport	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.report
IRTF	Rtf	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.rtf
IsecEnterprise	secEnterprise	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.authentication.enterprise
IsecLDAP	secLDAP	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.authentication.ldap
IsecWinAD	secWinAD	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.authentication.secwinad
IServer	Server	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.server
IServerGroup	ServerGroup	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.servergroup
IService	Service	CI_SYSTEMOBJECTS	com.businessobjects.sdk.plugin.desktop.service

Class	SI_KIND identifier	Repository table	Package
IServiceContainer	ServiceContainer	CI_SYSTEMOBJECTS	com.businessobjects.sdk.plugin.desktop.servicecontainer
IShortcut	Shortcut	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.shortcut
ISMTP	Smtp	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.destination.smtp
ITxt	Txt	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.txt
IUniverse	Universe	CI_APPOBJECTS	com.businessobjects.sdk.plugin.desktop.universe
IUser	User	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.user
IUserGroup	UserGroup	CI_SYSTEMOBJECTS	com.crystaldecisions.sdk.plugin.desktop.usergroup
IWebi	Webi	CI_INFOOBJECTS	com.businessobjects.sdk.plugin.desktop.webi
IWord	Word	CI_INFOOBJECTS	com.crystaldecisions.sdk.plugin.desktop.word

3.3.3 Accessing properties in query results

There are two ways to access the properties of an object returned in the query results:

- Use the accessor (get and set) methods in the object interface.
- Use the property bag in the object interface.

Using accessor methods

The object interface contains accessor methods that can be used to retrieve the values of certain object properties.

For example, if the `SI_NAME` property was included in the `SELECT` clause, use the `getTitle` method of `IInfoObject` to retrieve it.

> **Note**
>
> If you call an accessor method to get a property value, and the corresponding property was not included in the `SELECT` clause, an exception is thrown.

The following code retrieves all objects in the repository that match the specified condition. The `SELECT` clause of the query includes the `SI_NAME` and `SI_DESCRIPTION`, which specifies to include these properties in the results. For each object in the results, the `getTitle` and `getDescription` methods of `IInfoObject` are used to get the values of these properties.

```
String queryString = "SELECT SI_NAME, SI_DESCRIPTION FROM CI_INFOOBJECTS "
        + "WHERE SI_NAME LIKE 'My Report%'";
IInfoObjects infoObjects = infoStore.query(queryString);
Iterator infoObjectsIter = infoObjects.iterator();
while(infoObjectsIter.hasNext()) {
    IInfoObject infoObject = (IInfoObject) infoObjectsIter.next();
    out.println(infoObject.getTitle() + ": " + infoObject.getDescription() +
"<br/>");
}
```

The `IInfoObject` interface is the base interface for all objects in the CMS repository. It only contains accessor methods for properties that are common to all object types. To access type-specific accessor methods for an object, you must cast the object to the appropriate interface. For example, if you have a folder object and you want to use folder related accessor methods, cast the object to `IFolder`.

Using the property bag

The object interface also contains a property bag, which contains all the properties that were retrieved for the object.

Use the `properties` method of `IInfoObject` to return the property bag of the object. Then use the `getProperty` method of `IProperties` to retrieve the value from the property bag. `getProperty` returns an `IProperty` object that contains the value of the property with the specified name.

> **Note**
>
> When using `getProperty` to retrieve a property from a property bag, use the appropriate constant that is defined in `CePropertyID` rather than the literal property name. For example, use `CePropertyID.SI_PARENTID` instead of the literal string `SI_PARENTID`.

To set the value of a property or to add a property to a property bag, use the `setProperty` methods of `IProperties`.

Use the property bag only if the object interface does not have accessor methods for that property. For example, the `SI_CREATION_TIME` property can be accessed only through the property bag. The following code retrieves the value of the `SI_CREATION_TIME` property:

```
String queryString = "SELECT SI_NAME, SI_CREATION_TIME FROM CI_INFOOBJECTS "
        + "WHERE SI_KIND='CrystalReport' AND SI_NAME = 'Accessibility'";
IInfoObjects infoObjects = infoStore.query(queryString);
```

```
IInfoObject infoObject = (IInfoObject) infoObjects.get(0);
IProperties properties = infoObject.properties();
IProperty creationTimeProperty =
properties.getProperty(CePropertyID.SI_CREATION_TIME);
String value = (String) creationTimeProperty.getValue();
```

Processing and scheduling information properties

If an object contains scheduling information, use the `SI_PROCESSINFO` and `SI_SCHEDULEINFO` properties to access the processing and scheduling information for that object.

> **Note**
>
> If an object has never been scheduled and has not been configured with scheduling information, the `SI_PROCESSINFO` and `SI_SCHEDULEINFO` properties are undefined. Use the `SI_IS_SCHEDULABLE` property to determine if an object supports scheduling.

Unlike other properties, `SI_PROCESSINFO` and `SI_SCHEDULEINFO` contain nested properties (subproperties) that can be retrieved individually in queries.

The format for specifying a subproperty in a query is as follows:

```
<propertyName>.<subPropertyName>
```

where `<propertyName>` is either `SI_PROCESSINFO` or `SI_SCHEDULEINFO` and `<subPropertyName>` is the name of the subproperty.

For example, the following query retrieves only the specified values from the processing and scheduling properties.

```
SELECT SI_NAME, SI_PROCESSINFO.SI_DBNEEDLOGON,
SI_SCHEDULEINFO.SI_SUBMITTER FROM CI_INFOOBJECTS
WHERE SI_NAME LIKE 'Weekly Reports%';
```

> **Note**
>
> The `SI_PROCESSINFO` and `SI_SCHEDULEINFO` properties and their subproperties are not supported in the `WHERE` clause.

The `IInfoObject` class has two accessor methods for retrieving processing and scheduling information: `getProcessingInfo` and `getSchedulingInfo`.

- `getProcessingInfo` returns an `IProcessingInfo` object containing the processing information. To access properties in `IProcessingInfo`, use the property bag, which can be retrieved by using the `properties` method.
- `getSchedulingInfo` returns an `ISchedulingInfo` object containing the scheduling information. To access properties in `ISchedulingInfo`, use the accessor methods.

Related Information

Object types [page 26]

3.3.4 The SELECT clause

The `SELECT` clause contains the list of properties to include in the objects returned by the query.

Each `IInfoObject` in the results will have the corresponding property set to the value contained in the repository for that object. For example, if you include the `SI_GUID` and `SI_DESCRIPTION` properties in the `SELECT` clause, the objects returned by the query contain the `SI_GUID` and `SI_DESCRIPTION` properties.

```
SELECT SI_ID, SI_NAME, SI_GUID
```

The following table describes the most commonly used properties in the `SELECT` clause and shows the methods in `IInfoObject` that can be used to retrieve them:

Property name	Accessor methods	Description
`SI_ID`	`IInfoObject.getID()`	An integer containing the unique identifier of the object. Note that this is not a primary key. If the object is deleted from the repository, the identifier may be reassigned to another object at a later time.
`SI_NAME`	`IInfoObject.getTitle()`	A string containing the name of the object.
`SI_DESCRIPTION`	`IInfoObject.getDescription()`	A string containing the description of the object.

Many other properties can be included in the `SELECT` clause. Some of these properties apply to all types of objects. Many others are specific to the type of object you are retrieving.

> **Note**
>
> If a property name in the `SELECT` clause is misspelled, or if the property is undefined for the matching objects, the query parser ignores the property. If none of the property names in the `SELECT` clause are valid, the query will return an empty result set.

To select all the properties of an object, use `*` in the `SELECT` clause instead of the individual property names. For example, the following query returns all the properties of the folder objects whose IDs are within the specified range.

```
SELECT * FROM CI_INFOOBJECTS WHERE SI_KIND='Folder' AND SI_ID BETWEEN 1000 AND
1200
```

> **Note**
>
> Do not use the `*` in queries that return a large number of rows. It can cause a significant decrease in performance.

3.3.4.1 Functions

3.3.4.1.1 COUNT function

Retrieves the number of distinct values of a property. When you use the `COUNT` function in a `SELECT` clause, the result set contains only one `InfoObject` instance. The result of the `COUNT` function is stored in this object in a property bag named `SI_AGGREGATE_COUNT`. The name of the item in the property bag is the name of the property whose values were counted.

For example, the following query gets the number of objects in the `CI_INFOOBJECTS` table that have a value in the `SI_ID` property. (Note that all objects have an `SI_ID` property value, so this query retrieves the total number of objects in the table.)

```
SELECT COUNT(SI_ID) FROM CI_INFOOBJECTS
```

The following code executes the query and gets the value of the count.

```
String queryString = "SELECT COUNT(SI_ID) FROM CI_INFOOBJECTS";
IInfoObjects infoObjects = infoStore.query(queryString);
IInfoObject infoObject = (IInfoObject) infoObjects.get(0);
IProperties counts = infoObject.properties().getProperties("SI_AGGREGATE_COUNT");
int idCount = (int)counts.getInt("SI_ID");
```

You can retrieve multiple counts in a single query. For example, the following query retrieves the number of `SI_ID` values and `SI_DESCRIPTION` values. `SI_AGGREGATE_COUNT` property bag contains one property for each property name specified in the `COUNT` functions in the query.

```
SELECT COUNT(SI_ID), COUNT(SI_DESCRIPTION) FROM CI_INFOOBJECTS
```

In this case, `SI_AGGREGATE_COUNT` property bag contains an `SI_ID` item and an `SI_DESCRIPTION` item.

Do not use the `COUNT` function and the `TOP` function in the same query.

The `SI_SCHEDULEINFO` and `SI_PROCESSINFO` properties are not supported for use with the `COUNT` function.

> **Note**
>
> The `COUNT` function may not return the number of rows that match the query conditions. If the specified property is undefined for any of the matching rows, then that row is not included in the count. To return the number of rows that match the conditions, use `COUNT(SI_ID)`. The `SI_ID` property is defined for all objects in the repository.

3.3.4.1.2 TOP function

Specifies the maximum number of objects that are returned by the query. For example, the following query finds the first 20 objects in the `CI_INFOOBJECTS` table that have an `SI_ID` property:

```
SELECT TOP 20 * FROM CI_INFOOBJECTS
```

3.3.5 The FROM clause

The `FROM` clause specifies the tables that contain the objects you want to retrieve from the CMS repository.

The `FROM` clause can specify a single table or a list of tables. For example, specify a single table as follows:

```
FROM CI_INFOOBJECTS
```

To search multiple tables simultaneously, use a comma-separated list of table names. For example, the following query searches the `CI_INFOOBJECTS` and `CI_SYSTEMOBJECTS` tables:

```
SELECT SI_NAME, SI_ID
FROM CI_INFOOBJECTS, CI_SYSTEMOBJECTS
WHERE SI_ID =123 AND SI_CREATION_TIME='2000/03/31'
```

The following table describes the tables that can be used in the `FROM` clause.

Table	Description
`CI_INFOOBJECTS`	Contains objects that are often used to build the user desktop, such as favorites folders and reports.
`CI_SYSTEMOBJECTS`	Contains objects that are often used to build the admin desktop and internal system objects, such as servers, connections, users, and user groups.
`CI_APPOBJECTS`	Contains objects that represent BI platform applications. For example, the object representing the BI launch pad is stored in this table.

3.3.6 The WHERE clause

The optional `WHERE` clause lets you specify conditions (search criteria) to limit the number of items returned in the query results. Conditions reduce query execution time and make the result set easier to process because it reduces the number of irrelevant items.

> **Note**
>
> The `SI_SCHEDULEINFO` and `SI_PROCESSINFO` properties are not supported in the `WHERE` clause.

A condition in a `WHERE` clause usually consists of a property name, an operator, and a value. For example:

```
SI_ID = 9287
```

The value can be either another property name or a literal value.

The following `WHERE` clause contains a condition to retrieve all `CrystalReport` objects:

```
WHERE SI_KIND='CrystalReport'
```

> **Note**
>
> The `SI_PROGID` property has been deprecated in favor of the `SI_KIND` property.

You can specify multiple conditions in a single `WHERE` clause by using the `AND` keyword and `OR` keyword. The following example uses `AND` to combine two conditions to retrieve all objects that have the same parent and that were created through a scheduling process:

```
WHERE SI_PARENTID=231 AND SI_INSTANCE=1
```

Related Information

Conditions [page 36]

3.3.6.1 Commonly used properties in the WHERE clause

The following examples show the most common uses of the `WHERE` clause:

- To retrieve every info object of the class type `IReport`:

  ```
  "WHERE SI_KIND='CrystalReport'"
  ```

- To retrieve a specific InfoObject by ID:

  ```
  "WHERE SI_ID=9287"
  ```

- To retrieve every InfoObject that shares a common parent and that has been created through a scheduling process:

  ```
  "WHERE SI_PARENTID=231 And SI_INSTANCE=1"
  ```

In the `WHERE` clause, four columns are often used as criteria, the `SI_KIND` column, `SI_ID` column, `SI_PARENTID` column, and the `SI_INSTANCE` column.

> **Note**
>
> `SI_INSTANCE` is used when a boolean is required to distinguish whether the item is the result of a scheduling process.

These properties function as identifiers.

DB Column	Identity type	Value Type	IInfoObject accessor methods	Description
`SI_ID`	Unique	int	`getID()`	Identifies each InfoObject instance uniquely in the database. This is not a primary key; if the instance is deleted, the value may later be reassigned to a new instance.
`SI_KIND`	Group based on InfoObject extended class type	String identifier, such as "CrystalReport"	`getKind()`	Identifies each row by a particular InfoObject extended class type.
`SI_PARENTID`	Group based on the ID of the parent InfoObject.	int	`getParentID()`	Identifies the InfoObject instance that operates in a parent relationship to the current InfoObject. Typically, a report that is configured to be scheduled is a parent, and each report that is copied and stored when scheduled will view the source report as its parent.
`SI_INSTANCE`	Boolean	boolean	`isInstance()`	Identifies whether the item that is stored in the database row is an InfoObject that was created through scheduling (such as a nightly report) and is therefore an "instance".

3.3.6.2 Conditions

Conditions are criteria that allow you to refine searches against the CMS repository. You can use one or more conditions to reduce the number of objects that are returned by a query.

A condition has the following form:

```
<property> <operator> <value>
```

- `<property>` is a property that has been assigned a value on the object.
- `<operator>` is one of the supported operators, which include =, !=, >, <, >=, <=, `LIKE`, and `IN`.
- `<value>` is either a literal value to be compared to the property value.

For example, to search for objects whose `SI_KIND` property is set to "Server" , include the following condition in the query's `WHERE` clause:

```
SI_KIND = 'Server'
```

In this case, `SI_KIND` is the property, the = is the operator, and the literal string "Server" is the value.

If you have more than one condition, they can be connected by `AND` or `OR`.

- `AND` combines two conditions and evaluates to true when both of the conditions are true. For example, the following query combines two conditions to find objects whose `SI_KIND` is "CrystalReport" and whose `SI_ID` is within the specified range.

  ```
  SELECT SI_NAME, SI_ID, SI_DESCRIPTION
  FROM CI_INFOOBJECTS
  WHERE
      SI_KIND = 'CrystalReport'
  AND
      SI_ID BETWEEN 100 AND 800
  ```

- `OR` combines two conditions and evaluates to true when either condition is true. For example, the following query combines two conditions to find objects whose `SI_ID` is either within one range of values or is one of a non-contiguous set.

  ```
  SELECT SI_NAME, SI_ID, SI_DESCRIPTION
  FROM CI_INFOOBJECTS
  WHERE
      SI_ID BETWEEN 100 AND 200
  OR
      SI_ID IN(576, 578, 901)
  ```

To negate a condition, use the `NOT` keyword before the operator. For example, the following query searches for objects whose `SI_ID` is not one of a specific group.

```
SELECT SI_NAME, SI_ID, SI_DESCRIPTION
FROM CI_INFOOBJECTS
WHERE SI_ID NOT IN(576, 578, 901)
```

Related Information

Operators [page 37]

3.3.6.3 Operators

The following operators are supported by the query language:

- =

- !=
- >
- <
- >=
- <=
- (NOT) LIKE
- (NOT) IN
- (NOT) BETWEEN ... AND ...
- ALL

3.3.6.3.1 ALL

Use ALL to search for objects with properties that have the smallest or largest value among all objects.

You can use the following format, where <operator> can be =, !=, >, <, >=, or <=, and <property1> is the same type of property as <property2>:

property1operator ALL *property2*

Example

The following query returns the objects that have the largest number of children.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_CHILDREN >= ALL SI_CHILDREN
```

The following query returns the objects that have the smallest number of children.

```
SELECT SI_ID, SI_NAME
 FROM CI_INFOOBJECTS WHERE SI_CHILDREN <= ALL SI_CHILDREN
```

The previous examples may return more than one object, because many objects may have the same number of children. In some cases, however, you can use ALL to specify uniqueness. For example, a query that searches for SI_ID >= ALL SI_ID returns the object with the highest SI_ID.

You can also compare values of two different properties.

Example

The following query returns all objects that are not parents of other objects.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_ID != ALL SI_PARENTID
```

3.3.6.3.2 BETWEEN

Use `BETWEEN` to search for property values within a sequential range. The following query retrieves all objects in `CI_INFOOBJECTS` whose `SI_ID` property is between 100 and 300 inclusively.

```
SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_ID BETWEEN 100 and 300
```

It can be used with string values as well as with numeric values. The following query retrieves all objects in `CI_INFOOBJECTS` whose `SI_NAME` begins with a letter between I and P. (Note that the query does not match `SI_NAME` values beginning with R.)

```
SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_NAME BETWEEN 'I' and 'R'
```

3.3.6.3.3 IN

Use `IN` to search for properties that are within a certain set of values.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_ID IN (103,105,106)
```

3.3.6.3.4 IS NULL

Use `IS NULL` to search for objects that do not have a specific property defined.

> **Note**
>
> searching for properties that match `IS NULL` is not the same as searching for properties that match the empty string (''). If a property matches the `IS NULL` condition, then that property is not defined for that object. If a property matches the empty string (''), then that property has been defined and the value in that property is an empty string.

Example

The following query uses `IS NULL` to search for objects that do not have the `SI_DESCRIPTION` property defined.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION IS NULL
```

The following query uses `IS NOT NULL` to search for objects that have the `SI_DESCRIPTION` property defined, the value of which can be either empty or non-empty.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION IS NOT NULL
```

The following query uses the empty string as a search criteria to retrieve objects that have the `SI_DESCRIPTION` property value set to a non-empty string.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION != ''
```

> **Note**
>
> If you use `IS NULL` with a non-existent property, the condition will match all objects. This can cause your query to execute slowly (depending on other conditions in the query) and decrease performance. Before executing a query, verify that all property names are spelled correctly.

3.3.6.3.5 LIKE

Use `LIKE` to search for values that match a pattern.

The pattern can include the following wildcard characters:

Wildcard character	Description	Example
%	Any string of zero or more characters	`WHERE SI_NAME LIKE '%computer%'` finds all objects with the word 'computer' anywhere in the name.
_ (underscore)	Any single character	`WHERE SI_USERFULLNAME LIKE '_ean %'` finds all four-letter first names that end with ean (Dean, Sean, and so on).
[ ]	Any single character within the specified range ([a-f]) or set ([abcdef]).	`WHERE SI_USERFULLNAME LIKE '% [C-P]arsen'` finds all objects with last names that end with arsen and begin with any single character between C and P (for example Carsen, Larsen, Karsen, and so on).
[^]	Any single character not within the specified range ([^a-f]) or set ([^abcdef]).	`WHERE SI_NAME LIKE 'de[^l]%'` finds all objects whose names begin with de and where the following letter is not l.

To use a wildcard character as a literal character, enclose it in square brackets ([]). For example, the following query finds all objects that have the text "5%" at the end of the description.

```
SELECT SI_NAME FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION LIKE '%5[%]'
```

Normally, searches are case-insensitive. For example, the following query finds objects whose description contains the word "report," regardless of the case:

```
SELECT SI_NAME, SI_DESCRIPTION FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION LIKE
'%report%'
```

You can also use `LIKE` to do a search that is case-insensitive. To create a case-sensitive query, enclose the characters that you are searching for with square brackets:

```
SELECT SI_ID, SI_DESCRIPTION FROM CI_INFOOBJECTS WHERE SI_DESCRIPTION LIKE '%[R]
[E][P][O][R][T]%'
```

3.3.6.4 Literal values

3.3.6.4.1 Boolean values

In a query statement, a Boolean value must be written as 1 or 0, and not as True or False.

Example

The following query returns all objects that are instances.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_INSTANCE=1
```

3.3.6.4.2 Date/time values

The Date/time format in queries can include hours, minutes, and seconds, and can be specified in the following way:

```
yyyy.mm.dd.hh.mm.ss
```

You can omit anything, provided you start starting from the right; for example, you can omit seconds, or seconds and minutes.

> **Note**
>
> The time must be in the 24 hour clock format.

If you specify hh.mm.ss, they must be in UTC time (GMT, in standard time, not daylight savings time), where the separator can be any separator character, and each separator can be a different character. All of the following orders would work:

```
yyyy/mm/dd/hh/mm/ss
yyyy.mm.dd.hh.mm.ss
yyyy/mm/dd.hh.mm.ss
yyyy/mm/dd,hh:mm:ss
```

Example

The following query returns all objects that were updated after 6 PM on January 11, 2000.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_UPDATE_TS >
'2000.01.11.18:00:00'
```

The following query returns all objects that were not updated during the month of January, 2000.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS WHERE SI_UPDATE_TS != '2000.01'
```

3.3.6.4.3 Numeric values

Numeric values in the `WHERE` clause can be surrounded by single quotation marks, but it is not required. For example, the following queries are equivalent:

```
SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_ID BETWEEN 10 and 2000
```

```
SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_ID BETWEEN '10' and '2000'
```

3.3.6.4.4 String values

String values in the `WHERE` clause must be surrounded by single quotation marks. For example:

```
SELECT SI_ID, SI_NAME WHERE SI_NAME = 'Jonathan'
```

To include a single quotation mark inside a string, use the character twice. For example, if you are searching the repository for objects whose name contains **`Jonathan's`**, use two single quotation marks to represent the apostrophe:

```
SELECT SI_ID, SI_NAME WHERE SI_NAME LIKE 'Jonathan''s%'
```

3.3.7 The ORDER BY clause

Use an `ORDER BY` clause to sort the results of a query. The `ORDER BY` clause consists of one or more properties. The sequence of the properties determines the order of the objects in the result set.

You can sort query results on property values in either ascending (`ASC`) or descending (`DESC`) order. If an order is not specified, `ASC` is the default.

Example

The following example returns a set of objects sorted in ascending order on the `SI_NAME` property.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS ORDER BY SI_NAME
```

The following example returns a set of objects sorted on two properties. First, the results are sorted in ascending order on the `SI_ID` property. Next, the results are sorted in descending order on the `SI_NAME` property.

```
SELECT SI_ID, SI_NAME FROM CI_INFOOBJECTS ORDER BY SI_ID ASC, SI_NAME DESC
```

3.3.8 Query string best practices

Use the following recommendations to improve the scalability of your query strings. An efficient query can improve the performance of your system.

3.3.8.1 Include only the properties that you need in your query

A generic query such as "SELECT *", "SELECT DYNAMIC", or "SELECT STATIC" is extremely expensive and would be a major hindrance to the scalability of your application.

The most efficient query is one that retrieves only the properties that you need.

A few properties are dynamic. These properties do not map to a specific value in the underlying database. Instead, they are calculated dynamically at the time when the query string is passed to the CMS repository.

Dynamic properties are expensive. Unlike the static properties in the previous sections, dynamic properties require processing. Therefore they should be called only when necessary, and avoided whenever scalability is a concern.

The following list shows queries from simplest and fastest to the most expensive:

- Select only the ID property.

  ```
  "SELECT SI_ID"
  ```

  This is useful when setting security rights.
- Select the ID and Name properties.

  ```
  "SELECT SI_ID, SI_NAME"
  ```

  This is useful when displaying a list of items such as reports or users.
- Select all of the specific properties that you require, but do not include any dynamic properties.

  ```
  "SELECT SI_ID, SI_NAME, SI_DESCRIPTION, SI_SCHEDULEINFO.DESTINATION, ..."
  ```

This is the most common type of select when retrieving classes with detailed information. The select clause contains a long list of properties, but only those required for the particular class being retrieved. Indexed properties are at the front of the query.

- Select all of the specific properties that you require, and include dynamic properties when needed.

```
"SELECT SI_ID, SI_NAME, SI_DESCRIPTION, SI_SCHEDULEINFO.DESTINATION,
SI_PATH, ..."
```

This is the same as the previous query, except now a necessary dynamic property has been added. The dynamic property will add cost to the query, but is limited to only those dynamic properties included.

- Select all static properties.

```
"SELECT STATIC"
```

This will gather all static properties, but leave out dynamic properties. This should not be used in a production application, but only for proof-of-concept queries.

> **Note**
>
> In previous versions of this software that did not use dynamic properties, this query is equivalent to "SELECT *".

- Select all dynamic properties.

```
"SELECT DYNAMIC"
```

This would have a heavy toll on processing as every property that requires a calculation would be returned. Use only for proof-of-concept queries.

- Select all properties, including both static and dynamic.

```
"SELECT *"
```

This is the slowest query you can create. It would have a heavy toll on processing as every static property plus every dynamic property that requires a calculation would be returned. Use only for proof-of-concept queries.

3.3.8.2 Use the TOP N function

Be as specific as possible when you query the server. For example, when you use the asterisk operator (*) in a SELECT statement, always ensure that you use the TOP N function or the WHERE clause to limit the selection to a small number of objects. If the CMS is forced to select all objects, the system may become overloaded with requests; if it selects all properties for a particular object, it can take many times longer to retrieve the data.

It is recommended that you never use * without TOP N, unless you retrieve a small number of objects. For example, `SELECT * FROM CI_INFOOBJECTS WHERE SI_ID=128` will retrieve only one object.

- `SELECT *` returns an upper limit of 1000 objects. You can specifically ask for more than 1000 object with the TOP N function.
- The IInfoObjects collection's getResultSize() method contains the number of objects that are found on the CMS that match a particular query. This number can be compared to the actual number of objects that are returned from the CMS.

Example: Inefficient use

Consider the following example where a variety of report details must be retrieved from every report in the CMS repository:

```
Select * From CI_INFOOBJECTS Where SI_KIND = 'CrystalReport'
```

This query returns all properties for all report objects (with an upper limit of 1000 objects).

Example: Efficient use

If a large number of reports exists, it will be significantly faster to use the following statement with the TOP N operator attached to the asterisk:

```
Select TOP 100 * From CI_INFOOBJECTS Where SI_KIND = 'CrystalReport'
```

You could also use a statement with TOP N attached to a list of properties:

```
Select TOP 100 SI_NAME, SI_ID, SI_CHILDREN, SI_DESCRIPTION, SI_HASTHUMBNAIL _
From CI_INFOOBJECTS Where SI_KIND = 'CrystalReport'
```

Although this query may seem bulky, to replace the asterisk with the TOP N operator and the desired properties may mean a retrieval time that is measured in seconds rather than minutes.

If you use SELECT *, you can use TOP N to limit the number of records that are returned by a query. For example, you can do the following:

```
Select TOP 20 * From CI_SYSTEMOBJECTS Where SI_KIND = 'User' ORDER BY SI_NAME
```

That code sends back the first 20 users in alphabetical order (for example, "Anna" to "Joanna"). You can then retrieve the next 20 users:

```
Select TOP 20 * From CI_SYSTEMOBJECTS Where SI_KIND = 'User' AND SI_NAME > 'Joe'
ORDER BY SI_NAME
```

3.3.8.3 Use indexed properties

Use indexed properties to filter the result set of an infostore query and find objects faster.

For example, the potential result set of a query may contain 10,000 Report objects. To enhance the performance of such a query, filter down the result set. Use indexed properties, such as SI_PROGID, to construct an efficient query that returns a smaller result set and lessens the search time.

The following is a list of all properties that are indexed.

Column/Property
SI_CUID
SI_GUID
SI_HIDDEN_OBJECT
SI_ID
SI_INSTANCE_OBJECT
SI_KIND
SI_NAME
SI_NAMEDUSER
SI_NEXTRUNTIME
SI_OWNERID
SI_PARENTID
SI_PLUGIN_OBJECT
SI_PROGID
SI_RECURRING
SI_RUID
SI_RUNNABLE_OBJECT
SI_SCHEDULE_STATUS
SI_UPDATE_TS

> **Note**
>
> The SI_NAME property is indexed for all objects, including instances.

3.3.8.4 Avoid non-indexed properties when constructing the WHERE clause

The Central Management Server (CMS) examines the WHERE clause and determines whether a property is indexed. Based on this information, the CMS constructs an optimized query.

Indexed properties, such as SI_NAME='testName', are stored in relational data on a database. A query that contains an indexed property directly queries the database, and retrieves all object IDs that match the given search criteria (testName).

Non-indexed properties, such as SI_SERVER_ID='myServer', are stored in the binary large object (BLOB) data on the database. A BLOB is determined only by its size and location; its structure cannot be interpreted by a database management system. The CMS retrieves and examines the BLOB for every object in the database, and then determines if a match exists between the search criteria (myServer) and the object (the value of the BLOBs SI_SERVER_ID).

For example, the WHERE clause may contain both an indexed property and a non-indexed property:

To optimize this query, the CMS divides the search into two categories:

1. The CMS quickly retrieves the Object ID for every object whose name equals testObject.

2. The CMS retrieves the BLOBs (from only those objects whose name equals testObject), and then checks the SI_SERVER_ID for a match.

To improve query performance, when possible construct a WHERE clause that uses indexed properties and limits non-indexed properties.

- You can use ISchedulingInfo and IProcessingInfo properties directly in a WHERE clause; however, because these properties are not indexed, they slow the query process. Use them only with small result sets. To ensure a small result, place the appropriate indexed property filters in the WHERE clause, along with non-indexed properties.
- In a WHERE clause, you must specify Boolean and Date/Time values, with a particular format.

3.3.8.5 Don't use LIKE in the WHERE clause

Don't use LIKE in the WHERE clause.

The LIKE operator takes longer to process than the '=' operator. Substituting '=' for LIKE will significantly improve performance.

3.4 Using URI queries to retrieve objects

Use the `IInfoStore.getPagingQuery` method with a URI query string to search for and retrieve a set of paged results containing objects stored in the Central Management Server (CMS). For example, to retrieve all child folders of the root folder in your installation, you can execute the following URI path query:

```
path://InfoObjects/Root Folder/*[SI_KIND='Folder']
```

The path-based repository view offered by the URI query language maps closely to the view of user applications such as BI launch pad. This makes the language intuitive to use in the sense that InfoObjects have a folder-based structure.

The URI query language allows you to do the following:

- Directly access a resource through its path.
- Perform recursive searches in the repository folder hierarchy for a specific InfoObject.
- Search for InfoObjects that match a certain condition or property.

In this section, you learn how to use the URI query language to retrieve objects from the CMS repository. You learn about the most commonly used elements in a URI query and how SDK classes interact with the path query language.

Related Information

To run a URI path query [page 48]
Using the query language to retrieve objects [page 24]

3.4.1 Prior knowledge

The URI query language requires the knowledge of some key SQL query language concepts.

You should have a basic understanding of the following concepts:

- The query language properties, such as SI_NAME and SI_CUID.
- The syntax of the query language, such as the WHERE clause and the ORDER BY clause. These clauses are used in the URI query language, although in a different form.
- How to select only the properties that you need in your path query, in order to improve system performance.
- Indexed properties, and how they optimize your queries.

Related Information

Use indexed properties [page 64]

3.4.2 To run a URI path query

To run a URI path query against the CMS repository, you must have a valid `IEnterpriseSession` object.

URI queries are first processed by the `IInfoStore.getPagingQuery` method with the end goal of creating page-based results. This method performs paging operations on this URI query and returns a series of subqueries for each page of InfoObjects. You can then iterate through these page-based subqueries, convert each to a valid query language (SQL) query, and process the results for each page.

1. Call the `getService` factory method of the `IEntepriseSession` object, and set the argument to **`InfoStore`**. Cast the return object to `IInfoStore`.

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Create the query string.

The query string has the following format:

```
<protocol>://<query>
```

- `<protocol>` specifies the type of URI path query to run. There are four supported protocols:
  - `path`, which allows you to specify the URI path to an object or group of objects. The path can include filters for more precise results.
  - `cuid`, which allows you to specify one or more object CUIDs.
  - `search`, which allows you to specify search criteria.
  - `query`, which allows you to use the SQL-like query language.
- `<query>` is the protocol-specific query string.

The following code creates a query string that retrieves all system objects in the repository with IDs between 678 and 700.

```
String queryString =
      "path://SystemObjects/**/[SI_ID BETWEEN 678 AND 700]";
```

> **Note**
>
> Query language keywords, property names, and table names are not case-sensitive. For example, when you specify the protocol, `path` is the same as `PATH`.

3. Create a `PagingQueryOptions` object. In the constructor call, pass the maximum number of items that can appear on a page.

```
int maxItemsPerPage = 20;
PagingQueryOptions options =
        new PagingQueryOptions(maxItemsPerPage);
```

4. Pass the query string and the `PagingQueryOptions` object to the `getPagingQuery` method of the `IInfoStore` object.

```
IPageResult pageResult =
        infoStore.getPagingQuery(queryString, options);
```

The `getPagingQuery` method examines the query and the maximum number of items per page specified in the `PagingQueryOptions` object. It uses this information to determine how many pages are required and then generates a set of subqueries. Each subquery can be used to retrieve the items for a page.`getPagingQuery` returns an `IPageResult` collection that contains the subqueries.

5. Use the `iterator` method of the `IPageResult` collection to get an iterator object and retrieve the individual queries from the collection. Cast the objects returned by the iterator to `IStatelessPageInfo`.

```
Iterator pageResultIter = pageResult.iterator();
while(pageResultIter.hasNext()) {
    IStatelessPageInfo pageInfo = (IStatelessPageInfo) pageResultIter.next();
    ...
}
```

6. Call the `getPageSQL` of the `IStatelessPageInfo` object.

```
String pageQuery = pageInfo.getPageSQL();
```

This returns a query string that can be passed to the `query` method of `IInfoStore`.

7. Pass the query string to the `query` method of the `IInfoStore` object.

```
IInfoObjects pageInfoObjects = infoStore.query(pageQuery);
```

   The `query` method returns an `IInfoObjects` collection that contains the results of the query.

8. Use the `iterator` method of the `IInfoObjects` collection to get an iterator object. and retrieve individual objects from the collection. Cast the objects returned by the iterator to `IInfoObject`.

```
Iterator infoObjectsIter = pageInfoObjects.iterator();
while(infoObjectsIter.hasNext()) {
    IInfoObject infoObject = (IInfoObject) infoObjectsIter.next();
    ...
}
```

3.4.3 Object types

To retrieve an InfoObject with a URI path query, you need to know its path in the CMS repository. Each type of InfoObject has a different root table and base folder.

What is the path based on InfoObject type?

InfoObject type	Root table	Base folder
Calendar	`SystemObjects`	`Calendars`
Category	`InfoObjects`	`Categories`
Connection	`SystemObjects`	`Logon Sessions`
CrystalReport	`InfoObjects`	`Root Folder`
CrystalEnterprise.DiskUnmanaged	`SystemObjects`	`Plugins/Destination Plugins/`
Event	`SystemObjects`	`Events`
Excel	`InfoObjects`	`Root Folder`
FavoritesFolder	`InfoObjects`	`User Folders`
Flash	`InfoObjects`	`Root Folder`
Folder	`InfoObjects`	`Root Folder`
CrystalEnterprise.Ftp	`SystemObjects`	`Plugins/Destination Plugins/`

InfoObject type	Root table	Base folder
FullClient	InfoObjects	Root Folder
Hyperlink	InfoObjects	Root Folder
Inbox	InfoObjects	Inboxes
LicenseKey	SystemObjects	License Keys
CrystalEnterprise.Managed	SystemObjects	Plugins/Destination Plugins/
ObjectPackage	InfoObjects	Root Folder
Overload	AppObjects	Application Folder
Pdf	InfoObjects	Root Folder
PersonalCategory	InfoObjects	Personal Categories
Powerpoint	InfoObjects	Root Folder
Profile	SystemObjects	Profiles
Program	InfoObjects	Root Folder
Publication	InfoObjects	Root Folder
Rtf	InfoObjects	Root Folder
secEnterprise	SystemObjects	Plugins/Auth Plugins/
secLDAP	SystemObjects	Plugins/Auth Plugins/
secWinAD	SystemObjects	Plugins/Auth Plugins/
Server	SystemObjects	Servers
ServerGroup	SystemObjects	Server Groups
Shortcut	InfoObjects	Root Folder
CrystalEnterprise.Smtp	SystemObjects	Plugins/Destination Plugins/
Txt	InfoObjects	Root Folder
Universe	AppObjects	Application Folder
User	SystemObjects	Users

InfoObject type	Root table	Base folder
UserGroup	`SystemObjects`	`User Groups`
Webi	`InfoObjects`	`Root Folder`
Word	`InfoObjects`	`Root Folder`
Xcelsius	`InfoObjects`	`Root Folder`

Related Information

The path operator: / [page 56]

3.4.4 Protocols

URI queries support four types of protocols:

- `path://`
  Use this protocol to locate InfoObjects using their URI path. You can search for InfoObjects using wildcard characters.
- `cuid://`
  Use this protocol to locate InfoObjects using their CUID.
- `search://`
  Use this protocol to locate InfoObjects using more specific search conditions than wildcard characters.
- `query://`
  Use this protocol to locate InfoObjects using the traditional SQL query language.

> **Note**
>
> All protocols produce the same result: they return one or more InfoObject resources from the CMS. As such, the queries are completely interchangeable.

3.4.4.1 Path protocol

The path query language syntax is as follows:

```
path://[Root table]/[Base folder]/folder1/folder2/.../[Target resource SI_NAME]
```

The path begins with one of three root tables:

- InfoObjects

- SystemObjects
- AppObjects

From the root table, you specify a hierarchical path of folders separated by the forward slash character, down to the target resource.

The root table and base folder is different for each type of InfoObject.

You can combine the query with path query operators.

Example: Retrieve Crystal report

In this example, you retrieve a report named `My Report` from the CMS. It is assumed that you know the report is located in the `Feature Samples` folder.

> **Note**
>
> In cases where you do not know the full name or location of an InfoObject, you can use the path query language to perform searches.

For the `CrystalReport` InfoObject type, the root table is `InfoObjects` and the base folder is `Root Folder`. Therefore, the path query must begin with `path://InfoObjects/Root Folder`.

The `Feature Samples` folder is located in the base folder, so your finished path query looks like the following:

```
path://InfoObjects/Root Folder/Feature Samples/My Report
```

Example: Retrieve user

In this example, you retrieve the `Administrator` user object from the CMS.

For the `User` InfoObject type, the root table is in `SystemObjects` and the base folder is `Users`. Therefore, the path query must begin with `path://SystemObjects/Users`.

The `Administrator` user is located in the base folder, so your finished path query looks like the following:

```
path://SystemObjects/Users/Administrator
```

Related Information

Object types [page 50]
Operators [page 56]

3.4.4.1.1 Wildcard characters

You can substitute wildcard characters for folders and target resource names in your path query. There are two types of wildcard characters:

- Single asterisk (*).
  Matches any single element in your query.
- Double asterisk (**).
  Matches any element recursively.

> **Note**
>
> Each wildcard character adds an additional round trip call to the CMS, which reduces system performance.

Example: Return all InfoObjects in all tables (InfoObjects, AppObjects, SystemObjects) that match the name "My Report"

```
path://*/**/My Report
```

Example: Return all users starting with the character Y

```
path://SystemObjects/Users/Y*
```

Example: Search for all InfoObjects two levels down from the root folder

```
path://InfoObjects/Root Folder/*/*/*
```

Example: Search for all InfoObjects that start with the character C and are contained in a folder that starts with the character B

```
path://InfoObjects/Root Folder/B*/C*
```

3.4.4.2 CUID protocol

The CUID protocol syntax is as follows:

```
cuid://<CUID_1, CUID_2,..., CUID_N>
```

You can combine the query with operators.

Related Information

Operators [page 56]

3.4.4.3 Search protocol

The search protocol syntax is as follows:

```
search://{search_term_1 search_term_2 ... search_term_N}?search_option_1=[true |
false]&...search_option_n=[true | false]
```

A search term is in the form of a string. The search query returns all InfoObjects that match at least one search term. You can have a string with spaces if you enclose the whole string in single quotation characters. For example, the string `'My World'` is one term but the string `My World` is two terms.

Each search option has either a `true` or `false` value.

Example

The following query returns all InfoObjects that have an exact `SI_KEYWORDS` or `SI_NAME` match with either the string "Financial" or "My World".

```
search://{Financial 'My World'}?MatchExact=true&CaseSensitive=true
```

Related Information

The parameter operator: ? [page 58]

3.4.4.4 SQL protocol

The SQL protocol syntax is as follows:

```
query://{SELECT [Properties] FROM [Table] WHERE [Conditions] ORDER BY [Properties]}
```

Related Information

Using the query language to retrieve objects [page 24]

3.4.5 Operators

Path query operators are optional operators that you add to your path queries. Multiple operators can be combined in a single query.

3.4.5.1 The path operator: /

Use the path operator (/) at the end of the path query to return InfoObjects linked to the target resource through a relationship.

You can specify a relationship keyword after the operator. If you do not specify a relationship keyword, the default relationship of the InfoObject will be used.

Example: No path operator

This path query returns the `Report Samples` folder:

```
path://InfoObjects/Root Folder/Report Samples
```

Example: Path operator

If you add the path operator to the end of the query, you return the InfoObjects linked to the folder through its default relationship. The default relationship of a folder is its children objects. Therefore, this path query returns all InfoObjects located in the `Report Samples` folder:

```
path://InfoObjects/Root Folder/Report Samples/
```

Related Information

Relationships [page 60]

3.4.5.2 The parent inclusion operator: +

Use the parent inclusion operator (+) with the path operator (/) to include the parent InfoObject in your query result.

Example: Return the "My Report" object, and all instances that reference "My Report" as its parent

```
path://InfoObjects/Root Folder/Report Samples/My Report+/*
```

3.4.5.3 The attribute operator: @

Use the attribute operator (@) to specify the InfoObject properties you want to return with your query. If this operator is not specified, then the query returns all of an InfoObject's properties.

Example: Return the SI_CUID and SI_NAME properties of the "My Report" object

```
path://InfoObjects/Root Folder/Report Samples/My Report@SI_CUID, SI_NAME
```

3.4.5.4 The conditional operator: [ ]

Use the conditional operator ([ ]) to filter your query with a condition. The conditional operator can contain any valid SQL WHERE clause.

Example: Return all objects under the "Report Samples" folder with CUID greater than 123

```
path://InfoObjects/Root Folder/Report Samples/[SI_CUID > 123]
```

Example: Return all objects under the "Report Samples" folder that start with P

```
path://InfoObjects/Root Folder/Report Samples/[SI_NAME LIKE 'P%']
```

This is equivalent to the following query:

```
path://InfoObjects/Root Folder/Report Samples/P*
```

Related Information

The WHERE clause [page 34]

3.4.5.5 The parameter operator: ?

Use the parameter operator (?) to add sort order and search options to your query. Use this operator with one or more of the following keywords.

> **Note**
>
> You can combine the different search options with the ampersand (&) character.

OrderBy

The `OrderBy` parameter specifies how the query results are sorted.

```
path://[Path query]?OrderBy=[property_1 [ASC | DESC]], [property_2 [ASC | DESC]],
```

`ASC` specifies ascending order and `DESC` specifies descending order. The default sort order is ascending.

The default option is `OrderBy=SI_ID`. Sort by `SI_CUID` is not supported.

> **Note**
>
> Query results are paged by default, even if you do not specify any sort order. To page your query results, you must ensure that each InfoObject property in the `OrderBy` clause is an indexed property. If no paging occurs, then your results will be returned in one page. Returning a large number of results in one page dramatically reduces performance.

The following example returns all InfoObjects under the root folder and sorts them by `SI_NAME`, descending sort.

```
path://InfoObjects/Root Folder/**/*?OrderBy=SI_NAME DESC
```

The following example returns all InfoObjects under the root folder and sorts them by `SI_PARENTCUID` in descending order and then by `SI_NAME` in ascending order.

```
path://InfoObjects/Root Folder/**/*?OrderBy=SI_PARENTCUID DESC, SI_NAME ASC
```

The following search options are used with the search protocol.

SearchName

Specifies whether the `SI_NAME` property of the InfoObject should be searched.

The default value is true.

SearchKeywords

Specifies whether the `SI_KEYWORDS` property of the InfoObject should be searched.

The default value is true.

CaseSensitive

Specifies whether the case of the search terms must match the case of the InfoObject.

The default value is false.

MatchAllWords

Specifies whether all search terms must match the InfoObject.

> **Note**
>
> If `MatchAllWords` and `MatchExact` are both set to `true` with more than one search term, then the search will never return any results. There can never be an exact match with more than one search term, if all search terms are to match at the same time.

The default value is false.

FindWithoutWords

Specifies whether to search for InfoObjects that do not match any search terms.

For example, if you set `FindWithoutWords` to `true` and search for the InfoObject name `Kitty`, then the search returns all InfoObjects not named `Kitty`.

The default value is false.

MatchExact

Specifies whether the search term must be a complete match for the InfoObject. A partial substring match will be ignored.

The default value is false.

IncludeInstances

Specifies whether to include the instances of the InfoObjects in the search results.

The default value is false.

3.4.6 Relationships

One of the most common relationships is the relationship between the parent folder InfoObject and its children. In this case, the relationship is a hierarchical one, as the children are one level down from the parent folder.

Relationships also exist between InfoObjects not located together in a hierarchical structure. If you want to return InfoObjects that are not related hierarchically, you must use a relationship keyword in your query.

The relationship language syntax is as follows:

```
path://[Path query]/[Target resource SI_NAME]/[keyword-property combination]
```

The `keyword-property combination` is in the form `keyword[property]`, where `keyword` is the relationship keyword and `property` is the InfoObject property to apply the relationship to.

This section lists the relationship keywords that you can insert into your query.

3.4.6.1 Ancestors relationship

Syntax

`ancestors[property]`

Returns all InfoObjects that are higher in the hierarchy than the target resource, including the parents.

Example

Return all user groups with the Administrator user, including the user groups of the user group directly containing Administrator.

```
path://SystemObjects/Users/Administrator/ancestors[SI_USERGROUPS]
```

3.4.6.2 Descendants relationship

Syntax

```
descendants[property]
```

Returns all InfoObjects that are lower in the hierarchy than the target resource, including the children.

Example

Return all user groups under the Administrators user group.

```
path://SystemObjects/User Groups/Administrators/descendants[SI_GROUP_MEMBERS]
```

3.4.6.3 Connected components relationship

Syntax

```
connected components[property]
```

Returns all InfoObjects that are connected components to the target resource.

Example

Return all connected components that are associated with the My Report object.

```
path://InfoObjects/Root Folder/Report Samples/My Report/connected
components[SI_BUSINESSVIEWS]
```

3.4.6.4 Parents relationship

Syntax

```
parents[property]
```

Returns all InfoObjects that are parents of the the target resource.

Example

Return the user group with the Administrator user.

```
path://SystemObjects/Users/Administrator/parents[SI_USERGROUPS]
```

3.4.6.5 Children relationship

Syntax

```
children[property]
```

Returns all InfoObjects that are children of the target resource.

Example

Return the user groups directly under the Administrators user group, but not in any subgroups.

```
path://SystemObjects/User Groups/Administrators/children[SI_GROUP_MEMBERS]
```

3.4.6.6 Members relationship

Syntax

```
members[property]
```

Returns the InfoObject members contained in the target resource.

Example

Return the user group members of the Administrators user group.

```
path://SystemObjects/User Groups/Administrators/members[SI_GROUP_MEMBERS]
```

3.4.6.7 Groups relationship

Syntax

```
groups[property]
```

Returns all user groups that contain the target resource. The groups relationship is the counterpart to the members relationship.

Example

Return all user groups that contain the Administrator user.

```
path://SystemObjects/Users/Administrator/groups[SI_USERGROUPS]
```

3.4.6.8 Default relationships

If you do not specify a relationship keyword after the path operator, the default relationship of the InfoObject will be used. In most cases, the default relationship is the children relationship.

For example, the CrystalReport InfoObject has the children relationship as its default relationship. This means that the following two queries are equivalent:

```
path://InfoObjects/Root Folder/Feature Samples/My Report/
```

```
path://InfoObjects/Root Folder/Feature Samples/My Report/children[SI_PARENTID]
```

Both queries will return the instances that belong to the CrystalReport InfoObject.

If the InfoObject has no children, then nothing is returned.

What is the default relationship based on InfoObject type?

InfoObject type	Default relationship
User	`groups[SI_USERGROUPS]`
UserGroup	`members[SI_GROUP_MEMBERS]`
All other InfoObject types	`children[SI_PARENTID]`

3.4.7 URI query best practices

3.4.7.1 Include only the properties that you need in your query

The most efficient query is one that retrieves only the properties that you need. Use the attribute operator to return only the InfoObject properties that you plan to use.

Related Information

The attribute operator: @ [page 57]

3.4.7.2 Use indexed properties

Use indexed properties to filter the result set of a query and find objects faster.

For example, the potential result set of a query may contain 10,000 CrystalReport objects. To enhance the performance of such a query, filter down the result set. Use indexed properties, such as SI_NAME, to construct an efficient query that returns a smaller result set and lessens the search time.

The following is a list of all properties that are indexed.

Property
SI_CUID
SI_GUID
SI_HIDDEN_OBJECT

Property
SI_INSTANCE_OBJECT
SI_KIND
SI_NAME
SI_NAMEDUSER
SI_NEXTRUNTIME
SI_OWNERID
SI_PARENTID
SI_PLUGIN_OBJECT
SI_PROGID
SI_RECURRING
SI_RUID
SI_RUNNABLE_OBJECT
SI_SCHEDULE_STATUS
SI_UPDATE_TS

> **Note**
>
> Sort by SI_CUID is not supported. The SI_NAME property is indexed for all objects, including instances.

3.4.7.3 Catch and handle URIParserException exceptions

`URIParserException` is a special subclass of `SDKException` used to classify errors encountered when performing paged URI queries to the CMS. Such errors include malformed query expressions and invalid URI paths contained in a query. `URIParserException` contains numerous subclasses that are thrown when these errors are encountered. You can catch and handle specific `URIParserException` subclass exceptions first in order to increase the flexibility of your application.

For example, you can check for errors in the folder path when querying for documents:

```
try {
    ...
    String URIquery = "path://InfoObjects/Root Folders/Report Samples/
Demonstration/" + reportName;
    int maxItemsPerPage = 20;
    PagingQueryOptions options = new PagingQueryOptions(maxItemsPerPage);
    IPageResult pageResult = iStore.getPagingQuery(URIquery, options);
}
```

```
catch (URIParserException.InvalidPathNode e) {
    ...
}
catch (URIParserException e){
    ...
}
```

Refer to the `com.crystaldecisions.sdk.exception.URIParserException` class in the *SAP BusinessObjects Business Intelligence Platform Java API Reference* for a complete list of all SDK URI query exceptions.

3.5 Working with multilingual content

Although multilingual content cannot be created or modified through the BI platform Java SDK, you can use this SDK to retrieve and view multilingual content in a BI platform installation. Multilingual content is created and managed by the Translation Management Tool, which can be installed with the SAP BusinessObjects Business Intelligence Platform Client Tools Installer.

Multilingual InfoObject properties

InfoObjects have several properties that contain multilingual information:

- `SI_ML_NAME` - a property bag that contains localized InfoObject names.
- `SI_ML_DESCRIPTION` - a property bag that contains localized InfoObject descriptions.
- `SI_FILES` - a property bag that contains localized file properties for InfoObjects.

> **Note**
>
> When querying the CMS, use the `SI_CUID` property to identify the InfoObject you want to find. Each InfoObject can be uniquely identified by its `SI_CUID` property. Avoid using the `SI_NAME` property or the `SI_ML_NAME` property to identify InfoObjects in queries, because they are not guaranteed to be unique.

New multilingual APIs

Some of the APIs in this SDK have multilingual overloads that return localized content. These overloaded methods take the requested locale as a parameter and may use a fallback mechanism to return the content in another locale if the requested locale is not available.

The `IInfoObject` interface defines multilingual overloads for some methods:

- `getTitle` - Returns the `SI_ML_NAME` property for the requested locale.
- `getDescription` - Returns the `SI_ML_DESCRIPTION` property for the requested locale.
- `getFiles` - Returns the `SI_FILES` property for the requested locale.

The `IInfoObject` interface also defines methods that you can use to determine the available locales:

- `getTitleLocales` - Returns the available locales for the `SI_ML_NAME` property.
- `getDescriptionLocales` - Returns the available locales for the `SI_ML_DESCRIPTION` property.
- `getFileLocales` - Returns the available locales for the `SI_FILES` property.

There are no APIs for setting multilingual properties of InfoObjects. This is because the translation state is managed by the Translation Management Tool, which flags content that needs to be translated and performs other translation management tasks. If you plan to use this SDK to change the source language properties of an InfoObject in a multilingual environment, you will need to rerun the Translation Management Tool to update the InfoObject's multilingual properties.

Locale fallback mechanisms

When you request information for a specific language locale (for example: `en_US`, `en_UK`, `en_CA`), the system attempts to provide localized content in that locale. However, if there is no content available in the requested locale, the system will return the requested content in another locale (for example, `en_US` instead of `en_UK`), the fallback language, or the original content as determined by the fallback algorithm.

Each InfoObject defines a `LocaleOption` value that specifies how to determine the content locale. The `LocaleOption` value can be one of the following:

- `DEFAULT` - Returns content in the requested locale. If there is no content available in the requested locale, a `null` value is returned.
- `FALLBACK` - Returns content in the requested locale or the closest available locale as determined by the fallback algorithm.

Each plugin includes a `SI_FALLBACK_LOCALE` property, which defines the fallback language to use if content in the requested language is not available. When the `LocaleOption` value is set to `FALLBACK`, the system searches for localized content in the following order:

1. The requested language and locale, or another locale in the requested language.
2. The fallback language and locale, or another locale in the fallback language.
3. The original, untranslated content.

For example, consider a system that contains original content in `de_DE` that has been translated to `ja_JP`. If the `SI_FALLBACK_LOCALE` property is set to `en_US`, and the user requests content in `fr_FR`, then the system searches for content in the following order:

- Looks for content in the requested locale, `fr_FR`, but does not find any.
- Looks for content in any locale of the requested language, `fr_*`, but does not find any.
- Looks for content in the fallback locale, `en_US`, but does not find any.
- Looks for content in any locale of the fallback language, `en_*`, but does not find any.
- Returns the original, untranslated content in `de_DE`.

3.6 Best practices for this SDK

This section contains information on coding practices that improve the efficiency of your applications.

3.6.1 Cleanup unused sessions and resources

To ensure that the maximum number of licenses is available, always log off any users who no longer require a session. Each active session requires a license. When the maximum number of available licenses is exceeded, additional users will be unable to log on to the BI platform.

Additionally, it is beneficial to destroy all variables that are stored in the session and release resources on the server. Use the `Invalidate` method which destroys all variables in the application server's session object including the `IInfoStore` object.

This example assumes that you have already logged on and have stored the `IEnterpriseSession` object.

Example

```
IEnterpriseSession enterpriseSession =
(IEnterpriseSession)session.getAttribute("MyCESession");
session.removeAttribute("MyCESession");
enterpriseSession.logoff();
session.invalidate();
```

3.6.2 Store the IEnterpriseSession object

When viewing reports, the `IEnterpriseSession` object that is passed to the viewer must be stored in the session object. Each subsequent report viewer request must then use the stored `IEnterpriseSession` object. Failure to do so creates a new print job for every viewer request, and hinders overall system performance.

3.6.3 Determine supported interfaces

An object's supported interfaces can be determined through reflection. Determining a supported interface through reflection is usually done during the development phase. It allows you to determine what a particular `IInfoObject` or `Object` instance should be cast as when a method you are calling will consistently be returning an object of the same type.

Reflection

Many methods within this SDK will have `Object` or `IInfoObject` object as their return type. To determine the specific interface that the object supports, reflection must be used. Using reflection allows an object's supported interfaces to be determined. This allows the object to be manually cast as the determined type. Using reflection is preferred for cases where the return type will always be the same. This is primarily when querying using a specific ProgId, SI_ID, or SI_NAME.

The following example demonstrates how to use reflection to determine the specific type of an `IInfoObject` instance, in this case, an `IUserGroup` object:

```
import java.lang.reflect.*;
.
.
.
IInfoObjects objs = iStore.query("SELECT SI_NAME FROM CI_SYSTEMOBJECTS "
    + "WHERE SI_KIND='UserGroup'");
IInfoObject obj = (IInfoObject) objs.get(0);
Class[] interfaces = obj.getClass().getInterfaces();
for( int i = 0; i < interfaces.length; i++ )
{
    out.println( interfaces[i].getName() );
}
```

3.6.4 Catch and handle SDK exceptions

When errors are encountered, the SDK throws many different exceptions. Check for these exceptions in your application to handle different scenarios for your users.

SDKException class

`SDKException` is the base class that encapsulates all exceptions thrown by the SDK. `SDKException` contains numerous subclasses that are thrown when the SDK encounters errors. You can catch and handle specific subclass exceptions first in order to increase the flexibility of your application.

For example, suppose your application allows documents to be scheduled, certain users may not have the necessary rights to perform such an action. An `SDKException.NoRight` exception will be thrown by the SDK during this operation, which you can catch handle:

```
try {
    ...
    infoStore.schedule(infoObjects);
}
catch (SDKException.NoRight e) {
    ...
}
catch (SDKException e){
    ...
}
```

In this example, the more general `SDKException` class is caught as a last resort after checking to see if `SDKException.NoRight` was thrown.

Refer to the `com.crystaldecisions.sdk.exception.SDKException` class in the *SAP BusinessObjects Business Intelligence Platform Java API Reference* for a complete list of all SDK exceptions.

SDKServerException class

`SDKServerException` is a special subclass of `SDKException` used to classify errors encountered when making requests to the Central Management Server (CMS). Such errors include problems when interacting with objects in the BI platform repository using the `IInfoStore` interface, and security errors that can occur during authentication workflows.

The `SDKServerException` class captures various information about the specific server error encountered. This includes the error code string, which can be retrieved using the `getErrorCodeString` method. These error codes can be used to handle server errors appropriately in your application.

For example, suppose your application uses Trusted Authentication as a Single Sign-On solution. If the shared secret on the web server has expired, the CMS will encounter an error with error code `FWB 00022` and the SDK will throw an `SDKServerException` exception. After catching the exception, you can handle error `FWB 00022` accordingly before invoking more general exception-handling logic:

```
try {
    ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
    ITrustedPrincipal trustedPrincipal =
sessionMgr.createTrustedPrincipal(username, cms);
    IEnterpriseSession enterpriseSession = sessionMgr.logon(trustedPrincipal);
}
catch (SDKServerException e) {
    if (e.getErrorCodeString() == "FWB 00022")
    {
        ...
    }
    else
    {
        ...
    }
}
```

Refer to the "BI Platform Servers (FWB) Error Messages" section in the *Error Message Explained* guide for a complete list of all BI platform server error codes.

SDKRuntimeException class

The `SDKRuntimeException` class captures runtime errors encountered by the SDK. These exceptions generally do not need to be caught and handled in your application.

Refer to the `com.crystaldecisions.sdk.exception.SDKRuntimeException` class in the *SAP BusinessObjects Business Intelligence Platform Java API Reference* for a complete list of all SDK runtime exceptions.

3.6.5 Keep the CMS in a consistent state

As a best practice, make sure that operations requiring multiple CMS queries are done as a single transaction so that the CMS always remains in a consistent state.

To commit InfoObjects to the CMS in a single transaction:

1. Create a new InfoObject in memory.
2. Add files to the InfoObject.
3. Commit the InfoObject.

The following example adds two folders to an InfoObject, and then commits it to the CMS:

```
void createInfoObject(IEnterpriseSession enterpriseSession, String newIn
foObjectName) throws SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoS
tore");
  IInfoObjects newFolders = infostore.newInfoObjectCollection();
  IFolder newFolder1 = (IFolder) newFolders.add(IFolder1.FOLDER_KIND);
  IFolder newFolder2 = (IFolder) newFolders.add(IFolder2.FOLDER_KIND);
  infostore.commit(newFolders);
}
```

If multiple database transactions are required, you can maintain consistency by using temporary storage outside of the CMS:

1. Create an InfoObject in temporary storage, in a private area of the repository that is inaccessible to users.
2. Perform database transactions. For example, update Crystal report parameters.
3. Copy the InfoObject from the temporary storage area to an accessible folder in the CMS.

Related Information

To create an InfoObject [page 19]

3.7 Serializing objects in horizontal or vertical clusters

Horizontal Cluster

A horizontal cluster consists of a cluster of servers that are exposed to browser clients as a single virtual server. Horizontal clusters help to increase application scalability, performance, and robustness.

Vertical Cluster

A vertical cluster is like a horizontal cluster, except that rather than use several server machines linked together, vertical clusters use a single machine with multiple CPUs. Vertical clusters help to increase scalability on multiprocessor computers since they distribute work to several processes. Each process runs on a different CPU.

Serialization Requirement

Horizontal and vertical clusters are frequently configured to allow users to access any machine in the group. In this configuration, user session information needs to be shared across all machines.

To share user session information, it is transferred to an out-of-process session state server. Objects that are stored on out-of-process session state servers must be serializable; however, not all objects are serializable. If your objects are not serializable, you will need an alternative way to maintain object persistence.

The following classes can be serialized:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`

State Management

State management is how web page state information is maintained. The main approaches to state management are session and serialized string.

Session

Session allows you to persist any object for the entire length of a user's session by storing that object in the web server's memory.

Session is typically used to do either of these things:

- Store information that needs to persist its state during the entire length of a user session, such as logon or other information required as users navigate the Web application.
- Store an object that needs to persist its state only across a page reload or a functionally grouped set of pages.

The advantage of using session variables for persistence is that it preserves the user's state information on the Web server for access at any time from any page.

The following code retrieves an `IInfoObject` and stores it in an attribute of the `HttpSession`.

```
String reportQuery = "SELECT * FROM CI_INFOOBJECTS WHERE SI_ID=" + reportID;
IInfoObjects reports = infoStore.query(reportQuery);
IInfoObject report = reports.get(0);
session.setAttribute("MyReport", report);
```

Then, on another web page, you can retrieve the object from the `HttpSession`:

```
IInfoObject report = (IInfoObject) session.getAttribute("MyReport");
```

This method can be used for persisting `IInfoObject`, `IInfoObjects`, `IInfoStore`, and `IIEnterpriseSession` objects.

Serialized string

`IEnterpriseSession` objects can also be reconstructed from a serialized session string. Once you have a valid session, you can get the serialized session string as follows:

```
String serializedSession = enterpriseSession.getSerializedSession();
```

Then, you can pass the serialized string to another web page (using a cookie or a form attribute, for example) and reconstruct the session from it:

```
IEnterpriseSession enterpriseSession = sessionMgr.getSession(serializedSession);
```

3.8 Use the same CMS to read and write an object

If your deployment uses two or more instances of the Central Management Server (CMS), ensure that the same CMS instance is used to read and write the same object.

When you change an object using one CMS instance, that CMS instance updates the object before it can be read or modified again by the same CMS instance. The change is propagated to other CMS instances in the system landscape. However, if an object is modified using one CMS instance, and simultaneously read using another CMS instance, it is possible to read the object before the modification has been applied.

This may happen when all of the following conditions are met:

- The BI platform deployment uses two or more CMS instances.
- Your application uses more than one session.
- You are performing simultaneous or near-simultaneous access to the same object using different sessions that use different CMS instances.

> ⓘ Note
>
> This scenario is most likely to occur in multithreaded applications.

To avoid this scenario, ensure that you use the same CMS instance to read and write the same object. You can either use a single session to access a particular object, or you can use multiple sessions that use the same CMS instance.

Discovering which CMS your session uses

You can discover which CMS instance your session uses by retrieving the CMS name from the session. For example, if you are using the BI platform Java SDK, use the `IEnterpriseSession.getCMSName` method to retrieve the session name. If you are using the BI platform .NET SDK, read the `EnterpriseSession.CMSName` property.

> **Note**
>
> It is possible for a session to switch from one CMS to a another, even while the session is active. This is useful when, for example, a CMS is taken down for maintenance. It is recommended to verify which CMS your session uses immediately before accessing it.

4 Setting up the development environment

If you are developing against the SAP BusinessObjects Business Intelligence platform SDK, the Report Application Server SDK, or incorporating report viewing into your application using the Crystal Report viewers, there are many components to install and configure. Your `web.xml` file must be configured to take advantage of SDK settings, specific JAR files must be deployed with your web application to utilize the SDKs, and certain prerequisite software must be installed.

You must be familiar with the basics of developing and deploying Java Platform Enterprise Edition (Java EE) web components.

4.1 Setting your target JVM for compiling

When compiling SAP BusinessObjects Business Intelligence Suite 4.x applications that use the SDK, you must target JVM 1.5. For example, use the `-target` java compiler option, `javac -target 1.5`, or ensure that your JSP compiler targets JVM 1.5.

> ⓘ Note
>
> The target JVM version for compiling your Java applications is different than the JVM versions listed for supported application servers in the *Product Availability Matrix*.

4.2 Web application setup

For the web application to function correctly, your web application must be properly configured. Configuring your web application involves setting up the directory structure, creating a `web.xml` file, and copying the required support and library files into the web application.

The following procedure assumes that you are creating a web application from the beginning. However, you can also use the directions below to modify an existing web application to add viewer functionality.

4.2.1 To set up your web application

You can configure your web application to use libraries and classes needed to deploy web applications that use this SDK.

1. Create the following directory structure in the web applications directory of your web application server, where `web_application_name` is the name of your web application:

```
web_application_name
      WEB-INF
         lib
         classes
      crystalreportviewers
```

> **Note**
>
> Your web application directory (`web_application_name`) should be located in the main web application directory for your web application server (for Tomcat, this directory is typically called `webapps`). This is the root directory for the web application and contains all of the files that the web application needs.

   Directory Contents:

   - The `WEB-INF` directory contains the support files required for your web application.
   - The `WEB-INF\lib` directory contains the JAR files required by your web application.
   - The `crystalreportviewers` directory contains the support files required by the viewers.

2. Copy the support and library JAR files required for your web application into the `WEB-INF\lib` directory. For details on which JARs to deploy, see JAR files needed for deployment of SAP BusinessObjects software [page 83].

> **Note**
>
> Copy the dependent jar files from `java\lib\external` into `WEB-INF\lib`. You do not need to create a `WEB-INF\lib\external` subdirectory.

3. Create a `web.xml` file for your web application and save it in the `WEB-INF` directory (or modify an existing `web.xml` file if you are adding viewing capability to an existing web application).
   To learn how to set up the `web.xml` file, see Configuring your web.xml file [page 76].

4.2.2 Configuring your web.xml file

A `web.xml` file defines settings that are used in a Java web application. The `web.xml` file must be placed in the `WEB-INF` directory of your web application. The root element in the file must always be `<web-app>`. The following settings can be added to your `web.xml` file.

Web.xml setting reference table

Setting	Description
crystal_document_view [page 77]	Sets the view type of the report view pane.

Setting	Description
crystal_exception_info [page 78]	Allows you to display exception information
crystal_exception_log_file [page 78]	Allows you to log exception information
crystal_image_uri [page 78]	Sets the location of the `crystalreportsviewers` directory in your web application
crystal_image_use_relative [page 79]	Sets the interpretation of the `crystal_image_uri` to be relative to the web page, application, or server
crystal_max_number_parameter_default_values [page 79]	Sets the default number of values that will be returned in a list.
crystal_processing_indicator_delay [page 79]	Enables the Report Processing Indicator
crystal_processing_indicator_text [page 80]	Modifies the text displayed by the Report Processing Indicator
crystal_use_asynchronous_requests [page 80]	Enables asynchronous requests.
CrystalReportViewerServlet [page 80]	Allows you to view the parameter panel in your web application, handles report exporting, interactive parameters, images, and charts
Faces Servlet [page 81]	Enables the use of JavaServer Faces (JSF) components in your web application

crystal_document_view

Sets the view type of the report view pane. The `crystal_document_view` parameter value can be set to one of the following values:

- `printlayout` - Displays a grey background behind the report with a dropdown shadow cast by the report page. Shows the page layout of the report when printed.
- `weblayout` - Fills the entire report view pane with the report. There is no grey background.

> **Note**
>
> If the crystal_document_view parameter is not specified, it is set to `printlayout` by default.

```
<context-param>
        <param-name>crystal_document_view</param-name>
        <param-value>weblayout</param-value>
</context-param>
```

crystal_exception_info

Allows you to display exception information. If you do not display exception information you can still log the exception information if you use the `crystal_exception_log_file setting`. The `crystal_exception_info` parameter value can be set to one of the following values:

- `short` - Displays the exception information without the accompanying stack trace.
- `long` - Displays the exception information with the accompanying stack trace.
- `disable` - Exception information is not displayed. The user must handle the exception.

> **Note**
>
> If the `crystal_exception_info` parameter is not specified, it is set to `short` by default.

```
<context-param>
        <param-name>crystal_exception_info</param-name>
        <param-value>long</param-value>
</context-param>
```

crystal_exception_log_file

Allows you to log exception information. The parameter value specifies the location of the log file. Regardless of the setting of the `crystal_exception_info` parameter, the exception information output to the log file will be in the long format.

> **Note**
>
> By default, exceptions are not logged.

```
<context-param>
        <param-name>crystal_exception_log_file</param-name>
        <param-value>c:\temp\webreportingexception.log</param-value>
</context-param>
```

crystal_image_uri

Sets the location of the `crystalreportsviewers` directory in your web application. This setting must be included in the web.xml file if you are using a report viewer in your web application. The `crystalreportviewers` directory contains all files required by the viewer.

```
<context-param>
         <param-name>crystal_image_uri</param-name>
         <param-value>/web_application_name/crystalreportviewers</param-value>
</context-param>
```

crystal_image_use_relative

Sets the interpretation of the `crystal_image_uri` to be relative to the web page, application, or server. The parameter value can be set to one of the following values:

- `webapp` - Sets the uri as relative to the web application directory.
- `server` - Sets the uri as relative to the server root.
- `Page` - Sets the uri is relative to the page from which the resources in the crystalreportviewers directory are requested.

```
<context-param>
        <param-name>crystal_image_use_relative</param-name>
        <param-value>webapp</param-value>
</context-param>
```

crystal_max_number_parameter_default_values

By default, the Java DHTML viewers return a maximum of 200 values per list at a time. The `crystal_max_number_parameter_default_values` parameter value overrides this default size. If a list exceeds the default size of 200, or the number set by this parameter, a link to the remaining values is displayed.

```
<context-param>
         <param-name>crystal_max_number_parameter_default_values</param-name>
                  <param-value>100</param-value>
</context-param>
```

crystal_processing_indicator_delay

The Report Processing Indicator will present specific feedback regarding both general processing status and print processing status. By default, the Report Processing Indicator will display for any postback action longer than 500 ms. The `crystal_processing_indicator_delay` parameter value overwrites the default delay of 500 ms. You can specify any non-negative number in milliseconds. When the parameter value is set to 0, the report processing indicator is disabled.

> **Note**
>
> This feature works only when using a Java DHTML viewer. This feature is currently implemented only for Internet Explorer browsers.

```
<context-param>
        <param-name>crystal_processing_indicator_delay</param-name>
        <param-value>0</param-value>
</context-param>
<context-param>
        <param-name>crystal_processing_indicator_text</param-name>
           <param-value>some text</param-value>
</context-param>
```

crystal_processing_indicator_text

The Report Processing Indicator will present specific feedback regarding both general processing status and print processing status. The `crystal_processing_indicator_text` parameter value modifies the text displayed by the Report Processing Indicator.

> **Note**
>
> This feature works only when using a Java DHTML viewer. This feature is currently implemented only for Internet Explorer browsers.

```
<context-param>
        <param-name>crystal_processing_indicator_delay</param-name>
        <param-value>0</param-value>
</context-param>
<context-param>
        <param-name>crystal_processing_indicator_text</param-name>
          <param-value>some text</param-value>
</context-param>
```

crystal_use_asynchronous_requests

By default, asynchronous requests are permitted within the DHTML viewer. If you wish to enable the *Back* button in the user's browser to work with the DHTML viewer, you must disable asynchronous requests by setting this parameter to `false.`

> **Note**
>
> It is recommended that you do not set the `crystal_use_asynchronous_requests` parameter to `false`. Doing so will disable some viewer performance enhancemenets and enhanced appearance features.

```
<context-param>
    <param-name>crystal_use_asynchronous_requests</param-name>
    <param-value>false</param-value>
</context-param>
```

CrystalReportViewerServlet

The CrystalReportViewerServlet allows you to view the parameter panel in your web application. The servlet allows the Java DHTML viewers to export and print when the viewers are embedded in other content. The servlet also handles interactive parameters, images, and charts. This setting must be included in the web.xml file if you are using a report viewer in your web application.

```
<servlet>
        <servlet-name>CrystalReportViewerServlet</servlet-name>
```

```
        <servlet-
class>com.crystaldecisions.report.web.viewer.CrystalReportViewerServlet</servlet-
class>
</servlet>
<servlet-mapping>
        <servlet-name>CrystalReportViewerServlet</servlet-name>
        <url-pattern>/CrystalReportViewerHandler</url-pattern>
</servlet-mapping>
```

Faces Servlet

Enables the use of JavaServer Faces (JSF) components in your web application. This setting must be included in the web.xml file if you are using JSF components or beans in your web application.

- `load-on-startup` - When the value is set to 0, the servlet does not load on startup. When the value is set to 1, the servlet loads on startup.
- `url-pattern` – Sets the path to the `Faces Servlet`.
- `javax.faces.application.CONFIG_FILES` parameter value – Sets the location of the faces-config.xml file. You must create this file when setting up a JSF project. For more information see:

```
<listener>
     <listener-class>com.sun.faces.config.ConfigureListener</listener-class>
</listener>
<servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup> 1 </load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.faces</url-pattern>
</servlet-mapping>
<context-param>
        <param-name>javax.faces.application.CONFIG_FILES</param-name>
<param-value>/WEB-INF/faces-config.xml</param-value>
</context-param>
```

Example: Basic web.xml file

You must change the following parameters:

- `display-name` - Name of your web application.
- `crystal_image_uri` parameter value - Location of the `crystalreportviewers` folder.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
        PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
<display-name>web_application_name</display-name>
        <context-param>
                    <param-name>crystal_image_uri</param-name>
                    <param-value>/web_application_name/crystalreportviewers</
param-value>
```

```
            </context-param>
        <context-param>
                    <param-name>crystal_servlet_uri</param-name>
                     <param-value>/CrystalReportViewerHandler</param-value>
        </context-param>
        <servlet>
                       <servlet-name>CrystalReportViewerServlet</servlet-name>
                    <servlet-
class>com.crystaldecisions.report.web.viewer.CrystalReportViewerServlet</servlet-
class>
        </servlet>
        <servlet-mapping>
                    <servlet-name>CrystalReportViewerServlet</servlet-name>
                    <url-pattern>/CrystalReportViewerHandler</url-pattern>
        </servlet-mapping>
</web-app>
```

Example: Web.xml file using JSF

You must change the following parameters:

- `display-name` - Name of your web application.
- `crystal_image_uri` parameter value - Location of the `crystalreportviewers` folder.

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC
        "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
        <display-name>web_application_name</display-name>
        <listener>
                <listener-class>com.sun.faces.config.ConfigureListener</listener-
class>
        </listener>
        <servlet>
                <servlet-name>Faces Servlet</servlet-name>
                <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
                <load-on-startup> 1 </load-on-startup>
        </servlet>
        <servlet-mapping>
                <servlet-name>Faces Servlet</servlet-name>
                <url-pattern>*.faces</url-pattern>
        </servlet-mapping>
        <context-param>
                <param-name>javax.faces.application.CONFIG_FILES</param-name>
                <param-value>/WEB-INF/faces-config.xml</param-value>
        </context-param>
        <context-param>
                <param-name>crystal_image_uri</param-name>
                <param-value>/web_application_name/crystalreportviewers</param-
value>
        </context-param>
        <context-param>
                     <param-name>crystal_servlet_uri</param-name>
                     <param-value>/crystalreportviewerservlet</param-value>
        </context-param>
        <servlet>
                       <servlet-name>CrystalReportViewerServlet</servlet-name>
                     <servlet-
class>com.crystaldecisions.report.web.viewer.CrystalReportViewerServlet</servlet-
class>
        </servlet>
```

```
        <servlet-mapping>
                    <servlet-name>CrystalReportViewerServlet</servlet-name>
                    <url-pattern>/CrystalReportViewerHandler</url-pattern>
        </servlet-mapping>
</web-app>
```

4.2.3 JAR files needed for deployment of SAP BusinessObjects software

To create custom applications using the SAP BusinessObjects Business Intelligence platform SDKs, or to deploy the sample applications provided, you'll need to know which JAR files to use. To obtain these JAR files, run the SAP BusinessObjects Business Intelligence Platform Client Tools 4.1 installer and select the developer components for the SDKs that your application uses. All core JAR files and language resources JAR files are installed in the following directory:

`C:\Program Files (x86)\SAP Business Objects\SAP BusinessObjects Enterprise XI 4.0\java\lib`.

Dependent JAR files are installed in the following directory:

`C:\Program Files (x86)\SAP Business Objects\SAP BusinessObjects Enterprise XI 4.0\java\lib\external`.

Webservices JAR files are stored in the following directory:

`C:\Program Files (x86)\SAP Business Objects\ SAP BusinessObjects Enterprise XI 4.0\dataAccess\connectionServer\xml\drivers\webservices\`

For more information about each SDK, see http://help.sap.com.

Multilingual application development requires specific JAR files that contain language resources to be installed. Command phrases and error messages will be displayed in the appropriate language according to the user's default system locale. Appropriate JAR files must be included with the application to have the correct messages displayed.

This table provides extensions for the different languages that are available. Some languages require more JAR files to enable multilingual deployment. Replace the extension <xx> with the appropriate extension from the following table:

Language Extensions

English	en	Dutch	nl
French	fr	German	de
Italian	it	Spanish	es
Japanese	ja	Simplified Chinese	zh_CN
Russian	ru	Turkish	tr

> **Note**
>
> To ensure messages appear in the language that matches the locale setting for each client machine, system locale settings must be applied to both the current user account and to the default user profile.

Deployment in languages other than en, de, es, fr, it, ja, nl, ru, tr, and zh_CN requires more resources to be included with your application. For deployment in any of the additional languages, include all of the JAR files listed as resources for additional languages as well as those listed as core JAR files, dependent JAR files, and language resources.

BI Platform Administration

> **Note**
>
> The SAP BusinessObjects Business Intelligence platform Java Server Faces Components are deprecated as of release XI 4.1, and will not be available in future releases. To develop SAP BusinessObjects Business Intelligence platform Java applications, use the SAP BusinessObjects Business Intelligence platform Java SDK, or the SAP BusinessObjects Business Intelligence platform Web Services Consumer Java SDK.

SDK	Deployment in en, de, es, fr, it, ja, nl, ru, tr, zh_CN		Resources for Additional Languages
	Core JAR Files	Dependent JAR Files	
SAP BusinessObjects Business Intelligence platform Java SDK	• bcm.jar • biarengine.jar • ceaspect.jar • cecore.jar • celib.jar • ceplugins_core.jar • cesession.jar • corbaidl.jar • ebus405.jar • logging.jar • TraceLog.jar	• activation.jar • aspectjrt.jar • axiom-api-1.2.21.jar • axiom-impl-1.2.21.jar • axis2-adb-1.7.9.jar • axis2-kernel-1.7.9.jar • axis2-saaj-1.7.9.jar • com.sap.js.passport.api.jar • commons-logging-1.1.1.jar • derby.jar • freessl201.jar • log4j.jar • sapjce.jar **Note** As of 4.3 SP2, sapjce.jar replaces, certj-FIPS.jar, cryptoj-FIPS.jar, ssljFIPS.jar, jcmFIPS.jar and cryptojce.jar. Some scenarios require additional DLLs and configurations (sapcrypto.dll, slcryptokernel.dll and slcryptokernel.dll.sha256). Refer to SAP Note 3101582 on how to configure SAP JCE for different scenarios. • wsdl4j-1.6.2.jar • xmlSchema-core2.2.1.jar **Note** Starred items indicate JAR files required when using SSL between the web tier and the backend	• ceresprops_<xx>.jar • cecore_<xx>.jar • celib_<xx>.jar **Note** Include these files in addition to the JAR files listed in the previous columns. Replace <xx> with the appropriate language code.

SDK	Deployment in en, de, es, fr, it, ja, nl, ru, tr, zh_CN		Resources for Additional Languages
	Core JAR Files	Dependent JAR Files	
		servers, as well as between backend servers.	

SAP Crystal Reports

> **Note**
> SAP Crystal Reports deployment is self contained and does not require any dependent JAR files.

SDK	Deployment in en, de, es, fr, it, ja, nl, ru, tr, and zh_CN	Resources for Additional Languages
	Core JAR Files	
Report Application Server Java SDK	• CrystalReportsSDK.jar • cereports.jar* **Note** Starred items are used for integrating the Report Application Server Java SDK with the BI platform.	• CrystalReportsSDK_xx.jar • cereports_xx.jar* **Note** Replace <xx> with the appropriate language code. Starred items are used for integrating the Report Application Server Java SDK with the BI platform.

SDK	Deployment in en, de, es, fr, it, ja, nl, ru, tr, and zh_CN Core JAR Files	Resources for Additional Languages
Viewers Java SDK	• CrystalReportsSDK.jar • log4j.jar • logging.jar • TraceLog.jar • webreporting.jar • sap.com~tc~sec~csi.jar **Note** The following jars must be included when redistributing applications that view reports that are stored within the BI platform. • aspectjrt.jar • bcm.jar • ceaspect.jar • cecore.jar • celib.jar • ceplugins_core.jar • cereports.jar • cesession.jar • corbaidl.jar • sapjce.jar **Note** As of 4.3 SP2, sapjce.jar replaces, certj-FIPS.jar, cryptojFIPS.jar, ssljFIPS.jar, jcmFIPS.jar and cryptojce.jar. Some scenarios require additional DLLs and configurations (sapcrypto.dll, slcrypto-kernel.dll and slcryptokernel.dll.sha256). Refer to SAP Note 3101582 on how to configure SAP JCE for different scenarios. • CrystalReportsSDK.jar	• webreporting_<xx>.jar **Note** Replace <xx> with the appropriate language code.

SDK	Deployment in en, de, es, fr, it, ja, nl, ru, tr, and zh_CN Core JAR Files	Resources for Additional Languages
	• ebus405.jar • icu4j.jar • log4j.jar • logging.jar • ras21sdk.jar • SL_plugins.jar • TraceLog.jar • webreporting.jar • sap.com~tc~sec~csi.jar	

Web Services

JAR files needed for Web Services are installed in the following directory:

```
C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI
4.0\web services\en\dsws_consumer\data\dswsjavaApi
```

SDK	Deployment in en, de, es, fr, it, ja, nl, ru, tr, zh_CN Core JAR Files	Dependent JAR Files	Resources for Additional Languages
Web Services Java SDK	• dsws-bicatalog.jar • dsws-biplatform.jar • dsws-common.jar • dsws-common-util.jar • dsws-publish.jar • dsws-reportengine.jar • dsws-session.jar • TraceLog.jar	• axiom-api-1.2.21.jar • axiom-impl-1.2.21.jar • axis2-kernel-1.7.9.jar • axis2-xmlbeans-1.7.9.jar • commons-codec.jar • commons-httpclient-3.1.jar • commons-logging-1.1.1.jar • log4j.jar • mail-1.4.jar • wsdl4j-1.6.2.jar • xmlbeans-2.6.0.jar • xmlSchema-core2.2.1.jar	`

4.3 Setting up your SAP BusinessObjects Business Intelligence platform environment

The SAP BusinessObjects Business Intelligence platform SDK allows you to administer your reports, documents, and users in your BI platform installation. There are several required components and settings to configure in order to develop and test against this SDK.

4.3.1 Required SAP BusinessObjects Business Intelligence platform components

What components need to be installed?

Before you can use the SAP BusinessObjects Business Intelligence platform SDK, you must have the SAP BusinessObjects Business Intelligence platform installed on either your development machine or on the same network as your development machine. For more information about how to install the BI platform, see the *SAP BusinessObjects Business Intelligence Platform Installation Guide*.

What components must be running?

To run a BI platform web application, all of your client and server components must be running: your web server, application server, and your Central Management Server (CMS). If you are running on a Windows system, once you have installed the BI platform with its default settings, your server components are automatically run as a machine service. Unless you manually turn this service off, the servers will start whenever you turn on your computer. For a Unix system, the servers will automatically start once the installation is completed. However, the servers are not running as a service and will not automatically start next time the machine is turned on or rebooted. For more information about how to start and stop your servers, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide* .

> **Note**
>
> It is possible to have different server components installed on different machines. For example, your application server can be on a different machine from your web server. If this is the case, you should ensure that the server components that are on other machines are running before you use your web application.

4.3.2 Configuring SAP BusinessObjects Business Intelligence platform settings

If the SAP BusinessObjects Business Intelligence platform Java SDK is not installed to the default directory, various paths need to be specified so that the system can locate the appropriate files. To specify paths, use the

`CeEnterpriseContext` class. Two properties in this class need to be set if the default installation directory is not used. The following table lists these properties and their use.

- The `BOBJ_ENTERPRISE_HOME` property only needs to be set if you try to use the `IReport.refreshProperties()` and `IServer.manageServer()` methods.
- The `BOBJ_COMMON` property only needs to be set if you try to use the `IServer.manageServer()` method and the common files are not installed in the default directory.
- The `UNIX_BSHELL` property does not need to be set if you are working within a Windows environment.
- These properties take effect only if they are set before any user logs on.

Property Name	Description
`BOBJ_ENTERPRISE_HOME`	This property sets the path to the BI platform directory that is created upon installation, so that the system can find the servicemanagerjni.dll used to start and stop Windows-based servers. This property also finds the reportadd executable, so that the application server can create a new process to add or refresh report objects.
`BOBJ_COMMON`	This property is necessary only if the common files are not installed in the default directory.
`UNIX_BSHELL`	This property is necessary only if the Bourne shell is not installed to the default directory. (Unix only.)

4.4 Setting up your Report Application Server environment

The Report Application Server SDK allows you to programmatically create and modify Crystal Report files. There are several components required to develop and test against this SDK, as well as considerations to take into account depending on how you store and access your reports.

4.4.1 Required Report Application Server components

This section describes the components that are required to run and develop web applications that use the Report Application Server SDK.

What components need to be installed?

Before you can use the Report Application Server SDK, you must have SAP BusinessObjects Business Intelligence platform 4.1 installed on either your development machine or on the same network as your development machine. Use the SAP BusinessObjects Business Intelligence Client Tools 4.1 installer to install

the Report Application Server SDK developer components. For more information on how to install the BI platform, see the *SAP BusinessObjects Business Intelligence Platform Installation Guide*.

Specific JAR files must be deployed to your web application. For information on the JAR files required by your product, see JAR files needed for deployment of SAP BusinessObjects software [page 83].

> **Note**
>
> To create or modify a particular report through the Report Application Server SDK, you may require additional product activation keycodes, as well as rights to view, refresh, edit, and export reports. For details, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

What components must be running?

The components that must be started depend on your usage scenario.

- To run a web application that accesses unmanaged reports from a file path, all of your client and server components should be running, including your Report Application Server and web server. In addition, several steps must be completed in order to programmatically open unmanaged reports using the Report Application Server with the BI platform:
  - Enable the Guest account. As of version XI 3.0, the Guest account is disabled by default, and an error message indicating that the user account has been disabled will be displayed when calling the `ReportClientDocument.Open` method.
  - Using Report Application Server command line parameters in the Central Management Console (CMC), configure the `ipport` switch to specify a port number that you know to be free. For example, you can set the `ipport` switch to specify the default port 1566 with `-ipport "1566"`. For further details, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide* .
  - Also using Report Application Server command line parameters in the CMC, specify the full directory path containing the reports you want to view. For example, you can set the report directory path by adding `-reportdirectory "C:\My Documents\Reports"` to the beginning of the command line. If this option is not specified, the Report Application Server looks for reports in the following default directory: `C:\Program Files (x86)\SAP Business Objects\SAP BusinessObjects Enterprise XI 4.0\Samples\en\Reports` .
- To run a web application that accesses managed reports from the BI platform repository, all of your client and server components should be running, including your Report Application Server web server, your Web Component Server, and your CMS.
  After you have installed the BI platform system with its default settings, your server components are automatically run as a machine service. Unless you manually turn this service off, the servers start when you turn on your computer. For more information on how to start and stop your servers, see 'Managing Servers' in the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

  > **Note**
  >
  > You can have different server components installed on different machines. For example, your Web Component Server can be on a different machine from your web server. If this is the case, ensure that the server components that are on other machines are running, before starting your web application.

For more information about safely starting and stopping servers, refer to the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

4.4.2 Specifying Report Application Server locations

The location of the Report Application Server is defined in the Server attribute in the `clientSDKOptions.xml` file. It can also be set programmatically using the Report Application Server SDK. To set the Server attribute at run time, use the `ReportAppSession` object's `setReportAppServer` method.

During the Report Application Server installation, the `clientSDKOptions.xml` file is created in the SDK Jar files installation directory (see Directories [page 96] for details). You must modify this file if the Report Application Server you want to connect to is installed on a different computer.

You can specify the location of the `clientSDKOptions.xml` file either statically, by setting a classpath that points to the file, or dynamically, by specifying the location of this file. If you have multiple Report Application Servers, you can enable load balancing by specifying the location of all the Report Application Servers on the network in the `clientSDKOptions.xml` file. See *Installing the Report Application Server on Windows* for details.

Defining the location of the clientSDKOptions.xml file statically

To specify the location of the `clientSDKOptions.xml` file statically, you need to add its file path to your web application server's `CLASSPATH` environment variable. You may need to add the file path to the `CLASSPATH` on your local system and on your web server. Consult the your web server's documentation for information about adding classpaths.

Defining the location of the clientSDKOptions.xml file dynamically

You can specify the location of the `clientSDKOptions.xml` file at run-time. In your JSP or Java files, use the Java method `setProperty` from the System class. Set the system property indicated by the `ras.config` key to the specified directory as follows:

```
system.setProperty("ras.config","c:\temp")
```

This function call specifies that the `clientSDKOptions.xml` file in `c:\temp` is to be used for locating Report Application Servers.

To avoid hard coding the location for the `clientSDKOptions.xml` file throughout your program, you may use the `web.xml` file (located by default in the `\WEB-INF\` directory of your web application) to specify the location of the `clientSDKOptions.xml`.

4.4.3 Report location

The Report Application Server can access both unmanaged and managed report files. The managed access is designed to allow users to modify report objects that are managed by the BI platform InfoStore, whereas unmanaged access is designed for modifying report files (`*.rpt`) that reside on a local file system. When you

open and save reports, you need to specify which reports to open and the folders where you want them to be saved.

Reports and folders are identified as follows:

- To identify report objects that are managed by the BI platform, provide a reference to an InfoObject.
- To identify reports and folders when using a Report Application Server to access report files, provide a file path.

The Report Application Server SDK provides an application programming interface for all Report Application Server distributions; hence, it supports both ways of identifying objects. For example, the method signature of the `saveAs` method of the `ReportClientDocument` class is as follows:

```
public void saveAs(java.lang.String displayName, java.lang.Object parentFolder,
int options)
```

The second parameter specifies a folder where the report will be saved. This parameter can be one of the following:

- A `String` object that gives a file path.
- An InfoObject that represents a folder in the BI platform.

Specifying a report file path

The Crystal report (`.rpt`) files you want to access by a file path through the Report Application Server must be in the folder designated as the Report Directory for the Report Application Server. The Report Application Server can open and save reports only to this folder and its subfolders. Attempting to write to folders other than this results in an "access denied" error.

Set the Report Directory using the Central Configuration Manager. During the default installation process, the sample reports folder is set as the Report Directory.

To improve performance, it is highly recommended that you use a local folder on the Report Application Server machine for the Report Directory. Reports stored outside the Report Application Server machine slow down performance, because they must be serialized and sent to the Report Application Server for processing each time they are accessed.

> **Note**
>
> Typing an asterisk (*) in the Report Directory field lets you access any report anywhere on your local machine.

You can specify reports several ways:

- Using the `ras` prefix.
  You can use the `ras` prefix as follows: `ras://c:\directory\reportname.rpt` . This string gives the location of `reportname.rpt` on the Report Application Server machine.

  > **Note**
  >
  > The `ras` prefix is not necessary as it is assumed by default. `c:\reportname.rpt` is assumed to be a path relative to the Report Application Server machine.

- Using the `rassdk` prefix.
  You can use the `rassdk` prefix as follows:
  `rassdk://c:\directory\reportname.rpt` . This string provides the location of a report named `reportname.rpt` on the machine where the Report Application Server SDK is running. This is typically the web application server.

  > **Note**
  >
  > For performance reasons, it is not recommended that you store reports on the machine running the Report Application Server SDK, or on any machine other than the Report Application Server.

- Without using a prefix.
  You can specify a report without using a prefix. When you do so, it is assumed that the report is on a local folder on the Report Application Server. For example, `c:\directory\reportname.rpt` is the location of `reportname.rpt` on the `C` drive of the Report Application Server. This format is equivalent to specifying `ras://c:\directory\reportname.rpt`.

4.4.4 Horizontal and vertical clustering

The Report Application Server supports horizontal and vertical clustering. Horizontal clustering involves running multiple Java application servers that are run on two or more separate physical machines. Vertical clustering, however, consists of multiple Java application servers on a single physical machine.

`ReportClientDocument` is a serializable object that Java application servers can store in session. If one Java application server fails, other servers can use the sessioned objects and continue to process them. This process is transparent to the user. If a Java application server can be configured for horizontal or vertical clustering and is supported by the Report Application Server, the Report Application Server will run in a clustered environment.

> **Note**
>
> Note: For the Report Application Server to run in a horizontal or vertical clustered environment, the `ReportClientDocument` object must be stored in session.

Platforms

The Report Application Server supports a number of Java application servers. Consult the `\Platforms\platforms.txt` file for details. This file is distributed with your product.

Configuration

Configurations for server clustering can vary between Java application servers. Consult your Java application server's documentation for deployment instructions.

4.5 Setting up your SAP Crystal Reports viewers

The SAP Crystal Reports viewers allow you to develop and deploy thin-client, customized report viewing capabilities within Java Server Pages (JSP). This enables you to display and export report files on client machines that do not have SAP Crystal Reports installed. There are several required components that the viewers interact with to obtain and process a report.

4.5.1 Required viewers components

The viewers are dependent on other BI platform software components. These components are described here.

What components need to be installed?

If you are using the viewers with a Crystal Reports 2020 Processing Server or a Crystal Reports 2020 Report Application Server, you must have SAP BusinessObjects Business Intelligence platform installed on either your development machine or on the same network as your development machine and the appropriate JAR files available to your web application. For more information about how to install these products, refer to their respective installation guides.

> ⓘ Note
>
> To view a report using any of the viewers, you must have the appropriate product activation keycode, as well as the appropriate access rights for the report. For details, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

Which components must be running?

- **SAP BusinessObjects Business Intelligence platform environment**
  In a BI platform environment, your Crystal Reports 2020 Processing Server and/or your Crystal Reports 2020 Report Application Server must be installed and running. The particular server required depends on your deployment and the viewers you are using in your web application.
- **Web application server**
  Your web application server and web server must also be configured correctly and running. Refer to the documentation that came with the web application server and web server for installation and configuration information.

4.5.2 Side-by-side installation of the viewers SDK

You can have two different versions of the viewers SDK installed on a single machine (referred to as a side-by-side installation). The recommended method of achieving a side-by-side installation is to use the

`WEB-INF\lib` directory of your web server to house the JAR files required by the viewers SDK. See JAR files needed for deployment of SAP BusinessObjects software [page 83] for details. When a new version of the viewers SDK is installed, the new JAR files that are automatically installed do not overwrite the JAR files in your `WEB-INF\lib` directory. By maintaining the older version of the SDK, your web application can continue to use the older library until you are ready to migrate to the newer version.

To migrate to the new version, make the appropriate changes to your web application, and then copy the new JAR files into the `WEB-INF\lib` directory.

4.6 Directories

Various components are installed to the following default directory structure:

Product/Component	Installation Directory
SAP BusinessObjects Business Intelligence platform main installation directory (Windows).	`C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\`
SAP BusinessObjects Business Intelligence platform main installation directory (Unix)	Defined during install. Referred to as `<businessobjects_install_folder>` in this table.
SAP Crystal Reports viewers resource files	`C:\Program Files (x86)\SAP BusinessObjects\Crystal Reports 14.0\crystalreportviewers\`
SAP Crystal Reports viewers resource files	`<businessobjects_install_folder>/crystalreports_xi40/crystalreportviewers/`
SDK JAR files (Windows)	`C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\java\lib\`
SDK JAR files (Unix)	`<businessobjects_install_folder>/enterprise_xi40/java/lib/`
Processing Extensions (Windows)	`C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\win32_x86\ProcessExt\`
Processing Extensions (Unix)	`<businessobjects_install_folder>/enterprise_xi40/processext/`

5 Using the SDK

This section explains how to use the APIs of the `InfoObject` model to programmatically configure the Central Management Server, retrieve and manipulate `InfoObjects`. The topics contained in this section explain key concepts and illustrate how to implement these scenarios in your application through code examples.

5.1 Authentication

BI platform authentication is the ability to validate users and grant or deny access to the Central Management Server (CMS). The SDK has a specific authentication framework and various security protocols to implement user authentication. You can configure the BI platform to work with several kinds of security protocols during logon: Enterprise, LDAP, and WinAD. You can also enable Single Sign-On (SSO) Authentication and trusted authentication .

Classes used when authenticating users

- `com.crystaldecisions.sdk.plugin.authentication.enterprise.IsecEnterprise`
  Sets global security options for native BI platform user accounts.
- `com.crystaldecisions.sdk.plugin.authentication.ldap.IsecLDAP`
  Provides properties and methods that map LDAP principals (users and groups) to the BI platform.
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
  `IEnterpriseSession` is the first object to be acquired; it precedes the creation of any other SDK objects. It combines both the name service and the client-side security objects.
- `com.crystaldecisions.sdk.framework.ISessionMgr`
  Represents the access point to the CMS.
- `com.crystaldecisions.sdk.framework.ITrustedPrincipal`
  Used to log onto CMS with Trusted Authentication. Requires a username and the server's shared secret.
- `com.crystaldecisions.sdk.occa.security.IEnterpriseLogonInformation`
  Stores extra logon information, including the reported and the resolved hostname and IP.
- `com.crystaldecisions.sdk.occa.security.ILogonTokenMgr`
  Used to create custom logon tokens, and retrieve the default logon token.

5.1.1 Authentication basics

Framework

Two core framework classes handle authentication.

- `ISessionMgr` class
  The `ISessionMgr` class acts as a controller during authentication to pass in user authentication information, and return an `IEnterpriseSession` object.
- `IEnterpriseSession` class
  The `IEnterpriseSession` class is the key identifier for the user's current session. It is typically used to retrieve the `IInfoStore` instance, based on a string identifier that is passed to the `IEnterpriseSession` interface.

> **Note**
>
> Unlike most other interfaces, `ISessionMgr` and `IEnterpriseSession` do not inherit from the `IInfoObject` interface.

Class	Package
`ISessionMgr`	`com.crystaldecisions.sdk.framework`
`IEnterpriseSession`	`com.crystaldecisions.sdk.framework`

Security protocols

You can configure the BI platform to work with the following kinds of security protocols during logon:

- `IsecEnterprise` class
  This class lets you to specify password settings and other security options for users who log on to the BI platform with a native Enterprise account.
- `IsecLDAP` class
  The LDAP protocol accesses users and groups from an LDAP server.
- `IsecWinAD` class
  The Windows AD protocol accesses users and groups from a Windows Active Directory server.

These classes allow you to map a group from an external system to the BI platform, enabling users who belong to that group to log on to the BI platform with a third-party alias. They also allow you to enable Single Sign-On (SSO) Authentication. For more information on SSO authentication, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

Each of these classes inherit from the `IInfoObject`class.

Class	Package
`IsecEnterprise`	`com.crystaldecisions.sdk.plugin.authentication.enterprise`
`IsecLDAP`	`com.crystaldecisions.sdk.plugin.authentication.ldap`

Class	Package
IsecWinAD	com.crystaldecisions.sdk.plugin.authentication.secwinad

5.1.2 Logging on to the CMS

The first step in working with this SDK is to log on to the Central Management Server (CMS) and retrieve an `IEnterpriseSession` object. All communication between the SDK and the CMS is done in the context of a user session. There are several ways to do this.

- Basic logon
  Log on a user to the CMS, using a username and password.
- Trusted Authentication
  Log on a user to the CMS from a trusted web application, without providing a password. Requires setting up a shared secret on the CMS and the web server.
- Logon tokens
  Log on a user to the CMS without username or password, using a logon token. Logon tokens are created at runtime from an existing user session, and can be passed to other web applications.
- Serialized sessions
  Access an existing user session without providing username or password. Serialized sessions are created at runtime from an existing user session, and can be passed to other web applications.
- Logon with auditing information
  A method that uses logon tokens and basic logon, but also passes an `IEnterpriseLogonInformation` object that stores auditing information, including the reported and the resolved hostname and IP. This information is then stored in the auditing database to show the hostname and IP of the users connecting to the CMS.

5.1.2.1 Basic logon

The most basic way to logon to the Central Management Server (CMS) is to use the `ISessionMgr.logon` method. This method requires your server information, username, password, and authentication type. It retrieves an `IEnterpriseSession` object.

Example

```
IEnterpriseSession basicLogon() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseSession enterpriseSession = sessionManager.logon("username",
"password", "<cms>:<port>", "secEnterprise");
  return enterpriseSession;
```

```
}
```

> **Note**
>
> - Replace **`<cms>`** with the name of the CMS machine and **`<port>`** with the port number of your CMS.
> - Replace **`username`** with a valid user name and **`password`** with the corresponding password.
> - Use the `ISessionMgr.getInstalledAuthIDs` method to determine the list of all available authentication types. Typical values include `secEnterprise`, `secLDAP`, and `secWinAD`.

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.exception.SDKException`

5.1.2.2 Using logon tokens

You can use a logon token to create a user session without prompting the user for credentials. For example, if a user has successfully logged on to the CMS, you can create a logon token from the user's session and pass it to another web application. This allows the user to access the web application without entering a username or password.

There are two types of logon tokens: default logon tokens and custom logon tokens. You can modify the hostname, lifetime, and number of logons for custom logon tokens. Default logon tokens cannot be modified. Both logon tokens create a new user session, which increases the license count in concurrent user licensing systems. If one user session is logged off, the other session will continue to be valid.

> **Note**
>
> Remember to URL-encode the token if you pass it to another web application through a URL.

Example

Creating a default logon token.

Shows how to create a default logon token and use it to create a second user session. `enterpriseSesssion2` is valid after `enterpriseSession1` is logged off.

```
void defaultToken() throws SDKException
{
  IEnterpriseSession enterpriseSession1 =
CrystalEnterprise.getSessionMgr().logon("username", "password", "<cms>:<port>",
"secEnterprise");
  ILogonTokenMgr tokenMgr = enterpriseSession1.getLogonTokenMgr();
  String defaultLogonToken = tokenMgr.getDefaultToken();
  IEnterpriseSession enterpriseSession2 =
CrystalEnterprise.getSessionMgr().logonWithToken(defaultLogonToken);
  enterpriseSession1.logoff();
  String CMSName = enterpriseSession2.getCMSName();
```

```
}
```

Example

Creating a custom logon token.

Shows how to create a custom logon token and use it to create a second user session. The logon token can only be used from machine `Host1` , is valid for 120 minutes, and can be used for 10 logons. `enterpriseSesssion2` is valid after `enterpriseSession1` is logged off.

```
void customToken() throws SDKException
{
  IEnterpriseSession enterpriseSession1 =
CrystalEnterprise.getSessionMgr().logon("username", "password", "<cms>:<port>",
"secEnterprise");
  ILogonTokenMgr tokenMgr = enterpriseSession1.getLogonTokenMgr();
  String customLogonToken = tokenMgr.createLogonToken("Host1",120,10);
  IEnterpriseSession enterpriseSession2 =
CrystalEnterprise.getSessionMgr().logonWithToken(customLogonToken);
  enterpriseSession1.logoff();
  String CMSName = enterpriseSession2.getCMSName();
}
```

> **Note**
>
> - Replace **`<cms>`** with the name of the CMS machine and **`<port>`** with the port number of your CMS.
> - Replace **`username`** with a valid username and **`password`** with the corresponding password.
> - Use the `ISessionMgr.getInstalledAuthIDs` method to determine the list of available authentication types. Typical values include `secEnterprise`, `secLDAP`, and `secWinAD`
> - Make sure to call the `ILogonTokenMgr.releaseToken` method when the token is no longer needed. This method throws an SDKException error when it fails to release a logon token. You can use the method as shown here:
>
> ```
> tokenMgr.releaseToken(customLogonToken);
> ```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.security.ILogonTokenMgr`

Related Information

Using serialized sessions [page 102]

5.1.2.3 Using serialized sessions

A serialized session is a string that represents a user session in the CMS. It is used to access an existing user session without prompting the user for credentials. Unlike logon tokens, using a serialized session does not increase the license count. If a user logs off the session created from the serialized session, the original session will also be logged off.

> **Note**
>
> Remember to URL-encode the serialized session if you pass it to another web application through a URL.

Example

Retrieving a serialized session.

Shows how to create a serialized session from an existing user session. The serialized session can be passed to another application. `enterpriseSession2` and `enterpriseSession1` reference the same user session, so `enterpriseSession2` is logged off when `enterpriseSession1.logoff()` is called.

```
void defaultToken() throws SDKException
{
  IEnterpriseSession enterpriseSession1 =
CrystalEnterprise.getSessionMgr().logon("username", "password", "<cms>:<port>",
"secEnterprise");
  String serializedSession = enterpriseSession1.getSerializedSession();
  IEnterpriseSession enterpriseSession2 =
CrystalEnterprise.getSessionMgr().getSession(serializedSession);
  enterpriseSession1.logoff();
}
```

Related Information

Using logon tokens [page 100]

5.1.2.4 Using trusted authentication

Users prefer to log on to the system once, without needing to provide passwords several times during a session. Trusted authentication provides a single sign-on solution (SSO) for integrating your BI platform authentication with third-party authentication solutions. Applications that have established trust with the Central Management Server (CMS) can use trusted authentication to allow users to log on without providing their passwords.

If trusted authentication has already in the Central Management Console (CMC) , you can use the `ISessionMgr` interface to pass the trusted user name and the shared secret and then log on to the CMS.

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
ITrustedPrincipal trustedPrincipal =
sessionMgr.createTrustedPrincipal("userName", "<cms>:<port>", "sharedSecret");
IEnterpriseSession enterpriseSession = sessionMgr.logon(trustedPrincipal);
```

> **Note**
>
> - Replace **`username`** with a valid user name and **`sharedSecret`** with the trusted authentication shared secret that is configured on the CMS.
> - Replace **`<cms>`** with the name of the CMS machine and **`<port>`** with the port number of your CMS.

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.ITrustedPrincipal`

> **Note**
>
> Trusted Authentication should not be enabled without HTTPS due to security reasons. If you have enabled Trusted Authentication without https, it is considered as breach of security as the URL is exposed to unauthorized users. To avoid security breach, the user's information can be validated with a valid certificate. For more information, refer to 1388240.

5.1.2.4.1 To set up Trusted Authentication

This example requires you to have an account with administrative rights that can log on to the Central Management Console (CMC).

This example shows how to set up a shared secret on the CMS can be used to establish trust with your custom application.

> **Note**
>
> Trusted Authentication should not be enabled without HTTPS due to security reasons. If you have enabled Trusted Authentication without https, it is considered as breach of security as the URL is exposed to unauthorized users. To avoid security breach, the user's information can be validated with a valid certificate. For more information, refer to 1388240.

1. Log on to the CMC with administrative rights.
2. Go to the *Authentication* management area.
3. Click the *Enterprise* option.
   The *Enterprise* dialog-box opens.
4. Scroll down until you see *Trusted Authentication*.

a. Click *Trusted Authentication is enabled*.
b. Click *New Shared Secret*.
   The following message is displayed:

   ```
   Shared secret key is generated and ready for download
   ```
c. Click *Download Shared Secret*.

   > **Note**
   >
   > The shared secret is used by the web server and the CMS to establish trust.

   The *File Download* dialog opens.
d. Click *Save* and point to following directory to save the `TrustedPrincipal.conf` file:

   <INSTALLDIR>\SAP BusinessObjects Enterprise XI 4.0\win32_x86
e. To specify the number of days that your shared secret is valid, enter a value for the *Shared Secret Validity Period* field.
f. Specify a timeout value for your trusted authentication requests.

   > **Note**
   >
   > The timeout value is the maximum amount of time, in milliseconds, that the clock on the client and clock and the CMS can differ. If you enter 0, the amount of time the two clock times can differ is unlimited. It is not recommended you set this value to 0, as this can increase your vulnerability to replay attacks.

5. Click *Update* to commit the shared secret.

   > **Note**
   >
   > All modifications to trusted authentication parameters are not audited by SAP BusinessObjects Business Intelligence platform. It is recommended that you manually back up all your trusted authentication information.

You can now use the API calls in your custom application to pass a trusted user name and the shared secret to the CMS for authentication.

5.1.2.5 Changing the Password of Non-Enterprise Users

BI platform Java SDK provides you with APIs to change the password of non-enterprise users including BW users.

> **Note**
>
> On successful password change, the user is logged out of all his current sessions.

To change the password, perform the following steps in your development environment:

1. Create an `IsessionMgr` class object instance using the syntax: `ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();`

2. Use the `changePassword` method on the `IsessionMgr` object with the indicated arguments:`sessionMgr.changePassword(“Username”, “Authentication”, “cms”, “oldpassword”, “newPassword”);`

5.1.2.6 Logon with auditing information

You can pass additional auditing information using either of these methods: `ISessionMgr.logonEx` and `ISessionMgr.logonWithTokenEx`. These methods are similar to `ISessionMgr.logon` and `ISessionMgr.logonWithToken`, except they have an additional parameter: an `IEnterpriseLogonInformation` object. This object stores auditing information, including the reported and the resolved hostname and IP. When these methods are run, a logon auditing event is triggered by the SDK, allowing the auditing database to store the hostname and IP of the users connecting to the Central Management Server (CMS).

Example

Logging on with `logonEx`.

```
IEnterpriseSession logonEx() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseLogonInformation logonInfo = sessionManager.createLogonInfo();
  logonInfo.setReportedHostname("computerName");
  logonInfo.setReportedIP("computerIP");
  IEnterpriseSession enterpriseSession = sessionManager.logonEx("username",
"password", "<cms>:<port>", "secEnterprise", logonInfo);
  return enterpriseSession;
}
```

Example

Logging on with `logonWithTokenEx`.

```
IEnterpriseSession logonWithcustomTokenEx() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseSession enterpriseSession = sessionManager.logon ("username",
"password", "<cms>:<port>", "secEnterprise");
  IEnterpriseLogonInformation logonInfo = sessionManager.createLogonInfo();
  logonInfo.setReportedHostname("computerName");
  logonInfo.setReportedIP("computerIP");
  String customLogonToken =
enterpriseSession.getLogonTokenMgr().createLogonToken("",120,100);
  enterpriseSession = sessionManager.logonWithTokenEx(customLogonToken,
logonInfo);
  return enterpriseSession;
}
```

> **Note**
>
> - Replace **`computerName`** with the name of the connecting computer and **`computerIP`** with the IP of the connecting computer.
> - Replace **`<cms>`** with the name of the CMS machine and **`<port>`** with the port number of your CMS.
> - Replace **`username`** with a valid username and **`password`** with the corresponding password.
> - Use the `ISessionMgr.getInstalledAuthIDs` method to determine the list of all available authentication types. Typical values include `secEnterprise`, `secLDAP`, and `secWinAD`

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `com.crystaldecisions.sdk.occa.security.IEnterpriseLogonInformation`
- `com.crystaldecisions.sdk.occa.security.ILogonTokenMgr`

Related Information

Basic logon [page 99]
Using logon tokens [page 100]

5.1.3 Logging off and freeing up licenses

Each active session requires a license. When the maximum number of available licenses is reached, additional users will be unable to log on to the BI platform. To maximize the number of available licenses, always log off any users who no longer require a session.

Example

This example assumes that you have already logged on and have stored the `IEnterpriseSession` object.

```
IEnterpriseSession enterpriseSession =
(IEnterpriseSession)session.getAttribute("MySession");
  session.removeAttribute("MySession");
  enterpriseSession.logoff();
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`

- `com.crystaldecisions.sdk.exception.SDKException`

5.2 Enhanced Credential Mapping

In BI 4.2.X and earlier releases, an administrator could save only one set of database credentials for every user in CMC.

This functionality requires the administrator to maintain the same credentials for all their different databases. In BI 4.3, you can save multiple set of database credentials for each user through data source references.

Data source reference

An administrator creates a data source reference in the BI platform. This data source reference is then used in the user properties where the administrator defines one set of the database credentials against it. This data source reference is then used as part of Credential Mapping, which is a mode of authentication available in the Connections. An administrator gets an option to select the data source reference of their choice when Credential Mapping is selected as the mode of authentication. In a similar way, an administrator can create multiple data source references if they've multiple databases connecting to the BI platform and define unique credentials for each user.

For more information on Data source reference, refer to Enhanced Credential Mapping section in the Business Intelligence Platform Administrator Guide.

Examples of API usage

New InfoObject type (=DatasourceReference) will be created in CMC specific page, with properties: Title and Description.

```
ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
IEnterpriseSession enterpriseSession = sessionManager.logon(CMS_USER, CMS_PAWD,
CMS_HOST, CMS_AUTH);
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");

IInfoObjects infoobjects = infostore.newInfoObjectCollection();

IDatasourceReference dr =
(IDatasourceReference)infoobjects.add(IDatasourceReference.KIND);
final String title = "MyNewDatasourceReference";
dr.setTitle(title);
dr.setDescription("My New Datasource Reference");
infostore.commit(infoobjects);
```

Add a username and password to a User account for a given Datasource Reference.

```
ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
IEnterpriseSession enterpriseSession = sessionManager.logon(CMS_USER, CMS_PAWD,
CMS_HOST, CMS_AUTH);
```

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
IInfoObjects res = infostore.query(SELECT STATIC, RELATIONSHIPS FROM
CI_SYSTEMOBJECTS WHERE SI_KIND='User' AND SI_NAME='user'");
IUser user = (IUser)res.get(0);
res = infostore.query("SELECT STATIC, RELATIONSHIPS FROM CI_SYSTEMOBJECTS WHERE
SI_KIND='DatasourceReference' AND SI_NAME='credMapping'");
IDatasourceReference dr = (IDatasourceReference)res.get(0);
final UserDatasourceReference mapping = new UserDatasourceReference("username",
"password");
user.addDatasourceReference(dr.getID(), dr.getCUID(), mapping);
user.save();
```

Get username / password for a Datasource Reference, from a UserInfo instance.

```
ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
IEnterpriseSession enterpriseSession = sessionManager.logon(CMS_USER, CMS_PAWD,
CMS_HOST, CMS_AUTH);
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
IInfoObjects res = infostore.query("SELECT STATIC, RELATIONSHIPS FROM
CI_SYSTEMOBJECTS WHERE SI_KIND='DatasourceReference' AND SI_NAME='credMapping'");
IDatasourceReference dr = (IDatasourceReference)res.get(0);
UserDatasourceReference mapping =
enterpriseSession.getUserInfo().getUserDatasourceReference(dr.getCUID());
```

Examples of CMS Queries

List of all DSRs	SELECT STATIC, RELATIONSHIPS FROM CI_SYSTEMOBJECTS WHERE SI_KIND='`DatasourceReference`'
List of DSRs related to Administrator User	SELECT SI_ID, SI_KIND, SI_NAME, FROM CI_SYSTEMOBJECTS WHERE CHILDREN("SI_NAME="`User-DatasourceReference`", "SI_ID='12")
List of Users related to Default DSR (for legacy compatibility)	SELECT SI_ID, SI_KIND, SI_NAME, SI_DATA FROM CI_SYSTEMOBJECTS WHERE PARENTS("SI_NAME="`User-DatasourceReference`", "SI_CUID='AWRx3R5vojJLgqMKj9X5qhI")

Hardcoded CUID for Default DSR is 'AWRx3R5vojJLgqMKj9X5qhI'.

5.3 Security

SAP BusinessObjects Business Intelligence platform provides a security framework for managing user accounts and user memberships within groups. You can control access to applications, servers, and actions by granting users sufficient rights to those objects.

This section explains how to use the SAP BusinessObjects Business Intelligence platform SDK to programmatically manage users and the security settings on objects.

Classes used to manage user and object security

- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
  Provides methods and properties that let you administer the behavior of users.
- `com.crystaldecisions.sdk.plugin.desktop.usergroup.IUserGroup`
  Provides methods and properties that let you to administer a user group.
- `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2`
  Allows you to assign roles, rights, and limits on a single object to users and user groups.
- `com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr`
  Allows you to assign roles, rights, and limits on a multiple objects to users and user groups. Use this interface to batch multiple requests to the Central Management Server (CMS) for increased efficiency and performance.
- `com.businessobjects.sdk.plugin.desktop.customrole.ICustomRole`
  Represents an access level, which is a set of rights frequently needed by users and user groups. An access level allows you to efficiently and uniformly apply common security settings on an object.

5.3.1 Users and Groups

5.3.1.1 To create a user group

User groups are represented by the `com.crystaldecisions.sdk.plugin.desktop.usergroup.IUserGroup` interface and contain information about their users and subgroups. You can programmatically construct new user groups with a few simple API calls.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Call the `newInfoObjectCollection` method of the `IInfoStore` interface to create an `IInfoObjects` collection.

   ```
   IInfoObjects newGroups = infostore.newInfoObjectCollection();
   ```

3. Call the `add` method of the collection to create an `IUserGroup` object.

   Use the constant `IUserGroup.KIND` and cast the result as an `IUserGroup` object.

   ```
   IUserGroup newUserGroup = (IUserGroup) newGroups.add(IUserGroup.KIND);
   ```

4. Set various properties of the new group, including its name and description.

   ```
   newUserGroup.setTitle("New Group Name");
   newUserGroup.setDescription("New group description.");
   ```

5. Commit the new user group collection back to the Central Management Server (CMS) to save the changes.

   ```
   infostore.commit(newGroups);
   ```

The new user group is created. To retrieve a group from the CMS, query from the `CI_SYSTEMOBJECTS` table using the appropriate `SI_KIND` object type identifier.

```
IInfoObjects groups = infostore.query("SELECT SI_ID, SI_NAME "
    + "FROM CI_SYSTEMOBJECTS WHERE SI_KIND='UserGroup' AND SI_NAME='New Group
Name'");
IUserGroup group = (IUserGroup) groups.get(0);
```

Replace `New Group Name` with the name of the user group you want to search for.

Example

This example creates a BI platform user group:

```
void createUserGroup(IEnterpriseSession enterpriseSession, IUser user) throws
SDKException
{
    IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");
    IInfoObjects newGroups = infostore.newInfoObjectCollection();

    IUserGroup newUserGroup = (IUserGroup) newGroups.add(IUserGroup.KIND);
    newUserGroup.setTitle("New Group Name");
    newUserGroup.setDescription("New group description.");

    Set usersOfNewGroup = newUserGroup.getUsers();
    usersOfNewGroup.add(user.getID());
    infostore.commit(newGroups);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.usergroup.IUser`
- `com.crystaldecisions.sdk.plugin.desktop.usergroup.IUserGroup`
- `java.util.Set`

Related Information

To create a user [page 111]

5.3.1.2 To create a user

Users are represented by the `com.crystaldecisions.sdk.plugin.desktop.user.IUser` interface and contain information about their security credentials, group membership, and identifying information. You can programmatically construct new users with a few simple API calls.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Call the `newInfoObjectCollection` method of the `IInfoStore` interface to create an `IInfoObjects` collection.

   ```
   IInfoObjects newUsers = infostore.newInfoObjectCollection();
   ```

3. Call the `add` method of the collection to create an `IUser` object.

   Use the constant `IUser.KIND`, and cast the result as an `IUser` object.

   ```
   IUser newUser = (IUser) newUsers.add(IUser.KIND);
   ```

4. Set various properties of the new user, including its name, password, and connection type.

   > **Note**
   >
   > `setNewPassword` is an administrative method which requires that the caller has been granted rights to change a user's password. Use the `changePassword` method to provide users with the opportunity to confirm their existing password and make a change. There is no method to retrieve a user's existing password.

   ```
   newUser.setTitle("username");
   newUser.setNewPassword("password");
   ```

5. Set the license information for this user.

   > **Note**
   >
   > The connection type determines whether the user is configured to consume a named user license (`IUser.NAMED`) or a concurrent user license (`IUser.CONCURRENT`) .

   ```
   newUser.setConnection(IUser.CONCURRENT);
   ```

6. Commit the new user collection back to the Central Management Server (CMS) to save the changes.

   ```
   infostore.commit(newUsers);
   ```

The new user is created. To retrieve a user from the CMS, query from the CI_SYSTEMOBJECTS table using the appropriate `SI_KIND` object type identifier.

```
IInfoObjects users = infostore.query("SELECT SI_ID, SI_NAME "
    + "FROM CI_SYSTEMOBJECTS WHERE SI_KIND='User' AND SI_NAME='username'");
IUser user = (IUser) users.get(0);
```

Replace `username` with the name of the user account you want to search for.

Example

This example creates a BI platform user account for a concurrent user:

```
void createUser(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects newUsers = infostore.newInfoObjectCollection();
  IUser newUser = (IUser) newUsers.add(IUser.KIND);

  newUser.setTitle("username");
  newUser.setNewPassword("password");
  newUser.setConnection(IUser.CONCURRENT);
  infostore.commit(newUsers);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CeSecurityCUID.LicenseRestriction`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`

Related Information

To create a user group [page 109]

5.3.1.3 To delete a user or user group

You can programmatically delete a user or user group.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Retrieve a user from the repository.

   ```
   IInfoObjects infoObjects = infostore.query("Select SI_ID From
   CI_SYSTEMOBJECTS "
       + "Where SI_KIND='USER'AND SI_NAME='username'");
   IInfoObject user = (IInfoObject)infoObjects.get(0);
   ```

   Replace `username` with the name of the user account you want to delete.

3. Call the `delete` method of the `IInfoObjects` collection to remove the particular user.

   ```
   infoObjects.delete(user);
   ```

4. Commit the modified user collection back to the Central Management Server (CMS) to save the changes.

```
infostore.commit(infoObjects);
```

Example

This example deletes a BI platform user account called `username` from the repository. A similar approach lets you delete user groups.

```
void deleteUser(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_ID From CI_SYSTEMOBJECTS
Where SI_KIND='USER' AND SI_NAME='username'");
  IInfoObject user = (IInfoObject)infoObjects.get(0);
  infoObjects.delete(user);
  infostore.commit(infoObjects);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

Related Information

To create a user [page 111]
To create a user group [page 109]

5.3.2 Setting Rights

5.3.2.1 Object security overview

Objects in the BI platform repository, whether it be a folder, document, user, server, or other type, possess a set of security information that specifies who has access to the object and what actions they can perform on that object. Both the `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2` and `com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr` interfaces, depending on your needs, allow you to programmatically assign roles, rights, and limits on an object to users and user groups.

This section defines some key security terms. For more information about object security, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

Roles (access levels)

A role (or access level) is a predefined set of security settings that you apply to an object. They contain the most common cases of security settings. For example, you can apply a role for the sales group, and a role for the marketing group. The sales role, when applied, grants its group members different rights to access the object than the marketing role.

The BI platform includes several predefined roles. You can also create custom roles (also referred to as custom access levels) and specify which rights are granted or denied for each role.

You can assign rights to users and groups without assigning them roles. But, as a best practice it is recommended that first you assign roles, and then assign more granular rights as necessary. Right settings defined in a role can be overridden by explicitly granting or denying the same granular right for the principal.

For more information, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide* for a list of the granular rights granted for each role.

Rights

A right is a permission to access an object or perform an action on it. For example, users who are granted the Delete Object right to a report are permitted to delete the report from the system.

There are two kinds of rights:

1. Boolean rights
   A right that is either granted or denied.
2. Limit rights
   A number that limits the ability to perform an action. For example, the maximum number of instances of a report.

Rights are granted, denied, or not specified.

The BI platform automatically makes available a set of system rights that are known to the Central Management Server (CMS). For example, the `CeSecurityID.Right.VIEW` system right enables control of a user's ability to view an object. In addition, each type of object provides a custom set of object-specific rights. For example, Crystal Reports have a `CeReportRightID.PRINT` right, which enables users to schedule the report to a printer.

Available rights

Available rights are the combination of system rights that are supplied through the CMS, and other rights that are specific to each type of object (also referred to as plugin-specific rights).

A right that is not applicable to one object type can still be set on such an object with the scope set to `CeSecurityOptions.RightScope.DESCENDANTS`. This right becomes effective on the children of this object that are of the applicable type. For example, the `CeSecurityID.Right.Schedule` right can be assigned to a folder object with the `CeSecurityOptions.RightScope.DESCENDANTS`scope. Although the folder itself cannot be scheduled, the folder may contain reports which can be scheduled, and the right will be inherited by all reports in that folder.

Explicit and Inherited rights

Rights can be explicitly assigned to an object, derived from an access level set on an object, or they can be inherited from other objects. For example, when you assign rights to a folder, its subfolders inherit the rights of the parent folder. And when you assign rights to a group, members who are added to that group inherit the rights of that group.

Effective Rights

The effective rights for a principal on an object are the result of combining all explicit rights, inherited rights, and rights derived from an access level. To determine the effective rights, the BI platform applies the following rules:

- Rights that are not specified (they are neither granted or denied), either explicitly or through inheritance, are denied.
- The most granular level at which a right is set determines its access. This means that rights that have been set explicitly always override rights that have been assigned through inheritance. And rights that are inherited from a parent object override rights inherited from further up the inheritance hierarchy. For example, a user can be denied access to a parent folder, but the user can be explicitly granted access to an object in that folder.
- If an explicitly set right for a user or group contradicts another explicitly set right on the same object, the right is denied. For example, a user belongs to access levels A and B. If access level A grants the right on an object and access level B denies it, then the user is denied access.

> **Note**
>
> Prior to BusinessObjects Enterprise XI 3.x, if there was a conflict between an inherited right and an explicit right, then the effective right was always denied, unless right inheritance was disabled.

The following table indicates how effective rights are calculated:

Explicit parent Object group right	Explicit child right	Effective right
Granted	Not specified	Granted
Not specified	Granted	Granted
Not specified	Not specified	Denied
Denied	Granted	Granted
Granted	Denied	Denied

Owner rights

You can set owner rights on an object for a user. For example, `CeSecurityID.Right.OWNER_DELETE` gives a user the right to delete an object instance, but only if the user is the owner.

Limits

A limit is a specific type of right that automates how object instances are cleaned up in the system. While other rights are governed by boolean values that specify whether the right is granted or denied to a principal on an object, a limit is governed by an integer value. At the object level, you can limit the number of instances that remain on the system for the object or for each user or group; you can also limit the number of days that an instance remains on the system for a user or group. As with other rights, limits can also be effective or explicit.

Principals

Principals are users or user groups that have been assigned rights, roles, and limits to a particular object. Principals are either explicit or effective.

- Explicit principals are users or user groups that have been specifically assigned certain types of access (rights, roles, or limits) to an object.
- Effective principals are users or user groups with calculated access to an object based on both explicit rights and inherited rights from a parent object.
  For example, a user who does not have explicit rights on an object may belong to a user group that does have explicit rights. Therefore, the user will be an effective principal for that object.

Related Information

Choosing a security API [page 116]

5.3.2.2 Choosing a security API

Checking and setting rights can have an impact on application performance. The API you choose determines how security information is accessed and what approaches are available to maintain application performance. There are two distinct entry-points to object security in the SDK:

- `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2`
  This interface is used to make security queries and modifications on a single object.
- `com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr`
  This interface is used to batch multiple security queries for multiple objects.

The following sections provide details on using each interface and improving performance.

ISecurityInfo2 interface

The `ISecurityInfo2` interface is retrieved from an `IInfoObject` object; therefore, it is suitable for checking and setting rights on a single object.

Checking rights on an object against a principal may require several network calls to the Central Management Server (CMS). To improve performance, consider doing the following:

- Use the `checkRights` method to check multiple rights using a single call rather than using multiple calls to the `checkRight` method.
- Check the SDK cache first for parameter values you can use when checking rights. All `checkRight` and `checkRights` methods provide a Boolean parameter where you can specify to use any cached security information first before querying the repository.

The general workflow for using the `ISecurityInfo2` interface to retrieve security information is as follows:

1. Retrieve an object, such as a document, folder, user, or server.
2. Use the `getSecurityInfo2` method inherited from the `IInfoObject` interface to get an `ISecurityInfo2` object.
3. Retrieve a list of principals for that object, using either the `getEffectivePrincipals` or `getExplicitPrincipals` methods of the `ISecurityInfo2` interface.
   Explicit principals are users or user groups with explicit rights assigned to a particular object. Effective principals are users or user groups with calculated rights to an object, based on both explicit rights and inherited rights from a parent object. These are known as effective rights.
4. Retrieve the list of rights, roles, or limits for each individual principal.
5. Retrieve information for an individual right, role, or limit.

ISecurityInfoMgr interface

The `ISecurityInfoMgr` interface is used to batch multiple security queries into a single call.
Batching security queries reduces the number of network trips to the CMS. Another advantage of the `ISecurityInfoMgr` interface is that it provides a way to check rights on objects to which users have not been granted view rights. The general workflow for using the `ISecurityInfoMgr` interface to retrieve security information is as follows:

1. Retrieve the `ISecurityInfoMgr` object from the current session using the `IEnterpriseSession.getSecurityInfoMgr` method.
2. Batch a series of security queries for multiple objects.
   - Use the `ISecurityInfoMgr.getSecCache` method to return an `ISecCacheController` object. The `ISecCacheController.batch` method is used to batch a series of non-administrative security queries on objects that the user session has rights to.
   - Use the `ISecurityInfoMgr.getSecCacheAdmin` method to return an `ISecCacheControllerAdmin` object. The `ISecCacheControllerAdmin.batch` method is used to batch a series of administrative security queries on multiple objects. The user session must have appropriate administrative rights on these objects to query for security information and set it.
3. Commit the batched queries to the CMS. The results are loaded into the SDK cache.
   Use the `ISecCacheController.commit` or `ISecCacheControllerAdmin.commit` methods.

> **Note**
>
> You must catch errors and exceptions that are thrown during the batch and commit process. Use the `rollback` method to restore the batch state.

4. Retrieve the results of the batched queries from the SDK cache.
   - Use the `ISecurityInfoMgr.getRights` method to return an `ISecRights` object containing non-administrative query results.
   - Use the `ISecurityInfoMgr.getRightsAdmin` method to return an `ISecRightsAdmin` object containing administrative query results (methods inherited from `ISecRightsQueryAdmin`).

> **Note**
>
> The SDK cache expires after 1 minute; therefore, you must load the cache with query results and retrieve the information before the cache expires to avoid an extra round trip to the CMS. In addition, because the cache size is limited by available memory, items in the cache will be released based on the memory required.

Aside from querying for security information, you can also use the `ISecRightsAdmin` interface to commit a batch of security changes to multiple objects using the `batch` method, various set and remove methods, and the `commit` method.

Retrieval of security information from the cache relies on the calls you make to preload the cache with the correct results. Certain information is retrieved as an `ISecurityResult` object, and you must cast the value of its `getResult` method to the correct interface. The following tables show related method calls between the load and retrieval steps, as well as the correct interface to cast results objects to.

ISecCacheController (cache load)	ISecRights (retrieval)	Cast ISecurityResult.getResult As
`cacheLimit`	`checkLimit`	`ILimitResult`
`cacheRight`	`checkRight`	Not applicable. `checkRight` returns an integer.

ISecCacheControllerAdmin (cache load)	ISecRightsQueryAdmin (retrieval)	Cast ISecurityResult.getResult As
`cacheExplicitLimits`	`getExplicitLimits`	`ISecurityLimitAdmin[]`
`cacheExplicitPrincipals`	`getExplicitPrincipals`	`ISecurityPrincipal[]`
`cacheExplicitRights`	`getExplicitRights`	`ISecurityRightAdmin[]`
`cacheExplicitRoles`	`getExplicitRoles`	`ISecurityRoleAdmin[]`
`cacheKnownRightsByPlugin`	`getKnownRightsByPlugin`	`IPluginBasedRightIDs`
`cacheKnownSecurityInfo`	`getKnownRoles`	`IRoleID[]`
`cacheLimit`	`checkLimit`	`ILimitResult`
`cachePrincipals`	`getPrincipals`	`ISecurityPrincipal[]`
`cacheRight`	`checkRight`	Not applicable. `checkRight` returns an integer.
`cacheSecurityInfo(int)`	`getSecurityInfo(int)`	`ISecurityInfoResult`

ISecCacheControllerAdmin (cache load)	ISecRightsQueryAdmin (retrieval)	Cast ISecurityResult.getResult As
`cacheSecurityInfo(int, int)`	`getSecurityInfo(int, int)`	`ISecurityInfoResult`

Related Information

Setting rights on a single object [page 120]
Setting rights on multiple objects [page 128]

5.3.2.3 Type-specific rights

BI platform objects contain a set of rights that can be granted to users and user groups. You can verify whether a particular user or user group has been granted a right on an object using a combination of the `getKnownRightsByPlugin` and `checkRight` methods of the `ISecurityInfo2` or `ISecRightsQueryAdmin` (via `ISecurityInfoMgr`) interfaces. The `getKnownRightsByPlugin` method returns an `IPluginBasedRightIDs` object, which organizes the available rights according to object type-specific categories (`ProgID` values, for example `com.crystaldecisions.sdk.plugin.desktop.pdf.IPDF.PROGID`).

For example, suppose you check the rights on a folder object. Examples of the categories you will find rights organized by in the `IPluginBasedRightIDs` interface include the following:

- `Any`
  General rights that can be applied to the folder or any object within the folder.
- `CrystalEnterprise.Folder`
  Rights that can be applied to the folder or any sub-folders.
- `CrystalEnterprise.Pdf`
  Rights that can be applied to any PDF files contained in the folder.

Type-specific rights apply to all objects. For example, after retrieving the available rights for a Crystal Reports file, some of the categories within the retrieved `IPluginBasedRightIDs` object include the following:

- `Any` - general rights that can be applied to the report or any child instance of the report.
- `CrystalEnterprise.Report` - rights that can be applied to the report or any child instances of the report.
- `CrystalEnterprise.Pdf` - rights that can be applied to any child PDF instances of the report.

After you have an `IPluginBasedRightIDs` object, iterate through all the object types to retrieve an `IRightID` object representing the elements of a right. Use these elements to construct a `RightDescriptor` object representing the right, and then call the `checkRight` method on this object to confirm whether the right has been granted.

> **Note**
>
> For more information about type-specific rights, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

Related Information

To check rights on an object [page 123]

5.3.2.4 Setting rights on a single object

The `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2` interface allows you to assign roles, rights, and limits on a single object to users and user groups. This section shows you how to program common workflows using this interface.

Related Information

Setting rights on multiple objects [page 128]

5.3.2.4.1 To retrieve security information for an object

Security information for a single object can be accessed through the `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2` interface.

In the following example, you iterate through each user or user group that has effective roles, rights, and limits on a Crystal Reports file, and print out the security information.

> **Note**
>
> If you want to batch a series of security queries for multiple objects into one CMS call, use the `ISecurityInfoMgr` interface and retrieve the results from the cache.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the report from the repository.

```
IInfoObjects infoObjects = infostore.query("Select SI_ID From CI_INFOOBJECTS
"
    + "Where SI_NAME='World Sales Report' And SI_INSTANCE=0");
IInfoObject report = (IInfoObject)infoObjects.get(0);
```

3. Call the `getSecurityInfo2` method to retrieve the `ISecurityInfo2` object for the report.

```
ISecurityInfo2 securityInfo = report.getSecurityInfo2();
```

4. Call the `getEffectivePrincipals` method to retrieve all the users and user groups who have effective rights on the report.

```
IEffectivePrincipals effectivePrincipals =
securityInfo.getEffectivePrincipals();
```

5. Use the iterator to access each individual principal from the `IEffectivePrincipals` collection.

```
Iterator it = effectivePrincipals.iterator();
while (it.hasNext())
{
    IEffectivePrincipal effectivePrincipal = (IEffectivePrincipal)it.next();
    ...
}
```

6. Retrieve all effective roles (access levels) for each principal that has been assigned to the report.
   a. Call the `getRoles` method to get the collection of effective roles.

   ```
   IEffectiveRoles effectiveRoles = effectivePrincipal.getRoles();
   ```

   b. Use an iterator to retrieve each individual role.

   After retrieving a single role, use the individual accessor methods of the `IEffectiveRole` interface to access desired information about the role.

   ```
   Iterator roleIT = effectiveRoles.iterator();
   while (roleIT.hasNext())
   {
       IEffectiveRole effectiveRole = (IEffectiveRole)roleIT.next();
       ...
   }
   ```

7. Retrieve all effective rights for each principal that has been assigned to the report.
   a. Call the `getRights` method to get the collection of effective rights.

   ```
   IEffectiveRights effectiveRights = effectivePrincipal.getRights();
   ```

   b. Use an iterator to retrieve each individual right.

   After retrieving a single right, use the individual accessor methods of the `IEffectiveRight` interface to access information about the right.

   ```
   Iterator rightIT = effectiveRights.iterator();
   while (rightIT.hasNext())
   {
       IEffectiveRight effectiveRight = (IEffectiveRight)rightIT.next();
       ...
   }
   ```

8. Retrieve all effective limits for each principal that has been assigned to the report.
   a. Call the `getlimits` method to get the collection of effective limits.

   ```
   IEffectiveLimits effectiveLimits = effectivePrincipal.getLimits();
   ```

   b. Use an iterator to retrieve each individual limit.

   After retrieving a single limit, use the individual accessor methods of the `IEffectiveLimit` interface to access information about the limit.

   ```
   Iterator limitIT = effectiveLimits.iterator();
   while (limitIT.hasNext())
   {
       IEffectiveLimit effectiveLimit = (IEffectiveLimit)limitIT.next();
       ...
   }
   ```

Example

This example retrieves the effective security information for a report and returns a String object that displays the results in an HTML table.

```
String viewEffectiveSecurityInfo(IEnterpriseSession enterpriseSession) throws
SDKException, IOException
{
  String resultString = "";

  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_ID From CI_INFOOBJECTS
Where SI_NAME='World Sales Report' And SI_INSTANCE=0");


  IInfoObject report = (IInfoObject)infoObjects.get(0);
  ISecurityInfo2 securityInfo = report.getSecurityInfo2();
  IEffectivePrincipals effectivePrincipals =
securityInfo.getEffectivePrincipals();
  Iterator it = effectivePrincipals.iterator();
  while (it.hasNext())
  {
    IEffectivePrincipal effectivePrincipal = (IEffectivePrincipal)it.next();
    IEffectiveRoles effectiveRoles = effectivePrincipal.getRoles();
    Iterator roleIT = effectiveRoles.iterator();

    resultString += "<b>Effective Principal: </b>" +
effectivePrincipal.getName() + "<br>";
    resultString += "<b>Effective Roles:</b>";

    resultString += "<table  border='1'><tr><th>getTitle</th><th>isInherited</
th></tr>";
    while (roleIT.hasNext())
    {
      IEffectiveRole effectiveRole = (IEffectiveRole)roleIT.next();
      resultString += "<tr><td>" + effectiveRole.getTitle() + "</td><td>" +
effectiveRole.isInherited() + "</td></tr>";
    }
    resultString += "</table>";

    IEffectiveRights effectiveRights = effectivePrincipal.getRights();
    Iterator rightIT = effectiveRights.iterator();

    resultString += "<b>Effective Rights:</b>";
    resultString += "<table border='1'><tr><th>getDescription</
th><th>isInherited</th></tr>";
    while (rightIT.hasNext())
    {
      IEffectiveRight effectiveRight = (IEffectiveRight)rightIT.next();
      resultString += "<tr><td>" +
effectiveRight.getDescription(java.util.Locale.ENGLISH) + "</td><td>" +
effectiveRight.isInherited() + "</td></tr>";
    }
    resultString += "</table>";

    IEffectiveLimits effectiveLimits = effectivePrincipal.getLimits();
    Iterator limitIT = effectiveLimits.iterator();

    resultString += "<b>Effective Limits:</b>";
    resultString += "<table border='1'><tr><th>getDescription</th><th>getValue</
th></tr>";
    while (limitIT.hasNext())
    {
      IEffectiveLimit effectiveLimit = (IEffectiveLimit)limitIT.next();
```

```
      resultString += "<tr><td>" +
effectiveLimit.getDescription(java.util.Locale.ENGLISH) + "</td><td>" +
effectiveLimit.getValue() + "</td></tr>";
    }
    resultString += "</table> <hr>";
  }
  return resultString;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IEffectiveLimit`
- `com.crystaldecisions.sdk.occa.infostore.IEffectiveLimits`
- `com.crystaldecisions.sdk.occa.infostore.IEffectivePrincipal`
- `com.crystaldecisions.sdk.occa.infostore.IEffectivePrincipals`
- `com.crystaldecisions.sdk.occa.infostore.IEffectiveRight`
- `com.crystaldecisions.sdk.occa.infostore.IEffectiveRights`
- `com.crystaldecisions.sdk.occa.infostore.IEffectiveRole`
- `com.crystaldecisions.sdk.occa.infostore.IEffectiveRoles`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2`
- `java.io.IOException`
- `java.util.Iterator`

Related Information

Object security overview [page 113]
To retrieve security information for multiple objects [page 129]

5.3.2.4.2 To check rights on an object

In this example, you iterate through all available rights on a report, check whether a user has been granted each right, and print out the results.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the report from the repository.

```
IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_NAME='World Sales Report' And SI_INSTANCE=0");
IInfoObject report = (IInfoObject)reports.get(0);
```

3. Call the `getSecurityInfo2` method to retrieve the `ISecurityInfo2` object for the report.

```
ISecurityInfo2 securityInfo = report.getSecurityInfo2();
```

4. Call the `getKnownRightsByPlugin` method to retrieve an `IPluginBasedRightIDs` object.

```
IPluginBasedRightIDs pluginBasedRightIDs =
securityInfo.getKnownRightsByPlugin();
```

5. Call the `getPluginRights` method of the `IPluginBasedRightIDs` object and iterate through each type-specific set of rights.

   The `getPluginRights` method returns a `java.util.Map` object. The keys of the `Map` object are the `ProgID` strings (for example, `com.crystaldecisions.sdk.plugin.desktop.pdf.IPDF.PROGID`) representing the different object types. The values are `java.util.Set` objects containing `IRightID` objects for each individual, type-specific right.

```
java.util.Map pluginRights = pluginBasedRightIDs.getPluginRights();
java.util.Iterator it = pluginRights.keySet().iterator();
while (it.hasNext())
{
    Object progID = it.next();
    ...
}
```

6. Obtain the set of `IRightID` objects for each type and iterate through the set.

```
java.util.Set rightIDs = (java.util.Set)pluginRights.get(progID);
java.util.Iterator rIt = rightIDs.iterator();

while (rIt.hasNext())
{
    IRightID key = (IRightID)rIt.next();
    ...
}
```

7. For each `IRightID` object, construct a new `RightDescriptor` object representing which right to check.

   Each `IRightID` object contains a base ID for the right, the object type the right belongs to, and a flag indicating if the right applies to the owner of the object. This information is used as input to the `RightDescriptor` constructor.

```
RightDescriptor rightToCheck = new
RightDescriptor(key.getBaseID(), key.getRightPluginKind(), key.isOwner(),
CeSecurityOptions.RightScope.CURRENT_OBJECT, progID);
```

   The fourth parameter of the constructor specifies the scope, which is the level the right is assigned to. `CeSecurityOptions.RightScope.CURRENT_OBJECT` indicates that the right applies to the current object where the right is being set, whereas `CeSecurityOptions.RightScope.DESCENDANTS` indicates that the right applies to descendants of the current object. The last parameter specifies which types of objects the right applies to. For example, a right assigned to a folder and constructed as `new RightDescriptor(CeSecurityID.Right.View, null, false, CeSecurityOptions.RightScope.DESCENDANTS, IReport.KIND)` applies only to all Crystal Report objects contained in the folder.

8. Call the `checkRight` method using as input the newly constructed `RightDescriptor` object and the ID of the user as input.

```
securityInfo.checkRight(rightToCheck, user.getID(), true)
```

Example

This example iterates through all available rights that can be set on a report, and checks whether a particular user has these rights granted. The results are formatted in an HTML table and returned as a String object.

```
String checkRights(IEnterpriseSession enterpriseSession, IUser user) throws
SDKException, IOException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String resultString = "Rights for user '" + user.getTitle() + "' for 'World
Sales Report'.<p>";

  IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS Where
SI_NAME='World Sales Report' And SI_INSTANCE=0");
  IInfoObject report = (IInfoObject)reports.get(0);
  ISecurityInfo2 securityInfo = report.getSecurityInfo2();

  IPluginBasedRightIDs pluginBasedRightIDs =
securityInfo.getKnownRightsByPlugin();
  Map pluginRights = pluginBasedRightIDs.getPluginRights();
  Iterator it = pluginRights.keySet().iterator();

  while (it.hasNext())
  {
    Object progID = it.next();
    Set rightIDs = (java.util.Set)pluginRights.get(progID);
    Iterator rIt = rightIDs.iterator();

    resultString += "<p><p><b>Plugin Type: </b>" + progID.toString() + "<br>";
    resultString += "<table><tr><td><b>Right</b></td><td><b>Is granted</b></td></
tr>";
    while (rIt.hasNext())
    {
      IRightID key = (IRightID)rIt.next();
      RightDescriptor rightToCheck = new
RightDescriptor(key.getBaseID(), key.getRightPluginKind(), key.isOwner(),
CeSecurityOptions.RightScope.CURRENT_OBJECT, progID);
      boolean hasRight = securityInfo.checkRight(rightToCheck, user.getID(),
true);
      resultString += "<tr><td>" + key.toString()+ "</td><td>" + hasRight + "</
td></tr>";
    }
    resultString += "</table>";
  }
  return resultString;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IPluginBasedRightIDs`
- `com.crystaldecisions.sdk.occa.infostore.IRightID`
- `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2`
- `com.crystaldecisions.sdk.occa.infostore.RightDescriptor`
- `com.crystaldecisions.sdk.occa.security.CeSecurityOptions`

- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
- `java.io.IOException`
- `java.util.Iterator`
- `java.util.Map`
- `java.util.Set`

Related Information

Type-specific rights [page 119]
Object security overview [page 113]
To check rights on multiple objects [page 133]
To set an explicit right on an object [page 126]

5.3.2.4.3 To set an explicit right on an object

While access levels (roles) are used to apply common sets of rights on objects to many users and groups, individual rights can also be explicitly assigned on objects to users and groups.

In this example, you grant a user an explicit right to refresh the data in a Crystal report.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the report from the repository.

```
IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_NAME='World Sales Report' And SI_INSTANCE=0");
IInfoObject report = (IInfoObject)reports.get(0);
```

3. Call the `getSecurityInfo2` method to retrieve the `ISecurityInfo2` object for the report.

```
ISecurityInfo2 securityInfo = report.getSecurityInfo2();
```

4. Add a user to the collection of explicit principals on the report.

> **Note**
> This example assumes that you have retrieved the user account.

```
IExplicitPrincipal explicitPrincipal = explicitPrincipals.add(user.getID());
```

5. Create a new `RightDescriptor` object that represents the right to refresh report data on demand.

   The `CeReportRightID` interface contains constant field values that specify Crystal Reports security rights. The `CeSecurityID.Right` interface defines general object rights. Rights that are specific to

other object types are specified by interfaces such as `CeWebIRightID`, `CeFullClientRightID`, and `CeUniverseRightID`.

```
RightDescriptor refreshReportRight = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false);
```

> **Note**
>
> Rights created using this constructor will add two rights to an object, one with a `CeSecurityOptions.RightScope.CURRENT_OBJECT` scope and one with a `CeSecurityOptions.RightScope.DESCENDANTS` scope. Use this constructor only to set rights. To retrieve rights information, you must use the `RightDescriptor` constructor that also requires scope and applicable object type parameters.

6. Add the newly created right to the collection of explicit rights for the principal.

```
IExplicitRights reportRights = explicitPrincipal.getRights();
IExplicitRight reportRight = reportRights.add(refreshReportRight);
```

7. Use the `setGranted` method to grant the right to the user.

```
reportRight.setGranted(true);
```

8. Commit the modified report collection back to the Central Management Server (CMS) to save the changes.

```
infostore.commit(reports);
```

> **Note**
>
> It is the report that was changed and therefore committed back to the CMS, not the user account . Rights are set on objects such as reports and folders, instead of on the users and groups who access them.

Example

This example grants a user an explicit right to refresh the data in a report called `World Sales Report.rpt`.

```
void setExplicitRight(IEnterpriseSession enterpriseSession, IUser user) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS Where
SI_NAME='World Sales Report' And SI_INSTANCE=0");
  IInfoObject report = (IInfoObject)reports.get(0);

  ISecurityInfo2 securityInfo = report.getSecurityInfo2();
  IExplicitPrincipals explicitPrincipals = securityInfo.getExplicitPrincipals();
  IExplicitPrincipal explicitPrincipal = explicitPrincipals.add(user.getID());

  RightDescriptor refreshReportRight = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false);
  IExplicitRights reportRights = explicitPrincipal.getRights();
  IExplicitRight reportRight = reportRights.add(refreshReportRight);
  reportRight.setGranted(true);
  infostore.commit(reports);
```

```
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IExplicitPrincipal`
- `com.crystaldecisions.sdk.occa.infostore.IExplicitPrincipals`
- `com.crystaldecisions.sdk.occa.infostore.IExplicitRight`
- `com.crystaldecisions.sdk.occa.infostore.IExplicitRights`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISecurityInfo2`
- `com.crystaldecisions.sdk.occa.infostore.RightDescriptor`
- `com.crystaldecisions.sdk.plugin.desktop.report.CeReportRightID`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`

Related Information

Object security overview [page 113]
To set an explicit right on multiple objects [page 136]
To check rights on an object [page 123]

5.3.2.5 Setting rights on multiple objects

The `com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr` interface allows you to assign roles, rights, and limits on multiple objects to users and user groups. This section shows you how to use this interface to batch multiple requests to the Central Management Server (CMS) for increased efficiency and performance.

Related Information

Setting rights on a single object [page 120]

5.3.2.5.1 To retrieve security information for multiple objects

If you want to batch a series of security queries for multiple objects into one CMS call, use the `com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr` interface and retrieve the results from the cache.

In the following example, you retrieve the effective security information (roles, rights, and limits) on all Crystal reports in a folder for a particular user. The results are displayed in an HTML table.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Call the `getSecurityInfoMgr` method to retrieve the `ISecurityInfoMgr` object for the session.

```
ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
```

3. Retrieve all reports in the folder from the repository.

```
IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_PARENT_FOLDER=" + folder.getID() + "
    + "And SI_KIND='CrystalReport' And SI_INSTANCE=0");
```

4. Batch a series of queries to check rights, and commit them to the CMS.
   a. Call the `getSecCacheAdmin` method of the `ISecurityInfoMgr` interface to retrieve an `ISecCacheControllerAdmin` object.

```
ISecCacheControllerAdmin secCacheControllerAdmin =
secInfoMgr.getSecCacheAdmin();
```

   The `ISecCacheControllerAdmin` interface is used to batch multiple security requests and load the results into the SDK cache.

   b. Call the `batch` method to start the batching process.

```
secCacheControllerAdmin.batch();
```

   c. Create a `try` block to rollback all batch queries in the event of an error.

```
try {
    ...
}
catch (Throwable e) {
    secCacheControllerAdmin.rollback();
    ...
}
```

   d. Iterate through each report in the folder and call the `cacheSecurityInfo` method.

```
for (Object object : reports)
{
    IInfoObject report = (IInfoObject)object;
    secCacheControllerAdmin.cacheSecurityInfo(report.getID(),
user.getID());
}
```

The `cacheSecurityInfo` method is used to request all the effective security information (roles, rights, and limits) on an object for the given principal user. Each call to the method batches additional requests but does not submit a request to the CMS.

e. Call the `commit` method to submit the queries and load the results into the SDK cache.

```
secCacheControllerAdmin.commit();
```

The results of the batched queries are now loaded into the SDK cache, and can be retrieved using the `ISecurityInfoMgr.getRightsAdmin` method.

> **Note**
>
> The SDK cache expires after 1 minute; therefore, you must load the cache with query results and retrieve the information before the cache expires to avoid an extra round trip to the CMS. In addition, because the cache size is limited by available memory, items in the cache will be released based on the memory required.

5. Call the `getRightsAdmin` method of the `ISecurityInfoMgr` interface to retrieve an `ISecRightsAdmin` object.

```
ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
```

6. Create a loop to step through each report.

```
for (Object object : reports)
{
    IInfoObject report = (IInfoObject)object;
    ...
}
```

7. For each report, call the `getSecurityInfo` method of the `ISecRightsAdmin` object to retrieve the results of the security query.

```
ISecurityResult securityResult =
secRightsAdmin.getSecurityInfo(report.getID(), user.getID());
```

The results are retrieved as an `ISecurityResult` object, which consists of two main parts: a generic `java.lang.Object` object containing the query results and a flag indicating the status of the request.

8. Call the `getResult` method of the `ISecurityResult` object, and cast the result as a `ISecurityInfoResult` object.

```
ISecurityInfoResult securityInfoResult =
(ISecurityInfoResult)securityResult.getResult();
```

In this example, `ISecurityInfoResult` is the correct interface to cast the result to, which provides access to all the roles, rights, and limits on the report.

9. Retrieve all effective roles (access levels) on each report for the given principal user.

a. Call the `getRoles` method to get the collection of effective roles.

```
ISecurityRoleAdmin[] securityRoleAdminArray =
securityInfoResult.getRoles();
```

b. Create a loop to step through each role on the report.

```
for (ISecurityRoleAdmin securityRoleAdmin : securityRoleAdminArray)
{
    ...
```

```
}
```

After retrieving a single role, use the individual accessor methods of the `ISecurityRoleAdmin` interface to access desired information about the role.

10. Retrieve all effective rights on each report for the given principal user.
    a. Call the `getRights` method to get the collection of effective rights.

    ```
    ISecurityRightAdmin[] securityRightAdminArray =
    securityInfoResult.getRights();
    ```

    b. Create a loop to step through each right on the report.

    ```
    for (ISecurityRightAdmin securityRightAdmin : securityRightAdminArray)
    {
        ...
    }
    ```

    After retrieving a single right, use the individual accessor methods of the `ISecurityRightAdmin` interface to access desired information about the right.

11. Retrieve all effective limits on each report for the given principal user.
    a. Call the `getLimits` method to get the collection of effective limits.

    ```
    ISecurityLimitAdmin[] securityLimitAdminArray =
    securityInfoResult.getLimits();
    ```

    b. Create a loop to step through each limit on the report.

    ```
    for (ISecurityLimitAdmin securityLimitAdmin : securityLimitAdminArray)
    {
        ...
    }
    ```

    After retrieving a single limit, use the individual accessor methods of the `ISecurityRightAdmin` interface to access desired information about the limit.

Example

This example retrieves the effective security information (roles, rights, and limits) on all Crystal reports in a folder for a particular user. The results are formatted in an HTML table and returned as a String object.

```
String  viewSecurityInfo(IEnterpriseSession enterpriseSession, IUser user,
IFolder folder) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String resultString = "";
  ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
  IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS
Where SI_PARENT_FOLDER=" + folder.getID() + "And SI_KIND='CrystalReport' And
SI_INSTANCE=0");
  ISecCacheControllerAdmin secCacheControllerAdmin =
secInfoMgr.getSecCacheAdmin();
  secCacheControllerAdmin.batch();
  try
  {
    for (Object object : reports)
    {
      IInfoObject report = (IInfoObject)object;
```

```
      secCacheControllerAdmin.cacheSecurityInfo(report.getID(), user.getID());
    }
    secCacheControllerAdmin.commit();
  }
  catch (Throwable e)
  {
    secCacheControllerAdmin.rollback();
  }

  ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
  for (Object object : reports)
  {
    IInfoObject report = (IInfoObject)object;
    resultString += "<b>Report: <i>" + report.getTitle() + "</i></b><br>";

    ISecurityResult securityResult =
secRightsAdmin.getSecurityInfo(report.getID(), user.getID());
    ISecurityInfoResult securityInfoResult =
(ISecurityInfoResult)securityResult.getResult();
    resultString += "<b>Roles:</b>";
    resultString += "<table border='1'><tr><th>getTitle</
th><th>getPrincipalName</th><th>isInherited</th></tr>";
    ISecurityRoleAdmin[] securityRoleAdminArray = securityInfoResult.getRoles();
    for (ISecurityRoleAdmin securityRoleAdmin : securityRoleAdminArray)
    {
      resultString += "<tr><td>" + securityRoleAdmin.getTitle() +
"</td><td>" + securityRoleAdmin.getPrincipalName() + "</td><td>" +
securityRoleAdmin.isInherited() + "</td></tr>";
    }
    resultString += "</table>";
    resultString += "<b>Rights:</b><table border='1'><tr><th>getDescription</
th><th>isInherited</th></tr>";
    ISecurityRightAdmin[] securityRightAdminArray =
securityInfoResult.getRights();
    for (ISecurityRightAdmin securityRightAdmin : securityRightAdminArray)
    {
      String rightDescription =
securityRightAdmin.getDescription(java.util.Locale.ENGLISH);
      resultString += "<tr><td>" + rightDescription + "</td><td>" +
securityRightAdmin.isInherited() + "</td></tr>";
    }
    resultString += "</table>";

    resultString += "<b>Limits:</b><table border='1'><tr><th>getDescription</
th><th>getValue</th></tr>";
    ISecurityLimitAdmin[] securityLimitAdminArray =
securityInfoResult.getLimits();
    for (ISecurityLimitAdmin securityLimitAdmin : securityLimitAdminArray)
    {
      resultString += "<tr><td>" +
securityLimitAdmin.getDescription(java.util.Locale.ENGLISH) + "</td><td>" +
securityLimitAdmin.getValue() + "</td></tr>";
    }
    resultString += "</table><hr>";
  }
  return resultString;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

- `com.crystaldecisions.sdk.occa.infostore.ISecurityLimitAdmin`
- `com.crystaldecisions.sdk.occa.infostore.ISecurityRightAdmin`
- `com.crystaldecisions.sdk.occa.infostore.ISecurityRoleAdmin`
- `com.crystaldecisions.sdk.occa.security.ISecCacheControllerAdmin`
- `com.crystaldecisions.sdk.occa.security.ISecRightsAdmin`
- `com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr`
- `com.crystaldecisions.sdk.occa.security.ISecurityInfoResult`
- `com.crystaldecisions.sdk.occa.security.ISecurityResult`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
- `java.io.IOException`

Related Information

Object security overview [page 113]
To retrieve security information for an object [page 120]

5.3.2.5.2 To check rights on multiple objects

In this example, you iterate through all available Crystal reports in a folder, check whether a particular user has the right to refresh data (`CeReportRightID.REFRESH_ON_DEMAND`) on each report, and print out the results. This example assumes that the session has administrative rights on the reports and uses the `ISecurityInfoMgr` interface to batch a series of queries, load the information into the SDK cache, and retrieve the results.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Call the `getSecurityInfoMgr` method to retrieve the `ISecurityInfoMgr` object for the session.

```
ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
```

3. Retrieve all reports in the folder from the repository.

```
IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_PARENT_FOLDER=" + folder.getID() "
    + "And SI_KIND='CrystalReport' And SI_INSTANCE=0");
```

4. Construct a new `RightDescriptor` object representing the right to refresh a report.

```
RightDescriptor rightToCheck = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false,
CeSecurityOptions.RightScope.CURRENT_OBJECT, "CrystalEnterprise.Report");
```

   The fourth parameter of the constructor specifies the scope, which is the level the right is assigned to. `CeSecurityOptions.RightScope.CURRENT_OBJECT` indicates that the right applies to the

current object where the right is being set, whereas `CeSecurityOptions.RightScope.DESCENDANTS` indicates that the right applies to descendants of the current object. The last parameter specifies which types of objects the right applies to. For example, a right assigned to a folder and constructed as `new RightDescriptor(CeSecurityID.Right.View, null, false, CeSecurityOptions.RightScope.DESCENDANTS, IReport.KIND)` applies only to all Crystal report objects contained in the folder.

5. Batch a series of queries to check rights, and commit them to the CMS.
   a. Call the `getSecCacheAdmin` method of the `ISecurityInfoMgr` interface to retrieve an `ISecCacheControllerAdmin` object.

      ```
      ISecCacheControllerAdmin secCacheControllerAdmin =
      secInfoMgr.getSecCacheAdmin();
      ```

      The `ISecCacheControllerAdmin` interface is used to batch multiple security requests and load the results into the SDK cache.
   b. Call the `batch` method to start the batching process.

      ```
      secCacheControllerAdmin.batch();
      ```

   c. Create a `try` block to rollback all batch queries in the event of an error.

      ```
      try {
          ...
      }
      catch (Throwable e) {
          secCacheControllerAdmin.rollback();
          ...
      }
      ```

   d. Iterate through each report in the folder, and call the `cacheRight` method.

      ```
      for (Object object : reports)
      {
          IInfoObject report = (IInfoObject)object;
          secCacheControllerAdmin.cacheRight(rightToCheck, user.getID(),
      report.getID());
      }
      ```

      The `cacheRight` method is used to request whether each principal user has the indicated right on the report. Each call to the method batches additional requests but does not submit a request to the CMS.
   e. Call the `commit` method to submit the queries and load the results into the SDK cache.

      ```
      secCacheControllerAdmin.commit();
      ```

   The results of the batched queries are now loaded into the SDK cache, and can be retrieved using the `ISecurityInfoMgr.getRightsAdmin` method.

   > **Note**
   >
   > The SDK cache expires after 1 minute; therefore, you must load the cache with query results and retrieve the information before the cache expires to avoid an extra round trip to the CMS. In addition, because the cache size is limited by available memory, items in the cache will be released based on the memory required.

6. Retrieve the results of the batched queries from the SDK cache.

a. Call the `getRightsAdmin` method of the `ISecurityInfoMgr` interface to retrieve an `ISecRightsAdmin` object.

```
ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
```

The `ISecRightsAdmin` interface in this example is used to retrieve loaded results from the SDK cache. This interface can also be used to batch multiple administrative security modifications and commit the changes to the CMS.

b. Iterate through each report in the folder and call the `checkRight` method.

The `checkRight` method returns an integer indicating whether the principal user has the indicated right granted on the report (2=granted, 3=denied).

```
String resultString = "";
for (Object object : reports)
{
  IInfoObject report = (IInfoObject)object;
  resultString += "<b>Report: </b>" + report.getTitle() + " -
<i>User has right granted? </i>" + secRightsAdmin.checkRight(rightToCheck,
user.getID(), report.getID()) + "<br>";
}
```

Example

This example iterates through all available Crystal reports in a folder, and checks whether a particular user has the right to refresh data (`CeReportRightID.REFRESH_ON_DEMAND`) granted on each report. The results are formatted in an HTML table and returned as a String object.

```
String checkRights(IEnterpriseSession enterpriseSession, IUser user, IFolder
folder) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String resultString = "";

  ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
  IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS
Where SI_PARENT_FOLDER=" + folder.getID() + "And SI_KIND='CrystalReport' And
SI_INSTANCE=0");
  RightDescriptor rightToCheck = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false,
CeSecurityOptions.RightScope.CURRENT_OBJECT, "CrystalEnterprise.Report");
  ISecCacheControllerAdmin secCacheControllerAdmin =
secInfoMgr.getSecCacheAdmin();
  secCacheControllerAdmin.batch();
  try
  {
    for (Object object : reports)
    {
      IInfoObject report = (IInfoObject)object;
      secCacheControllerAdmin.cacheRight(rightToCheck, user.getID(),
report.getID());
    }
    secCacheControllerAdmin.commit();
  }
  catch (Throwable e)
  {
    secCacheControllerAdmin.rollback();
  }
  ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
```

```
  resultString += "<b>Right: <i>CeReportRightID.REFRESH_ON_DEMAND (granted=2,
denied=3)</i></b><br>";
  for (Object object : reports)
  {
    IInfoObject report = (IInfoObject)object;
    resultString += "<b>Report: </b>" + report.getTitle() + " - <i>User has
right granted? </i>" + secRightsAdmin.checkRight(rightToCheck, user.getID(),
report.getID()) + "<br>";
  }
  return resultString;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.RightDescriptor`
- `com.crystaldecisions.sdk.occa.security.CeSecurityOptions`
- `com.crystaldecisions.sdk.occa.security.ISecCacheControllerAdmin`
- `com.crystaldecisions.sdk.occa.security.ISecRightsAdmin`
- `com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`
- `com.crystaldecisions.sdk.plugin.desktop.report.CeReportRightID`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
- `java.io.IOException`

Related Information

Object security overview [page 113]
To check rights on an object [page 123]
To set an explicit right on multiple objects [page 136]

5.3.2.5.3 To set an explicit right on multiple objects

While access levels (roles) are used to apply common sets of rights on objects to many users and groups, individual rights can also be explicitly assigned on objects to users and groups.

In this example, you grant a principal user an explicit right to refresh the data (`CeReportRightID.REFRESH_ON_DEMAND`) on all Crystal reports in a folder. This example assumes that the session has administrative rights on the reports and uses the `ISecurityInfoMgr` interface to batch a series of operations to set rights, and commit them to the Central Management Server (CMS).

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Call the `getSecurityInfoMgr` method to retrieve the `ISecurityInfoMgr` object for the session.

```
ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
```

3. Retrieve all reports in the folder from the repository.

```
IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_PARENT_FOLDER=" + folder.getID() "
    + "And SI_KIND='CrystalReport' And SI_INSTANCE=0");
```

4. Construct a new `RightDescriptor` object representing the right to refresh a report.

   The `CeReportRightID` interface contains constant field values that specify Crystal Reports security rights. The `CeSecurityID.Right` interface defines general object rights. Rights that are specific to other object types are specified by interfaces such as `CeWebIRightID`, `CeFullClientRightID`, and `CeUniverseRightID`.

```
RightDescriptor refreshReportRight = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false);
```

> **Note**
>
> Rights created using this constructor add two rights to an object, one with a `CeSecurityOptions.RightScope.CURRENT_OBJECT` scope and one with a `CeSecurityOptions.RightScope.DESCENDANTS` scope. Use this constructor only to set rights. To retrieve rights information, use the `RightDescriptor` constructor that also requires scope and applicable object type parameters.

5. Batch a series of requests to set rights, and commit them to the CMS.
   a. Call the `getRightsAdmin` method of the `ISecurityInfoMgr` interface to retrieve an `ISecRightsAdmin` object.

```
ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
```

   The `ISecRightsAdmin` interface in this example is used to batch multiple administrative security modifications and commit the changes to the CMS. This interface can also be used to retrieve loaded results from the SDK cache.

   b. Call the `batch` method to start the batching process.

```
secRightsAdmin.batch();
```

   c. Create a `try` block to rollback all batch modifications in the event of an error.

```
try {
    ...
}
catch (Throwable e) {
    secRightsAdmin.rollback();
}
```

   d. Iterate through each report in the folder and call the `setRight` method.

The fourth parameter is a Boolean value that indicates whether to grant or deny the principal the right described by the `RightDescriptor` object.

```
for (Object object : reports)
{
    IInfoObject report = (IInfoObject)object;
    secRightsAdmin.setRight(refreshReportRight, user.getID(),
report.getID(), true);
}
```

e. Call the `commit` method to submit the batch of rights changes.

```
secRightsAdmin.commit();
```

Example

This example iterates through all available Crystal reports in a folder and grants a particular user an explicit right to refresh data (`CeReportRightID.REFRESH_ON_DEMAND`) on each report.

```
void setExplicitRight(IEnterpriseSession enterpriseSession, IUser user, IFolder
folder) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  ISecurityInfoMgr secInfoMgr = enterpriseSession.getSecurityInfoMgr();
  IInfoObjects reports = infostore.query("Select SI_ID From CI_INFOOBJECTS "
    + "Where SI_PARENT_FOLDER=" + folder.getID()
    + "And SI_KIND='CrystalReport' And SI_INSTANCE=0");

  RightDescriptor refreshReportRight = new
RightDescriptor(CeReportRightID.REFRESH_ON_DEMAND, IReport.KIND, false);
  ISecRightsAdmin secRightsAdmin = secInfoMgr.getRightsAdmin();
  secRightsAdmin.batch();
  try
  {
          for (Object object : reports)
    {
      IInfoObject report = (IInfoObject)object;
      secRightsAdmin.setRight(refreshReportRight, user.getID(), report.getID(),
true);
    }
    secRightsAdmin.commit();
  }
  catch (Throwable e)
  {
    secRightsAdmin.rollback();
  }
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.RightDescriptor`

- `com.crystaldecisions.sdk.occa.security.ISecRightsAdmin`
- `com.crystaldecisions.sdk.occa.security.ISecurityInfoMgr`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`
- `com.crystaldecisions.sdk.plugin.desktop.report.CeReportRightID`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
- `java.io.IOException`

Related Information

Object security overview [page 113]
To set an explicit right on an object [page 126]
To check rights on multiple objects [page 133]

5.3.2.6 To create a custom access level

Access levels (roles) are used to apply common sets of rights on objects to many users and groups. The BI platform is installed with several predefined access levels. However, you can also create and customize your own access levels; this approach can greatly reduce administration and maintenance costs associated with security. You can programmatically construct new custom access levels with the SDK.

In this example, you create a custom access level that grants the right to delete any object from the repository.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Call the `newInfoObjectCollection` method of the `IInfoStore` interface to create an `IInfoObjects` collection.

   ```
   IInfoObjects newCustomRoles = infostore.newInfoObjectCollection();
   ```

3. Call the `add` method of the collection to create an `ICustomRole` object.

   Use the constant `ICustomRole.KIND`, and cast the result as an `ICustomRole` object.

   ```
   ICustomRole newCustomRole = (ICustomRole)
   newCustomRoles.add(ICustomRole.KIND);
   ```

4. Set various properties of the new access level, including its name and description.

   ```
   newCustomRole.setTitle("Delete Objects");
   newCustomRole.setDescription("This access level grants members to delete
   objects from the repository.");
   ```

5. Create a new `RightDescriptor` object that represents the right to delete objects from the repository.

   When creating a custom access level, use the `RightDescriptor` constructor that specifies the scope of the right with a `CeSecurityOptions.RightScope` object.

The `CeSecurityID.Right` interface defines general object rights. Rights that are specific to other object types are specified by interfaces such as `CeReportRightID`, `CeWebIRightID`, `CeFullClientRightID`, and `CeUniverseRightID`.

```
RightDescriptor deleteObjectsRight = new
RightDescriptor(CeSecurityID.Right.DELETE, "", false,
CeSecurityOptions.RightScope.CURRENT_OBJECT, CeSecurityOptions.ANY_OBJTYPE);
```

6. Add the newly created right to the collection of explicit rights for the access level.

> **Note**
>
> The Boolean argument `true` in the `addRoleRight` method call grants the right for the access level. You do not need to use the `IRoleRight.setGranted` method in this case.

```
IRoleRights roleRights = newCustomRole.getRoleRights();
IRoleRight roleRight = roleRights.addRoleRight(deleteObjectsRight, true);
```

7. Commit the new custom access level back to the Central Management Server (CMS) to save the changes.

```
infostore.commit(newCustomRoles);
```

The new custom access level is created. To assign this access level to an existing principal on an object such as a report, add it to the list of existing access levels for the principal:

```
IExplicitRoles customRoles = principal.getRoles();
IExplicitRole customRole = customRoles.add(newCustomRole.getID());
```

This code snippet assumes that you have an `IExplicitPrincipal` instance called `principal`. After adding the access level, the report or object that you modified security settings for must be committed back to the CMS using the `IInfoStore.commit` method.

Example

This example creates a custom access level that grants the right to delete any object from the repository.

```
void createCustomAccessLevel(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects newCustomRoles = infostore.newInfoObjectCollection();
  ICustomRole newCustomRole = (ICustomRole) newCustomRoles.add(ICustomRole.KIND);

  newCustomRole.setTitle("Delete Objects");
  newCustomRole.setDescription("This access level grants members to delete
objects from the repository.");

  RightDescriptor deleteObjectsRight = new
RightDescriptor(CeSecurityID.Right.DELETE, "", false,
CeSecurityOptions.RightScope.CURRENT_OBJECT, CeSecurityOptions.ANY_OBJTYPE);
  IRoleRights roleRights = newCustomRole.getRoleRights();
  IRoleRight roleRight = roleRights.addRoleRight(deleteObjectsRight, true);
  infostore.commit(newCustomRoles);
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.customrole.ICustomRole`
- `com.businessobjects.sdk.plugin.desktop.customrole.IRoleRight`
- `com.businessobjects.sdk.plugin.desktop.customrole.IRoleRights`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CeSecurityID`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.RightDescriptor`
- `com.crystaldecisions.sdk.occa.security.CeSecurityOptions`

Related Information

Object security overview [page 113]
To set an explicit right on an object [page 126]
To set an explicit right on multiple objects [page 136]

5.4 Scheduling

Scheduling is a process which allows you to select an `InfoObject` stored within the Central Management Server (CMS) and run it automatically at specified times. Reports and documents are the most commonly scheduled type of `InfoObject`, but you can also schedule other `InfoObjects` such as programs and object packages.

You schedule `InfoObjects` in various ways. You can specify custom values, file formats, destinations, and parameter values. You can also schedule reports using calendars.

A variety of options are available around scheduling:

- The `InfoObject` may be scheduled to run immediately after the schedule has been set, or to run at some point in the future.
- The `InfoObject` may be scheduled to run once, or to recur on a regular basis.
- The `InfoObject` may be scheduled to export as a Crystal report file, an excel file, a word file, a PDF file, or a number of other export formats.
- The `InfoObject` may be scheduled to be stored internally on the file repository server. Or, it may be scheduled to a number of output destinations.

To manage specific scheduling information you want to access the `SchedulingInfo` property that is associated with each instance of an `IInfoObject` object. Once the scheduling information has been determined and committed to the CMS repository, you can then choose to schedule the objects using the `IInfoStore.schedule` method. This method then creates runnable instances of the objects in the collection. A runnable instance is the child of the parent `InfoObject` that has been scheduled. For more information on scheduling see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide.*

Classes used when scheduling

- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
  Contains the scheduling properties of an object, including recurrence and destination options.
- `com.crystaldecisions.sdk.occa.infostore.CeScheduleType`
  Specifies the frequency that the object will be run.
- `com.crystaldecisions.sdk.plugin.desktop.calendar.ICalendar`
  Provides methods and properties that allow you to access a collection of business calendars and information related to their run days.

5.4.1 Scheduling workflow

The basic workflow for scheduling a report using the SDK is as follows:

To schedule a report

1. Retrieve a report object through the `InfoStore` object query.
2. Retrieve the `SchedulingInfo` object through the `InfoObject.SchedulingInfo` property.
3. Set the scheduling options for the report.
4. Set the output format of the report using the `ReportFormatOptions.format` property.
5. Set the destination and destination options for the report.
6. Schedule the report.

> **Note**
> The destination options set in the report will override the destination options set in the Job Server.

5.4.2 Output destinations

When you schedule an object, the instance of that object is sent to a particular destination. Supported destinations include the following:

Destination	Description
FTP	The instance can be copied to a server using FTP.
SMTP	The instance can be emailed to a person.
Unmanaged disk	The instance can be copied to a local or network file system.
Inbox (Managed)	The instance can be scheduled to the Enterprise inboxes of users.
Printer	The instance can be printed to a local or network printer.

> **Note**
>
> - Inbox is an example of a managed destination. A managed destination is a location that resides within the BI platform.
> - Printer is a Crystal report-specific destination that prints from the RPT format. All other destinations can apply to any scheduled object and supported format types. Printer options for a Crystal report are set directly through the `com.crystaldecisions.sdk.plugin.desktop.report.IReport` interface.

Destinations are configured in two ways:

- Using the default destination settings of the appropriate job server.
  The default destinations of a job server can be set through the `IServerDestination` interface.
- Using the destination plugin to set a destination on the object at schedule-time.
  The destinations for a scheduled object are set through the appropriate subinterfaces of the `IDestinationPlugin` interface:
  - `com.crystaldecisions.sdk.plugin.destination.ftp.IFTP`
  - `com.crystaldecisions.sdk.plugin.destination.smtp.ISMTP`
  - `com.crystaldecisions.sdk.plugin.destination.managed.IManaged`
  - `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanaged`

> **Note**
>
> You must enable the destination on the appropriate job server before scheduling a job to that destination. For more information, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

Objects that can be scheduled or sent to a destination:

Object Type	Schedule	Send
Crystal Report	Yes	Yes
Object Package	Yes	Yes
Program	Yes	Yes
Excel File	No	Yes
Word File	No	Yes
PDF File	No	Yes
Text File	No	Yes
RTF File	No	Yes
Powerpoint File	No	Yes

Object Type	Schedule	Send
Hyperlink	No	Yes
Web Intelligence document	Yes	Yes
Publication	Yes	Yes

What are the available destinations based on object type?

Object Type	Unmanaged Disk	FTP	SMTP (File) | (Link)	Inbox (File) (Link)	Favorites Folder (File) (Link)
Crystal Report	Yes	Yes	Yes | Yes	Yes | Yes	Yes | Yes
Object Package	No	No	No | No	Yes | Yes	Yes | Yes
Program	Yes	Yes	Yes | Yes	Yes | Yes	Yes | Yes
Excel File	Yes	Yes	Yes | Yes	Yes | Yes	Yes | Yes
Word File	Yes	Yes	Yes | Yes	Yes | Yes	Yes | Yes
PDF File	Yes	Yes	Yes | Yes	Yes | Yes	Yes | Yes
Text File	Yes	Yes	Yes | Yes	Yes | Yes	Yes | Yes
RTF File	Yes	Yes	Yes | Yes	Yes | Yes	Yes | Yes
Powerpoint File	Yes	Yes	Yes | Yes	Yes | Yes	Yes | Yes
Hyperlink	No	No	No | Yes	Yes | Yes	Yes | Yes
Web Intelligence document	No	No	No | Yes	Yes | Yes	Yes | Yes

5.4.2.1 Setting destination options

You can set options for each destination through the corresponding options interface:

- `com.crystaldecisions.sdk.plugin.destination.ftp.IFTPOptions`
- `com.crystaldecisions.sdk.plugin.destination.smtp.SMTPOptions`
- `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanagedOptions`
- `com.crystaldecisions.sdk.plugin.destination.managed.IManagedOptions`

Options are applied globally using the default destination for a job server, or applied individually using the destination plugins.

Global job server destination options

Global destination options are the default destination options set on a job server. The default destination options of the job server are used when you do not set the destination options through the destination plugin. There are two main cases to consider:

- Objects scheduled using the `IInfoStore.schedule` method.
  The default destination options of the job server that processes the scheduled object is used. For example, the Crystal Reports Job Server.
- Objects sent using the `IInfoStore.sendTo` method.
  The default destination options of the Destination Job Server are used. The Destination Job Server is unlike other job servers in that it does not handle scheduling or processing.

You can programmatically modify the default destination options on a job server using the `IServerDestination.getConfigGlobalOptions` method.

The following example enables and sets the default destination of all Crystal Reports Job Servers to the `C:\` folder on the local file system. The scheduling information for the object is then set to `CrystalEnterprise.DiskUnmanaged` so that this job server destination is used.

```
void setJobServerDefaultDestination(IEnterpriseSession enterpriseSession,
IInfoObject report) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects servers = infostore.query( "Select SI_ID, SI_HOSTED_SERVICES From
CI_SYSTEMOBJECTS "
    + "Where SI_KIND='Server' And SI_SERVER_KIND='jobserver' "
    + "And SI_NAME LIKE '%CrystalReportsJobServer%'");

  for (int i = 0; i < servers.size(); i++)
  {
    IServer crJobServer = (IServer)servers.get(i);
    IServerDestinations serverDestinations = crJobServer.getServerDestinations();
    IServerDestination diskUnmanagedDestination =
serverDestinations.getServerDestination("CrystalEnterprise.DiskUnmanaged");

    diskUnmanagedDestination.setRequestedStatus(true);

    IDiskUnmanagedOptions diskUnmanagedOptions =
(IDiskUnmanagedOptions)diskUnmanagedDestination.getConfigGlobalOptions();
    List outputFilePaths = diskUnmanagedOptions.getDestinationFiles();
    outputFilePaths.clear();
    outputFilePaths.add("C:\\");
  }
  infostore.commit(servers);

  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  scheduleInfo.getDestinations().add("CrystalEnterprise.DiskUnmanaged");
}
```

Destination plugin options

Destination plugins provide a way to set destination options that override the default options on a job server. Query for the appropriate plugin from the Central Management Server (CMS), and use the `IDestinationPlugin.getScheduleOptions` method to access and modify the options for the appropriate destination type.

After modifying the destination options for the plugin, you can then apply the destination plugin to an object using the `IDestination.setFromPlugin` method, and then schedule or send the object.

The following example queries for the unmanaged disk destination plugin, and sets its delivery options to the `C:\` folder on the local file system. The destination plugin is then applied to the scheduling information for the object.

```
void setDestinationToUnmanagedDisk(IEnterpriseSession enterpriseSession,
IInfoObject report) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects diskUnmanagedInfoObjects = infostore.query("Select
SI_DEST_SCHEDULEOPTIONS, SI_PROGID "
    + "From CI_SYSTEMOBJECTS Where SI_NAME = 'CrystalEnterprise.DiskUnmanaged'");
  IInfoObject infoObject = (IInfoObject)diskUnmanagedInfoObjects.get(0);
  IDestinationPlugin destinationPlugin = (IDestinationPlugin)infoObject;
  IDiskUnmanaged diskUnmanaged = (IDiskUnmanaged)destinationPlugin;
  IDiskUnmanagedOptions diskUnmanagedOptions =
(IDiskUnmanagedOptions)diskUnmanaged.getScheduleOptions();
  List outputFilePaths = diskUnmanagedOptions.getDestinationFiles();
  outputFilePaths.clear();
  outputFilePaths.add("C:\\");

  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  IDestination destination =
scheduleInfo.getDestinations().add("CrystalEnterprise.DiskUnmanaged");
  destination.setFromPlugin(destinationPlugin);
}
```

> **Note**
>
> You must enable the destination on the appropriate job server before scheduling a job to that destination. For more information, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IDestination`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationPlugin`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServerDestination`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServerDestinations`

- `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanaged`
- `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanagedOptions`
- `java.io.IOException`
- `java.util.List`

5.4.2.2 To set the destination of a scheduled object

You can set the destination for a scheduled object using the destination plugin (`IDestinationPlugin` and its subinterfaces) retrieved by querying the Central Management Server (CMS). Setting the destination in this way overrides the default settings on the job server. These examples assume that you have retrieved the `ISchedulingInfo` object using the `getSchedulingInfo` method.

Example: Set destination to FTP

This example specifies that the scheduled object be delivered as a file named `scheduledReport.pdf` to the `ftpdir` folder on an FTP server named `servername.domain` that is listening at port 21.

```
void setDestinationToFTP(IEnterpriseSession enterpriseSession, IInfoObject
report) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();

  IInfoObjects ftpInfoObjects = infostore.query("Select SI_DEST_SCHEDULEOPTIONS,
SI_PROGID From CI_SYSTEMOBJECTS Where SI_NAME = 'CrystalEnterprise.Ftp'");
  IInfoObject ftpInfoObject = (IInfoObject)ftpInfoObjects.get(0);

  IDestinationPlugin destinationPlugin = (IDestinationPlugin)ftpInfoObject;
  IFTP ftp = (IFTP)destinationPlugin;

  IFTPOptions ftpOptions = (IFTPOptions)ftp.getScheduleOptions();
  ftpOptions.setServerName("servername.domain");
  ftpOptions.setPort(Integer.parseInt("21"));
  ftpOptions.setUserName("username");
  ftpOptions.setPassword("password");
  List files = ftpOptions.getDestinationFiles();
  files.add("ftpdir\\scheduledReport.pdf");
  IDestination destination =
scheduleInfo.getDestinations().add("CrystalEnterprise.Ftp");
  destination.setFromPlugin(destinationPlugin);
}
```

> **Note**
>
> For more information about options that can be set when scheduling to an FTP server, see the `com.crystaldecisions.sdk.plugin.destination.ftp.IFTPOptions` interface in the *SAP BusinessObjects Business Intelligence Platform Java API Reference*.

Example: Set destination to SMTP

This example specifies that the scheduled object be emailed to two addresses, relayed through an SMTP server named `servername.domain` that is listening at port 25.

```
void setDestinationToSMTP(IEnterpriseSession enterpriseSession, IInfoObject
report) throws SDKException, IOException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();

  IInfoObjects smtpInfoObjects = infostore.query("Select
SI_DEST_SCHEDULEOPTIONS, SI_PROGID From CI_SYSTEMOBJECTS Where SI_NAME =
'CrystalEnterprise.Smtp'");
  IInfoObject infoObject = (IInfoObject)smtpInfoObjects.get(0);

  IDestinationPlugin destinationPlugin = (IDestinationPlugin)infoObject;
  ISMTP smtp = (ISMTP)destinationPlugin;

  ISMTPOptions smtpOptions = (ISMTPOptions)smtp.getScheduleOptions();
  smtpOptions.setDomainName("domain");
  smtpOptions.setServerName("servername");
  smtpOptions.setPort(Integer.parseInt("25"));
  smtpOptions.setSMTPAuthenticationType(ISMTPOptions.CeSMTPAuthentication.LOGIN);
  smtpOptions.setSMTPUserName("username");
  smtpOptions.setSMTPPassword("password");
  smtpOptions.setSubject("Email subject");
  smtpOptions.setMessage("This is the email message.");

  List toAddresses = smtpOptions.getToAddresses();
  toAddresses.add("user1@emaildomain.com");
  toAddresses.add("user2@emaildomain.com");*/

  IDestination destination =
scheduleInfo.getDestinations().add("CrystalEnterprise.Smtp");
  destination.setFromPlugin(destinationPlugin);
}
```

> **Note**
>
> For more information about options that can be set when scheduling to an email destination, see the `com.crystaldecisions.sdk.plugin.destination.smtp.ISMTPOptions` interface in the *SAP BusinessObjects Business Intelligence Platform Java API Reference*.

Example: Set destination to unmanaged disk

This example specifies that the scheduled object be delivered to the `C:\` folder on the local file system.

```
void setDestinationToUnmanagedDisk(IEnterpriseSession enterpriseSession,
IInfoObject report) throws SDKException
{
  IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects diskUnmanagedInfoObjects = infostore.query("Select
SI_DEST_SCHEDULEOPTIONS, SI_PROGID From CI_SYSTEMOBJECTS Where SI_NAME =
'CrystalEnterprise.DiskUnmanaged'");
  IInfoObject infoObject = (IInfoObject)diskUnmanagedInfoObjects.get(0);
```

```
  IDestinationPlugin destinationPlugin = (IDestinationPlugin)infoObject;
  IDiskUnmanaged diskUnmanaged = (IDiskUnmanaged)destinationPlugin;

  IDiskUnmanagedOptions diskUnmanagedOptions =
(IDiskUnmanagedOptions)diskUnmanaged.getScheduleOptions();
  List outputFilePaths = diskUnmanagedOptions.getDestinationFiles();
  outputFilePaths.clear();
  outputFilePaths.add("C:\\");

  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  IDestination destination =
scheduleInfo.getDestinations().add("CrystalEnterprise.DiskUnmanaged");
  destination.setFromPlugin(destinationPlugin);
}
```

> **Note**
>
> For more information about options that can be set when scheduling to a file system location, see the `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanagedOptions` interface in the *SAP BusinessObjects Business Intelligence Platform Java API Reference*.

Example: Set destination to inbox

This example specifies that the scheduled object be copied to a user's inbox.

```
void setDestinationToInbox(IEnterpriseSession enterpriseSession, IUser user,
IInfoObject report) throws SDKException, IOException
{
   IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");
   ISchedulingInfo scheduleInfo = report.getSchedulingInfo();

   IInfoObjects managedInfoObjects = infostore.query("Select
SI_DEST_SCHEDULEOPTIONS, SI_PROGID From CI_SYSTEMOBJECTS Where SI_NAME =
'CrystalEnterprise.Managed'");
   IInfoObject infoObject = (IInfoObject)managedInfoObjects.get(0);

   IDestinationPlugin destinationPlugin = (IDestinationPlugin)infoObject;
   IManaged managed = (IManaged)destinationPlugin;

   IManagedOptions managedOptions =
(IManagedOptions)managed.getScheduleOptions();

managedOptions.setDestinationOption(IManagedOptions.CeDestinationOption.ceInbox);
   managedOptions.setSendOption(IManagedOptions.CeManagedSendOption.ceCopy);
   managedOptions.setIncludeInstance(true);
   managedOptions.getDestinations().add(user.getID());

   IDestination destination =
scheduleInfo.getDestinations().add("CrystalEnterprise.Managed");
   destination.setFromPlugin(destinationPlugin);
}
```

> **Note**
>
> For more information about options that can be set when scheduling to an inbox, see the `com.crystaldecisions.sdk.plugin.destination.managed.IManagedOptions` interface in the *SAP BusinessObjects Business Intelligence Platform Java API Reference*.

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IDestination`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationPlugin`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
- `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanaged`
- `com.crystaldecisions.sdk.plugin.destination.diskunmanaged.IDiskUnmanagedOptions`
- `com.crystaldecisions.sdk.plugin.destination.ftp.IFTP`
- `com.crystaldecisions.sdk.plugin.destination.ftp.IFTPOptions`
- `com.crystaldecisions.sdk.plugin.destination.managed.IManaged`
- `com.crystaldecisions.sdk.plugin.destination.managed.IManagedOptions`
- `com.crystaldecisions.sdk.plugin.destination.smtp.ISMTP`
- `com.crystaldecisions.sdk.plugin.destination.smtp.ISMTPOptions`
- `java.io.IOException`
- `java.util.List`

5.4.3 To schedule a report

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Query for and retrieve the specified report object.

```
IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE=0");
IReport report = (IReport) infoObjects.get(0);
```

3. Call the `getSchedulingInfo` method to retrieve an instance of the `ISchedulingInfo` object.

```
ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
```

4. Set the scheduling options.

   The report will be set to run immediately and only once.

```
scheduleInfo.setType(CeScheduleType.ONCE);
scheduleInfo.setRightNow(true);
```

5. Obtain the report instance's format by calling the `getReportFormatOptions` method.

```
IReportFormatOptions reportFormatOptions = report.getReportFormatOptions();
```

6. Set the format type calling the `setFormat` method. The format type can be one of the values from the `CeReportFormat` class.

   > **Note**
   >
   > You may also set the properties of your chosen output format.

   ```
   reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPO
   RT);
   ```

7. To print a copy of the Crystal report to the printer, modify the printer options for the report.

   Only Crystal reports can be scheduled to print to the printer. Replace `"\\\\servername\ \printername"` with your printer name.

   > **Note**
   >
   > To access the printer, the Server Intelligence Agent must be running under a user account that has sufficient network privileges to access the printer. For more information about managing and configuring servers, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

   ```
   IReportPrinterOptions printerOptions = report.getReportPrinterOptions();
   printerOptions.setCopies(1);
   printerOptions.setEnabled(true);
   printerOptions.setFromPage(1);
   printerOptions.setToPage(1);
   printerOptions.setPrinterName("\\\\servername\\printername");
   ```

8. Schedule the report using the `IInfoStore` object.

   ```
   IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
   objectsToSchedule.add(report);
   infostore.schedule(objectsToSchedule);
   ```

Example

This example retrieves a Crystal report, schedules the report to its default destination, and prints the first page of the report to the printer.

```
void scheduleReport(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE=0");
  IReport report = (IReport) infoObjects.get(0);
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  scheduleInfo.setType(CeScheduleType.ONCE);
  scheduleInfo.setRightNow(true);

  IReportFormatOptions reportFormatOptions = report.getReportFormatOptions();

reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPORT)
;
  IReportPrinterOptions printerOptions = report.getReportPrinterOptions();
  printerOptions.setCopies(1);
  printerOptions.setEnabled(true);
```

```
  printerOptions.setFromPage(1);
  printerOptions.setToPage(1);
  printerOptions.setPrinterName("\\\\servername\\printername");

  IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
  objectsToSchedule.add(report);
  infostore.schedule(objectsToSchedule);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CeScheduleType`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions`
- `com.crystaldecisions.sdk.plugin.desktop.common.IReportPrinterOptions`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`

Related Information

To set the destination of a scheduled object [page 147]

5.4.4 To schedule a report with custom values

When you schedule a report it is possible to add custom values to the instances that are created from it. For example, an administrator could embed a user's email address or user name in the instances to keep track of which instances were created for a particular user. You can do this by using the `InfoObject`. This example will use `InfoObject` to store an email address. Later on, it will query for all instances on the CMS that contain that particular email address.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Query for and retrieve the specified report object.

   ```
   IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
   From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
   Report' and SI_INSTANCE = 0");
   IInfoObject report = (IInfoObject) infoObjects.get(0);
   ```

3. Use the `InfoObject.SchedulingInfo` property to retrieve the `SchedulingInfo` object.

   ```
   ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
   ```

4. Set the scheduling options.

   The report will be set to run immediately and only once.

```
scheduleInfo.setType(CeScheduleType.ONCE);
scheduleInfo.setRightNow(true);
```

5. Obtain the report instance's format using the `ReportFormatOptions` property.

```
IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();
```

6. Set the format type using the `Format` property. The format type can be one of the values from `CeReportFormat`.

   > **Note**
   >
   > You may also set the properties of your chosen output format.

```
reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPO
RT);
```

7. Use the InfoObject's property bag to add a custom value to the report.

   > **Note**
   >
   > To add a custom value, you must also give it a key. The key allows you to use the InfoStore Query method to retrieve the custom value. To retrieve reports containing the custom value use the key, in the Where clause of your query. If the property does not exist in the property bag, no results will be returned.

   The key in this example, which is a string with a value "Email", will be used later to retrieve the value.

```
report.properties().setProperty("Email", email);
```

8. Schedule the report using the `InfoStore` object.

```
IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
objectsToSchedule.add(report);
infostore.schedule(objectsToSchedule);
```

Example

This example retrieves a Crystal Report, adds an email address to the report, and schedules the report to its default destination.

```
void scheduleWithCustomValues(IEnterpriseSession enterpriseSession, String
email) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE = 0");
  IInfoObject report = (IInfoObject) infoObjects.get(0);
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  scheduleInfo.setType(CeScheduleType.ONCE);
```

```
  scheduleInfo.setRightNow(true);

  IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();

reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPORT)
;
  report.properties().setProperty("Email", email);

  IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
  objectsToSchedule.add(report);
  infostore.schedule(objectsToSchedule);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CeScheduleType`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`

Related Information

To set the destination of a scheduled object [page 147]

5.4.5 To schedule a report containing specific parameter values

This example shows you how to schedule a report with specific parameter values. You do not have to commit the modified parameters to the CMS in order to schedule the report with these parameters. When the report is scheduled, it uses parameters in the current `ReportInterface` and not those parameters stored on the CMS.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Query for and retrieve the specified report object.

   ```
   IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
   From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
   Report' and SI_INSTANCE = 0");
   IInfoObject report = (IInfoObject) infoObjects.get(0);
   ```

3. Use the `InfoObject.SchedulingInfo` property to retrieve an instance of the `SchedulingInfo` object.

   ```
   ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
   ```

4. Set the scheduling options.

   The report will be set to run immediately and only once.

```
scheduleInfo.setType(CeScheduleType.ONCE);
scheduleInfo.setRightNow(true);
```

5. Obtain the report instance's format using the `ReportFormatOptions` property.

```
IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();
```

6. Set the format type using the `Format` property. The format type can be one of the values from `CeReportFormat`.

   > **Note**
   >
   > You may also set the properties of your chosen output format.

```
reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPO
RT);
```

7. Retrieve and manipulate report parameters.

   The following code iterates through each parameter in the report and modifies discrete value String parameters.

```
for (int i = 0; i < ((IReport)report).getReportParameters().size(); i++)
{
  IReportParameter reportParameter = (IReportParameter)
((IReport)report).getReportParameters().get(i);
  if (reportParameter.isDiscreteValueSupported()
&& reportParameter.getValueType() ==
IReportParameter.ReportVariableValueType.STRING)
  {
    IReportParameterValues parameterValue =
reportParameter.getCurrentValues();
    IParameterFieldDiscreteValue discreteValue =
(IParameterFieldDiscreteValue)parameterValue.getValues(1).getValue(0);
    discreteValue.setValue("A String");
  }
}
```

   > **Note**
   >
   > This sample specifies `end_date` as the parameter field for the `getParameterName` method. The new parameter value `end_date` is then added by the `addSingleValue` method. If a parameter value exists, use the `get` method to retrieve existing parameter values rather than using the `addSingleValue` method.

8. Schedule the report using the `InfoStore` object.

```
IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
objectsToSchedule.add(report);
infostore.schedule(objectsToSchedule);
```

Example

This example retrieves a Crystal Report, modifies the report parameters, and schedules the report to its default destination.

```
void scheduleWithParameters(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE = 0");
  IInfoObject report = (IInfoObject) infoObjects.get(0);
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  scheduleInfo.setType(CeScheduleType.ONCE);
  scheduleInfo.setRightNow(true);

  IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();

reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPORT)
;

  for (int i = 0; i < ((IReport)report).getReportParameters().size(); i++)
  {
    IReportParameter reportParameter = (IReportParameter)
((IReport)report).getReportParameters().get(i);
    if (reportParameter.isDiscreteValueSupported()
&& reportParameter.getValueType() ==
IReportParameter.ReportVariableValueType.STRING)
    {
      IReportParameterValues parameterValue = reportParameter.getCurrentValues();
      IParameterFieldDiscreteValue discreteValue =
(IParameterFieldDiscreteValue)parameterValue.getValues(1).getValue(0);
      discreteValue.setValue("A String");
    }
  }

  IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
  objectsToSchedule.add(report);
  infostore.schedule(objectsToSchedule);
}
```

> **Note**
>
> This sample uses the `getReportParameters` method which has dependencies on `rascore.jar` and `serialization.jar` files. Therefore, you must import both files to your project for this sample to run.

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CeScheduleType`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.occa.report.data.IParameterFieldDiscreteValue`

- `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions`
- `com.crystaldecisions.sdk.plugin.desktop.common.IReportParameter`
- `com.crystaldecisions.sdk.plugin.desktop.common.IReportParameterValues`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`

Related Information

To set the destination of a scheduled object [page 147]

5.4.6 To schedule a report using calendars

This tutorial shows you how to perform operations with calendars. A business calendar offers you a way to generate a custom schedule that incorporates specific run days that are based on certain business requirements. For example, a user may create a business calendar that sets every week day (Monday through Friday) to be a run day.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve an instance of the `PluginManager` object and create a new `Calendar` object.

```
IPluginMgr pluginMgr = infostore.getPluginMgr();
IPluginInfo calendarPlugin =
pluginMgr.getPluginInfo("CrystalEnterprise.Calendar");
IInfoObjects newInfoObjects = infostore.newInfoObjectCollection();
newInfoObjects.add (calendarPlugin);
IInfoObject iObject = (IInfoObject)newInfoObjects.get(0);
```

3. Set the calendar title and description.

```
iObject.setTitle("My Calendar");
iObject.setDescription("A description of my calendar");
```

4. Add dates and times to the calendar.

   This code sets the calendar to run every Monday from January 1, 2010, to December 31, 2010.

```
int startDay = 01;
int startMonth = 0;
int startYear = 2010;
int endDay = 31;
int endMonth = 11;
int endYear = 2010;

int dayOfWeek = java.util.Calendar.MONDAY;
int weekOfMonth = IBusinessCalendarDay.ALL;
ICalendar calendar = (ICalendar)iObject;
IBusinessCalendarDays days = calendar.getDays();
int calendarID = days.getNextGroupID();
days.add(startDay, startMonth, startYear, endDay, endMonth, endYear,
dayOfWeek, weekOfMonth, calendarID);
```

5. Commit the `infoObject` to the CMS.

```
infostore.commit(newInfoObjects);
```

6. Query for and retrieve the specified report object.

```
InfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS "
  + "Where SI_KIND='CrystalReport' and SI_NAME='"
  + reportName + "' and SI_INSTANCE = 0");  IInfoObject report =
(IInfoObject) infoObjects.get(0);
IInfoObject report = (IInfoObject) infoObjects.get(0);
```

7. Use the `InfoObject.SchedulingInfo` property to retrieve the `SchedulingInfo` object.

```
ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
```

8. Set the scheduling options to use the calendar.

```
scheduleInfo.setType(CeScheduleType.CALENDAR);
scheduleInfo.setRightNow(false);
scheduleInfo.setCalendarTemplate(calendarID);
```

9. Obtain the report instance's format using the `ReportFormatOptions` property.

```
IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();
```

10. Set the format type using the `Format` property. The format type can be one of the values from `CeReportFormat`.

> **Note**
>
> You may also set the properties of your chosen output format.

```
reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPO
RT);
```

11. Schedule the report using the `InfoStore` object.

```
IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
objectsToSchedule.add(report);
infostore.schedule(objectsToSchedule);
```

Example

This example creates a new calendar, retrieves a Crystal report, and then schedules the report using calendar dates. The calendar schedules the report to run every Monday, from January 1, 2010, to December 31, 2010.

```
void scheduleWithCalendar(IEnterpriseSession enterpriseSession , String
reportName) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IPluginMgr pluginMgr = infostore.getPluginMgr();
  IPluginInfo calendarPlugin =
pluginMgr.getPluginInfo("CrystalEnterprise.Calendar");
```

```
  IInfoObjects newInfoObjects = infostore.newInfoObjectCollection();
  newInfoObjects.add (calendarPlugin);

  IInfoObject iObject = (IInfoObject)newInfoObjects.get(0);
  iObject.setTitle("My Calendar");
  iObject.setDescription("A description of my calendar");
  int startDay = 01;
  int startMonth = 0;
  int startYear = 2010;
  int endDay = 31;
  int endMonth = 11;
  int endYear = 2010;

  int dayOfWeek = java.util.Calendar.MONDAY;
  int weekOfMonth = IBusinessCalendarDay.ALL;

  ICalendar calendar = (ICalendar)iObject;
  IBusinessCalendarDays days = calendar.getDays();
  int calendarID = days.getNextGroupID();
  days.add(startDay, startMonth, startYear, endDay, endMonth, endYear,
dayOfWeek, weekOfMonth, calendarID);

  infostore.commit(newInfoObjects);
  IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS "
          + "Where SI_KIND='CrystalReport' and SI_NAME='"
          + reportName + "' and SI_INSTANCE = 0");
  IInfoObject report = (IInfoObject) infoObjects.get(0);
  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();

  scheduleInfo.setType(CeScheduleType.CALENDAR);
  scheduleInfo.setRightNow(false);
  scheduleInfo.setCalendarTemplate(calendarID);

  IReportFormatOptions reportFormatOptions =
((IReport)report).getReportFormatOptions();

reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPORT)
;

  IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
  objectsToSchedule.add(report);
  infostore.schedule(objectsToSchedule);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CeScheduleType`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.occa.pluginmgr.IPluginInfo`
- `com.crystaldecisions.sdk.occa.pluginmgr.IPluginMgr`
- `com.crystaldecisions.sdk.plugin.desktop.calendar.IBusinessCalendarDay`
- `com.crystaldecisions.sdk.plugin.desktop.calendar.IBusinessCalendarDays`
- `com.crystaldecisions.sdk.plugin.desktop.calendar.ICalendar`

- `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`
- `java.util.Calendar`

5.4.7 To schedule a report using events

You can schedule a report to be processed when an event is triggered. For example, you can schedule a report to run after another report has successfully completed processing, or when a custom event is manually triggered. In order for the report to be processed, all of the scheduling conditions must be met. If the event has been triggered but the other scheduling conditions have not been met, then the report will not be processed.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Query for and retrieve a report to be scheduled.

```
IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE=0");
IReport report = (IReport) infoObjects.get(0);
```

3. Query for and retrieve the event that should occur before the report is processed.

   This example queries for an event called `My Event`.

```
IInfoObjects events = infostore.query("Select SI_ID from CI_SYSTEMOBJECTS
where SI_NAME = 'My Event'");
IEvent event = (IEvent)events.get(0);
```

4. Call the `getSchedulingInfo` method of the `IReport` object to retrieve an instance of the `ISchedulingInfo` object.

```
ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
```

5. Set the scheduling options for the report.

   This example sets the report to run once, and immediately.

```
scheduleInfo.setType(CeScheduleType.ONCE);
scheduleInfo.setRightNow(true);
```

6. Set the output format for the scheduled report.

   This example sets the output format to be a Crystal report.

```
IReportFormatOptions reportFormatOptions = report.getReportFormatOptions();
reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPO
RT);
```

7. Retrieve the list of the dependent events that must be triggered before the report is processed.

```
IEvents scheduleEvents = scheduleInfo.getDependencies();
```

8. Add the event that must occur before the report is processed to the list of dependent events.

```
int eventID = event.getID();
scheduleEvents.add(eventID);
```

9. Schedule the report.

```
IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
objectsToSchedule.add(report);
infostore.schedule(objectsToSchedule);
```

Example

This example schedules a report to run once and immediately when the event `My Event` is triggered.

```
void scheduleUsingEvent(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_PROCESSINFO.SI_PROMPTS
From CI_INFOOBJECTS Where SI_KIND='CrystalReport' and SI_NAME='World Sales
Report' and SI_INSTANCE=0");
  IReport report = (IReport) infoObjects.get(0);

  IInfoObjects events = infostore.query("Select SI_ID from CI_SYSTEMOBJECTS
where SI_NAME = 'My Event'");
  IEvent event = (IEvent)events.get(0);

  ISchedulingInfo scheduleInfo = report.getSchedulingInfo();
  scheduleInfo.setType(CeScheduleType.ONCE);
  scheduleInfo.setRightNow(true);

  IReportFormatOptions reportFormatOptions = report.getReportFormatOptions();

reportFormatOptions.setFormat(IReportFormatOptions.CeReportFormat.CRYSTAL_REPORT)
;
  IEvents scheduleEvents = scheduleInfo.getDependencies();
  int eventID = event.getID();
  scheduleEvents.add(eventID);
  IInfoObjects objectsToSchedule = infostore.newInfoObjectCollection();
  objectsToSchedule.add(report);
  infostore.schedule(objectsToSchedule);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CeScheduleType`
- `com.crystaldecisions.sdk.occa.infostore.IEvents`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.occa.pluginmgr.IPluginInfo`
- `com.crystaldecisions.sdk.occa.pluginmgr.IPluginMgr`
- `com.crystaldecisions.sdk.plugin.desktop.event.IEvent`
- `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`

- `java.util.Calendar`

5.5 Events and alerting

Events capture actions that occur in the SAP BusinessObjects Business Intelligence platform. Examples of events include when a job finishes processing, a system file is created, a document alert is triggered, or a performance threshold is crossed. Events are defined by objects in the Central Management Server (CMS) that implement the `com.crystaldecisions.sdk.plugin.desktop.event.IEvent` interface.

You can use events with job scheduling to run jobs when an event is triggered. You can also have a user notified when an event is triggered by making the event alertable and subscribing the user to the event. For example, you can create an event that is triggered when report processing fails. When the event is triggered, you can have it send alert notifications to subscribed users and start another job that runs an error report.

You can use this SDK to create schedule, file, and custom events. You can also enable alerting on events, subscribe users to alerting-enabled events, and view alert notifications.

You can create several types of events with this SDK:

- Schedule events
  Schedule events are triggered when a scheduled job finishes processing. You set the event to be triggered when a job finishes successfully, when a job finishes but fails, or when it finishes either way. Schedule events are represented by the `com.crystaldecisions.sdk.plugin.desktop.event.IScheduleEvent` interface.
- File events
  File events are triggered when a file is created. For example, you could define a file event to be triggered when a log file appears on the system. File events are represented by the `com.crystaldecisions.sdk.plugin.desktop.event.IFileEvent` interface.
- Custom events
  Custom events are triggered manually, either through the UI or by using the SDK. Custom events are represented by the `com.crystaldecisions.sdk.plugin.desktop.event.IUserEvent` interface.

Related Information

To schedule a report using events [page 160]

5.5.1 To create a schedule event

You can create schedule events that are triggered when a scheduled job finishes processing. You can set the event to be triggered when the job finishes successfully, when a job finishes but fails, or when it finishes either way.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. Create a new `IInfoObject` collection, and add a new `IScheduleEvent` object to it.

```
IInfoObjects infoObjects = infostore.newInfoObjectCollection();
IScheduleEvent event =
(IScheduleEvent)infoObjects.add(IScheduleEvent.SPECIFIC_KIND);
```

3. Set the event title and description.

```
event.setTitle("Schedule Event");
event.setDescription("Schedule event description");
```

4. To enable alerting for the event, call the `setAlertEnabled` method.

```
event.setAlertEnabled(true);
```

5. Set the scheduling outcome that triggers the event.

   Use `IScheduleEvent.SUCCESS` to trigger the event when a job completes successfully, use `IScheduleEvent.FAILURE` to trigger the event when the job fails, and use `IScheduleEvent.BOTH` to trigger the event when the job finishes, whether or not it is successful.

```
event.setDependencyType(IScheduleEvent.SUCCESS);
```

6. Commit the new event collection back to the Central Management Server (CMS) to save the changes.

```
infostore.commit(infoObjects);
```

Example

This example creates a schedule event that occurs when the job completes successfully:

```
void createScheduleEvent(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.newInfoObjectCollection();
  IScheduleEvent event =
(IScheduleEvent)infoObjects.add(IScheduleEvent.SPECIFIC_KIND);
  event.setTitle("Schedule Event");
  event.setDescription("Schedule event description");
  event.setAlertEnabled(true);
  event.setDependencyType(IScheduleEvent.SUCCESS);
  infostore.commit(infoObjects);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.plugin.desktop.event.IScheduleEvent`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

5.5.2 To create a file event

You can create file events that are triggered when a file is created. For example, you can use a file event to schedule a report to run when a log file from an external process appears on the file system.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. Create a new `IInfoObject` collection, and add a new `IFileEvent` object to it.

```
IInfoObjects infoObjects = infostore.newInfoObjectCollection();
IFileEvent event = (IFileEvent)infoObjects.add(IFileEvent.SPECIFIC_KIND);
```

3. Set the event title and description.

```
event.setTitle("File Event");
event.setDescription("File event description.");
```

4. Set the path to the file to watch.

   This example sets the file path to `c:\log.txt`. The event is triggered when this file appears on the file system.

```
event.setFileName("C:\\log.txt");
```

5. Set the name of the event server that monitors the event.

   Replace `MyNodeName.EventServer` with the name of the event server that monitors the event.

```
event.setServerFriendlyName("MyNodeName.EventServer");
```

6. To enable alerting for the event, call the `setAlertEnabled` method.

```
event.setAlertEnabled(true);
```

7. To add an alert message, add a string to the collection returned by the `getDefaultAlertMessages` method.

```
event.getDefaultAlertMessages().add("The event was triggered.");
```

8. Commit the new event collection back to the Central Management Server (CMS) to save the changes.

```
infostore.commit(infoObjects);
```

Example

This example creates a file event that is triggered when the file `c:\log.txt` appears on the file system:

```
void createFileEvent(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.newInfoObjectCollection();
  IFileEvent event = (IFileEvent)infoObjects.add(IFileEvent.SPECIFIC_KIND);
  event.setTitle("File Event");
  event.setDescription("File event description.");
  event.setFileName("C:\\log.txt");
```

```
  event.setServerFriendlyName("MyNodeName.EventServer");
  event.setAlertEnabled(true);
  event.getDefaultAlertMessages().add("The event was triggered.");
  infostore.commit(infoObjects);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.plugin.desktop.event.IFileEvent`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

5.5.3 To create a custom event

You can create custom events and manually trigger them as needed. For example, you can configure a scheduling job to wait for a custom event to be manually triggered. By triggering an alerting-enabled custom event, you can send alert notifications to event subscribers.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
   ```

2. Create a new `IInfoObject` collection, and add a new `IUserEvent` object to it.

   ```
   InfoObjects infoObjects = infostore.newInfoObjectCollection();
   IUserEvent event = (IUserEvent)infoObjects.add(IUserEvent.SPECIFIC_KIND);
   ```

3. Set the event title and description.

   ```
   event.setTitle("My Event");
   event.setDescription("Event description.");
   ```

4. To enable alerting for the event, call the `setAlertEnabled` method.

   ```
   event.setAlertEnabled(true);
   ```

5. To add an alert message, add a string to the collection returned by the `getDefaultAlertMessages` method.

   ```
   event.getDefaultAlertMessages().add("The event was triggered.");
   ```

6. Commit the new event collection back to the Central Management Server (CMS) to save the changes.

   ```
   infostore.commit(infoObjects);
   ```

Example

This example creates a custom event named `My Event` that can be manually triggered:

```
void createCustomEvent(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.newInfoObjectCollection();
  IUserEvent event = (IUserEvent)infoObjects.add(IUserEvent.SPECIFIC_KIND);
  event.setTitle("My Event");
  event.setDescription("Event description.");
  event.setAlertEnabled(true);
  event.getDefaultAlertMessages().add("The event was triggered.");
  infostore.commit(infoObjects);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.plugin.desktop.event.IUserEvent`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

5.5.4 To enable alerting for an event

This example shows how to enable alerting for an event. When an alerting-enabled event is triggered, alert notifications are sent to subscribers of the event.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
   ```

2. Query for and retrieve an event from the Central Management Server (CMS).

   This example retrieves an event named `My Event`.

   ```
   IInfoObjects events = infostore.query("Select SI_ID from CI_SYSTEMOBJECTS
   where SI_KIND = 'Event' and SI_NAME = 'My Event'");
   IEvent event = (IEvent)events.get(0);
   ```

3. Enable alerting on the event by calling the `setAlertEnabled` method.

   ```
   event.setAlertEnabled(true);
   ```

4. Commit the modified event collection back to the CMS to save the changes.

   ```
   infostore.commit(events);
   ```

Example

This example shows how to enable alerting on an event and sets the alert message:

```
void enableAlerting(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects events = infostore.query("Select SI_ID from CI_SYSTEMOBJECTS
where SI_KIND = 'Event' and SI_NAME = 'My Event'");
  IEvent event = (IEvent)events.get(0);
  event.setAlertEnabled(true);
  infostore.commit(events);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.plugin.desktop.event.IEvent`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

5.5.5 To subscribe to an event

An event must be made alertable before a user or group can be subscribed to it.

You can subscribe a user or group to an event. When an event is triggered, alert notifications are sent to subscribers of the event.

> **Note**
>
> If you subscribe a group to an event, you can unsubscribe individual members of the group by adding them to the list returned by the `getExcludedSubscribers` method of the `IEventBase` class.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
   ```

2. Query for and retrieve an event from the InfoStore.

   This example retrieves an event named `My Event`.

   ```
   IInfoObjects events = infostore.query("Select SI_ID, SI_EVENT_SUBSCRIBERS
   from CI_SYSTEMOBJECTS where SI_KIND = 'Event' and SI_NAME = 'My Event'");
   IEvent event = (IEvent)events.get(0);
   ```

3. Query for and retrieve an `IUser` object from the InfoStore.

   This example queries for the `Administrator` user.

   ```
   IInfoObjects subscribers = infostore.query("Select SI_ID from
   CI_SYSTEMOBJECTS where SI_NAME = 'Administrator' and SI_KIND = 'User'");
   IUser subscriber = (IUser)subscribers.get(0);
   ```

4. Retrieve the user ID from the subscriber object.

```
int subscriberID = subscriber.getID();
```

5. Add the user ID to the list of alert subscribers for this event.

```
IAlertSubscriptions subscriptions = event.getAlertSubscriptions();
IAlertSubscription subscription = subscriptions.add(subscriberID);
```

6. Set the alert notification delivery options.

   This example sets the delivery method to Inbox.

```
Set<String> deliveryMethods = subscription.getDeliveryMethods();
deliveryMethods.add(IAlertSubscription.CeAlertDeliveryMethod.INBOX);
```

7. Commit the modified event collection back to the Central Management Server (CMS) to save the changes.

```
infostore.commit(events);
```

Example

This example shows how to subscribe a user to an alerting-enabled event named `My Event`:

```
void subscribeAlert(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects events = infostore.query("Select SI_ID, SI_EVENT_SUBSCRIBERS from
CI_SYSTEMOBJECTS where SI_KIND = 'Event' and SI_NAME = 'My Event'");
  IEvent event = (IEvent)events.get(0);

  IInfoObjects subscribers = infostore.query("Select SI_ID from CI_SYSTEMOBJECTS
where SI_NAME = 'Administrator' and SI_KIND = 'User'");
  IUser subscriber = (IUser)subscribers.get(0);
  int subscriberID = subscriber.getID();

  IAlertSubscriptions subscriptions = event.getAlertSubscriptions();
  IAlertSubscription subscription = subscriptions.add(subscriberID);
  Set<String> deliveryMethods = subscription.getDeliveryMethods();
  deliveryMethods.add(IAlertSubscription.CeAlertDeliveryMethod.INBOX);

  infostore.commit(events);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.plugin.desktop.common.IAlertSubscription`
- `com.crystaldecisions.sdk.plugin.desktop.common.IAlertSubscriptions`
- `com.crystaldecisions.sdk.plugin.desktop.event.IEvent`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

- `java.util.Set`

5.5.6 To trigger a custom event

Before you can trigger a custom event, it must already have been committed to the Central Management Server (CMS). You cannot create a custom event and trigger it in the same commit action.

You can programmatically trigger a custom event to schedule objects that are waiting for the event, or to send alert notifications to users and groups subscribed to the event.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. Query for and retrieve a user event from the InfoStore.

```
IInfoObjects events = infostore.query("Select SI_ID,
SI_DEFAULT_ALERT_MESSAGES * from CI_SYSTEMOBJECTS where SI_SPECIFIC_KIND =
'UserEvent' and SI_NAME = 'My Event'");
IUserEvent event = (IUserEvent) events.get(0);
```

3. Trigger the event.

```
event.trigger();
```

4. Commit the modified event collection back to the CMS to save the changes.

```
infostore.commit(events);
```

Example

This example shows how to trigger a custom event named `My Event`:

```
void triggerUserEvent(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects events = infostore.query("Select SI_ID, SI_DEFAULT_ALERT_MESSAGES
from CI_SYSTEMOBJECTS where SI_SPECIFIC_KIND = 'UserEvent' and SI_NAME = 'My
Event'");
  IUserEvent event = (IUserEvent) events.get(0);
  event.trigger();
  infostore.commit(events);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.event.IUserEvent`

5.5.7 To view alert notifications

When an alerting-enabled event is triggered, alert notifications are sent to the users and groups subscribed to the event. Alert notifications contain the alert message, the ID of the event that triggered the alert, and a list of users and groups subscribed to the event. This example shows how to get the alert notifications that have been sent to a user.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. Query for and retrieve a user, and include the `SI_RECEIVED_ALERTNOTIFICATIONS` property in your query.

   This example queries for the Administrator user.

```
IInfoObjects users = infostore.query( "Select SI_ID,
SI_RECEIVED_ALERTNOTIFICATIONS from CI_SYSTEMOBJECTS where SI_KIND='user' and
SI_NAME='Administrator'");
IUser user = (IUser)users.get(0);
```

3. Get the received alert notifications for this user.

```
IReceivedAlertNotifications receivedAlertNotifications =
user.getReceivedAlertNotifications();
IReceivedAlertNotification notification =
receivedAlertNotifications.iterator().next();
```

4. Get the ID of an alert notification.

```
int alertID = notification.getID();
```

5. Query for and retrieve an alert notification by using the alert notification's ID to find the alert notification, and include the `SI_ALERT_MESSAGES` property in your query.

```
IInfoObjects alertNotifications = infostore.query("Select SI_ID,
SI_ALERT_MESSAGES from CI_SYSTEMOBJECTS where SI_KIND = 'AlertNotification'
AND SI_ID = " + alertID);
IAlertNotification alertNotification = (IAlertNotification)
alertNotifications.iterator().next();
```

6. Get the alert messages by using the `getMessages` method of the `IAlertNotification` class.

```
List<String>alertMessages = alertNotification.getMessages();
String alertMessage = (String) alertMessages.get(0);
...
```

Example

This example shows how to view alert notifications that have been sent to a user:

```
void getAlertNotifications(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

```
  IInfoObjects users = infostore.query( "Select SI_ID,
SI_RECEIVED_ALERTNOTIFICATIONS from CI_SYSTEMOBJECTS where SI_KIND='user' and
SI_NAME='Administrator'");
  IUser user = (IUser)users.get(0);
  IReceivedAlertNotifications receivedAlertNotifications =
user.getReceivedAlertNotifications();
  IReceivedAlertNotification notification =
receivedAlertNotifications.iterator().next();
  int alertID = notification.getID();
  IInfoObjects alertNotifications = infostore.query("Select SI_ID,
SI_ALERT_MESSAGES from CI_SYSTEMOBJECTS where SI_KIND = 'AlertNotification' AND
SI_ID = " + alertID);
  IAlertNotification alertNotification = (IAlertNotification)
alertNotifications.iterator().next();
  List<String>alertMessages = alertNotification.getMessages();
  String alertMessage = (String) alertMessages.get(0);
    ...
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.alertnotification.IAlertNotification`
- `com.businessobjects.sdk.plugin.desktop.common.IRecievedAlertNotification`
- `com.businessobjects.sdk.plugin.desktop.common.IRecievedAlertNotifications`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`
- `java.util.List`

5.6 Publications

A publication is a set of objects that can be scheduled and distributed to multiple recipients on a regular basis. Recipients can be either BI platform users or external users, who can receive the publication by e-mail. The set of objects includes one or more report documents (Crystal Reports or Web Intelligence documents) and may also include static objects, such as Excel spreadsheets and images. The contents of the report documents in the publication can be personalized, so that recipients only see information that is relevant to them.

A publication contains a list of recipients (users and user groups) who will receive the publication when it is scheduled. E-mail recipients can also be retrieved from a dynamic data source, such as a Crystal Reports or Desktop Intelligence document.

A publication can also contain list of profiles and profile targets, which allows the creator of the publication to personalize the content of the reports in the publication based on the profile values assigned to each recipient.

Users can create and manage publications through either the Central Management Console (CMC) or BI launch pad. It is also possible to create and manage publications programmatically. In this section, you will learn how to use the SDK to create and manage publications.

> **Note**
>
> For more information on the publishing feature, see the *SAP BusinessObjects Business Intelligence Platform User's Guide*.

Classes used for managing publications

- `com.businessobjects.sdk.plugin.desktop.publication`
  Contains the `IPublication` interface and several supporting classes.
- `com.crystaldecisions.sdk.occa.infostore`
  Contains the classes for configuring destinations.
- `com.crystaldecisions.sdk.plugin.desktop.common`
  Contains the classes for configuring output formats and dynamic recipients.

5.6.1 Publishing workflow

The basic workflow for publishing documents using the SDK is as follows:

1. Create the publication in the repository.
2. Add report documents to the publication. (You can include one or more Crystal reports or Web Intelligence documents, but all documents must be of the same kind.)
3. Add recipients to the publication. You can subscribe BI platform users and user groups to receive the publication. You can also subscribe dynamic recipients by specifying a data source, such as a Crystal Reports or Desktop Intelligence document, that provides the name, e-mail address, and (optionally) profile values for each dynamic recipient.
4. Specify the profile values that will be used to personalize the report for each BI platform user.
5. Configure the destination(s) to which the publication will be delivered and, for each destination, specify the output format of each publication document.
6. Add static documents to the publication.
7. Schedule the publication.

The topics in this section explain in detail how to perform each of these steps.

5.6.2 To create a publication

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Execute a query on the CMS repository to retrieve the folder object where you want to create the publication.

   ```
   String folderQuery = "SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_KIND= '"
   ```

```
  + IFolder.FOLDER_KIND + "' AND SI_NAME = 'Feature Samples'";
IInfoObjects folders = infostore.query(folderQuery);
IInfoObject folder = (IInfoObject) folders.get(0);
```

3. Create a publication info object and set the parent ID to the ID of the folder object.

```
IInfoObjects newInfoObjects = infostore.newInfoObjectCollection();
IPublication publication = (IPublication)
newInfoObjects.add(IPublication.KIND);
publication.setTitle("Test Publication");
publication.setDescription("This is a sample publication");
publication.setParentID(folder.getID());
```

4. Save the publication to the CMS repository.

```
publication.save();
```

Example

The following code creates a publication in the `Feature Samples` folder of the CMS.

```
public void createPublication(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");

  String folderQuery = "SELECT SI_ID FROM CI_INFOOBJECTS WHERE SI_KIND= '"
     + IFolder.FOLDER_KIND + "' AND SI_NAME = 'Feature Samples'";
  IInfoObjects folders = infostore.query(folderQuery);
  IInfoObject folder = (IInfoObject) folders.get(0);

  IInfoObjects newInfoObjects = infostore.newInfoObjectCollection();
  IPublication publication = (IPublication)
newInfoObjects.add(IPublication.KIND);

  publication.setTitle("Test Publication");
  publication.setDescription("This is a sample publication");
  publication.setParentID(folder.getID());
  publication.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`

Related Information

To add a report document to a publication [page 175]
To add a static document to a publication [page 176]
To add a user to a publication [page 179]
To add a group to a publication [page 181]
To personalize a document for dynamic recipients [page 188]
To configure a destination and output format [page 193]

5.6.3 Adding documents to a publication

5.6.3.1 Report documents and static documents

A publication can contain two types of documents:

- Report (or dynamic) documents.
- Static documents.

The report document is the main component of a publication. A publication must contain at least one report document.

Publications can support Crystal reports and Web Intelligence documents. The kind of the report object, which is specified in the `SI_KIND` property of the report object, must be one of the following values:

- `CrystalReport`
- `WebI`
- `FullClient`
- `FullClientAddin`
- `FullClientTemplate`

> **Note**
>
> If there are multiple report documents in a publication, they must all be the same kind.

A static document is any kind of non-report document. This can include Excel spreadsheets, images, PDFs, and other kinds of supporting documents. A publication can contain any number of static documents.

Related Information

To add a report document to a publication [page 175]
To add a static document to a publication [page 176]

5.6.3.2 To add a report document to a publication

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Execute a query on the CMS repository to retrieve the publication object, and cast the object to `com.businessobjects.sdk.desktop.plugin.publication.IPublication`. The query must include the `SI_PUBLICATION_DOCUMENTS` property.

```
String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
  + "WHERE SI_KIND = '" + IPublication.KIND + "' AND SI_NAME =
'MyPublication' AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. Execute a query to retrieve the report object that you want to add to the publication. The query must include the `SI_PROCESSINFO` property.

   Publications can support Crystal reports and Web Intelligence documents. The kind of the report object must be one of the following values:

   - `CrystalReport`
   - `WebI`
   - `FullClient`
   - `FullClientAddin`
   - `FullClientTemplate`

```
String documentQuery = "SELECT SI_ID, SI_PROCESSINFO FROM CI_INFOOBJECTS "
  + "WHERE SI_NAME='World Sales Report' AND SI_INSTANCE = 0";
IInfoObjects documents = infostore.query(documentQuery);
IInfoObject document = (IInfoObject) documents.get(0);
```

4. Call the `getDocuments` method of the publication object and add the report object ID to the collection.

```
Collection publicationDocuments = publication.getDocuments();
Integer documentID = new Integer(document.getID());
publicationDocuments.add(documentID);
```

5. Use the `setDocumentProcessingInfo` method of the publication object to copy the document processing info from the report object to the publication.

   The first parameter to `setDocumentProcessingInfo` is the ID of the report document. The second parameter is the document kind. The third parameter is the property bag containing the processing info.

```
publication.setDocumentProcessingInfo(documentID, document.getKind(),
document.getProcessingInfo().properties());
```

6. Save the publication to the CMS repository.

```
publication.save();
```

Example

```
public void addReportToPublication(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
    + "WHERE SI_KIND = '" + IPublication.KIND + "' AND SI_NAME = 'MyPublication'
AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  String documentQuery = "SELECT SI_ID, SI_PROCESSINFO FROM CI_INFOOBJECTS "
    + "WHERE SI_NAME='World Sales Report' AND SI_INSTANCE = 0";
  IInfoObjects documents = infostore.query(documentQuery);
  IInfoObject document = (IInfoObject) documents.get(0);

  Collection publicationDocuments = publication.getDocuments();
  Integer documentID = new Integer(document.getID());
  publicationDocuments.add(documentID);

  publication.setDocumentProcessingInfo(documentID, document.getKind(),
document.getProcessingInfo().properties());

  publication.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `java.util.Collection`

Related Information

Report documents and static documents [page 174]
To add a static document to a publication [page 176]

5.6.3.3 To add a static document to a publication

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Execute a query on the CMS repository to retrieve the publication object. The query must include the `SI_PUBLICATION_DOCUMENTS` and `SI_SCHEDULEINFO` properties.

```
String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS,
SI_SCHEDULEINFO "
  + "FROM CI_INFOOBJECTS WHERE SI_KIND = '" + IPublication.KIND
  + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. Execute a query to retrieve the static document object that you want to add to the publication.

```
String staticDocumentQuery = "Select SI_ID FROM CI_INFOOBJECTS WHERE
SI_NAME='a_jpg_image'";
IInfoObjects staticDocuments = infostore.query(staticDocumentQuery);
IInfoObject staticDocument = (IInfoObject) staticDocuments.get(0);
```

4. Call the `getDocuments` method of the publication object and add the ID of the static document to the collection.

```
Collection publicationDocuments = publication.getDocuments();
Integer staticDocumentID = new Integer(staticDocument.getID());
publicationDocuments.add(staticDocumentID);
```

5. Retrieve the destinations from the scheduling info of the publication. At least one destination must already be configured for the publication.

```
ISchedulingInfo schedulingInfo = publication.getSchedulingInfo();
IDestinations destinations = schedulingInfo.getDestinations();
```

6. Add the static document ID to the static documents collection of the appropriate destination(s).

```
Iterator destinationsIter = destinations.iterator();
while(destinationsIter.hasNext())
{
  IDestination destination = (IDestination) destinationsIter.next();
  IDestinationStaticDocuments destinationDocs =
destination.getDestinationStaticDocuments();
  IDestinationStaticDocument newStaticDoc = destinationDocs.add();
  newStaticDoc.setStaticDocumentID(staticDocumentID);
}
```

7. Save the publication.

```
publication.save();
```

Example

```
public void addStaticDocumentToPublication(IEnterpriseSession enterpriseSession)
throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS,
SI_SCHEDULEINFO "
    + "FROM CI_INFOOBJECTS WHERE SI_KIND = '" + IPublication.KIND
    + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);
```

```
  String staticDocumentQuery = "Select SI_ID FROM CI_INFOOBJECTS WHERE
SI_NAME='a_jpg_image'";
  IInfoObjects staticDocuments = infostore.query(staticDocumentQuery);
  IInfoObject staticDocument = (IInfoObject) staticDocuments.get(0);

  Collection publicationDocuments = publication.getDocuments();
  Integer staticDocumentID = new Integer(staticDocument.getID());
  publicationDocuments.add(staticDocumentID);

  ISchedulingInfo schedulingInfo = publication.getSchedulingInfo();
  IDestinations destinations = schedulingInfo.getDestinations();

  Iterator destinationsIter = destinations.iterator();
  while(destinationsIter.hasNext())
  {
    IDestination destination = (IDestination) destinationsIter.next();
    IDestinationStaticDocuments destinationDocs =
destination.getDestinationStaticDocuments();
    IDestinationStaticDocument newStaticDoc = destinationDocs.add();
    newStaticDoc.setStaticDocumentID(staticDocumentID);
  }
  publication.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IDestination`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationStaticDocument`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationStaticDocuments`
- `com.crystaldecisions.sdk.occa.infostore.IDestinations`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `java.util.Collection`
- `java.util.Iterator`

Related Information

To configure a destination and output format [page 193]
Report documents and static documents [page 174]
To add a report document to a publication [page 175]

5.6.4 Adding recipients to a publication

5.6.4.1 Enterprise recipients and dynamic recipients

Publications can be delivered to two different types of recipients:

- Enterprise recipients.
- Dynamic recipients.

Enterprise recipients are individuals who have access to SAP BusinessObjects Business Intelligence platform and have been granted at least read permission on the publication. These recipients typically receive publications in their personal Inboxes, delivered through the managed destination.

Dynamic recipients are individuals who do not have user accounts in SAP BusinessObjects Business Intelligence platform. These recipients are identified through some external data source, which provides the unique identifier, e-mail address, and profile information (if any) for each recipient. Dynamic recipients typically receive publications by e-mail, delivered through the SMTP destination.

This section explains the various ways to include these different recipients in publications.

Related Information

To add a user to a publication [page 179]
To add a group to a publication [page 181]
To add dynamic recipients to a publication [page 182]

5.6.4.2 To add a user to a publication

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Execute a query on the CMS repository to retrieve the publication object, and cast the object to `com.businessobjects.sdk.desktop.plugin.publication.IPublication`. The query must include the `SI_PRINCIPALS` property.

```
String publicationQuery = "SELECT SI_ID, SI_PRINCIPALS FROM CI_INFOOBJECTS
WHERE SI_KIND = '"
  + IPublication.KIND
  + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. Execute a query to retrieve the user that you want to add to the publication.

```
String userQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
  + IUser.KIND + "' AND SI_NAME = 'Administrator'";
IInfoObjects users = infostore.query(userQuery);
IInfoObject user = (IInfoObject) users.get(0);
```

4. Call the `subscribe` method of the publication object. Set the parameter to the ID of the user object.

> **Note**
>
> In order to subscribe to a publication, the user must have the View Object system right on that publication.

```
Integer userID = new Integer(user.getID());
publication.subscribe(userID);
```

5. Save the publication.

```
publication.save();
```

Example

```
public void addUserToPublication(IEnterpriseSession enterpriseSession, String
username) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String publicationQuery = "SELECT SI_ID, SI_PRINCIPALS FROM CI_INFOOBJECTS
WHERE SI_KIND = '"
    + IPublication.KIND
    + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  String userQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
    + IUser.KIND + "' AND SI_NAME = '" + username + "'";
  IInfoObjects users = infostore.query(userQuery);
  IInfoObject user = (IInfoObject) users.get(0);

  Integer userID = new Integer(user.getID());
  publication.subscribe(userID);

  publication.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.user.IUser`

Related Information

To add a group to a publication [page 181]

5.6.4.3 To add a group to a publication

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Execute a query on the CMS repository to retrieve the publication object, and cast the object to `com.businessobjects.sdk.desktop.plugin.publication.IPublication`. The query must include the `SI_PRINCIPALS` property.

```
String publicationQuery = "SELECT SI_ID, SI_PRINCIPALS FROM CI_INFOOBJECTS
WHERE SI_KIND = '"
  + IPublication.KIND + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. Execute a query to retrieve the user group that you want to add to the publication.

> **Note**
>
> In order to subscribe to a publication, the user must have the View Object system right on that publication.

```
String groupQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
  + IUserGroup.KIND + "' AND SI_NAME = 'Administrators'";
IInfoObjects groups = infostore.query(groupQuery);
IInfoObject group = (IInfoObject) groups.get(0);
```

4. Retrieve the principals collection of the publication object and add the ID of the group to the collection.

```
Integer groupID = new Integer(group.getID());
Collection principals = publication.getPrincipals();
principals.add(groupID);
```

5. Save the publication.

```
publication.save();
```

Example

```
public void addGroupToPublication(IEnterpriseSession enterpriseSession, String
groupName) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  String publicationQuery = "SELECT SI_ID, SI_PRINCIPALS FROM CI_INFOOBJECTS
WHERE SI_KIND = '"
    + IPublication.KIND + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  String groupQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
    + IUserGroup.KIND + "' AND SI_NAME = '" + groupName + "'";
  IInfoObjects groups = infostore.query(groupQuery);
  IInfoObject group = (IInfoObject) groups.get(0);
```

```
  Integer groupID = new Integer(group.getID());
  Collection principals = publication.getPrincipals();
  principals.add(groupID);

  publication.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.usergroup.IUserGroup`
- `java.util.Collection`

Related Information

To add a user to a publication [page 179]

5.6.4.4 To add dynamic recipients to a publication

- You must have a Crystal Reports or Web Intelligence document that contains dynamic recipient data. The document includes:
  - A column containing unique identifiers. These identifiers are used internally by the publishing engine and are not displayed or included in the publication content.
  - A column containing the recipient e-mail addresses. (This column is required if the publication is delivered to the SMTP destination. Otherwise, this column is optional.)
  - A column containing full name of each recipient. (Optional.)
  - Columns containing personalization information. (Optional.)

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Execute a query on the CMS repository to retrieve the publication object, and cast the object to `com.businessobjects.sdk.desktop.plugin.publication.IPublication`. The query must include the `SI_PUBLICATION_EVENT_HANDLERS` property.

```
String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
  + "WHERE SI_KIND = '" + IPublication.KIND + "' "
  + "AND SI_NAME = 'MyPublication' "
  + "AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. Execute a query on the CMS repository to retrieve the document that contains the dynamic recipient information.

```
String providerDocumentName = "HowTo_CreatePublication_DynamicRecipients";
String providerDocumentQuery = "Select SI_ID FROM CI_INFOOBJECTS WHERE
SI_NAME='" + providerDocumentName + "'";
IInfoObjects providerDocuments = infostore.query(providerDocumentQuery);
IInfoObject providerDocument = (IInfoObject) providerDocuments.get(0);
int providerDocumentID = providerDocument.getID();
```

4. Retrieve the collection of publication event handlers associated with the publication, and add a handler to retrieve dynamic recipients.

```
IPublicationEventHandlers handlers =
publication.getPublicationEventHandlers();
IPublicationEventHandler handler =
handlers.addPublicationEventHandler(CePropertyID.SI_ON_READ_RECIPIENTS);
```

Publication event handlers are used to invoke publication extensions (also referred to as plugins), which allow you to customize the publishing process. The `addPublicationEventHandler` method of the `IPublicationEventHandlers` collection takes one parameter that specifies the type of event handler to add. The `CePropertyID.SI_ON_READ_RECIPIENTS` constant specifies that the event handler will be invoked when the publication builds the list of recipients.

5. Configure the publication event handler to use a dynamic recipient data provider, and specify the ID of the document that contains the dynamic recipient information.

```
IPublicationEventHandlerInfo handlerInfo = handler.add();
handlerInfo.setPluginClassName("com.businessobjects.publisher.dynamicrecipient
s.crystalreports.CRDataProvider");
handlerInfo.setDynamicDataDocumentID(providerDocumentID);
```

The `setPluginClassName` method of `IPublicationEventHandlerInfo` sets the fully-qualified class name of the data provider that will be invoked to handle the event. There are two built-in data providers that can be used for reading dynamic recipients:

- The `com.businessobjects.publisher.dynamicrecipients.crystalreports.CRDataProvider` class, which reads dynamic recipient information from a Crystal Reports document.
- The `com.businessobjects.publisher.dynamicrecipients.rebean.REDataProvider` class, which reads dynamic recipient information from a Web Intelligence document.

You can also use a custom dynamic recipient data provider.

6. Retrieve the profile value mappings collection and specify the columns in the dynamic recipient document that contain the name, full name, and e-mail address of each dynamic recipient.

```
IPublicationDynaRecipientProfileValueMappings valueMappings =
publication.getDynamicRecipientsProfileValueMappings();
valueMappings.setName("person.ID");
valueMappings.setFullName("person.full_name");
valueMappings.setEmailAddress("person.email");
```

These settings are used during publishing to provide the values for the `%SI_OWNER%`, `%SI_USERFULLNAME%`, and `%SI_EMAIL_ADDRESS%` placeholders.

> **Note**
>
> The column name you pass to `setName` must specify a column that contains the unique identifiers of the dynamic recipients. These values are used internally and are not included in the output of the publication.
>
> The column name you pass to `setEmailAddress` must specify a column that contains a valid e-maill address for each dynamic recipient.

7. Save the publication.

```
publication.save();
```

Example

The following example reads dynamic recipients from a Crystal Reports document adds them to a publication.

```
public void addDynamicRecipientsToPublication(IEnterpriseSession
enterpriseSession, String publicationName) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
    + "WHERE SI_KIND = '" + IPublication.KIND + "' "
    + "AND SI_NAME = '" + publicationName + "' "
    + "AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  String providerDocumentName = "HowTo_CreatePublication_DynamicRecipients";
  String providerDocumentQuery = "Select SI_ID FROM CI_INFOOBJECTS WHERE
SI_NAME='" + providerDocumentName + "'";
  IInfoObjects providerDocuments = infostore.query(providerDocumentQuery);
  IInfoObject providerDocument = (IInfoObject) providerDocuments.get(0);
  int providerDocumentID = providerDocument.getID();

  IPublicationEventHandlers handlers = publication.getPublicationEventHandlers();
  IPublicationEventHandler handler =
handlers.addPublicationEventHandler(CePropertyID.SI_ON_READ_RECIPIENTS);

  IPublicationEventHandlerInfo handlerInfo = handler.add();

handlerInfo.setPluginClassName("com.businessobjects.publisher.dynamicrecipients.c
rystalreports.CRDataProvider");
  handlerInfo.setDynamicDataDocumentID(providerDocumentID);

  IPublicationDynaRecipientProfileValueMappings valueMappings =
publication.getDynamicRecipientsProfileValueMappings();
  valueMappings.setName("person.ID");
  valueMappings.setFullName("person.full_name");
  valueMappings.setEmailAddress("person.email");
  publication.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`

- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationDynaRecipientProfileValueMappings`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandler`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandlerInfo`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandlers`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CePropertyID`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

Related Information

Dynamic recipient data providers [page 202]

5.6.5 Personalizing publication documents

5.6.5.1 To personalize a document for enterprise recipients

To personalize a document for enterprise recipients, profiles must exist and profile values must be assigned to users.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Execute a query on the CMS repository to retrieve the publication object, and cast the object to `com.businessobjects.sdk.desktop.plugin.publication.IPublication`. The query must include the`SI_PUBLICATION_DOCUMENTS` and `SI_PUBLICATION_DOCUMENTPROFILE_TARGETS` property.

```
String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
  + "WHERE SI_KIND = '" + IPublication.KIND + "' "
  + "AND SI_NAME = 'MyPublication"' "
  + "AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. Call the `getSchedulableDocuments` method of the publication object and get the ID of the report document from the collection.

```
Collection documents = publication.getSchedulableDocuments();
IInfoObject document = (IInfoObject) documents.toArray()[0];
Integer documentID = new Integer(document.getID());
```

4. Execute a query to retrieve the profile that you want to add to the report.

```
String profileQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
  + IProfile.KIND + "' AND SI_NAME = 'country'";
IInfoObjects profiles = infostore.query(profileQuery);
IInfoObject profile = (IInfoObject) profiles.get(0);
```

5. Call the `getPublicationDocumentProfileTargets` method of the publication object.

   This returns an `IPublicationDocumentProfileTargets` collection containing the profile targets that have been associated with the publication. Each profile target contains a collection of variable mappings for a report document in the publication.

```
IPublicationDocumentProfileTargets targets =
publication.getPublicationDocumentProfileTargets();
```

6. Call the `add` method of the profile targets collection. Set the argument to the ID of the report document.

   This creates a new profile target for the specified report document and adds it to the collection.

```
IPublicationDocumentProfileTarget target = targets.add(documentID.intValue());
```

7. Call the `getVariableMappings` method of the target object.

   This returns an `IPublicationDocumentVariableMappings` collection containing the variable mappings that have been associated with the report document.

```
IPublicationDocumentVariableMappings variableMappings =
target.getVariableMappings();
```

8. Call the `add` method of the variable mappings collection.

   This creates a new variable mapping and adds it to the collection.

```
IPublicationDocumentVariableMapping variableMapping = variableMappings.add();
```

9. Call the `setVariableName` method of the variable mapping object. Set the argument to the name of the column in the report document that contains the values to be compared to the profile values.

   > **Note**
   >
   > Crystal Reports column names must be surrounded by curly braces. For example, if the column name is **`Customer.Country`**, then set the variable name to **`{Customer.Country}`**.

```
variableMapping.setVariableName("{Customer.Country}");
```

10. Call the `setProfileID` method of the variable mapping object. Set the argument to the ID of the profile.

```
variableMapping.setProfileID(profile.getID());
```

11. Save the publication.

```
publication.save();
```

Example

The following code adds a profile to a report document in a publication and maps the profile to the **`{Customer.Country}`** column in the report.

```
public void personalizeDocumentForEnterpriseRecipients(IEnterpriseSession
enterpriseSession, String publicationName, String profileName) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
      + "WHERE SI_KIND = '" + IPublication.KIND + "' "
      + "AND SI_NAME = '" + publicationName + "' "
      + "AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  Collection documents = publication.getSchedulableDocuments();
  IInfoObject document = (IInfoObject) documents.toArray()[0];
  Integer documentID = new Integer(document.getID());

  String profileQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '"
    + IProfile.KIND + "' AND SI_NAME = '" + profileName + "'";
  IInfoObjects profiles = infostore.query(profileQuery);
  IInfoObject profile = (IInfoObject) profiles.get(0);

  IPublicationDocumentProfileTargets targets =
publication.getPublicationDocumentProfileTargets();
  IPublicationDocumentProfileTarget target = targets.add(documentID.intValue());

  IPublicationDocumentVariableMappings variableMappings =
target.getVariableMappings();
  IPublicationDocumentVariableMapping variableMapping = variableMappings.add();
  variableMapping.setVariableName("{Customer.Country}");
  variableMapping.setProfileID(profile.getID());

  publication.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.profile.IProfile`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationDocumentProfileTarget`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationDocumentProfileTargets`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.common.IPublicationDocumentVariableMapping`
- `com.crystaldecisions.sdk.plugin.desktop.common.IPublicationDocumentVariableMappings`

- `java.util.Collection`

Related Information

To add dynamic recipients to a publication [page 182]
To personalize a document for dynamic recipients [page 188]

5.6.5.2 To personalize a document for dynamic recipients

To personalize a document for dynamic recipients, you must have a publication that includes dynamic recipients.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Execute a query on the CMS repository to retrieve the publication object, and cast the object to `com.businessobjects.sdk.desktop.plugin.publication.IPublication`. The query must include the `SI_PUBLICATION_DOCUMENTS` and `SI_DYNAMIC_RECIPIENTS_PROFILE_VALUE_MAPPINGS` properties.

```
 String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS,
SI_DYNAMIC_RECIPIENTS_PROFILE_VALUE_MAPPINGS "
  + " FROM CI_INFOOBJECTS WHERE SI_KIND = '" + IPublication.KIND + "' "
  + " AND SI_NAME = '" + publicationName + "' " + "AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. Call the `getSchedulableDocuments` method of the publication object and get the ID of the report document from the collection.

```
Collection reportDocuments = publication.getSchedulableDocuments();
IInfoObject reportDocument = (IInfoObject) reportDocuments.toArray()[0];
Integer reportDocumentID = reportDocument.getID();
```

4. Retrieve the collection of dynamic recipient profile value mappings for the publication, and specify the columns in the dynamic recipient document that contain the name, full name, and e-mail address of each dynamic recipient.

```
IPublicationDynaRecipientProfileValueMappings valueMappings =
publication.getDynamicRecipientsProfileValueMappings();
valueMappings.setName("person.given_name");
valueMappings.setFullName("person.family_name");
valueMappings.setEmailAddress("person.email");
```

5. Retrieve the provider columns collection, and add the column from the dynamic recipients document that contains the profile values to be compared against the profile targets in the report document.

```
IDynamicRecipientProviderColumns providerColumns =
valueMappings.getProviderColumns();
providerColumns.addProviderColumn(1, "person.given_name");
providerColumns.addProviderColumn(2, "person.family_name");
providerColumns.addProviderColumn(3, "person.email");
```

```
providerColumns.addProviderColumn(4, "person.country");
```

The `addProviderColumn` method of the `IDynamicRecipientProviderColumns` collection takes two parameters; the first parameter is a 1-based index used to identify the column within the collection and the second specifies the name of the column within the dynamic recipient provider document.

6. Retrieve the per-document profile mappings.

   This collection is used to store collections of dynamic recipient profile value mappings. Each collection contains the profile value mappings for one schedulable document in the publication.

```
IDynamicRecipientPerDocProfileMappings perDocProfileMappings =
valueMappings.getPerDocProfileValueMappings();
```

7. Add a document profile mapping collection. Set the source document ID of the mapping to the ID of the schedulable document.

```
IDynamicRecipientPerDocProfileMapping mapping = perDocProfileMappings.add();
mapping.setSourceDocumentID(reportDocumentID);
```

8. Retrieve the variable mappings collection associated with the document profile mapping collection, and add a variable mapping to it.

   The `addVariableMapping` method of `IDynamicRecipientVariableMappings` takes three parameters; the first parameter specifies the index of the provider column within the `IDynamicRecipientProviderColumns` collection and the second is the name of the variable within the dynamic recipient provider document. The third parameter must be set to the same value as the first parameter.

```
IDynamicRecipientVariableMappings varMappings =
mapping.getDynamicRecipientVariableMappings();
varMappings.addVariableMapping(4, "{Customer.Country}", 4);
```

> **Note**
>
> Variable names must use the same syntax conventions as the source document. For example, Crystal Reports column names must be surrounded by curly braces.

9. Save the publication.

```
publication.save();
```

Example

```
public void personalizeDocumentForDynamicRecipients(IEnterpriseSession
enterpriseSession, String publicationName) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS,
SI_DYNAMIC_RECIPIENTS_PROFILE_VALUE_MAPPINGS "
    + " FROM CI_INFOOBJECTS WHERE SI_KIND = '" + IPublication.KIND + "' "
    + " AND SI_NAME = '" + publicationName + "' " + "AND SI_INSTANCE = 0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  Collection reportDocuments = publication.getSchedulableDocuments();
```

```
  IInfoObject reportDocument = (IInfoObject) reportDocuments.toArray()[0];
  Integer reportDocumentID = reportDocument.getID();

  IPublicationDynaRecipientProfileValueMappings valueMappings =
publication.getDynamicRecipientsProfileValueMappings();
  valueMappings.setName("person.given_name");
  valueMappings.setFullName("person.family_name");
  valueMappings.setEmailAddress("person.email");

  IDynamicRecipientProviderColumns providerColumns =
valueMappings.getProviderColumns();
  providerColumns.addProviderColumn(1, "person.given_name");
  providerColumns.addProviderColumn(2, "person.family_name");
  providerColumns.addProviderColumn(3, "person.email");
  providerColumns.addProviderColumn(4, "person.country");

  IDynamicRecipientPerDocProfileMappings perDocProfileMappings =
valueMappings.getPerDocProfileValueMappings();

  IDynamicRecipientPerDocProfileMapping mapping = perDocProfileMappings.add();
  mapping.setSourceDocumentID(reportDocumentID);

  IDynamicRecipientVariableMappings varMappings =
mapping.getDynamicRecipientVariableMappings();
  varMappings.addVariableMapping(4, "{Customer.Country}", 4);
  publication.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.publication.IDynamicRecipientPerDocProfileMapping`
- `com.businessobjects.sdk.plugin.desktop.publication.IDynamicRecipientPerDocProfileMappings`
- `com.businessobjects.sdk.plugin.desktop.publication.IDynamicRecipientProviderColumns`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationDynaRecipientProfileValueMappings`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.common.IDynamicRecipientVariableMappings`
- `java.util.Collection`

Related Information

To add dynamic recipients to a publication [page 182]
Dynamic recipient data providers [page 202]

5.6.5.3 Create and manage a profile

This JSP sample shows you how to create and manage a profile.

Example

```
<!-- begin JSP segment -->
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="com.businessobjects.sdk.plugin.desktop.profile.*" %>
<%@ page import="com.businessobjects.sdk.plugin.desktop.common.*" %>
<%@ page import="com.crystaldecisions.sdk.exception.SDKException" %>
<%@ page import="com.crystaldecisions.sdk.occa.infostore.*" %>
<%@ page import="com.crystaldecisions.sdk.framework.*" %>
<%@ page import="java.util.*"%>
<%!
/* Creates a new profile.
 *
 * Parameters:
 *     iStore - the IInfoStore object from the current session.
 *  name - The name of the profile.
 *  description - The description of the profile.
 *
 * Returns:
 *  String - the output String that contains the relevant information.
 */
String addProfile(IInfoStore iStore, String name, String description)
{
    String outString = "";
    try
    {
        /* Create a new profile object. */
        IInfoObjects newcollection = iStore.newInfoObjectCollection();
        IProfile newProfile = (IProfile)newcollection.add("Profile");

        // Set the profile object's InfoObject properties.
        newProfile.setTitle(name);
        newProfile.setDescription(description);

        // Save the new profile to the CMS.
        iStore.commit(newcollection);

        outString = "<B>Profile created.</B><BR>" +
                    "<B>Name:</B> " + name + "<BR>" +
                    "<B>Description:</B> " + description + "<BR><BR>";
    }
    catch(SDKException e)
    {
        throw new Error("An error has occurred: "
                        + e.getMessage());
    }

    return outString;
}
/* Assigns a profile value to a group.
 *
 * Parameters:
 *     iStore - the IInfoStore object from the current session.
 *  profileName - The name of the profile.
 *  groupName - The name of the group.
 *  profileValue - The profile value.
 *
```

```
 * Returns:
 *  String - the output String that contains the relevant information.
 */
String assignProfileValueToGroup(IInfoStore iStore, String profileName, String
groupName, String profileValue)
{
    String outString = "";
    try
    {
        // Retrieve the profile ID from the profile name.
        IInfoObjects profiles = iStore.query("SELECT SI_ID FROM CI_SYSTEMOBJECTS
WHERE SI_KIND = 'Profile' AND SI_NAME = '" + profileName + "'");
        IProfile profile = (IProfile) profiles.get(0);
        int profileID = profile.getID();

        // Retrieve the group as an ISystemPrincipal object.
        IInfoObjects groups = iStore.query("SELECT SI_ID FROM CI_SYSTEMOBJECTS
WHERE SI_KIND = 'UserGroup' AND SI_NAME = '" + groupName + "'");
        ISystemPrincipal groupPrincipal = (ISystemPrincipal) groups.get(0);

        // Create the association between principal and profile.
        IProfileValues profileValues = groupPrincipal.getProfileValues();
        IProfileValue profileValueObject = profileValues.add(profileID);

        // Set the profile value.
        profileValueObject.setFormula(profileValue);

        // Save the changes to the CMS.
        iStore.commit(groups);

        outString = "<B>Profile value assigned to group.</B><BR>" +
                    "<B>Group:</B> " + groupName + "<BR>" +
                    "<B>Value:</B> " + profileValue + "<BR><BR>";
    }
    catch(SDKException e)
    {
        throw new Error("An error has occurred: "
                        + e.getMessage());
    }

    return outString;
}
%>
<!-- end JSP segment -->
<!-- begin HTML segment -->
<html>
<head>
</head>
<body>
<%
/* Retrieve the IInfoStore object from the current session.
 */
IInfoStore iStore = (IInfoStore) session.getAttribute("InfoStore");
/* Enter your profile name here. */
String PROFILE_NAME = "new_profile";
/* Enter your profile description here. */
String PROFILE_DESCRIPTION = "new_description";
/* Enter your group name here. */
String GROUP_NAME = "group_name";
/* Enter your profile value here. */
String PROFILE_VALUE = "=\"Vancouver\"";
out.println(addProfile(iStore, PROFILE_NAME, PROFILE_DESCRIPTION));
out.println(assignProfileValueToGroup(iStore, PROFILE_NAME, GROUP_NAME,
PROFILE_VALUE));
%>
</body>
</html>
<!-- end HTML segment -->
```

5.6.6 Configuring destinations and output formats

5.6.6.1 To configure a destination and output format

To configure destinations and output formats for documents in a publication, you must have a ublication that includes at least one report document.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Execute a query on the CMS repository to retrieve the publication object, and cast the object to `com.businessobjects.sdk.desktop.plugin.publication.IPublication`. The query must include the `SI_SCHEDULEINFO` property.

```
String publicationQuery = "SELECT SI_ID, SI_PROCESSINFO, SI_SCHEDULEINFO, "
  + "SI_PUBLICATION_DOCUMENTS FROM CI_INFOOBJECTS "
  + "WHERE SI_KIND = '" + IPublication.KIND
  + "' AND SI_NAME = 'MyPublication' AND SI_INSTANCE=0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

3. Execute a query to retrieve the destination plugin object.

```
String pluginQuery = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_PARENTID=29 AND
"
  + "SI_NAME = '" + IManaged.PROGID + "'";
IInfoObjects plugins = infostore.query(pluginQuery);
IDestinationPlugin inboxPlugin = (IDestinationPlugin) plugins.get(0);
```

4. Add the destination to the scheduling info of the publication.

```
ISchedulingInfo schedulingInfo = publication.getSchedulingInfo();
IDestinations destinations = schedulingInfo.getDestinations();
IDestination inboxDestination = destinations.add(IManaged.PROGID);
inboxDestination.setFromPlugin(inboxPlugin);
inboxDestination.setDeliverPerUser(true);
```

5. Get the report document from the collection of schedulable documents.

```
Collection documents = publication.getSchedulableDocuments();
IInfoObject document = (IInfoObject) documentIter.next();
Integer documentID = new Integer(document.getID());
```

6. Add the output format information to the destination.

   Supported formats are defined in the `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions.CeReportFormat` interface. Each document kind supports different output formats. For more information, see Supported output formats [page 196].

```
IDestinationFormat formats = inboxDestination.getDestinationFormats();
IDestinationFormat format = formats.add();
format.setSourceDocumentID(documentID);
IFormatInfo formatInfo = format.getFormatInfos().add();
formatInfo.setSourceDocumentKind(document.getKind());
formatInfo.setFormat(document.getKind(),
IReportFormatOptions.CeReportFormat.PDF);
```

7. Copy the document processing info from the report to the publication.

```
IProcessingPublicationInfo processingPubInfo =
        (IProcessingPublicationInfo)
publication.getDocumentProcessingInfoObject(documentID);
IInfoObject reportDoc = (IInfoObject) processingPubInfo;
IFormatInfo formatInfo = processingPubInfo.getFormatInfos().add();
formatInfo.setSourceDocumentKind(document.getKind());
formatInfo.setFormat(document.getKind(),
IReportFormatOptions.CeReportFormat.PDF);
publication.setDocumentProcessingInfo(documentID, document.getKind(),
        reportDoc.getProcessingInfo().properties());
```

   Use the `setDocumentProcessingInfo` method of the publication object to copy the document processing info from the report to the publication. The first parameter to `setDocumentProcessingInfo` is the ID of the report document. The second parameter is the document kind. The third parameter is the property bag containing the processing info.

8. Save the publication.

```
publication.save();
```

Example

The following example configures a publication to deliver each document to the inbox destination and sets the output format to PDF.

```
public void configurePublicationDestination(IEnterpriseSession
enterpriseSession, String publicationName) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String publicationQuery = "SELECT SI_ID, SI_PROCESSINFO, SI_SCHEDULEINFO, "
    + "SI_PUBLICATION_DOCUMENTS FROM CI_INFOOBJECTS "
    + "WHERE SI_KIND = '" + IPublication.KIND
    + "' AND SI_NAME = '" + publicationName + "' AND SI_INSTANCE=0";
  IInfoObjects publications = infostore.query(publicationQuery);
  IPublication publication = (IPublication) publications.get(0);

  String pluginQuery = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_PARENTID=29 AND "
    + "SI_NAME = '" + IManaged.PROGID + "'";
  IInfoObjects plugins = infostore.query(pluginQuery);
  IDestinationPlugin inboxPlugin = (IDestinationPlugin) plugins.get(0);

  ISchedulingInfo schedulingInfo = publication.getSchedulingInfo();
  IDestinations destinations = schedulingInfo.getDestinations();
  IDestination inboxDestination = destinations.add(IManaged.PROGID);
  inboxDestination.setFromPlugin(inboxPlugin);
  inboxDestination.setDeliverPerUser(true);

  Collection documents = publication.getSchedulableDocuments();
  Iterator documentIter = documents.iterator();

  while(documentIter.hasNext())
  {
    IInfoObject document = (IInfoObject) documentIter.next();
    Integer documentID = new Integer(document.getID());

    IDestinationFormat format = inboxDestination.getDestinationFormats().add();
    format.setSourceDocumentID(documentID);
    IFormatInfo formatInfo = format.getFormatInfos().add();
```

```
    formatInfo.setSourceDocumentKind(document.getKind());
    formatInfo.setFormat(document.getKind(),
IReportFormatOptions.CeReportFormat.PDF);

    IProcessingPublicationInfo processingPubInfo =(IProcessingPublicationInfo)
publication.getDocumentProcessingInfoObject(documentID);
    IInfoObject reportDoc = (IInfoObject) processingPubInfo;
    formatInfo = processingPubInfo.getFormatInfos().add();
    formatInfo.setSourceDocumentKind(document.getKind());
    formatInfo.setFormat(document.getKind(),
IReportFormatOptions.CeReportFormat.PDF);
    publication.setDocumentProcessingInfo(documentID, document.getKind(),
    reportDoc.getProcessingInfo().properties());
   }
   publication.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IDestination`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationFormat`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationPlugin`
- `com.crystaldecisions.sdk.occa.infostore.IDestinations`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.plugin.desktop.common.IFormatInfo`
- `com.crystaldecisions.sdk.plugin.desktop.common.IProcessingPublicationInfo`
- `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOption`
- `com.crystaldecisions.sdk.plugin.destination.managed.IManaged`
- `java.util.Collection`
- `java.util.Iterator`

After a destination is configured, you can add any static documents that need to be delivered to that destination.

Related Information

Supported output formats [page 196]
To add a static document to a publication [page 176]

5.6.6.2 Supported output formats

The supported output formats are defined in the `com.crystaldecisions.sdk.plugin.desktop.common.IReportFormatOptions.CeReportFormat` interface.

The following table shows all supported output formats and which report document kinds support each one.

Output format constant	Crystal Reports	Web Intelligence
CRYSTAL_REPORT	yes	
EXCEL	yes	yes
EXCEL_DATA_ONLY	yes	
MHTML	yes	yes
PDF	yes	yes
RTF	yes	
RTF_EDITABLE	yes	
TEXT_CHARACTER_SEPARATED	yes	
TEXT_PAGINATED	yes	
TEXT_PLAIN	yes	
TEXT_TAB_SEPARATED	yes	
TEXT_TAB_SEPARATED_TEXT	yes	
USER_DEFINED	n/a	n/a
WORD	yes	
XML	yes	

For more information on the specific output formats, see the *SAP BusinessObjects Business Intelligence Platform User's Guide*.

Related Information

To configure a destination and output format [page 193]

5.6.7 Troubleshooting publication exceptions

Before starting a publication, you can perform an integrity check by calling the `checkPublicationIntegrity` method of `IPublication`. The integrity check analyzes the publication object to see if there are any configuration errors that would prevent it from publishing. This can be useful if you want to avoid exceptions due to configuration errors, especially for large, long-running publications.

The following table describes common exceptions that are thrown during the integrity checking and publishing processes, and explains how to correct them.

Exception	Description and solution
`MultipleDocumentKinds`	This exception is thrown when the publication contains more than one kind of schedulable document. A publication may contain multiple schedulable documents, but those documents must all be of the same kind. For example, you can have two Web Intelligence documents in the same publication, but you cannot have one Web Intelligence document and one Crystal Reports document. To correct the problem, modify the collection of documents returned by the `getDocuments` method of `IPublication` so that it only contains one kind of schedulable document (only Crystal Reports, or only Web Intelligence documents). Create additional publications to publish the other kinds of scheduled documents. Note that this restriction does not apply to static documents; the publication can contain any number of documents of other types.
`MissingProcessingInfo`	The publication contains a schedulable document that is missing processing information. There are two possible solutions: • Remove the document from the publication. • Use the `setDocumentProcessingInfo` method of `IPublication` to add processing information for the document.
`InvalidFormatKind`	The publication contains format information that specifies an invalid source document kind. There are two possible solutions: • Remove the invalid `IFormatInfo` object from the `IFormatInfos` collection. If the entry exists in the document processing info, use `IProcessingPublicationInfo.getFormatInfos()` to get the collection. If the entry exists in the destination, use `IDestinationFormat.getFormatInfos()`. • Use the `setSourceDocumentKind` method of the invalid `IFormatInfo` object to change the kind to one of CrystalReports, WebIntelligence, and DesktopIntelligence, as appropriate.
`DuplicateFormatInfo`	The publication contains duplicate document format information.

Exception	Description and solution
	Remove all entries in the `IFormatInfos` collection that contain duplicate kind or format information.
`InvalidDestinationDocument`	The publication contains an invalid destination document. There are two possible solutions: • Remove the invalid document from the `IDestinationFormats` collection. • Add the document to the collection returned by the `getDocuments` method of `IPublication`.
`InvalidDestinationFormat`	The publication contains a destination document with invalid format information. There are two possible solutions: • Add an `IFormatInfo` entry for the document that matches the format described in the destination. • Remove the invalid `IFormatInfo` object from the `IDestinationFormats` collection.
`DuplicateDestinationMergeFormat`	The publication contains a destination with a duplicate merge format. Remove the duplicate merge format from the `IDestinationPluginArtifactFormats` collection.
`InvalidDestinationStaticDocument`	The publication has a destination that contains an invalid static document. There are two possible solutions: • Remove the static document from the `IDestinationStaticDocuments` collection. • Add the static document to the collection returned by the `getDocuments` method of `IPublication`.
`DuplicateDestinationStaticDocument`	The publication has a destination that contains a duplicate static document. To correct the problem, remove the duplicate entry from the `IDestinationStaticDocuments` collection.

5.6.8 Publication extensions

5.6.8.1 What is a publication extension?

Publication extensions let you customize the publication process. For example, you can use a post-delivery publication extension to clean up scope batches for high-volume publications. Or you can use a pre-delivery publication extension to include additional artifacts in the personalized publication content.

There are three types of publication extensions:

- Dynamic recipient data providers, which allow you to add publication recipients from an external data source.
- Post-processing (pre-delivery) plugins, which allow you to add processing logic after publication contents have been personalized.
- Distribution complete (post-delivery) plugins, which allow you to add processing logic after publication contents have been delivered.

In this section, you learn about the different types of publication extensions and how to create, deploy, and invoke them programmatically.

5.6.8.2 Building and deploying custom publication extensions

To build a publication extension, you must include `pub_common.jar` in your class path in addition to the standard SAP BusinessObjects Business Intelligence platform SDK libraries. On Windows, this file is located by default in `C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\java\lib`.

> **Note**
>
> In addition to `pub_common.jar` and the standard libraries, if you are creating a post-processing plugin and you are using the `PostProcessingPluginHelper` class, you must also include `pub_processing.jar` in your class path. This library is located in the same directory as `pub_common.jar`.

After the publication extension is built, deploy it in the publication extensions directory. On Windows, the default location of this directory is `C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\java\lib\publishingPlugins`. Then restart the Adaptive Processing Server.

5.6.8.3 To invoke a publication extension in a publication

Before you can invoke a publication extension in a publication, the extension jar file must be deployed correctly.

1. Execute a query on the CMS repository to retrieve the publication object, and cast the object to `com.businessobjects.sdk.desktop.plugin.publication.IPublication`. The query must include the `SI_PUBLICATION_DOCUMENTS` `SI_PUBLICATION_EVENT_HANDLERS`, and `SI_SCHEDULEINFO` properties.

```
String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS, "
        + "SI_PUBLICATION_EVENT_HANDLERS, SI_SCHEDULEINFO FROM CI_INFOOBJECTS
"
        + "WHERE SI_KIND = '" + IPublication.KIND + "' AND SI_NAME =
'MyPublication' "
        + "AND SI_INSTANCE = 0";
IInfoObjects publications = infostore.query(publicationQuery);
IPublication publication = (IPublication) publications.get(0);
```

2. Retrieve the collection of publication event handlers associated with the publication, and add an event handler that corresponds to the type of the publication extension you want to invoke.

   The `addPublicationEventHandler` method of the `IPublicationEventHandlers` collection takes one parameter that specifies the type of event handler to add. The following constants defined in `com.crystaldecisions.sdk.occa.infostore.CePropertyID` are supported:

Constant	Description
`SI_ON_READ_RECIPIENTS`	Invokes a dynamic recipient data provider when the publishing engine builds the list of recipients for the publication.
`SI_ON_POST_PROCESS_PERSONALIZED_DOCS`	Invokes a post-processing plugin after the publishing engine finishes personalizing the publication content and is ready to deliver it to the recipients.
`SI_ON_AFTER_DELIVER_SCOPE_BATCH`	Invokes a distribution complete plugin after the publishing engine delivers the publication content to the recipients.

```
IPublicationEventHandlers handlers =
publication.getPublicationEventHandlers();
IPublicationEventHandler handler =

handlers.addPublicationEventHandler(CePropertyID.SI_ON_READ_RECIPIENTS);
```

> **Note**
>
> A publication can only contain one event handler of each type. If you add an event handler and the collection already contains an event handler for that type, the existing event handler is overwritten.

3. Configure the publication extension name and class name in the event handler.

   The `setName` method of `IPublicationEventHandlerInfo` sets the name to associate with the publication extension within the context of this publication. (For extensions that can be associated with publications through the SAP BusinessObjects Business Intelligence platform UI, this is the name that is displayed.)

   The `setPluginClassName` method of `IPublicationEventHandlerInfo` sets the fully-qualified class name of the publication extension that will be invoked to handle the event. The class name can specify one of the built-in extensions or a custom extension.

```
String pluginClass =

"com.businessobjects.publisher.dynamicrecipients.crystalreports.CRDataProvider
";
IPublicationEventHandlerInfo handlerInfo = handler.add();
handlerInfo.setName("My Plugin");
handlerInfo.setPluginClassName(pluginClass);
```

4. If you need to pass additional information to the publication extension, use the generic parameter.

```
handlerInfo.setGenericParameter("parameterValue");
```

5. If the publication extension is a post-processing plugin, configure artifact output formats for each destination that the plugin supports.

```
IDestinations destinations =
publication.getSchedulingInfo().getDestinations();
Iterator destinationsIter = destinations.iterator();
while(destinationsIter.hasNext())
{
    IDestination destination = (IDestination) destinationsIter.next();
    IDestinationPluginArtifactFormats formats =
destination.getDestinationPluginArtifactFormats();
    IDestinationPluginArtifactFormat format = formats.add();
    format.setFormat(CeFormatNames.PDF);

format.setDistributionMode(IDestinationPluginArtifactFormat.CeDistributionMode
.FILTER_EXCLUDE_SOURCE_DOCUMENTS);
}
```

The publication extension is invoked at the appropriate stage of the publication process, as determined by the event handler type.

Example

The following example retrieves a publication and adds an event handler that invokes a custom data provider publication extension.

```
void setCustomDynamicRecipientDataProviderForPublication(IInfoStore infostore,
String publicationName) throws SDKException
{
  String publicationQuery = "SELECT SI_ID, SI_PUBLICATION_DOCUMENTS FROM
CI_INFOOBJECTS "
                + "WHERE SI_KIND = '"+ IPublication.KIND + "' "+ "AND SI_NAME =
'"+ publicationName
                + "' "    + "AND SI_INSTANCE = 0";

        IInfoObjects publications = infostore.query(publicationQuery);
        IPublication publication = (IPublication) publications.get(0);
        IPublicationEventHandlers handlers =
publication.getPublicationEventHandlers();
        IPublicationEventHandler handler =
handlers.addPublicationEventHandler(CePropertyID.SI_ON_READ_RECIPIENTS);

        IPublicationEventHandlerInfo handlerInfo = handler.add();
        handlerInfo.setName("MySQL Recipient Provider");

handlerInfo.setPluginClassName("com.businessobjects.sample.MySqlRecipientProvider
");

handlerInfo.setGenericParameter("localhost;3306;dbname;dbuser;dbpassword;select
* from person");

        IPublicationDynaRecipientProfileValueMappings valueMappings =
publication.getDynamicRecipientsProfileValueMappings();
        valueMappings.setName("given_name");
        valueMappings.setFullName("family_name");
        valueMappings.setEmailAddress("email");
        publication.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.publication.IPublication`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationDynaRecipientProfileValueMappings`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandler`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandlerInfo`
- `com.businessobjects.sdk.plugin.desktop.publication.IPublicationEventHandlers`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.CePropertyID`
- `com.crystaldecisions.sdk.occa.infostore.IDestination`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationPluginArtifactFormat`
- `com.crystaldecisions.sdk.occa.infostore.IDestinationPluginArtifactFormats`
- `com.crystaldecisions.sdk.occa.infostore.IDestinations`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.common.CeFormatNames`
- `java.util.Iterator`

Related Information

Building and deploying custom publication extensions [page 199]
Post-processing plugins [page 205]
Distribution complete plugins [page 207]

5.6.8.4 Dynamic recipient data providers

A dynamic recipient data providers is a publication extension that retrieves recipients from an external data source and subscribes them to a publication. The BI platform includes the following built-in dynamic recipient data providers:

- `com.businessobjects.publisher.dynamicrecipients.crystalreports.CRDataProvider`, which reads dynamic recipient information from a Crystal Reports document.
- `com.businessobjects.publisher.dynamicrecipients.rebean.REDataProvider`, which reads dynamic recipient information from an Web Intelligence document.

You can create custom data providers to read dynamic recipient information from other data sources.

> **Note**
>
> Custom dynamic recipient data providers are not available from the Central Management Console or BI launch pad. They can only be added to publications programmatically. For more information on adding dynamic recipients to a publication programmatically, see To add dynamic recipients to a publication [page 182].

A custom data provider consists of a Java class that implements the `com.businessobjects.publisher.dynamicrecipients.IDataProvider` interface. This interface contains four methods, which must be implemented by the data provider class:

Method	Description
`getRecipients`	This method must return a `java.sql.ResultSet` containing the dynamic recipients. The parameter is a `com.businessobjects.publisher.dynamicrecipients.IDataProviderContext` object, which contains information about the context, including any parameters that were used to invoke the extension.
`getSources`	This method is implemented by internal data providers. Custom data providers must implement this method to return null and must not contain any other logic.
`keepAlive`	This method is invoked at regular intervals to ping the data source and ensure it is still active. Any code required to keep the data source connection alive should be implemented within this method. If the data source connection is still valid, the method must return true.
`close`	This method can be used to perform clean-up. For example, any code required to close database connections should be implemented within this method.

For detailed information on the `IDataProvider` and `IDataProviderContext` interfaces, see the *SAP BusinessObjects Business Intelligence Platform Java API Reference*.

Example

The following data provider class retrieves dynamic recipients from a MySQL database. It accepts a semi-colon delimited string via the generic parameter. The format of the string is:

```
<dbhost>;<port>;<dbname>;<user>;<password>;<query>
```

where:

- `dbhost` is the MySQL database server host name.
- `port` is the MySQL database server port number.
- `dbname` is the name of the database that contains the dynamic recipients.
- `user` is the username of a MySQL user who has at elast read privileges on the specified database.
- `password` is the password of the specified user.
- `query` is the SQL query that retrieves the dynamic recipient data.

> **Note**
>
> This example requires the Connector/J 5.0 MySQL JDBC driver. A General Public License (GPL) version of this driver can be downloaded from `http://www.mysql.com/products/connector/j`. Make sure that the `mysql-connector-java-5.0.4-bin.jar` library is on the class path.

```
package com.businessobjects.sample;
```

```
import com.businessobjects.publisher.dynamicrecipients.*;
import com.businessobjects.foundation.logging.*;
import java.util.List;
import java.sql.*;
public class MySqlRecipientProvider implements IDataProvider {
    private ILogger logger =
LoggerManager.getLogger(MySqlRecipientProvider.class);
    private static final long serialVersionUID = 1L;
 private Connection connection;
 private Statement statement;
 private ResultSet resultSet;

 public ResultSet getRecipients(IDataProviderContext context) throws
SQLException, Exception
 {
   String param = context.getGenericParameter();
   String[] parts = param.split(";");

   if(parts.length < 6)
   {
     logger.error("Missing parameter values or incorrect parameter format. "
               + "Expected format:
<dbhost>;<port>;<dbname>;<user>;<password>;<query>");
     return null;
   }

   String host = parts[0];
   String port = parts[1];
   String dbname = parts[2];
   String user = parts[3];
   String password = parts[4];
   String query = parts[5];

   new com.mysql.jdbc.Driver();

   String dbUrl = "jdbc:mysql://" + host + ":" + port + "/" + dbname;
   connection = (Connection) DriverManager.getConnection(dbUrl, user, password);

   statement = connection.createStatement();
   resultSet = statement.executeQuery(query);
   return resultSet;
 }
 public List getSources(IDataProviderContext context) throws SQLException,
Exception
 {
   return null;
 }
 public boolean keepAlive() throws SQLException, Exception
 {
   return true;
 }
 public void close() throws SQLException, Exception
 {
   resultSet.close();
   statement.close();
   connection.close();
  }
}
```

Related Information

Enterprise recipients and dynamic recipients [page 179]
To add dynamic recipients to a publication [page 182]

Building and deploying custom publication extensions [page 199]

5.6.8.5 Post-processing plugins

A post-processing plugin is a publication extension that is invoked during the personalization stage of the publishing process, usually to manipulate the publication contents.

Post-processing plugins can be added to a publication either programmatically or from the Central Management Console (CMC).

> **Note**
>
> The BI launch pad application does not allow you to specify post-processing plugins for a publication through the UI.

The BI platform includes the following built-in post-processing plugins:

- `com.businessobjects.publisher.postprocessing.plugin.ZipMergePlugin`, which merges content into a single zip archive.
- `com.businessobjects.publisher.postprocessing.plugin.PdfMergePlugin`, which merges content into a single Adobe Acrobat Reader document.

You can create custom post-processing plugins to add custom behavior to the publishing process.

A custom post-processing plugin consists of a Java class that implements the `com.businessobjects.publisher.postprocessing.IPublicationPostProcessingPlugin` interface. This interface contains two methods, which must be implemented by the plugin class:

Method	Description
`handle`	This method is invoked one time for each destination and scope. It must return an `IInfoObjects` collection containing the artifacts created by the plugin that will be delivered to the recipients associated with the current destination and scope. The parameter is a `com.businessobjects.publisher.postprocessing.IPublicationPostProcessingContext` object, which contains information about the context, including the current scope and destination, and any parameters that were used to invoke the plugin.
`getTargetDestinations`	This method must return a collection of `com.businessobjects.publisher.postprocessing.PluginTargetDestination` representing the destinations for which the plugin will be invoked.

The `com.businessobjects.publisher.postprocessing.PostProcessingPluginHelper` provides several static methods that can be used in post-processing plugins. For example, to add new artifacts, use `createInfoObject` method. The following code creates a text file info object.

```
ITxt textInfoObject = (ITxt)
PostProcessingPluginHelper.createInfoObject(context,
```

```
                 ITxt.PROGID, "text/plain", null, null);
```

For detailed information on `IPublicationPostProcessingPlugin`, `IPublicationPostProcessingContext`, `PluginTargetDestination`, and `PostProcessingPluginHelper`, see the *SAP BusinessObjects Business Intelligence Platform Java API Reference* .

> **Note**
>
> In addition to `pub_common.jar` and the standard SAP BusinessObjects Business Intelligence Platform SDK libraries, if you use `PostProcessingPluginHelper`, you must also include `pub_processing.jar` in your class path.

Example

The following post-processing plugin class adds a text file to each personalized version of the publication. The text file contains a list of the artifacts created within the current scope, excluding the text file itself.

```
package org.example.bobj.boe.sdk.publication;
import java.util.*;
import java.io.*;
import com.crystaldecisions.sdk.occa.infostore.*;
import com.crystaldecisions.sdk.plugin.desktop.txt.*;
import com.crystaldecisions.sdk.plugin.destination.managed.*;
import com.crystaldecisions.sdk.plugin.destination.smtp.*;
import
com.businessobjects.publisher.postprocessing.IPublicationPostProcessingContext;
import
com.businessobjects.publisher.postprocessing.IPublicationPostProcessingPlugin;
import com.businessobjects.publisher.postprocessing.PluginTargetDestination;
import com.businessobjects.publisher.postprocessing.PostProcessingPluginHelper;
import com.businessobjects.publisher.common.ILoggerAdapter;
public class PubExtension implements IPublicationPostProcessingPlugin
{
  public IInfoObjects handle(IPublicationPostProcessingContext context) throws
Exception
  {
          ILoggerAdapter logger = context.getAdminLogAdapter();
          IDestination destination = context.getDestination();
          int scopeID = context.getScopeID();
          logger.info("Currently processing Scope ID " + scopeID
                  + " for destination '" + destination.getName() + "'");
          int publicationID = context.getPublication().getID();
          String filename = (String) context.getOptions();
          if (filename == null || filename.trim().length() == 0)
    {
               filename = "c:\\manifest-publication-" + publicationID + "-scope-"
                        + scopeID + "_" + System.currentTimeMillis() + ".txt";
          }

          ArrayList documents = context.getDocuments();
          if (documents == null || documents.size() == 0)
    {
               throw new Exception("No documents to process.");
    }

          File file = new File(filename);
          FileOutputStream fileOutputStream = new FileOutputStream(file);
          PrintWriter printWriter = new PrintWriter(fileOutputStream);
```

```
          printWriter.println("PubExtension manifest for publication with SI_ID "
                  + publicationID + ", scope ID " + scopeID);

          Iterator documentIter = documents.iterator();
          while (documentIter.hasNext())
    {
               IInfoObject document = (IInfoObject) documentIter.next();
               String message = "SI_CUID: " + document.getCUID() + ", SI_ID: "
                       + document.getID() + ", " + document.getTitle();
               printWriter.println(message);
          }
          printWriter.close();

          ArrayList<File> files = new ArrayList<File>();
          files.add(file);
          ITxt textInfoObject = (ITxt)
PostProcessingPluginHelper.createInfoObject(context,
      ITxt.PROGID, "text/plain", null, null);
          textInfoObject.getFiles().addFile(file);
          IInfoObjects newInfoObjects =
context.getInfoStore().newInfoObjectCollection();
          newInfoObjects.add(textInfoObject);
          return newInfoObjects;
     }
    public Collection getTargetDestinations() throws SDKException
 {
         PluginTargetDestination inboxDestination = new PluginTargetDestination(
                 IManaged.PROGID,

IDestinationPluginArtifactFormat.CeDistributionMode.FILTER_EXCLUDE_SOURCE_DOCUMEN
TS);
         PluginTargetDestination smtpDestination = new PluginTargetDestination(
              ISMTP.PROGID,

IDestinationPluginArtifactFormat.CeDistributionMode.FILTER_EXCLUDE_SOURCE_DOCUMEN
TS);
         PluginTargetDestination[] destinations = {inboxDestination,
smtpDestination};
         return Arrays.asList(destinations);
     }
}
```

Related Information

Building and deploying custom publication extensions [page 199]

5.6.8.6 Distribution complete plugins

A distribution complete plugin is a publication extension that executes after the publishing engine has finished processing a scope batch and has delivered the personalized publication contents to the recipients associated with that scope batch. Distribution complete plugins can be added to a publication either programmatically or from the Central Management Console (CMC).

> **Note**
>
> The BI launch pad application does not allow you to specify distribution complete plugins for a publication through the UI.

You can create custom distribution complete plugins to perform specific actions after scope batch completion.

A custom distribution complete plugin consists of a Java class that implements the `com.businessobjects.publisher.distribution.IDistributionCompletePlugin` interface. This interface contains one method, which must be implemented by the plugin class:

Method	Description
`onDistributionComplete`	This method must return true containing the dynamic recipients. The parameter is a `com.businessobjects.publisher.distribution.IDistributionCompleteContext` object, which contains information about the context, including the current scope batch and the SAP BusinessObjects Business Intelligence platform session.

For detailed information on the `IDistributionCompletePlugin` and `IDistributionCompleteContext` interfaces, see the *SAP BusinessObjects Business Intelligence Platform Java API Reference*.

Example

The following example deletes all the artifacts that were created for the scope batch and then deletes the scope batch.

```
package com.businessobjects.publisher.distribution;
import com.businessobjects.publisher.common.ILoggerAdapter;
import com.businessobjects.sdk.plugin.desktop.scopebatch.IScopeBatch;
import com.crystaldecisions.sdk.exception.SDKException;
import com.crystaldecisions.sdk.occa.infostore.IInfoObject;
import com.crystaldecisions.sdk.occa.infostore.IInfoObjects;
public class CleanScopeBatch implements IDistributionCompletePlugin
{
  public boolean onDistributionComplete(IDistributionCompleteContext context)
  {
    private ILoggerAdapter logger = context.getAdminLogAdapter();
    try
    {
      IScopeBatch scopebatch = context.getScopeBatch();
      if (scopebatch == null)
      {
        logger.error("No valid ScopeBatch object in
DistributionCompleteContext");
        return false;
      }
      StringBuffer queryString = new StringBuffer("select SI_ID from
CI_INFOOBJECTS where children(\"SI_NAME='PublicationScopeBatch-Artifact'\",
\"SI_ID=");
      queryString.append(scopebatch.getID());
      queryString.append("\")");
      logger.debug("Query for artifacts to delete: "    +
queryString.toString());
      IInfoObjects artifacts =
context.getInfoStore().query(queryString.toString());
      if (artifacts == null || artifacts.size() == 0)
      {
        logger.debug("There were no artifacts to delete for ScopeBatch ["+
scopebatch.getID() + "]");
        return false;
      }
      else
```

```
      {
        logger.info("Deleting " + artifacts.size()+ " artifacts for ScopeBatch
["    + scopebatch.getID() + "]");
      }
      for (int i = 0; i < artifacts.size(); i++)
      {
        IInfoObject artifact = (IInfoObject) artifacts.get(i);
        artifact.deleteNow();
      }
      scopebatch.deleteNow();
    }
    catch (SDKException e)
    {
      return false;
    }
    return true;
  }
}
```

Related Information

Building and deploying custom publication extensions [page 199]

5.7 Server administration

Server administration tasks are generally performed in the Central Management Console (CMC) web application. You can also perform these tasks programmatically by using the Server Intelligence APIs.

Server Intelligence is the server management framework used in BusinessObjects Enterprise XI 3.x and later. It simplifies the administration and deployment of SAP BusinessObjects Business Intelligence platform servers and services, making it easier to optimize performance and fault tolerance, especially in complex deployments. For more information on Server Intelligence, see the *SAP BusinessObjects Business Intelligence Platform Server Administration Guide*.

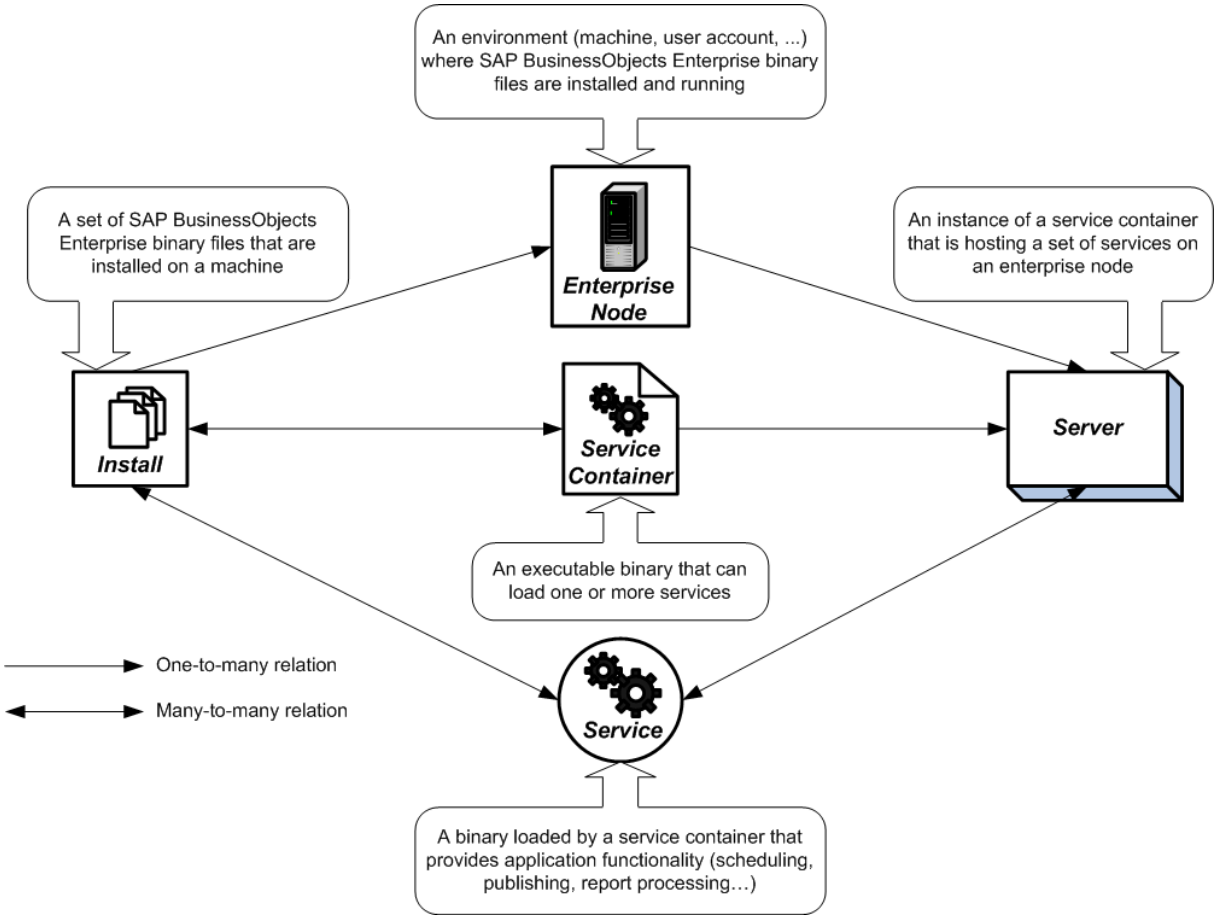
The Server Intelligence framework exposes APIs that allow you to efficiently manage the servers in your BI platform installation. This section explains how to use the Server Intelligence APIs.

Classes used for server administration

- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
  Provides general information on the server, and allows you to start, stop, restart, enable, or disable the server.
- `com.crystaldecisions.sdk.plugin.desktop.servergroup.IServerGroup`
  Manages a server group and specifies which members belong to it.

5.7.1 Server Intelligence architecture

The following diagram represents the components of the Server Intelligence architecture.



5.7.2 Server Intelligence components

A BI platform installation consists of the following components:

- Enterprise nodes.
- Installs.
- Service containers.
- Services.
- Servers.

Enterprise nodes

An enterprise node is the context of a BI platform installation. It contains:

- The physical location of the installed binaries (host machine and hard drive).
- The base installation directory.
- The operating system user account under which binaries are run.
- Placeholders for values used by the Server Intelligence framework, including the absolute paths to the JRE `bin` directory and the directory where temporary audit log files are stored.
- The Server Intelligence Agent process, which monitors the status of all the servers in the deployment.

The simplest type of deployment, where all the components are installed on a single machine, has only one enterprise node. A complex deployment has multiple enterprise nodes, one for each physical machine.

The `com.businessobjects.sdk.plugin.desktop.enterprisenode.IEnterpriseNode` interface is used to represent enterprise nodes.

Installs

An install is a set of BI platform binaries. It defines placeholders for the platform-specific relative paths and filenames these binaries.

Note that an install does not represent an actual installation. It is similar to a manifest containing a list of binaries and their relative paths. The Server Intelligence framework uses this information in conjunction with the information in an enterprise node to determine the absolute paths to the actual binaries on a physical machine.

The simplest type of deployment has only one enterprise node, which would have one install. A complex deployment could have many enterprise nodes and one install, in which case each node would have the same binaries installed. Alternatively, a complex deployment could have multiple installs, in which case different nodes could have different sets of binaries installed.

The `com.businessobjects.sdk.plugin.desktop.install.IInstall` interface is used to represent BI platform installations.

Service containers

A service container is an executable binary file that can be configured to run one or more services. A service container can be used by multiple servers. A server is associated with only one service container.

The `com.businessobjects.sdk.plugin.desktop.servicecontainer.IServiceContainer` interface is used to represent BI platform executables.

Services

A service is a binary file that is loaded by a service container to provide some application functionality, such as report caching, auditing, or report scheduling. A server can host one or more services. The service container determines which services can be hosted by the server.

The `com.businessobjects.sdk.plugin.desktop.service.IService` interface is used to represent BI platform services.

Servers

A server is a running instance of a service container. It contains the configuration information used to launch the associated service container and host one or more services.

The `com.crystaldecisions.sdk.plugin.desktop.server.IServer` interface is used to represent BI platform servers. Each server contains:

- An `com.businessobjects.sdk.plugin.desktop.common.IConfiguredContainer` object, which can be retrieved by calling the `getConfiguredContainer` method of the `IServer` interface. The `IConfiguredContainer` object contains the settings that are used to configure the service container executable when the server is started.
- An `com.businessobjects.sdk.plugin.desktop.common.IConfiguredServices` collection, which can be retrieved by calling the `getConfiguredServices` method of the `IServer` interface. `IConfiguredServices` contains one `com.businessobjects.sdk.plugin.desktop.common.IConfiguredService` object for each service that is hosted by the server. Each `IConfiguredService` object contains the configuration information that is passed to the service when the server is started.

For more information on these interfaces and their members, see the *SAP BusinessObjects Business Intelligence Framework Java API Reference*.

5.7.3 Managing servers

This section explains how to use the Server Intelligence APIs to perform the following server management tasks:

- Starting servers.
- Stopping servers.
- Restarting servers.
- Cloning servers.
- Getting server metrics.

5.7.3.1 To start a server

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the server object from the repository, and cast it to `IServer`.

```
String serverQuery = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_ID=" + serverID;
```

```
IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. Call the `setExpectedRunState` of the server object, and set the argument to `ExpectedRunState.RUNNING`.

```
server.setExpectedRunState(ExpectedRunState.RUNNING);
```

4. Save the server object .

```
server.save();
```

This sets the `SI_EXPECTED_RUN_STATE` property of the server to indicate that the server should be running. If it is not currently running, then the Server Intelligence Agent begins the process of starting the server.

Example

```
void startServer(IEnterpriseSession enterpriseSession, int serverID) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_ID=" + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);

  server.setExpectedRunState(ExpectedRunState.RUNNING);
  server.save();
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState`

5.7.3.2 To stop a server

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the server object from the repository, and cast it to `IServer`.

```
String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. Call the `setExpectedRunState` of the server object, and set the argument to `ExpectedRunState.STOPPED`.

```
server.setExpectedRunState(ExpectedRunState.STOPPED);
```

4. Save the server object .

```
server.save();
```

This sets the`SI_EXPECTED_RUN_STATE` property of the server to indicate that the server should be stopped. If it is currently running, then the Server Intelligence Agent begins the process of stopping the server.

Example

```
void stopServer(IEnterpriseSession enterpriseSession, int serverID) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
  server.setExpectedRunState(ExpectedRunState.STOPPED);
  server.save();
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState`

5.7.3.3 To restart a server

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the server object from the repository, and cast it to `IServer`.

```
String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. Call the `setExpectedRunState` of the server object, and set the argument to `ExpectedRunState.RESTART`.

```
server.setExpectedRunState(ExpectedRunState.RESTART);
```

4. Save the server object .

```
server.save();
```

This sets the`SI_EXPECTED_RUN_STATE` property of the server to indicate that the server should be restarted. If it is currently running, then the Server Intelligence Agent begins the process of stopping the server and starting it again. If the server is not currently running, then the Server Intelligence Agent begins the process of starting it.

Example

```
void restartServer(IEnterpriseSession enterpriseSession, int serverID) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
  server.setExpectedRunState(ExpectedRunState.RESTART);
  server.save();
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState`

5.7.3.4 To clone a server

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the server object from the repository, and cast it to `IServer`.

```
String serverQuery = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_ID=" + serverID;
IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. Call the `newInfoObjectCollection` method of the `IInfoStore` object to create an `IInfoObjects` collection.

```
IInfoObjects cloneInfoObjects = infoStore.newInfoObjectCollection();
```

4. Call the static `copy` method of the `IInfoObjects` collection to create a clone of the server. Set the first argument to the server object that you want to clone. Set the second argument to the constant `IInfoObjects.CopyModes.COPY_NEW_OBJECT_NEW_FILES`.

```
IServer serverClone = (IServer) cloneInfoObjects.copy(server,
  IInfoObjects.CopyModes.COPY_NEW_OBJECT_NEW_FILES);
```

5. Configure properties on the clone object.

> **Note**
>
> If the server is an Input File Repository server, the friendly name must be prefixed with "`Input.`". If it is an Output File Repository server, the friendly name must be prefixed with "`Output.`".

6. Save the clone object .

```
serverClone.save();
```

Example

The following example creates a clone of an existing server.

```
void cloneServer(IEnterpriseSession enterpriseSession, int serverID) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String queryString = "SELECT * FROM CI_SYSTEMOBJECTS WHERE SI_ID=" + serverID;
  IInfoObjects serverInfoObjects = infostore.query(queryString);
  IServer server = (IServer) serverInfoObjects.get(0);
  IInfoObjects cloneInfoObjects = infostore.newInfoObjectCollection();
  IServer serverClone = (IServer) cloneInfoObjects.copy(server,
  IInfoObjects.CopyModes.COPY_NEW_OBJECT_NEW_FILES);

  serverClone.setTitle("Clone_" + System.currentTimeMillis() + "_"
    + server.getTitle());
  serverClone.setFriendlyName("Clone_" + System.currentTimeMillis() + "_"
    + server.getFriendlyName());
  serverClone.setExpectedRunState(ExpectedRunState.RUNNING);
  serverClone.setDisabled(true);
  serverClone.save();
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState`

5.7.3.5 To get server metrics

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the server object from the repository, and cast it to `IServer`.

   The query must include the `SI_METRICS` property.

```
String serverQuery = "SELECT SI_METRICS FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. Retrieve the metric descriptions object from the repository, and cast it to `com.businessobjects.sdk.plugin.desktop.metricdescriptions.IMetricDescriptions`.

```
String metricDescriptionsQuery =
  "SELECT SI_ID, SI_METRIC_DESCRIPTIONS FROM CI_SYSTEMOBJECTS WHERE
SI_KIND='MetricDescriptions'";
IMetricDescriptions metricDescriptions = (IMetricDescriptions)
  infostore.query(metricDescriptionsQuery).get(0);
```

4. Get the server metrics collection.

```
IServerMetrics serverMetrics = server.getMetrics();
```

5. Get the service interface metrics.

   > **Note**
   >
   > To get the names of the available service interfaces, use the `getServiceInterfaceNames` method of the `IServerMetrics` object.

```
IMetrics serviceMetrics = serverMetrics.getMetrics(serviceName);
```

6. Use the iterator to get each metric object from the `IServerMetrics` collection.

```
Iterator serviceMetricsIter = serviceMetrics.iterator();
while(serviceMetricsIter.hasNext())
{
  IMetric metric = (IMetric) serviceMetricsIter.next();
  String metricName = metric.getName();
  ...
}
```

   Note that the `getName` method of `IMetric` returns the internal name of the metric. The following code demonstrates how to retrieve the localized name of a metric.

```
IMLDescriptions descriptions =
metricDescriptions.getMetricDescriptions(serviceName);
IPropertyRenderTemplate propertyRenderTemplate =
descriptions.getPropertyRenderTemplate(metricName);
String localizedMetricName = propertyRenderTemplate.getLabel(Locale.ENGLISH);
```

Example

The following code retrieves the metrics for all service interfaces on a server and displays them in an HTML table.

```
String getServerMetrics(IEnterpriseSession enterpriseSession, int serverID)
throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_METRICS FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);

  String metricDescriptionsQuery = "SELECT SI_ID, SI_METRIC_DESCRIPTIONS FROM
CI_SYSTEMOBJECTS WHERE SI_KIND='MetricDescriptions'";
  IMetricDescriptions metricDescriptions = (IMetricDescriptions)
infostore.query(metricDescriptionsQuery).get(0);
  IServerMetrics serverMetrics = server.getMetrics();
  Set serviceNames = serverMetrics.getServiceInterfaceNames();
  Iterator serviceNamesIter = serviceNames.iterator();
  String resultString = "";
  while(serviceNamesIter.hasNext())
  {
    String serviceName = (String) serviceNamesIter.next();
    IMetrics serviceMetrics = serverMetrics.getMetrics(serviceName);
    resultString += "<div class=\"bordered\"><h5>" + serviceName + " metrics</
h5>\n<table>"
      + "<tr><th align=\"left\">Name</th><th align=\"left\">Value</th></tr>\n";
    Iterator serviceMetricsIter = serviceMetrics.iterator();

    while(serviceMetricsIter.hasNext())
    {
      IMetric metric = (IMetric) serviceMetricsIter.next();
      String metricName = metric.getName();
      IMLDescriptions descriptions =
metricDescriptions.getMetricDescriptions(serviceName);
      IPropertyRenderTemplate propertyRenderTemplate =
descriptions.getPropertyRenderTemplate(metricName);
      String localizedMetricName =
propertyRenderTemplate.getLabel(Locale.ENGLISH);
      resultString += "<tr><td valign=\"top\">" + localizedMetricName + "</
td><td valign=\"top\">"
        + metric.getValue() + "</td></tr>\n";
    }
    resultString += "</table></div>";
  }
  return resultString;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServerMetrics`
- `com.businessobjects.sdk.plugin.desktop.common.IMetric`
- `com.businessobjects.sdk.plugin.desktop.common.IMetrics`

- com.businessobjects.sdk.plugin.desktop.metricdescriptions.IMetricDescriptions
- com.businessobjects.sdk.plugin.desktop.metricdescriptions.IMLDescriptions
- com.businessobjects.sdk.plugin.desktop.metricdescriptions.IPropertyRenderTemplate
- java.util.Iterator
- java.util.Locale
- java.util.Set

5.7.4 Adding servers

To programmatically add a server to a deployment, you must first determine which enterprise node will host the server. Next, you must choose the service container that the server will use. The service container determines which services can be hosted by the server, so make sure that you select a service container that can support the services you want to host. Finally, you must decide which services will be provided by the server.

This section explains how to add a server to a deployment, and describes the different services and service containers you can use to create a server.

Related Information

To get all service containers [page 219]
To get all services [page 221]
To add a server [page 223]

5.7.4.1 To get all service containers

When adding a new server, you must associate it with a service container. The service container determines which services the server can host. You can query the Central Management Server (CMS) for all service containers in the deployment and retrieve information such as the CUID, name, and supported services for each container.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. Query for and retrieve all service containers from the CMS.

   The service containers are found in folder 55 of the CMS. Retrieve the `SI_ML_DESCRIPTION` property bag which contains the locale-specific descriptions of each container, and the `SI_SERVICE4SC` property bag which contains the IDs of all supported services for the container.

```
String servicesQuery = "SELECT SI_ID, SI_NAME, SI_CUID, SI_ML_DESCRIPTION,
SI_SERVICE4SC FROM CI_SYSTEMOBJECTS WHERE SI_KIND='ServiceContainer' AND
SI_PARENT_FOLDER=55";
```

```
IInfoObjects serviceContainers = (IInfoObjects)
infostore.query(servicesQuery);
```

3. Use an iterator to access each service container from the `IInfoObjects` collection.

```
Iterator it = serviceContainers.iterator();
while (it.hasNext())
{
  IServiceContainer serviceContainer = (IServiceContainer) it.next();
  ...
}
```

4. Call the `getDescription` and `getCUID` methods of the `IServiceContainer` class to retrieve identifying information for each container.

> **Note**
>
> The `getDescription` method takes in a locale as a parameter. This locale-specific description is displayed in the Central Management Console (CMC) UI.

```
resultString += "<b>Service Container Description:</b> " +
serviceContainer.getDescription(Locale.ENGLISH) + " - ";
resultString += "<b>CUID: </b>" + serviceContainer.getCUID() + "<br>";
```

5. Call the `getServices` method of the `IServiceContainer` class to retrieve the IDs of all supported services for that container.

```
Set supportedServices = serviceContainer.getServices();
```

6. Use an iterator to access each supported service ID.

```
Iterator servIt = supportedServices.iterator();
while (servIt.hasNext())
{
  Integer serviceID = (Integer) servIt.next();
  ...
}
```

Example

The following example retrieves the description, CUID, and supported service IDs for each service container in the CMS.

```
String getServiceContainers(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String servicesQuery = "SELECT SI_ID, SI_NAME, SI_CUID, SI_ML_DESCRIPTION,
SI_SERVICE4SC FROM CI_SYSTEMOBJECTS WHERE SI_KIND='ServiceContainer' AND
SI_PARENT_FOLDER=55";
  IInfoObjects serviceContainers = (IInfoObjects) infostore.query(servicesQuery);
  String resultString = "";
  Iterator it = serviceContainers.iterator();
  while (it.hasNext())
  {
    IServiceContainer serviceContainer = (IServiceContainer) it.next();
    resultString += "<b>Service Container Description:</b> " +
serviceContainer.getDescription(Locale.ENGLISH) + " - ";
```

```
    resultString += "<b>CUID: </b>" + serviceContainer.getCUID() + "<br>";
    resultString += "<b>Supported Service IDs:</b><br>";

    Set supportedServices = serviceContainer.getServices();
    Iterator servIt = supportedServices.iterator();
    while (servIt.hasNext())
    {
      Integer serviceID = (Integer) servIt.next();
      resultString += "    " + serviceID + "<br>";
    }
    resultString += "<br>";
  }
  return resultString;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.businessobjects.sdk.plugin.desktop.servicecontainer.IServiceContainer`
- `java.util.Iterator`
- `java.util.Locale`
- `java.util.Set`

5.7.4.2 To get all services

When adding a new server, you must decide what services you want the server to host. You can query the Central Management Server (CMS) for all services in the deployment and retrieve information such as the description and CUID.

> **Note**
>
> The new services you add must be supported by the service container associated with your new server. You can use the `IService.getContainers` method to obtain the IDs of all service containers that are configured to host that particular service.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
   ```

2. Query for and retrieve all services from the CMS.

   The services are found in folder 52 of the CMS. Retrieve the `SI_ML_DESCRIPTION` property bag which contains the locale-specific descriptions of each service.

   ```
   String servicesQuery = "SELECT SI_ID, SI_NAME, SI_CUID, SI_ML_DESCRIPTION
   FROM CI_SYSTEMOBJECTS WHERE SI_KIND='Service' AND SI_PARENT_FOLDER=52";
   IInfoObjects services = (IInfoObjects) infostore.query(servicesQuery);
   ```

3. Use an iterator to access each service from the `IInfoObjects` collection.

   ```
   Iterator it = services.iterator();
   ```

```
while (it.hasNext())
{
  IService service = (IService) it.next();
  ...
}
```

4. Call the `getDescription` and `getCUID` methods of the `IService` class to retrieve identifying information for each service.

> **Note**
>
> The `getDescription` method takes in a locale as a parameter. This locale-specific description is displayed when adding services to servers through the Central Management Console (CMC) UI.

```
resultString += "<b>Service Description:</b> " +
service.getDescription(Locale.ENGLISH) + " - ";
resultString += "<b>CUID: </b>" + service.getCUID() + "<br>";
```

Example

The following example retrieves the description and CUID for each service in the CMS.

```
String getServices(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infostore =
(IInfoStore)enterpriseSession.getService("InfoStore");
  String servicesQuery = "SELECT SI_ID, SI_NAME, SI_CUID, SI_ML_DESCRIPTION FROM
CI_SYSTEMOBJECTS WHERE SI_KIND='Service' AND SI_PARENT_FOLDER=52";
  IInfoObjects services = (IInfoObjects) infostore.query(servicesQuery);
  String resultString = "";
  Iterator it = services.iterator();
  while (it.hasNext())
  {
    IService service = (IService) it.next();
    resultString += "<b>Service Description:</b> " +
service.getDescription(Locale.ENGLISH) + " - ";
    resultString += "<b>CUID: </b>" + service.getCUID() + "<br>";
  }
  return resultString;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.businessobjects.sdk.plugin.desktop.service.IService`
- `java.util.Iterator`
- `java.util.Locale`

5.7.4.3 To add a server

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Call the `newInfoObjectCollection` method of the `IInfoStore` object to create an `IInfoObjects` collection.

```
IInfoObjects servers = infostore.newInfoObjectCollection();
```

3. Call the `add` method of the collection to create an `IServer` object.

```
IServer server = (IServer) servers.add(IServer.KIND);
server.setTitle(newServerName);
server.setDescription("Sample adaptive processing server");
```

4. Call `setFriendlyName` to specify a friendly name for the server.

```
server.setFriendlyName(newServerName);
```

> **Note**
>
> If the server is an Input File Repository server, the friendly name must be prefixed with "`Input.`". If it is an Output File Repository server, the friendly name must be prefixed with "`Output.`".

5. Call `setExpectedRunState` to specify whether you want the server to start automatically after it is created.

   The `com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState` class contains the constants for the expected run state.

```
server.setExpectedRunState(ExpectedRunState.RUNNING);
```

6. Call `setDisabled` to specify whether you want the server to be disabled by default.

```
server.setDisabled(true);
```

7. Call `setAutoBoot` to specify whether you want the server to be managed by the Server Intelligence Agent.

```
server.setAutoBoot(true);
```

8. Save the server object to the repository and re-fetch it.

```
server.save();
String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
server.getID();
server = (IServer) infostore.query(serverQuery).get(0);
```

> **Note**
>
> When you save changes to an object, it is recommended that you re-fetch it after saving. This ensures that all properties contain valid values.

9. Connect a service container to the server.

```
String serviceContainerQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE
SI_PROGID = '"
```

```
  + IServiceContainer.KIND + "' AND SI_NAME =
'AdaptiveProcessingServiceContainer'";
IServiceContainer serviceContainer = (IServiceContainer)
infoStore.query(serviceContainerQuery).get(0);
server.setContainer(serviceContainer.getID());
```

The service container determines which types of services can be hosted on the server. You can use the `getServices` method of the `IServiceContainer` class to retrieve the IDs of all supported services for that container.

10. Connect the server to the enterprise node that you want to host it.

```
String enterpriseNodeQuery = "SELECT TOP 1 SI_ID FROM CI_SYSTEMOBJECTS WHERE
SI_KIND = '"
  + IEnterpriseNode.KIND + "'";
IEnterpriseNode enterpriseNode = (IEnterpriseNode)
infostore.query(enterpriseNodeQuery).get(0);
server.setEnterpriseNode(enterpriseNode.getID());
```

11. Retrieve the services that you want to host on the server.

```
String servicesQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE "
  + "SI_NAME IN('AuditingService', 'PublishingService', 'AuditProxyService')";
IInfoObjects services = infostore.query(servicesQuery);
```

> **Note**
>
> The `SI_NAME` value is not the service description visible in the Central Management Console (CMC) when selecting services for a server. The service description in the UI refers to the locale-specific value in the `SI_ML_DESCRIPTION` property bag. You can query for a list of all services in the CMS using the SDK to obtain the `SI_NAME` and `SI_CUID` values of the services you want to add.

12. Add each service to the list of services hosted by the server.

```
IConfiguredServices configuredServices = server.getHostedServices();
Iterator serviceIter = services.iterator();
while(serviceIter.hasNext())
{
  IService service = (IService) serviceIter.next();
  configuredServices.add(service.getID());
  ...
}
```

13. Save the server object.

```
server.save();
```

You can now access the server through the *Servers* page of the Central Management Console.

Example

```
void addServer(IEnterpriseSession enterpriseSession, String newServerName)
throws SDKException
 {
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects servers = infostore.newInfoObjectCollection();
  IServer server = (IServer) servers.add(IServer.KIND);
  server.setTitle(newServerName);
```

```
  server.setDescription("Sample adaptive processing server");
  server.setFriendlyName(newServerName);
  server.setExpectedRunState(ExpectedRunState.RUNNING);
  server.setDisabled(true);
  server.setAutoBoot(true);
  server.save();
  String serverQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE SI_ID=" +
server.getID();
  server = (IServer) infostore.query(serverQuery).get(0);
  String serviceContainerQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE
SI_KIND = '"
    + IServiceContainer.KIND + "' AND SI_NAME =
'AdaptiveProcessingServiceContainer'";
  IServiceContainer serviceContainer = (IServiceContainer)
infostore.query(serviceContainerQuery).get(0);
  server.setContainer(serviceContainer.getID());

  String enterpriseNodeQuery = "SELECT TOP 1 SI_ID FROM CI_SYSTEMOBJECTS WHERE
SI_KIND = '"
    + IEnterpriseNode.KIND + "'";
  IEnterpriseNode enterpriseNode = (IEnterpriseNode)
infostore.query(enterpriseNodeQuery).get(0);
  server.setEnterpriseNode(enterpriseNode.getID());

  String servicesQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS WHERE "
    + "SI_NAME IN('AuditingService', 'PublishingService', 'AuditProxyService')";
  IInfoObjects services = infostore.query(servicesQuery);

  IConfiguredServices configuredServices = server.getHostedServices();
  Iterator serviceIter = services.iterator();
  while(serviceIter.hasNext())
  {
    IService service = (IService) serviceIter.next();
    configuredServices.add(service.getID());
    server.save();
  }
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.ExpectedRunState`
- `com.businessobjects.sdk.plugin.desktop.enterprisenode.IEnterpriseNode`
- `com.businessobjects.sdk.plugin.desktop.service.IService`
- `com.businessobjects.sdk.plugin.desktop.servicecontainer.IServiceContainer`
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredService`
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredServices`

5.7.5 Configuring services and service containers

Before proceeding, it is important to know about the different parts of a service configuration.

A service configuration consists of:

- Configuration properties.
- Execution properties.
- Local files.

Related Information

Configuration properties [page 226]
Execution properties [page 230]
Local files [page 231]

5.7.5.1 Configuration properties

Configuration properties control the behavior of a service or service container. The values of these properties can be changed at any time. However, depending on the property, the change may not be applied immediately. Some properties require that you restart the server in order for the change to take effect.

The configuration properties of a service are stored in a `com.businessobjects.sdk.plugin.desktop.common.IServiceConfigProperties` object, which can be retrieved by calling `getConfigProps` on an `IConfiguredService` object.

The configuration properties of a service container are stored in a `com.businessobjects.sdk.plugin.desktop.common.IServiceContainerConfigProperties` object, which can be retrieved by calling `getConfigProps` on an `IConfiguredContainer` object.

5.7.5.1.1 Actual and requested configuration properties

Some configuration property changes cannot be applied while the server is active. The Server Intelligence framework tracks both the current, or actual, configuration and the requested configuration changes that will be applied when the server hosting is restarted.

The `getActualConfigProps` method of `IConfiguredService` and `IConfiguredContainer` returns an `IActualConfigProperties` collection, which inherits from `IConfigProperties`. This collection contains the configuration properties that are currently applied to the service hosted on the server.

5.7.5.1.2 Configuration property defaults

When a service container or a service is associated with a server, the default configuration property values are automatically copied from the configuration in the `IService` or `IServiceContainer` object. If you make changes to the configuration properties that cause problems, you can revert to the default configuration property values in the `IService` or `IServiceContainer` object.

5.7.5.1.3 To retrieve configuration properties

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the server object from the repository, and cast it to `IServer`.

   The query must include the `SI_HOSTED_SERVICES` property.

```
String serverQuery = "SELECT SI_ID, SI_HOSTED_SERVICES FROM "
  + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. Call the `getHostedServices` method of the server object.

   This returns an `IConfiguredServices` collection, which contains configuration information for the services that are hosted by the server.

```
IConfiguredServices cfgServices = server.getHostedServices();
```

4. Call the `get` method of the `IConfiguredServices` collection to retrieve a single configured service.

```
Set cfgServiceIDs = cfgServices.getConfiguredServiceIDs();
Integer cfgServiceID = (Integer) cfgServiceIDs.toArray()[0];
IConfiguredService cfgService = cfgServices.get(cfgServiceID.intValue());
```

   This example uses the `getConfiguredServiceIDs` method of the `IConfiguredServices` collection and retrieves the first ID from the resulting set. Set the argument to the ID of the service for which you want to retrieve the configuration information. This returns an `IConfiguredService` object, which contains the configuration of the specified service on this server.

5. Call the `getConfigProps` method of the `IConfiguredService` object and iterate through the names and values of the configuration properties for the selected service.

```
IConfigProperties configProps = (IConfigProperties)
cfgService.getConfigProps();
String[] propertyNames = configProps.getPropNames();
String resultString = "";
int size = propertyNames.length;
for(int i = 0; i < size; i++)
{
  IConfigProperty property = configProps.getProp(propertyNames[i]);
  String name = property.getDisplayName(Locale.ENGLISH);
  String value = property.getValue().toString();
  ...
}
```

   The `getConfigProps` method returns an `IConfigurationProperties` collection, which contains the individual configuration properties. Use the `getPropNames` method to return an array containing the names of the configuration properties for the service. Then use the `getProp` method of the `IConfigurationProperties` collection to retrieve the property from the collection.

Example

The following example retrieves the configuration properties for a configured service and prints their localized display names and values to the standard output.

```
String retrieveConfigProperties(IEnterpriseSession enterpriseSession, int
serverID) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID, SI_HOSTED_SERVICES FROM "
    + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);

  IConfiguredServices cfgServices = server.getHostedServices();
  Set cfgServiceIDs = cfgServices.getConfiguredServiceIDs();
  Integer cfgServiceID = (Integer) cfgServiceIDs.toArray()[0];
  IConfiguredService cfgService = cfgServices.get(cfgServiceID.intValue());
  IConfigProperties configProps = (IConfigProperties)
cfgService.getConfigProps();
  String[] propertyNames = configProps.getPropNames();
  String resultString = "";
  int size = propertyNames.length;
  for(int i = 0; i < size; i++)
  {
    IConfigProperty property = configProps.getProp(propertyNames[i]);
    String name = property.getDisplayName(Locale.ENGLISH);
    String value = property.getValue().toString();
    resultString += name + ": " + value + "<br>";
  }
  return resultString;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredServices`
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredService`
- `com.businessobjects.sdk.plugin.desktop.common.IConfigProperties`
- `com.businessobjects.sdk.plugin.desktop.common.IConfigProperty`
- `java.util.Locale`
- `java.util.Set`

5.7.5.1.4 To reset configuration properties

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Retrieve the server object from the repository, and cast it to `IServer`.

   The query must include the `SI_HOSTED_SERVICES` property.

```
String serverQuery = "SELECT SI_ID, SI_HOSTED_SERVICES FROM "
  + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. Call the `getHostedServices` method of the server object.

   This returns an `IConfiguredServices` collection, which contains configuration information for the services that are hosted by the server.

```
IConfiguredServices cfgServices = server.getHostedServices();
```

4. Call the `get` method of the `IConfiguredServices` collection to retrieve a single configured service.

```
Set cfgServiceIDs = cfgServices.getConfiguredServiceIDs();
Integer cfgServiceID = (Integer) cfgServiceIDs.toArray()[0];
IConfiguredService cfgService = cfgServices.get(cfgServiceID.intValue());
```

   This example uses the `getConfiguredServiceIDs` method of the `IConfiguredServices` collection and retrieves the first ID from the resulting set. Set the argument to the ID of the service for which you want to retrieve the configuration information. This returns an `IConfiguredService` object, which contains the configuration of the specified service on this server.

5. Call the `resetToFactoryConfigProps` method of the `IConfiguredService` object to reset all the values in the service configuration properties back to the default values defined in the service.

```
cfgService.resetToFactoryConfigProps();
```

6. Save the server object.

```
server.save();
```

> **Note**
>
> The changes take effect after the next time the server is restarted.

Example

The following example resets the configuration properties for a configured service.

```
void resetConfigProperties(IEnterpriseSession enterpriseSession, int serverID)
throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID, SI_HOSTED_SERVICES FROM "
    + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
  IConfiguredServices cfgServices = server.getHostedServices();
  Set cfgServiceIDs = cfgServices.getConfiguredServiceIDs();
  Integer cfgServiceID = (Integer) cfgServiceIDs.toArray()[0];
  IConfiguredService cfgService = cfgServices.get(cfgServiceID.intValue());
  cfgService.resetToFactoryConfigProps();
  server.save();
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredServices`
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredService`
- `java.util.Set`

5.7.5.2 Execution properties

Similar to configuration properties, execution properties control the behavior of a service or service container. The values of these properties are passed as command line arguments to the service container binary. Changes to execution properties are only applied when the server is restarted.

The execution properties of a service are stored in a `com.businessobjects.sdk.plugin.desktop.common.IExecProps` object, which can be retrieved by calling `getConfigProps` on an `IConfiguredService` object.

The execution properties of a service container are stored in a `com.businessobjects.sdk.plugin.desktop.common.IExecProps` object, which can be retrieved by calling `getExecProps` on an `IConfiguredContainer` object.

5.7.5.2.1 To retrieve execution properties

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Retrieve the server object from the repository, and cast it to `IServer`.

   The query must include the `SI_HOSTED_SERVICES` property.

   ```
   String serverQuery = "SELECT SI_ID, SI_CONFIGURED_CONTAINERS FROM "
     + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
     IServer server = (IServer) infostore.query(serverQuery).get(0);
   ```

3. Call the `getContainer` method of the server object to return the server container associated with this server.

   ```
   IConfiguredContainer cfgContainer = server.getContainer();
   ```

4. Call the `getExecProps` method of the `IConfiguredContainer` object.

This method returns an `IExecProps` object, which contains the methods for retrieving the command line information that is passed to the container when the server is started.

```
IExecProps execProps = cfgContainer.getExecProps();
String resultString = "";
resultString += "File paths: " + execProps.getFilePath() + "<br>";
resultString += "Working directory: " + execProps.getWorkDir()+ "<br>";
resultString += "Execution flags: " + execProps.getExecFlags() + "<br>";
resultString += "Command line arguments: " + execProps.getArgs() + "<br>";
```

Example

The following example retrieves the execution properties for a configured service.

```
String retrieveExecutionProperties(IEnterpriseSession enterpriseSession, int
serverID) throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID, SI_CONFIGURED_CONTAINERS FROM "
    + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
  IConfiguredContainer cfgContainer = server.getContainer();
  IExecProps execProps = cfgContainer.getExecProps();
  String resultString = "";
  resultString += "File paths: " + execProps.getFilePath() + "<br>";
  resultString += "Working directory: " + execProps.getWorkDir()+ "<br>";
  resultString += "Execution flags: " + execProps.getExecFlags() + "<br>";
  resultString += "Command line arguments: " + execProps.getArgs() + "<br>";
  return resultString;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredContainer`
- `com.businessobjects.sdk.plugin.desktop.common.IExecProps`

5.7.5.3 Local files

Local files are files located on the enterprise node that hosts the service or service container. These files contain additional information required by the service or service container.

The local files of a service or service container are stored in a `com.businessobjects.sdk.plugin.desktop.common.IStringProps` object, which can be retrieved by calling `getLocalFiles` on an `IConfiguredService` or `IConfiguredContainer` object.

5.7.5.3.1 To retrieve local files

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the server object from the repository, and cast it to `IServer`.

   The query must include the `SI_HOSTED_SERVICES` property.

```
String serverQuery = "SELECT SI_ID, SI_CONFIGURED_CONTAINERS FROM "
  + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);
```

3. Call the `getContainer` method of the server object to return the server container associated with this server.

```
IConfiguredContainer cfgContainer = server.getContainer();
```

4. Call the `getLocalFiles` method of the `IConfiguredContainer` object.

   This method returns an `IStringProps` object, which can be iterated through to obtain the names of the file system files that are used by the container.

```
IStringProps localFiles = cfgContainer.getLocalFiles();
String resultString = "";
Iterator localFilesIter = localFiles.iterator();
while(localFilesIter.hasNext())
{
  String filename = (String) localFilesIter.next();
  resultString += filename;
}
```

   Use the `add` and `remove` methods of `IStringProps` to add or remove local files.

Example

The following example retrieves the local files for a configured service and prints the filenames to the standard output.

```
String retrieveLocalFiles(IEnterpriseSession enterpriseSession, int serverID)
throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_ID, SI_CONFIGURED_CONTAINERS FROM "
    + "CI_SYSTEMOBJECTS WHERE SI_ID = " + serverID;
  IServer server = (IServer) infostore.query(serverQuery).get(0);

  IConfiguredContainer cfgContainer = server.getContainer();
  IStringProps localFiles = cfgContainer.getLocalFiles();

  String resultString = "";
  Iterator localFilesIter = localFiles.iterator();
  while(localFilesIter.hasNext())
  {
    String filename = (String) localFilesIter.next();
    resultString += filename;
  }
  return resultString;
```

```
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.businessobjects.sdk.plugin.desktop.common.IConfiguredContainer`
- `com.businessobjects.sdk.plugin.desktop.common.IStringProps`
- `java.util.Iterator`

5.7.6 Managing server groups

This section explains how to programatically perform the following server group management tasks:

- Creating server groups.
- Deleting server groups.
- Adding servers to groups.
- Removing servers from groups.

5.7.6.1 To create a new server group

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Call the `newInfoObjectCollection` method of the `IInfoStore` interface to create an `IInfoObjects` collection.

   ```
   IInfoObjects newServerGroups = infostore.newInfoObjectCollection();
   ```

3. Call the `add` method of the collection to create an InfoObject, and cast the result to the appropriate plugin interface.

   ```
   IServerGroup newServerGroup = (IServerGroup)
   newServerGroups.add(IServerGroup.KIND);
   ```

   The `add` method takes an `IServerGroup.KIND` field value representing the type of InfoObject to create. This example creates a server group and casts the result to the `IServerGroup` plugin interface.

4. Set the properties of the `IInfoObject`.

   ```
   newServerGroup.setTitle(newServerGroupName);
   newServerGroup.setDescription("New server group description");
   ```

5. Commit the `IInfoObjects` collection to the repository.

```
infostore.commit(newServerGroups);
```

Example

```
void createNewServerGroup(IEnterpriseSession enterpriseSession, String
newServerGroupName) throws SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");

  IInfoObjects newServerGroups = infostore.newInfoObjectCollection();
  IServerGroup newServerGroup = (IServerGroup)
newServerGroups.add(IServerGroup.KIND);

  newServerGroup.setTitle(newServerGroupName);
  newServerGroup.setDescription("New server group description");

  infostore.commit(newServerGroups);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.servergroup.IServerGroup`

5.7.6.2 To add a server to a server group

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfostore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Query for and retrieve the `IInfoObjects` collection that contains the specified `ServerGroup` object.

```
String serverGroupQuery = "Select SI_GROUP_MEMBERS From CI_SYSTEMOBJECTS "
  + "Where SI_ID=" + serverGroupID;
IInfoObjects serverGroups = infostore.query(serverGroupQuery);
IServerGroup serverGroup = (IServerGroup) serverGroups.get(0);
```

3. Query for and retrieve the `IInfoObjects` collection that contains the specified `Server` object.

```
String serverQuery = "Select SI_ID, SI_ML_NAME From CI_SYSTEMOBJECTS "
  + "Where SI_PROGID='CrystalEnterprise.Server' " + "AND SI_ID=" + serverID;
IInfoObjects servers = infostore.query(serverQuery);
IServer server = (IServer) servers.get(0);
```

4. Add the server to the server group.

```
serverGroup.getServers().add(server.getTitle());
```

> **Note**
>
> Groups can also be added to a server group. To add a group to a server group use the `ServerGroup.getSubGroups().add` method.

5. Commit the `IInfoObjects` collection to the repository.

```
infostore.commit(serverGroups);
```

Example

```
void addServerToServerGroup(IEnterpriseSession enterpriseSession, int serverID,
int serverGroupID) throws SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String serverGroupQuery = "Select SI_GROUP_MEMBERS From CI_SYSTEMOBJECTS "
    + "Where SI_ID=" + serverGroupID;
  IInfoObjects serverGroups = infostore.query(serverGroupQuery);
  IServerGroup serverGroup = (IServerGroup) serverGroups.get(0);
  String serverQuery = "Select SI_ID, SI_ML_NAME From CI_SYSTEMOBJECTS "
    + "Where SI_PROGID='CrystalEnterprise.Server' " + "AND SI_ID=" + serverID;
  IInfoObjects servers = infostore.query(serverQuery);
  IServer server = (IServer) servers.get(0);
  serverGroup.getServers().add(server.getTitle());
  infostore.commit(serverGroups);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.servergroup.IServerGroup`

5.8 Federation

Federation allows administrators to configure automatic replication of content between multiple BI platform installations. Content can be created and managed on one deployment (the origin site) and can be configured to replicate to other deployments (the destination sites) at scheduled intervals.

This replication can be either unidirectional or bidirectional. In a unidirectional configuration, destination sites pull changes from the origin site. Any changes to content on the origin site overwrite changes on

the destination sites. In a bidirectional configuration, synchronization occurs on both the origin and the destination: if destination content has been updated, the origin content is updated accordingly; if origin content has been updated, the destination content is updated accordingly.

Federation also supports remote scheduling, which allows administrators to configure reports on destination sites and have the actual report processing done on the origin site.

The ability to replicate content across deployments has many benefits, including simplified content management and more consistent application and enforcement of rights policy across multiple offices.

Federation can be configured by using either the Central Management Console (CMC) or the SDK. This section demonstrates how to use the SDK to configure federation.

> **Note**
>
> For more information on federation and how to configure federation using the CMC, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

5.8.1 Configuring federation

The basic steps for configuring federation are as follows:

- At the origin site, create a replication list. The replication list specifies the content that you want to replicate to destination sites.
- At each destination site, create a remote connection. The remote connection contains information about the origin site, such as the CMS host name, and the username and password used to establish a connection.
- At each destination site, create a replication job. The replication job uses the information in the remote connection to establish a connection with the origin site, retrieve the replication list, and synchronize the appropriate content.

Related Information

To create a replication list [page 237]
To create a remote connection [page 239]
To create and schedule a replication job [page 241]

5.8.2 Classes used for federation

The following packages contain classes for configuring federation:

- `com.businessobjects.sdk.plugin.desktop.manifest`, which contains classes for configuring replication lists.
- `com.businessobjects.sdk.plugin.desktop.remotecluster`, which contains classes for configuring remote connections.

- `com.businessobjects.sdk.plugin.desktop.replication`, which contains classes for configuring replication jobs.
- `com.crystaldecisions.sdk.occa.infostore`, which contains classes for accessing the repository. It also contains the `ISchedulable` interface, which is used to configure the scheduling options for replication jobs.

For detailed information on the APIs described in this section, see the *SAP BusinessObjects Business Intelligence Platform Java API Reference*.

5.8.3 To create a replication list

The replication list must be created at the origin site. It specifies which content will be replicated to destination sites.

1. Log on to the origin site and get the InfoStore service.

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
IEnterpriseSession originEnterpriseSession =
sessionMgr.logon("originUsername", "originPassword", "originCMS",
"secEnterprise");
IInfoStore infostore = (IInfoStore)
originEnterpriseSession.getService("InfoStore");
```

2. Create an `IInfoObjects` collection and add an `IManifest` object to it.

   The `com.businessobjects.sdk.plugin.desktop.manifest.IManifest` interface is used to represent replication list objects in the Central Management Server (CMS). The corresponding plugin kind is `IManifest.KIND`.

```
IInfoObjects replicationLists = infostore.newInfoObjectCollection();
IManifest replicationList = (IManifest) replicationLists.add(IManifest.KIND);
replicationList.setTitle(replicationListName);
```

3. Add the IDs of the objects that will be replicated to the replicable objects collection.

```
Collection replicableObjects = replicationList.getReplicableObjects();
replicableObjects.add(folderID);
```

   The `getReplicableObjects` method returns a collection containing the IDs of the objects on the origin that will be replicated.

   > **Note**
   >
   > There are restrictions on the types of objects that can be replicated. For example, top-level folders and publication objects cannot be replicated. For more information, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

4. Add the dependency types that will be replicated to the included dependencies collection.

```
IIncludedDependencies includedDependencies =
replicationList.getIncludedDependencies();
includedDependencies.addDependency("Profile-Principal",
CeRelationshipDirection.PARENT_CHILDREN);
includedDependencies.addDependency("DataConnection-Universe",
CeRelationshipDirection.CHILD_PARENTS);
```

The `getIncludedDependencies` method returns an `IIncludedDependencies` collection. Use the `addDependency` method to add the dependency types that you want to be replicated. The first parameter specifies the relation name. The second parameter is the direction (parent to child or child to parent) and must be set to one of the constants in the `com.businessobjects.sdk.plugin.desktop.common.CeRelationshipDirection` interface.

5. Save the replication list to the repository.

> **Note**
>
> If you plan to create replication jobs programmatically, you will need to provide the CUID of the replication list. You can retrieve the CUID of the replication list after you save it to the repository.

```
replicationList.save();
```

Example

The following code creates a replication list that contains a folder (identified by the ID in the variable `folderID`) and is configured to include all object dependencies during replication.

```
public void createReplicationList(String replicationListName, int folderID)
throws SDKException
{
  ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
  IEnterpriseSession originEnterpriseSession =
sessionMgr.logon("originUsername", "originPassword", "originCMS",
"secEnterprise");

  IInfoStore infostore = (IInfoStore)
originEnterpriseSession.getService("InfoStore");

  IInfoObjects replicationLists = infostore.newInfoObjectCollection();
  IManifest replicationList = (IManifest) replicationLists.add(IManifest.KIND);
  replicationList.setTitle(replicationListName);

  Collection replicableObjects = replicationList.getReplicableObjects();
  replicableObjects.add(folderID);

  IIncludedDependencies includedDependencies =
replicationList.getIncludedDependencies();
  includedDependencies.addDependency("Profile-Principal",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("DataConnection-
Universe",CeRelationshipDirection.CHILD_PARENTS);
  includedDependencies.addDependency("Category-Document",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("User-Inbox",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("Webi-Universe",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("User-Favorites",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("UserGroup-User",
CeRelationshipDirection.PARENT_CHILDREN);
  includedDependencies.addDependency("Universe(Core)-
Universe",CeRelationshipDirection.CHILD_PARENTS);
  includedDependencies.addDependency("EnterpriseData-
Flash",CeRelationshipDirection.CHILD_PARENTS);
  includedDependencies.addDependency("Universe-
UserGroup",CeRelationshipDirection.PARENT_CHILDREN);
```

```
  includedDependencies.addDependency("CustomRole-Object",
CeRelationshipDirection.CHILD_PARENTS);
  includedDependencies.addDependency("User-
PersonalCategory",CeRelationshipDirection.PARENT_CHILDREN);
  replicationList.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.common.CeRelationshipDirection`
- `com.businessobjects.sdk.plugin.desktop.manifest.IIncludedDependencies`
- `com.businessobjects.sdk.plugin.desktop.manifest.IManifest`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `java.util.Collection`

Related Information

Dependency types [page 245]

5.8.4 To create a remote connection

Before you create a remote connection, make sure you have the following information:

- The CMS host name and port of the origin site.
- The username and password of the BI platform user account at the origin site that will be used to establish the connection.
- The cluster (or web service) URI of the origin site.

The remote connection is used by the destination site to connect to the origin site. Before you can create replication jobs, you must create a remote connection at the destination site.

1. Log on to the destination site and get the InfoStore service.

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
IEnterpriseSession destinationEnterpriseSession =
sessionMgr.logon("destinationUsername", "destinationPassword",
"destinationCMS", "secEnterprise");
IInfoStore infostore = (IInfoStore)
destinationEnterpriseSession.getService("InfoStore");
```

2. Create an `IInfoObjects` collection and add an `IRemoteCluster` object to it.

The `com.businessobjects.sdk.plugin.desktop.remotecluster.IRemoteCluster` interface is used to represent remote connection objects in the Central Management Server (CMS). The corresponding plugin kind is `IRemoteCluster.KIND`.

```
IInfoObjects remoteClusters = infostore.newInfoObjectCollection();
IRemoteCluster remoteCluster = (IRemoteCluster)
remoteClusters.add(IRemoteCluster.KIND);
remoteCluster.setTitle(remoteClusterName);
```

3. Configure the required settings on the remote connection object.

   The username, password, and authentication type are used to log on to the origin site. The cluster URI specifies the base URI for invoking web services on the origin. The default is `http://<hostname>:<port>/dswsbobje`. Replace `<hostname>` with the host name of the origin application server (the server that is hosting the web service provider) and `<port>` with the port number.

```
remoteConnection.setCMS("originCMS");
remoteConnection.setUserName("originUsername");
remoteConnection.setPassword("originPassword");
remoteConnection.setAuthType("secEnterprise");
remoteConnection.setClusterURI("http://originCMS:8080/dswsbobje");
```

4. Save the remote connection to the repository.

   > **Note**
   >
   > If you plan to create replication jobs programmatically, you will need to provide the ID of the remote connection. You can retrieve the ID of the remote connection after you save it to the repository.

```
remoteConnection.save();
```

Example

The following code creates a remote connection.

```
public void createRemoteConnection(String connectionName) throws SDKException
{
  ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
  IEnterpriseSession destinationEnterpriseSession =
sessionMgr.logon("destinationUsername", "destinationPassword", "destinationCMS",
"secEnterprise");
  IInfoStore infostore = (IInfoStore)
destinationEnterpriseSession.getService("InfoStore");
  IInfoObjects remoteConnections = infostore.newInfoObjectCollection();

  IRemoteCluster remoteConnection = (IRemoteCluster)
remoteConnections.add(IRemoteCluster.KIND);
  remoteConnection.setTitle(connectionName);

  remoteConnection.setCMS("originCMS");
  remoteConnection.setUserName("originUsername");
  remoteConnection.setPassword("originPassword");
  remoteConnection.setAuthType("secEnterprise");
  remoteConnection.setClusterURI("http://originCMS:8080/dswsbobje");

  remoteConnection.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.remotecluster.IRemoteCluster`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

5.8.5 To create and schedule a replication job

Before creating a replication job, the following must exist:

- A remote connection must exist in the repository of the destintation site. The remote connection must contain all the information required to connect to the origin site.
- A replication list must exist in the repository of the origin site. The list must contain the IDs of the items to be replicated.

Replication jobs are responsible for scheduling the replication of objects between the origin and the destination.

1. Log on to the destination site and get the InfoStore service.

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();

IEnterpriseSession destinationSession =
sessionMgr.logon("destinationUserName", "destinationPassword",
"destinationCMS", "secEnterprise");
IInfoStore destinationInfoStore = (IInfoStore)
destinationSession.getService("InfoStore");
```

2. Create an `IInfoObjects` collection and add an `IReplication` object to it.

   The `com.businessobjects.sdk.plugin.desktop.replication.IReplication` interface is used to represent replication job objects in the Central Management Server (CMS). The corresponding plugin kind is `IReplication.KIND`.

```
IInfoObjects replicationJobs = destinationInfoStore.newInfoObjectCollection();
IReplication replicationJob = (IReplication)
replicationJobs.add(IReplication.KIND);
replicationJob.setTitle("replicationJobName");
```

3. Set the parent ID of the replication object to the ID the remote connection that you want to use to connect to the origin site.

```
replicationJob.setParentID(remoteConnectionID);
```

4. Set the remote manifest of the replication object to the CUID of the replication list on the origin site.

```
replicationJob.setRemoteManifest(replicationListCUID);
```

5. Configure the scheduling information for the replication job.

`IReplication` implements the `com.crystaldecisions.sdk.occa.infostore.ISchedulable` interface, and can be scheduled in much the same way as reports. (You do not configure destinations or formats for replication jobs.)

```
ISchedulingInfo schedulingInfo = replicationJob.getSchedulingInfo();
schedulingInfo.setRightNow(true);
```

6. Specify whether you want to perform one-way or two-way replication.

   The `setPushEnabled` method lets you specify whether two-way replication is enabled. If set to false, one-way replication is used. If set to true, two-way replication is enabled.

   If two-way replication is enabled, you must also specify the method to use for resolving conflicts. (A conflict occurs when the same item has been updated on both the origin and the destination.) Use `setConflictResolutionMode` to specify whether changes on the origin (`IReplication.CeConflictResolutionMode.MASTER_WINS`) or on the destination (`IReplication.CeConflictResolutionMode.SLAVE_WINS`) take precedence.

```
replicationJob.setPushEnabled(true);
replicationJob.setConflictResolutionMode(IReplication.CeConflictResolutionMode
.MASTER_WINS);
```

7. Save the replication job to the repository and schedule it.

```
replicationJob.save();
replicationJob.schedule();
```

Example

The following code creates a two-way replication job where changes on the origin always overwrite changes on the destination in the event of conflict.

```
public void createScheduleReplicationJob() throws SDKException, IOException
{
  ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();

  IEnterpriseSession destinationSession =
sessionMgr.logon("destinationUserName", "destinationPassword", "destinationCMS",
"secEnterprise");
  IInfoStore destinationInfoStore = (IInfoStore)
destinationSession.getService("InfoStore");

  String remoteConnectionQuery = "SELECT SI_ID FROM CI_SYSTEMOBJECTS Where
SI_KIND = '" + IRemoteCluster.KIND
    + "' AND SI_NAME = '" + "remoteConnectionName" + "' "
    + " AND SI_INSTANCE = 0";
  IInfoObjects remoteConnections =
destinationInfoStore.query(remoteConnectionQuery);
  IRemoteCluster remoteConnection = (IRemoteCluster) remoteConnections.get(0);
  int remoteConnectionID = remoteConnection.getID();
  IEnterpriseSession originEnterpriseSession =
sessionMgr.logon("originUserName", "originPassword", "originCMS",
"secEnterprise");
  IInfoStore originInfoStore = (IInfoStore)
originEnterpriseSession.getService("InfoStore");
  String replicationListQuery = "SELECT SI_ID, SI_CUID FROM CI_SYSTEMOBJECTS
WHERE SI_KIND = '" + IManifest.KIND
    + "' AND SI_NAME = '" + "replicationListName" + "' "
    + " AND SI_INSTANCE = 0";
```

```
  IInfoObjects replicationLists = originInfoStore.query(replicationListQuery);
  IManifest replicationList = (IManifest)replicationLists.get(0);
  String replicationListCUID = replicationList.getCUID();

  IInfoObjects replicationJobs = destinationInfoStore.newInfoObjectCollection();
  IReplication replicationJob = (IReplication)
replicationJobs.add(IReplication.KIND);
  replicationJob.setTitle("replicationJobName");
  replicationJob.setParentID(remoteConnectionID);
  replicationJob.setRemoteManifest(replicationListCUID);
  replicationJob.enableCleanup(true);
  replicationJob.setPushEnabled(true);

replicationJob.setConflictResolutionMode(IReplication.CeConflictResolutionMode.MA
STER_WINS);

  ISchedulingInfo schedulingInfo = replicationJob.getSchedulingInfo();
  schedulingInfo.setRightNow(true);
  replicationJob.save();
  replicationJob.schedule();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.manifest.IManifest`
- `com.businessobjects.sdk.plugin.desktop.remotecluster.IRemoteCluster`
- `com.businessobjects.sdk.plugin.desktop.replication.IReplication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `java.io.IOException`

Related Information

To create a replication list [page 237]
To create a remote connection [page 239]

5.8.6 Remote scheduling

Remote scheduling allows you to schedule a report on a destination site and process it on the origin site.

> **Note**
>
> For more information on remote scheduling, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

To enable remote scheduling for a replication job, the *Replicate remote schedules* and *Two-way replication* options must be enabled. The following code shows how to do this programmatically.

```
public void enableRemoteScheduling(IEnterpriseSession enterpriseSession, String
replicationJobName) throws SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String replicationJobQuery = "SELECT SI_ID, SI_REPLICATE_REMOTESCHEDULES FROM
CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IReplication.KIND
    + "' AND SI_NAME = '" + replicationJobName + "' "
    + " AND SI_INSTANCE = 0";

  IInfoObjects replicationJobs = infostore.query(replicationJobQuery);
  IReplication replicationJob = (IReplication) replicationJobs.get(0);
  if(!replicationJob.getReplicateRemoteSchedules())
  {
    replicationJob.setReplicateRemoteSchedules(true);
    replicationJob.setPushEnabled(true);
    replicationJob.save();
  }
}
```

For each report that you want to run remotely, you must enable the *Run at origin site* option in the *Schedule* > *Scheduling Server Group* settings of the report. The following code shows how to do this programmatically.

```
public void setRunAtOriginSite(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String reportQuery = "SELECT SI_ID, SI_SCHEDULEINFO FROM CI_INFOOBJECTS "
    + "WHERE SI_NAME = 'World Sales Report' AND SI_KIND = '"
    + IReport.KIND + "' AND SI_INSTANCE = 0";

  IInfoObjects reports = infostore.query(reportQuery);
  IReport report = (IReport) reports.get(0);
  ISchedulingInfo schedulingInfo = report.getSchedulingInfo();
  schedulingInfo.setRunOnOriginatingCluster(true);
  report.save();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.replication.IReplication`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.ISchedulingInfo`
- `com.crystaldecisions.sdk.plugin.desktop.report.IReport`

Related Information

To create and schedule a replication job [page 241]

5.8.7 Dependency types

When you create a replication list, you can specify what types of dependencies to include during replication. The `getIncludedDependencies` method of `com.businessobjects.sdk.plugin.desktop.manifest.IManifest` returns an `IIncludedDependencies` collection. Use the `addDependency` method to add the dependency types that you want to be replicated. The first parameter specifies the relation name. The second parameter is the direction (parent to child or child to parent) and must be set to one of the constants in the `com.businessobjects.sdk.plugin.desktop.common.CeRelationshipDirection` interface.

The following table shows the supported relation/direction combinations and the corresponding descriptions.

Relation name	Direction	Description
`Profile-Principal`	`PARENT_CHILDREN`	Include profiles for selected users and user groups
`DataConnection-Universe`	`CHILD_PARENTS`	Include connections used by selected universes
`Category-Document`	`PARENT_CHILDREN`	Include documents for selected categories
`User-Inbox`	`PARENT_CHILDREN`	Include inboxes for selected users
`Webi-Universe`	`PARENT_CHILDREN`	Include universes for selected reports
`User-Favorites`	`PARENT_CHILDREN`	Include personal folders for selected users
`UserGroup-User`	`PARENT_CHILDREN`	Include members of selected user groups
`Universe(Core)-Universe`	`CHILD_PARENTS`	Include universes required by selected universes
`EnterpriseData-Flash`	`CHILD_PARENTS`	Include supported dependencies for selected Flash objects
`Universe-UserGroup`	`PARENT_CHILDREN`	Include user groups for selected universes
`CustomRole-Object`	`CHILD_PARENTS`	Include access levels set on selected objects
`User-PersonalCategory`	`PARENT_CHILDREN`	Include personal categories for selected users

Related Information

To create a replication list [page 237]

5.9 Monitoring

The monitoring application allows users to keep track of the performance of the BI platform servers and helps them ensure the availability of key services. Probes and watches are the two main features in the monitoring application, which helps you check the performance and availability of servers and services within the BI platform.

Probes

Probes are tools that simulate end user workflows. Users can execute a probe to verify that a server or service is functioning normally. When the probe is executed, it returns the result of the probe and the time taken to execute the probe. Probes can be scheduled to run at specified intervals. The result and roundtrip time of executing each probe is stored in a database. You can create a trending graph from the information in this database to assist you with capacity planning.

The monitoring application contains several default probes. The BI platform allows you to create customized probes and add them to the monitoring application. You create a new probe by using the provided SDKs, through the command-line interface, or through DFOs.

5.9.1 Adding a new probe

You can create new probes and add them to the monitoring application by using the provided SDKs. In the monitoring application, each probe is represented by an InfoObject. To create a new probe and integrate the new probe within the monitoring application, write an implementation class and then create a new InfoObject of type `Probe`. You can create an InfoObject either through command-line interface, through DFOs, or through the SDK.

The following sections describe how to add a new probe:

Writing an implementation class

Perform the following steps to write an implementation class:

1. Create a java class using the provided probe SDK and save it as a `.java` file. For example, create the `NewProbe.java` file.
2. Compile the `NewProbe.java` file.
3. Create a JAR file by using the following command:

   ```
   jar -cvf NewProbe.jar
                   NewProbe.class
   ```

4. Create a directory named `probes` in the directory `C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\java\lib`.

5. Deploy the JAR file and the dependent JAR files to the directory: `C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\java\lib\probes`.

Creating an InfoObject

An InfoObject can be created through a command-line interface (CLI), DFOs, or the SDK. The following steps describe how you can create an InfoObject.

To create an InfoObject through SDK:

1. Get the probe InfoObject plugin.

```
IPluginInfo plugInfo =

infoStore.getPluginMgr().getPluginInfo("CrystalEnterprise.MON.Probe")
```

> **Note**
>
> `CrystalEnterprise.MON.Probe` is the name of the plugin.

2. Create a new InfoObject.

```
IInfoObject newProbe;
             newProbe = infoObjects.add(plugInfo);
             IProbeInfoObject probe = (IProbeInfoObject) newProbe;
```

3. Add the following properties to the new InfoObject.

Property	Command
Name of the probe	`String name = "Test Probe";` `probe.setTitle(name);`
Class name of the probe	`String className = "com.test.probe" ;` `probe.setProbeClassName(className);`
Timeout in seconds	`long timeout = 10;` `probe.setTimeout(timeout);`

Input parameters (optional)	`Map < String, Object > iMap = new HashMap < String, String > ();`
	`iMap.put("Test1", "Value1");`
	`iMap.put("Test2", true);`

Property	Command
	`iMap.put("Test3", new Integer(11));` `probe.setInputParameters (iMap);`

> **Note**
> The class name you provide for the InfoObject should be the same as the name provided for the implementation class.

4. Save the new probe.

```
probe.save();
```

To create an InfoObject through CLI:

1. In the command prompt, enter the following command: `cd C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\win64_x64\scripts`.
2. Execute the `probeAdd` command along with the required attributes and parameters as provided in the table below:

Attributes/Parameters	Description
`-auth`	Authentication type
`-classname`	Fully qualified class name of the probe
`-cms`	CMS name
`-help`	Print help for this application
`-inputparam`	Input parameters for the probe. For example, key1:type1:value1; key2:type2:value2 supports Integer/Long/Boolean/String
`-name`	Name of the probe
`-password`	Password to run the probe
`-timeout`	Timeout interval in seconds
`-username`	UserName to run the probe
`-type`	Defines the type of the probe. The values 1, 2, and 3 may appear, which correspond to: • 1: Health probe • 2: Diagnostic probe • 3: Probe that function as both health and diagnostic probes
`-probeInputType`	This can be commandLine or script upload. if (commandLine) Key:CommandLine

Attributes/Parameters	Description
	type:String value: <the command> if (scriptUpload) `-scriptLocation` and then specify the script location
`EnableVirtualMetrics`	key:EnableVirtualMetrics type:Boolean value:true/false
`Delimiter`	key:Delimiter type:String value:,
`FirstRowhasColumnNames`	key:FirstRowhasColumnNames type:Boolean value:true/false
`metriccolumns`	key:Metric_Columns type:Integer value:any integer
`anchorcolumn`	key:AnchorColumn `type:Integer value:any integer`

The following command shows you how to create a Java probe :

```
probeAdd -username Administrator -name "Test" -password ""
             -timeout 100 -classname "com.test.java" -cms "localhost" -auth
secEnterprise -
             inputparam
"documented:Integer:1000;refresh:Boolean:false;TestParam:String:Test
             String".
```

The following command shows you how to create a script based probe:

```
probeAdd.bat -auth secEnterprise  -cms localhost -password Password1 -timeout
100 -username Administrator -type 3 -name testprobe -scriptbasedinput
scriptupload=C:\monitoring_junit\testScript.bat;enablevirtualmetrics=true;delimit
er=,;firstrowhascolumnnames=true;anchorcolumn=1;metriccolumns=5
```

To create an InfoObject through CLI:

1. In the command prompt, enter the following command: `cd C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\win64_x64\scripts`.

2. Execute the `probeAdd` command along with the required attributes and parameters as provided in the table below:

Attributes/Parameters	Description
`-auth`	Authentication type
`-classname`	Fully qualified class name of the probe
`-cms`	CMS name
`-help`	Print help for this application
`-inputparam`	Input parameters for the probe. For example, key1:type1:value1; key2:type2:value2 supports Integer/Long/Boolean/String
`-name`	Name of the probe
`-password`	Password to run the probe
`-timeout`	Timeout interval in seconds
`-username`	UserName to run the probe
`-type`	Defines the type of the probe. The values 1, 2, and 3 may appear, which correspond to: • 1: Health probe • 2: Diagnostic probe • 3: Probe that function as both health and diagnostic probes

`-probeInputType`	This can be commandLine or script upload.
	if (commandLine)
	Key:CommandLine
	type:String
	value: <the command>
	if (scriptUpload)
	`-scriptLocation` and then specify the script location

`EnableVirtualMetrics`	key:EnableVirtualMetrics type:Boolean value:true/false
`Delimiter`	key:Delimiter type:String value:,
`FirstRowhasColumnNames`	key:FirstRowhasColumnNames

Attributes/Parameters	Description
	type:Boolean value:true/false
`metriccolumns`	key:Metric_Columns type:Integer value:any integer
`anchorcolumn`	key:AnchorColumn `type:Integer value:any integer`

To create an InfoObject through DFO:

1. Create a DFO file for the new probe InfoObject and place it in the following directory: `C:\Program Files\Business Objects\BusinessObjects Enterprise 12.0\Packages`.
2. Restart the CMS.

5.9.1.1 Classes used for adding a new probe

The following packages are exposed by the the BI platform for the addition of new probes or modifying the properties of existing probes in the monitoring application:

- `com_businessobjects_sdk_monitoring_plugin_desktop_probe`, which contains the following classes that are used for creating and modifying the probe InfoObject:

  `IProbeInfoObject.java`
  `IProbeInfoObjectBase.java`
- `com_businessobjects_sdk_monitoring_probe`, which contains the following classes that are used for implementing the probe:

  `AbstractProbe.java`
  `DiagnosticProbeResult.java`
  `HealthProbeResult.java`
  `IDiagnosticProbe.java`
  `IDiagnosticProbeResult.java`
  `IHealthProbe.java`
  `IHealthProbeResult.java`

5.10 Platform Search

Platform Search is a scalable, high-performance optimized search service that enables users to search content within the BI platform repository. It refines the search results by grouping them into categories and ranking them in order of their relevance. Platform Search supports both continuous and schedule-based indexing.

Platform Search suggests a query to create a document in case the document is not already available. It enables you to search multiple content types.

5.10.1 Classes used for Platform Search

The following packages contain interfaces and classes for retrieving search results and creating a Web Intelligence document using the search results:

- `com.sap.businessobjects.platform.search` contains the `IPlatformSearchService` interface used for submitting search requests and getting search results.
- `com.sap.businessobjects.platform.search.common` contains the following classes used for processing search results and creating a Web Intelligence document:

  `SearchIndexRequest`
  `SearchIndexResponse`
  `CreateDocumentRequest`
  `CreateDocumentResponse`

For more information about these APIs, see the *SAP BusinessObjects Business Intelligence Platform Java API Reference*.

5.10.2 To retrieve search results

The Platform Search SDK enables you to retrieve the search results and use them to create a new Web Intelligence document using the queries generated from Universe hits. It also allows you to edit the query and generate a report with the search response.

Perform the following steps to retrieve the search results and optionally create a Web Intelligence document from them:

1. Get the Search Service.

```
IPlatformSearchService searchClient =
(IPlatformSearchService)
session.getService("PlatformSearchServiceOCA");
```

2. Create an instance of the search request object.

```
SearchIndexRequest request = new SearchIndexRequest();
```

3. Set the properties of the request object using the following methods:

Method	Usage
setQuery()	`request.setQuery("Sales");`
setMaxInstances()	`request.setMaxInstances(4);`
setCategoryTypes()	`request.setCategoryTypes(catTypes);`
setGroupFacetsByType()	`request.setGroupFacetsByType(true);`
setCategories()	`request.setCategories(new String[] {"FolderPath/Reports", "Country/US", "Country/India"});`
setPageSize()	`request.setPageSize(20);`
setPage()	`request.setPage(2);`
setHighLightingInfoEnabled()	`request.setHighLightingInfoEnabled(true)`
setSuggestedQueriesEnabled()	`request.setSuggestedQueriesEnabled(true)`
setCategoryInfoEnabled()	`request.setCategoryInfoEnabled(true)`
setHitLinksEnabled()	`request.setHitLinksEnabled(true)`
setHitPromptsEnabled()	`request.setHitPromptsEnabled(true)`
setMaxHitLinks()	`request.setMaxHitLinks(1)`
setMaxDocuments()	`request.setMaxDocuments(1000)`
setSearchAgents()	`request.setSearchAgents(new String[] {"AXk3RFXWBDRDtH5pfOJ6ilE","ATe1Gx2MGE5As8s0y1Z9ssE"})`
setMaxCategoriesShown()	`request.setMaxCategoriesShown(5);`

Method	Usage
`setMaxSubCategoriesShown()`	`request.setMaxSubCategoriesShown(5);`

4. Invoke the `search()`, `fullSearch()` or `quickSearch()` methods in `IPlatformSearchService` interface to retrieve search results from the Search Service.
   - ```
     SearchIndexResponse response = searchClient.search(request)
     ```
   - ```
     SearchIndexResponse fullSearchResponse = searchClient.fullSearch(queryString, clientLocale);
     ```
   - ```
     SearchIndexResponse quickSearchResponse = searchClient.quickSearch(queryString, clientLocale, maxInstances, page, pageSize, categories);
     ```
5. Get the search results from the search response using the following:
   - ```
     IInfoObjects Hitobjs = response.getHitObjects();
     ```
   - ```
     IInfoObjects universeObjects = response.getUniverseObjects();
     ```
   - ```
     PropertyBag categories = response.getCategoryBag()
     ```
   - ```
     PropertyBag suggestions = response.getSuggestionBag()
     ```
   - ```
     int instanceCount = response.getTotalInstCount()
     ```
   - ```
     int documentCount = response.getTotalDocCount()
     ```

   > **Note**
   >
   > You need to perform the following steps in case you want to create a Web Intelligence document using the response retrieved from the search request.

6. Create an instance of the document request.

   ```
   CreateDocumentRequest cdRequest = new CreateDocumentRequest();
   ```

7. Set the properties of the Create Document request instance using the following methods:

Method	Usage
`setUniverseCUID()`	`cdRequest.setUniverseCUID(universeCuid)`
`setTitle()`	`cdRequest.setTitle(title)`
`setAutoRun()`	`cdRequest.setAutoRun(true);`
   ```

Method	Usage
`setAutoDrill()`	`cdRequest.setAutoDrill(true);`
`addResultObjectName()`	`cdRequest.addResultObjectName(name)`
`addFilter()`	`cdRequest.addFilter(String objectName, String[] values)`

8. Call the `createDocument()` method using the search service.

```
CreateDocumentResponse cdResponse = searchClient. createDocument(cdRequest);
```

9. Get the ID of the folder where the temporary Web Intelligence Desktop document is created.

```
String folderID = cdResponse.getTempFolderId();
```

> **Note**
>
> You can either save the newly created Web Intelligence Desktop document in the repository or discard it.

Example: Sample code

The following example retrieves the search results and enables you to create a Web Intelligence Desktop document.

```
 IEnterpriseSession session = null;
 String queryString = "sales" ;
 try{
                session =
CrystalEnterprise.getSessionMgr().logon("administrator", "", "localhost",
"secEnterprise");
                IPlatformSearchService searchClient =
(IPlatformSearchService)session.getService("PlatformSearchServiceOCA");
                SearchIndexRequest searchRequest = new SearchIndexRequest();
                //Setting values for request
                searchRequest.setQuery(queryString);
                searchRequest.setCategories(new String[] {"FolderPath/Reports",
"Country/US", "Country/India"});
                searchRequest.setMaxInstances(4);
                searchRequest.setPageSize(20);
                searchRequest.setPage(2);
                searchRequest.setHighLightingInfoEnabled(true);
                searchRequest.setSuggestedQueriesEnabled(true);
                searchRequest.setCategoryInfoEnabled(true);
                searchRequest.setHitLinksEnabled(true);
                searchRequest.setHitPromptsEnabled(true);
                searchRequest.setCategoryTypes (new CategoryTypes[]
{SearchIndexRequest.CategoryTypes.METADATA_CATEGORY});
                searchRequest.setGroupFacetsByType(true);
    searchRequest.setMaxDocuments(10);
    searchRequest.setMaxHitLinks(10);
    searchRequest.setMaxCategoriesShown(5);
    searchRequest.setMaxSubCategoriesShown(5);
```

```
    SearchIndexResponse searchResponse = searchClient.search(searchRequest);
    SearchIndexResponse fullSearchResponse =
searchClient.fullSearch(queryString, new Locale("en" , "US"));
    SearchIndexResponse quickSearchResponse = searchClient.quickSearch
(queryString, new Locale("en" , "US"), 4,2,20, new String[] {"FolderPath/
Reports"});
//Using Hit Objects
IInfoObjects hitObjs = searchResponse.getHitObjects();
System.out.println("====Number of objects hit: "+hitObjs.size());
for(int nHitIndex =0;nHitIndex < hitObjs.size(); nHitIndex++)
{

        IInfoObject obj = (IInfoObject) hitObjs.get(nHitIndex);
        System.out.println("Content found = " + obj.getTitle() + " ID = "+
obj.getID());
     System.out.println("Folder Path for ObjID = " +
obj.properties().getProperty(PropertyIDs.nameToID("SI_FOLDER_PATH")).getValue().t
oString()+"\n");

}
//Using Category
PropertyBag categories = searchResponse.getCategoryBag();
int subBagCount = categories.getInt("SI_TOTAL");
for(int nBag = 1 ; nBag<=subBagCount ; ++nBag) {
        PropertyBag subBag =
categories.getPropertyBag(nBag).getPropertyBag(String.valueOf(CategoryTypes.METAD
ATA_CATEGORY));
        System.out.println("====No of Category: " + subBag.getInt("SI_TOTAL"));
 }
//Using Suggestion
PropertyBag suggestions = searchResponse.getSuggestionBag();
int nCategorySize = suggestions.getInt("SI_TOTAL");
System.out.println("\n====Number of Suggestions: "+nCategorySize);
for (int i=0; i<nCategorySize; i++) {
     PropertyBag topicBag =
suggestions.getItem(Integer.toString(i+1)).getPropertyBag();
     System.out.println("Did you mean: " +
topicBag.getString(CePropertyID.SI_HIT_TEXT) + "?");
}
//NOTE: This property bag will be set only when there will be no hits.
//Using Universe Object
IInfoObjects unvObjs = searchResponse.getUniverseObjects();
System.out.println("\n====Number of Universe Objects: " + unvObjs.size());
for(int i=0; i < unvObjs.size(); i++){
        IInfoObject unvObj = (IInfoObject)unvObjs.get(i);
//You need to perform the following steps if you want to create a new Web
Intelligence document from the search results.
//Using Create Document
CreateDocumentRequest cdRequest = new CreateDocumentRequest();
cdRequest.setUniverseCUID(unvObj.getCUID());
cdRequest.setTitle("Test");
cdRequest.setAutoRun(true);
//Create Document Request accepts filter name and filter values which are
returned as a part of the universe response
IProperties filterProp =
unvObj.properties().getProperties(PropertyIDs.nameToID("SI_HIT_FILTERS"));
PropertyArrayHelper filterHelper = new
PropertyArrayHelper((PropertyBag)filterProp, PropertyIDs.SI_TOTAL);
for(int j=0; j < filterHelper.size(); j++){
        IProperties filterNameProp = (IProperties)filterHelper.get(j);
        IProperty property =
(IProperty)filterNameProp.getProperty(PropertyIDs.nameToID("SI_HIT_OBJECT"));
        String filtername = (String)property.getValue();
        System.out.println("\n====Filter Name: "+filtername);
        IProperties filterValueProp =
filterNameProp.getProperties(PropertyIDs.nameToID("SI_HIT_VALUES"));
```

```
        PropertyArrayHelper valuesHelper = new
PropertyArrayHelper((PropertyBag)filterValueProp, PropertyIDs.SI_TOTAL);
        String[] filtervalue = new String[valuesHelper.size()];
        for(int k=0; k < valuesHelper.size(); k++){
                filtervalue[k] = (String)valuesHelper.get(k);
                System.out.println("====Filter Value: "+filtervalue[k]);
}
cdRequest.addFilter(filtername, filtervalue);
}
//Create Document Response accepts name of the hit-object which is returned as a
part of the universe response
IProperties nameProp =
unvObj.properties().getProperties(PropertyIDs.nameToID("SI_HIT_OBJECTS"));
PropertyArrayHelper nameHelper = new PropertyArrayHelper((PropertyBag)nameProp,
PropertyIDs.SI_TOTAL);
for(int l=0; l < nameHelper.size(); l++){
        cdRequest.addResultObjectName((String)nameHelper.get(l));
}
CreateDocumentResponse createResponse = searchClient.createDocument(cdRequest);
}
} catch (Exception e) {
         e.printStackTrace();
}
```

5.11 Auditing

Servers in a BI platform installation can be configured to log information about events that they trigger. For example, when a user logs on, the Central Management Server (CMS) triggers a logon event. Administrators can configure which server events are tracked as well as the behavior of the auditing systems. It is possible for you to programmatically manipulate the auditing configuration of the system.

Auditing events are centrally configured by the `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo` interface. This interface contains methods for retrieving all auditing events, which can be organized by either logical categories, or by cumulative, pre-configured levels.

Auditing levels are collections of auditing events that can be enabled or disabled as a group. All auditing levels are cumulative and contain all auditing events from the preceding levels. You can manage auditing events using either custom or pre-configured auditing levels:

- Four pre-configured auditing levels can be used to set which events are captured.
- The *Custom* auditing level lets you enable or disable collection of specific events.

Also, the details that are collected when events are captured can be tailored to the needs of each system.

Once an auditing event is triggered, the system captures details about the event, such as the name of the user responsible for triggering it. The details that are captured are configured independently from the auditing levels.

> **Note**
>
> For more information about auditing, refer to the *SAP Business Objects Business Intelligence Platform Administrator Guide*.

Classes used for auditing

- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
  Is the entrypoint interface auditing, and contains methods to retrieve other auditing object types.
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvents` and `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvent`
  Contain methods to retrieve and configure the auditing events.
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetails` and `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetail`
  Contain methods to retrieve and configure the auditing details that are captured.
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditLevels` and `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditLevel`
  Contain methods to retrieve and deploy pre-configured auditing levels.
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditCategories` and `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditCategory`
  Contain methods to retrieve and inspect pre-configured auditing categories.
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditConnectionInfo`
  Contains methods to retrieve and configure the credentials used to connect to the auditing database. It also defines which driver to use, based on the SQL database type.

5.11.1 To retrieve an auditing event

Auditing events are stored and managed centrally by an `IAuditEventInfo` object. This example demonstrates retrieval of a specific auditing event, along with a description and auditing level that the event belongs to.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the auditing events collection from the repository, and cast it to the `IAuditEventInfo` interface.

   Use the corresponding `SI_KIND` constant from the `IAuditEventInfo` interface to retrieve the collection. The query must also include a reference to the `SI_AUDIT_EVENTS` property, because it contains the complete list of auditing events in the system.

```
String auditQuery = "SELECT SI_AUDIT_EVENTS FROM CI_SYSTEMOBJECTS WHERE " +
  "SI_KIND ='" + IAuditEventInfo.KIND + "'";
IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);
```

3. Call the `getAuditEvents` method of the `IAuditEventInfo` interface.

```
IAuditEvents auditEvents = auditEventInfo.getAuditEvents();
```

4. Call the `getEvent` method of the `IAuditEvents` object with an ID of a particular event type.

   The *View* auditing event has an event type ID of 1002. Other event type IDs are found in the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

```
IAuditEvent viewEvent = auditEvents.getEvent(1002);
```

5. Retrieve a description of the event, as well as its auditing level.

```
String viewEventProperties = "Event Description: " +
viewEvent.getDescription() +
  "<br>Audit Level: " + viewEvent.getAuditLevel();
```

> **Note**
>
> The integer values for the auditing levels correspond to constant field values in the `IAuditEventInfo.AuditEventLevel` subinterface.

Example

The following example returns a `String` object containing a description and the auditing level for the *View* auditing event.

```
String getViewEvent(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENTS FROM CI_SYSTEMOBJECTS WHERE " +
    "SI_KIND ='" + IAuditEventInfo.KIND + "'";
  IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);

  IAuditEvents auditEvents = auditEventInfo.getAuditEvents();
  IAuditEvent viewEvent = auditEvents.getEvent(1002);

  String viewEventProperties = "Event Description: " +
viewEvent.getDescription() +
    "<br>Audit Level: " + viewEvent.getAuditLevel();
  return viewEventProperties;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvents`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvent`

Related Information

Auditing [page 257]

5.11.2 To retrieve an auditing event detail

Auditing event details are stored and managed centrally by an `IAuditEventInfo` object. Auditing event details determine what data is collected when an auditing event is triggered, and some details are associated with multiple event types. This example shows how to retrieve a specific auditing event detail object.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the auditing events collection from the repository, and cast it to the `IAuditEventInfo` interface.

   Use the corresponding `SI_KIND` constant from the `IAuditEventInfo` interface to retrieve the collection. The query must also include a reference to the `SI_AUDIT_EVENT_DETAILS` property because it contains the complete list of auditing events in the system.

```
String auditQuery = "SELECT SI_AUDIT_EVENT_DETAILS FROM CI_SYSTEMOBJECTS
WHERE " +
        "SI_KIND = '" + IAuditEventInfo.KIND + "'";
IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);
```

3. Call the `getAuditDetails` method of the `IAuditEventInfo` interface.

```
IAuditDetails auditDetails = auditEventInfo.getAuditDetails();
```

4. Call the `getDetail` method of the `IAuditDetails` object.

   The *Query* auditing detail has an event detail ID of 5. Other detail IDs are found in the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

```
IAuditDetail queryEventDetail = auditDetails.getDetail(5);
```

5. Retrieve a description of the detail and also a flag indicating whether collection of the detail is enabled.

```
String queryEventDetailDescription = "Event Detail Description: " +
queryEventDetail.getDescription() +
  "<br>Collection of this event detail is enabled?: " +
queryEventDetail.isEnabled();
```

Example

The following example returns a `String` object containing a description for the *Query* auditing event detail.

```
String getQueryEventDetail(IEnterpriseSession enterpriseSession) throws
SDKException
{
  InfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENT_DETAILS FROM CI_SYSTEMOBJECTS WHERE
" +
    "SI_KIND ='" + IAuditEventInfo.KIND + "'";
  IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);

  IAuditDetails auditDetails = auditEventInfo.getAuditDetails();
  IAuditDetail queryEventDetail = auditDetails.getDetail(5);
```

```
  String queryEventDetailDescription = "Event Detail Description: " +
queryEventDetail.getDescription() +
    "<br>Collection of this event detail is enabled?: " +
queryEventDetail.isEnabled();
  return queryEventDetailDescription;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetails`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetail`

Related Information

Auditing [page 257]
To enable an auditing event detail [page 265]

5.11.3 To change the current auditing level

Auditing levels are defined by field values in the `IAuditEventInfo.AuditEventLevel` subinterface as follows:

Field	int value
`COMPLETE`	4
`CUSTOM`	0
`DEFAULT`	3
`MINIMAL`	2
`OFF`	1

Each auditing event belongs to an auditing level. Auditing levels are cumulative and contain all auditing events from the preceding levels.

The `AuditEventLevel.CUSTOM` auditing level allows you to choose which individual auditing events are captured. This example changes the current auditing level defined by the CMS to this custom option.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the auditing events collection from the repository, and allocate it to an `IInfoObjects` collection.

   Use the corresponding `SI_KIND` constant from the `IAuditEventInfo` interface to retrieve the collection. The query must also include references to the `SI_AUDIT_EVENTS` and `SI_AUDIT_LEVELS` properties, because they contain details used by the `IAuditLevel` interface.

   ```
   String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_LEVELS FROM
   CI_SYSTEMOBJECTS " +
     "WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
   IInfoObjects infoObjects = infoStore.query(auditQuery);
   ```

3. Retrieve the `IAuditEventInfo` object from the `IInfoObjects` collection.

   ```
   IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
   ```

4. Call the `setCurrentAuditLevel` method of the `IAuditEventInfo` interface, and pass in a value from `IAuditEventInfo.AuditEventLevel`.

   ```
   auditEventInfo.setCurrentAuditLevel(AuditEventLevel.CUSTOM);
   ```

5. Commit the changes back to the repository.

   ```
   infoStore.commit(infoObjects);
   ```

Example

The following example sets the current auditing level to `CUSTOM`, allowing customized auditing events to be set.

```
void changeAuditLevel(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_LEVELS FROM
CI_SYSTEMOBJECTS " +
    "WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  auditEventInfo.setCurrentAuditLevel(AuditEventLevel.CUSTOM);
  infoStore.commit(infoObjects);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo.AuditEventLevel`

Related Information

To enable an auditing event [page 263]
To restore custom auditing events [page 266]
Auditing [page 257]

5.11.4 To enable an auditing event

This task assumes that you have already set your current auditing level to custom (`AuditEventLevel.CUSTOM`).

Beyond the events which are configurable with the pre-set auditing levels, other events can be individually enabled or disabled. Enabled events are stored as a set of IDs which reference auditing events which are being logged.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the auditing events collection from the repository, and allocate it to an `IInfoObjects` collection.

   Use the corresponding `SI_KIND` constant from the `IAuditEventInfo` interface to retrieve the collection. The query must also include references to `SI_AUDIT_EVENTS` and `SI_ENABLED_AUDIT_EVENTS`, because they contain the information necessary to make changes to the set of enabled events.

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_ENABLED_AUDIT_EVENTS FROM " +
  "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. Retrieve the `IAuditEventInfo` object from the `IInfoObjects` collection.

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. Call the `getAuditEvents` method of the `IAuditEventInfo` interface, and store the resulting `IAuditEvents` object.

```
IAuditEvents auditEvents = auditEventInfo.getAuditEvents();
```

5. Call the `getEvent` method of the `IAuditEvents` interface, and store the resulting `IAuditEvent` object.

   The *Retrieve* auditing event has an event ID of 1013. Other event type IDs are found in the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

```
IAuditEvent retrieveEvent = auditEvents.getEvent(1013);
```

6. Retrieve the `java.util.Set` collection of currently enabled auditing events from the `IAuditEventInfo` object.

```
Set enabledEvents = auditEventInfo.getEnabledEventTypeIDs();
```

7. Add the event to the `java.util.Set` collection of enabled auditing events.

```
enabledEvents.add(retrieveEvent.getID());
```

8. Commit the changes back to the repository.

```
infoStore.commit(infoObjects);
```

Example

The following example adds the *Retrieve* event to the set of enabled auditing events on the system.

```
void enableRetrieveEvent(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_ENABLED_AUDIT_EVENTS FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);

  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  IAuditEvents auditEvents = auditEventInfo.getAuditEvents();
  IAuditEvent retrieveEvent = auditEvents.getEvent(1013);
  Set enabledEvents = auditEventInfo.getEnabledEventTypeIDs();
  enabledEvents.add(retrieveEvent.getID());

  infoStore.commit(infoObjects);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvents`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEvent`
- `java.util.Set`

Related Information

To change the current auditing level [page 261]
To retrieve an auditing event [page 258]
To restore custom auditing events [page 266]
Auditing [page 257]

5.11.5 To enable an auditing event detail

Each auditing event is associated with auditing event details. The details determine which information is captured when the event is triggered. Some details are mandatory while others are optional and may be enabled or disabled. Details are configured independently from auditing levels.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the auditing events collection from the repository, and allocate it to an `IInfoObjects` collection.

   Use the corresponding `SI_KIND` constant from the `IAuditEventInfo` interface to retrieve the collection. The query must also include references to the parameters `SI_AUDIT_EVENT_DETAILS` and `SI_DISABLED_AUDIT_EVENT_DETAILS` because they contain the information necessary to make changes to the set of enabled details.

```
String auditQuery = "SELECT SI_AUDIT_EVENT_DETAILS,
SI_DISABLED_AUDIT_EVENT_DETAILS FROM " +
  "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. Retrieve the `IAuditEventInfo` object from the `IInfoObjects` collection.

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. Call the `getAuditDetails` method of the `IAuditEventInfo` interface, and store the resulting `IAuditDetails` object.

```
IAuditDetails auditDetails = auditEventInfo.getAuditDetails();
```

5. Call the `getDetail` method of the `IAuditDetails` interface with an ID of a particular event detail.

   The *Query* auditing event detail has a detail ID of 25. Other event detail IDs are found in the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

```
IAuditDetail queryDetail = auditDetails.getDetail(25);
```

6. Retrieve the `java.util.Set` collection of currently enabled auditing event details from the `IAuditEventInfo` object.

```
Set enabledDetails = auditEventInfo.getEnabledDetailTypeIDs();
```

7. Add the detail to the `java.util.Set` collection of enabled auditing event details.

```
enabledDetails.add(queryDetail.getID());
```

8. Commit the changes back to the repository.

```
infoStore.commit(infoObjects);
```

Example

The following example adds the *Query* event detail to the set of enabled auditing events on the system.

```
void enableQueryDetail(IEnterpriseSession enterpriseSession) throws SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String auditQuery = "SELECT SI_AUDIT_EVENT_DETAILS,
SI_DISABLED_AUDIT_EVENT_DETAILS " +
    "FROM CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  IAuditDetails auditDetails = auditEventInfo.getAuditDetails();
  IAuditDetail queryDetail = auditDetails.getDetail(25);
  Set enabledDetails = auditEventInfo.getEnabledDetailTypeIDs();
  enabledDetails.add(queryDetail.getID());
  infoStore.commit(infoObjects);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetails`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditDetail`
- `java.util.Set`

Related Information

To retrieve an auditing event detail [page 260]
Auditing [page 257]

5.11.6 To restore custom auditing events

If the `CUSTOM` auditing level is used and configured for your environment, the custom settings are saved, making them retrievable if the auditing level is inadvertently modified. This example shows how to change the auditing level to `CUSTOM` with the same auditing events and settings used previously.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Retrieve the auditing events collection from the repository, and allocate it to an `IInfoObjects` collection.

   Use the corresponding `SI_KIND` constant from the `IAuditEventInfo` interface to retrieve the collection. The query must also include references to the `SI_AUDIT_EVENTS` and `SI_AUDIT_LEVELS` properties,

because they contain the information necessary to revert to the saved custom events and change the auditing level.

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_LEVELS FROM " +
        "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. Retrieve the `IAuditEventInfo` object from the `IInfoObjects` collection.

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. Call the `setCurrentAuditLevel` method of the `IAuditEventInfo` interface, and pass in the `CUSTOM` value from `IAuditEventInfo.AuditEventLevel`.

```
auditEventInfo.setCurrentAuditLevel(AuditEventLevel.CUSTOM);
```

5. Commit the changes back to the repository.

```
infoStore.commit(infoObjects);
```

6. Use `IInfoStore` to query for an updated set of `IInfoObjects`.

   Use the corresponding `SI_KIND` constant from the `IAuditEventInfo` interface to retrieve the collection. The query must also include references to the `SI_AUDIT_EVENTS` and `SI_PREVIOUS_ENABLED_AUDIT_EVENTS` properties, because they contain the information necessary to restore the custom events.

```
auditQuery = "SELECT SI_AUDIT_EVENTS, SI_PREVIOUS_ENABLED_AUDIT_EVENTS FROM "
+
        "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
infoObjects = infoStore.query(auditQuery);
```

7. Retrieve the updated `IAuditEventInfo` object from the `IInfoObjects` collection.

```
auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

8. Use the `java.util.List` collection to retrieve a list of previously enabled custom events from the `IAuditEventInfo` object.

```
List previousEventsList =
auditEventInfo.getPreviousEnabledCustomAuditEvents();
```

9. Use the `java.util.Set` collection to retrieve a set of currently enabled custom events from the `IAuditEventInfo` object.

```
Set enabledEventsSet = auditEventInfo.getEnabledEventTypeIDs();
```

10. Clear the enabled events from the set to remove any events added by the change in auditing level.

```
enabledEventsSet.clear();
```

11. Add the list of previously enabled custom events to the empty set of enabled custom events.

```
enabledEventsSet.addAll(previousEventsList);
```

12. Commit the changes back to the repository.

```
infoStore.commit(infoObjects);
```

Example

The following example will change the auditing level to custom, and then enable the same events that were previously configured.

```
void restoreCustomEvents(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_PREVIOUS_ENABLED_AUDIT_EVENTS
FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);

  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  auditEventInfo.setCurrentAuditLevel(AuditEventLevel.CUSTOM);
  infoStore.commit(infoObjects);
  auditQuery = "SELECT SI_AUDIT_EVENTS, SI_PREVIOUS_ENABLED_AUDIT_EVENTS FROM "
+
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  infoObjects = infoStore.query(auditQuery);

  auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  List previousEventsList = auditEventInfo.getPreviousEnabledCustomAuditEvents();

  Set enabledEventsSet = auditEventInfo.getEnabledEventTypeIDs();
  enabledEventsSet.clear();
  enabledEventsSet.addAll(previousEventsList);

  infoStore.commit(infoObjects);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo.AuditEventLevel`
- `java.util.Set`
- `java.util.List`

Related Information

To change the current auditing level [page 261]
To enable an auditing event [page 263]
Auditing [page 257]

5.11.7 To retrieve auditing categories

Auditing events can be organized by categories that are logically similar. This example shows retrieval of an auditing category and some of its properties.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the auditing events collection from the repository, and cast it to the `IAuditEventInfo` interface.

   Use the corresponding `SI_KIND` constant from the `IAuditEventInfo` interface to retrieve the collection. The query must also include a reference to the `SI_AUDIT_EVENTS` and `SI_AUDIT_CATEGORIES` properties, because they contain details used by the `IAuditCategory` interface.

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_CATEGORIES FROM " +
        "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);
```

3. Retrieve the `IAuditCategories` collection from the `IAuditEventInfo` object.

```
IAuditCategories auditCategories = auditEventInfo.getAuditCategories();
```

4. Retrieve the *Common Events* category by using the `getCategory` method of the `IAuditCategories` interface.

   The ID for the *Common Events* category is 1.

```
IAuditCategory commonEvents = auditCategories.getCategory(1);
```

5. Retrieve a description of the category.

```
String commonEventDescription = commonEvents.getDescription();
```

6. Retrieve the `java.util.Set` collection of related auditing event IDs, and cast it to a `String` object.

```
String commonEventIDs = commonEvents.getEventIDs().toString();
```

Example

The following example returns an array of `String` objects containing the localized description of the*Common Events* auditing category, as well as a set of IDs of associated auditing events.

```
String[] commonEventCategoryInfo(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_CATEGORIES FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IAuditEventInfo auditEventInfo = (IAuditEventInfo)
infoStore.query(auditQuery).get(0);
  IAuditCategories auditCategories = auditEventInfo.getAuditCategories();
  IAuditCategory commonEvents = auditCategories.getCategory(1);

  String[] commonEventProperties = new String[2];
```

```
    commonEventProperties[0] = commonEvents.getDescription();
    commonEventProperties[1] = commonEvents.getEventIDs().toString();

    return commonEventProperties;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditCategories`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditCategory`

Related Information

To retrieve an auditing event [page 258]
Auditing [page 257]

5.11.8 To retrieve a server auditing metric

You can retrieve a number of server metrics from the Central Management Server (CMS), including metrics relating to auditing. This example retrieves the number of auditing events remaining to be collected, as polled from the CMS.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Retrieve a server object from the repository, and cast it to the `IServer` object type.

   To guarantee a result, match the description in `SI_DESCRIPTION` to the one used for the CMS. Also, the query must include a reference to the`SI_METRICS` property, because it contains the information necessary to access metrics data.

   ```
   String serverQuery = "SELECT SI_METRICS FROM CI_SYSTEMOBJECTS " +
           "WHERE SI_DESCRIPTION = 'Central Management Server'";
   IServer server = (IServer) infoStore.query(serverQuery).get(0);
   ```

3. Retrieve the `IServerMetrics` collection from the `IServer` object.

   ```
   IServerMetrics serverMetrics = server.getMetrics();
   ```

4. Retrieve the `AuditAdmin` `IMetrics` object from the `IServerMetrics` collection.

   Other auditing metrics are available from the `APSAdmin` `IMetrics` object.

   ```
   IMetrics auditMetrics = serverMetrics.getMetrics("AuditAdmin");
   ```

5. Retrieve the *auditeeNumberOfEventsRemaining* `IMetric` object from the `IMetrics` collection.

   On the CMS, the *auditeeNumberOfEventsRemaining* `IMetric` object holds the first position in the `AuditAdmin` `IMetrics` collection.

```
IMetric numEvents = (IMetric) auditMetrics.get(0);
```

6. Retrieve the name and value from the `IMetric` object.

```
numEvents.getName();
numEvents.getValue();
```

Example

The following example returns the values for the number of auditing events remaining to be collected from the set of auditing metrics, as polled from the CMS.

```
String getServerAuditMetric(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String serverQuery = "SELECT SI_METRICS FROM CI_SYSTEMOBJECTS " +
    "WHERE SI_DESCRIPTION = 'Central Management Server'";
  IServer server = (IServer) infoStore.query(serverQuery).get(0);

  IServerMetrics serverMetrics = server.getMetrics();
  IMetrics auditMetrics = serverMetrics.getMetrics("AuditAdmin");
  IMetric numEvents = (IMetric) auditMetrics.get(0);
  return numEvents.getName() + " " + numEvents.getValue().toString();
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServer`
- `com.crystaldecisions.sdk.plugin.desktop.server.IServerMetrics`
- `com.businessobjects.sdk.plugin.desktop.common.IMetric`
- `com.businessobjects.sdk.plugin.desktop.common.IMetrics`

Related Information

Auditing [page 257]

5.11.9 To retrieve auditing data store connection information

Many attributes of the auditing database connection are programmatically changeable. The following example shows how to retrieve some properties of the auditing database connection. You can set different values

using the corresponding setter methods and committing the changed `IInfoObjects` collection to the Central Management Server (CMS). See the *SAP BusinessObjects Java API Reference* for details on these methods.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the auditing events collection from the repository, and allocate it to an `IInfoObjects` collection.

   Use the corresponding `SI_KIND` constant from the `IAuditEventInfo` interface to retrieve the collection. The query must also include a reference to `SI_AUDIT_EVENTS` and `SI_AUDIT_DB_CONNECTION` because these properties contain the information necessary to retrieve and make changes to the database connection properties.

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUBIT_DB_CONNECTION FROM " +
  "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. Retrieve the `IAuditEventInfo` object from the `IInfoObjects` collection.

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. Retrieve the `IAuditConnectionInfo` object from the `IAuditEventInfo` collection.

```
IAuditConnectionInfo connectionInfo = auditEventInfo.getAuditConnectionInfo();
```

5. Get the user name by calling the `getUserName` method of the `IAuditConnectionInfo` interface.

```
connectionInfo.getUserName()
```

> **Note**
>
> There is a `setPassword` method to programmatically set the password used to logon to the auditing database. The password is not retrievable through the SDK.

6. Get the data source name by calling the `getDsnName` method of the `IAuditConnectionInfo` interface.

```
connectionInfo.getDsnName()
```

7. Get the database driver type for the auditing database by calling the `getDatabaseDriver` method of the `IAuditConnectionInfo` interface.

```
connectionInfo.getDatabaseDriver()
```

> **Note**
>
> Six pre-configured driver identifiers can be used with the equivalent setter method. Select the driver that is suitable for the type of database that will store the auditing events.
>
> - `IAuditConnectionInfo.CeAuditDatabaseDriver.DB2_AUDIT_DB_DRIVER`
> - `IAuditConnectionInfo.CeAuditDatabaseDriver.MAXDB_AUDIT_DB_DRIVER`
> - `IAuditConnectionInfo.CeAuditDatabaseDriver.MYSQL_AUDIT_DB_DRIVER`
> - `IAuditConnectionInfo.CeAuditDatabaseDriver.ORACLE_AUDIT_DB_DRIVER`
> - `IAuditConnectionInfo.CeAuditDatabaseDriver.SQL_SERVER_AUDIT_DB_DRIVER`
> - `IAuditConnectionInfo.CeAuditDatabaseDriver.SYBASE_AUDIT_DB_DRIVER`

> If the `CeAuditDatabaseDriver.SQL_SERVER_AUDIT_DB_DRIVER` is used, you can set a single sign-on (trusted connection) variable using the `setUseTrustedConnection` method of the `IAuditConnectionInfo` interface.

Example

The following example retrieves several properties from the active auditing database connection and returns them as a `String` object.

```
String getAuditDBConnectionInfo(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_DB_CONNECTION FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
  IAuditConnectionInfo connectionInfo = auditEventInfo.getAuditConnectionInfo();
  String adsConnectionInfo = "User name: " + connectionInfo.getUserName() +
"<br>" +
    "DSN name: " + connectionInfo.getDsnName() + "<br>" +
    "Database driver: " + connectionInfo.getDatabaseDriver();

  return adsConnectionInfo;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditConnectionInfo`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditConnectionInfo.CeAuditDatabaseDriver`

Related Information

Auditing [page 257]

5.11.10 To change the duration for which auditing events are stored

Auditing events are stored in the auditing database for a certain number of days and then automatically deleted. This value is typically configured in the Central Management Console (CMC).This example shows how to programmatically change the duration that events are stored.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
```

2. Retrieve the auditing events collection from the repository, and allocate it to an `IInfoObjects` collection.

   Use the corresponding `SI_KIND` constant from the `IAuditEventInfo` interface to retrieve the collection. The query must also include references to `SI_AUDIT_EVENTS` and `SI_AUDIT_DELETE_AFTER_N_DAYS`, because they contain information necessary for the system to change the log storage duration.

```
String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_DELETE_AFTER_N_DAYS
FROM " +
  "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
IInfoObjects infoObjects = infoStore.query(auditQuery);
```

3. Retrieve the `IAuditEventInfo` object from the `IInfoObjects` collection.

```
IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

4. Call the `setDeleteAfterNDays` method, and pass in a positive integer representing the number of days that auditing events are stored.

   > ⚠ Caution
   >
   > Data older than the number of days set here will be permanently deleted from the auditing database; it cannot be recovered. You may want to consider periodically moving records to an archive database if you wish to maintain long-term records.

```
auditEventInfo.setDeleteAfterNDays(365);
```

5. Commit the changes back to the repository.

```
infoStore.commit(infoObjects);
```

Example

The following example changes the duration that auditing events are stored to 365 days.

```
void setAuditEventStorageDuration(IEnterpriseSession enterpriseSession) throws
SDKException
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
  String auditQuery = "SELECT SI_AUDIT_EVENTS, SI_AUDIT_DELETE_AFTER_N_WEEKS
FROM " +
    "CI_SYSTEMOBJECTS WHERE SI_KIND = '" + IAuditEventInfo.KIND + "'";
  IInfoObjects infoObjects = infoStore.query(auditQuery);
  IAuditEventInfo auditEventInfo = (IAuditEventInfo) infoObjects.get(0);
```

```
  auditEventInfo.setDeleteAfterNDays(365);
  infoStore.commit(infoObjects);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.businessobjects.sdk.plugin.desktop.auditeventinfo2.IAuditEventInfo`

Related Information

Auditing [page 257]

5.11.11 To enhance traceability for auditing through custom applications

A new logon method has been added with SAP BusinessObjects Business Intelligence platform 4.1 which takes an `IEnterpriseLogonInformation` object as an additional parameter. If used, auditing events will record the CUID and name of the custom application supplied to the `IEnterpriseLogonInformation` object when events are triggered. For a table of all usable `CUIDs` and their custom application names, see: Custom application CUID values and names [page 276].

1. Retrieve an `ISessionMgr` object by calling the `getSessionMgr` method of the `static CrystalEnterprise` interface.

   ```
   ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
   ```

2. Retrieve an `IEnterpriseLogonInformation` object by calling the `createLogonInfo` method of the `ISessionMgr` interface.

   ```
   IEnterpriseLogonInformation logonInfo = sessionManager.createLogonInfo();
   ```

3. Set the client type by calling the `setClientType` method of the `IEnterpriseLogonInformation` interface.

   There are twenty-five custom application identifiers specifically reserved for use by your applications. They follow the naming convention `Custom Application 1`, through `Custom Application 25`, each with a specific CUID that is passed as the parameter to the `setClientType` method.

   ```
   logonInfo.setClientType("AZ4HLbS0loRMpE2twk442RU");
   ```

4. Retrieve an `IEnterpriseSession` object by calling the `logonEx` method of the `ISessionMgr` interface.

   ```
   IEnterpriseSession enterpriseSession = sessionManager.logonEx("username",
   "password", "myCMS:6400", "secEnterprise", logonInfo);
   ```

The logon event and all further events triggered by this session will now include the name of the custom application used to access the system. Use the `IEnterpriseSession` as you normally would.

Example

The following example demonstrates using the `logonEx` method to access the CMS.

```
IEnterpriseSession logonExDemonstration() throws SDKException
{
  ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
  IEnterpriseLogonInformation logonInfo = sessionManager.createLogonInfo();
  logonInfo.setClientType("AZ4HLbS0loRMpE2twk442RU");

  IEnterpriseSession enterpriseSession = sessionManager.logonEx("username",
"password", "myCMS:6400", "secEnterprise", logonInfo);
  return enterpriseSession;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `com.crystaldecisions.sdk.occa.security.IEnterpriseLogonInformation`

Related Information

Auditing [page 257]
Custom application CUID values and names [page 276]

5.11.12 Custom application CUID values and names

CUID values for custom applications

These reserved CUID values can be passed to the `IEnterpriseLogonInformation.setClientType` method when you implement logon calls to your application that take in an `IEnterpriseLogonInformation` object. Recorded auditing events will then indicate the custom application CUID and name for the originating session that resulted in an auditing event being triggered.

CUID	Application Name
AZ4HLbS0loRMpE2twk442RU	Custom Application 1
AR6QLgQwJjxElm502FT_b4A	Custom Application 2
AWYcldOex_tAodjJRrlogIk	Custom Application 3

CUID	Application Name
ASahxem3v_xGjXsQKQaskyk	Custom Application 4
AbKQQkLhQ9RKpbMEV7DECkc	Custom Application 5
AcBrUi5J0rVCiFZ.M35hCYM	Custom Application 6
Afbw5UJ0UYdJINbhDusG1mc	Custom Application 7
AT084MQMNHIIs1fUB9Y4x8k	Custom Application 8
AT0F5dLN5rZJq9IuT7pILTU	Custom Application 9
AXKjeUbzVq1Gm9XS4kIF1Ho	Custom Application 10
ARc7Bcof_V1JpaH.LaBmKMM	Custom Application 11
AZFkpRO4waBGvZNBt4R85YY	Custom Application 12
AZETCpaYYmRIingE_hCnWFw	Custom Application 13
ARAGSiINIAtMvkpuANjWU6I	Custom Application 14
AcfWGfuiPeREmYs3drOx0sQ	Custom Application 15
AUKOYrZfNBpNimTvy.m_WxU	Custom Application 16
Acdt0d02A6VNq3o8DWHWpH8	Custom Application 17
ASRvIZ.mkeVJrhmSxnCqFb8	Custom Application 18
AXydQKHjGVROt_8czecFvxE	Custom Application 19
Ae25eOj9rcFMj31xm.i_5U8	Custom Application 20
AU0U6I7_vcJKsSBk8eZBQBM	Custom Application 21
AUPgOLoPI9JGI5M19R1SeXw	Custom Application 22
Aa7yGnOcE1pPmfkDuxznG7U	Custom Application 23
AQ8m0QHV9LNArZcRo6a2xuM	Custom Application 24
ASyT3vZzFuZKkE7iXkp_tuI	Custom Application 25

Related Information

Auditing [page 257]
To enhance traceability for auditing through custom applications [page 275]

5.12 Localization and language packs

Managing the preferred viewing locale

You can store the preferred viewing locale of the current user and retrieve it at a later time. This can be useful if you need to personalize locale-specific elements of your application, such as formatting of numbers and currency in reports.

The preferred viewing locale is stored in the `IEnterpriseSession` object of the user. By default, this value is not set and it is not stored in the Central Management Server (CMS), so it must be set by your application if required.

Viewing installed and supported language packs

The SAP BusinessObjects Business Intelligence platform SDK also provides methods for retrieving the installed and supported language packs.

Classes used for managing the preferred viewing locale and language packs

- `com.crystaldecisions.sdk.occa.security.IUserInfo`
  Contains the information that identifies the currently logged on user and allows you to get or set the user's preferred locale.
- `com.crystaldecisions.sdk.framework.ILanguageMgr`
  Provides access to information about the locales that are supported and which locales (language packs) are installed for this deployment.

5.12.1 To set the preferred viewing locale

1. Get the current user from a valid `IEnterpriseSession` session object.

```
IUserInfo userInfo = enterpriseSession.getUserInfo();
```

2. Set the preferred viewing locale for the current user.

```
userInfo.setPreferredViewingLocale(Locale.SIMPLIFIED_CHINESE);
```

Example

```
public void setPreferredViewingLocale(IEnterpriseSession enterpriseSession)
throws SDKException
{
  IUserInfo userInfo = enterpriseSession.getUserInfo();
  userInfo.setPreferredViewingLocale(Locale.SIMPLIFIED_CHINESE);
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.security.IUserInfo`
- `java.util.Locale`

5.12.2 To display the current preferred viewing locale

1. Get the current user from a valid `IEnterpriseSession` session object.

   ```
   IUserInfo userInfo = enterpriseSession.getUserInfo();
   ```

2. Get the preferred viewing locale for the current user.

   ```
   Locale preferredViewingLocale = userInfo.getPreferredViewingLocale();
   ```

3. Get the preferred viewing locale display name.

   ```
   String localeName = preferredViewingLocale.getDisplayName();
   ```

Example

```
public String getPreferredViewingLocale(IEnterpriseSession
enterpriseSession)throws SDKException
{
  IUserInfo userInfo = enterpriseSession.getUserInfo();
  Locale preferredViewingLocale = userInfo.getPreferredViewingLocale();
  String localeName = preferredViewingLocale.getDisplayName();
  return localeName;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.security.IUserInfo`
- `java.util.Locale`

5.12.3 To display installed language packs

1. Retrieve an instance of the `ISessionMgr` object.

```
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
```

2. Retrieve an instance of the `ILanguageMgr` object.

```
ILanguageMgr languageMgr = sessionMgr.getLanguageMgr();
```

3. Use the `ILanguageMgr` object to retrieve the installed language packs.

```
 List installedLocales = languageMgr.getInstalledLocales();
```

4. Iterate through the installed language packs and retrieve their display names.

```
String list = new String("");
Iterator installedLocalesIter = installedLocales.iterator();
while (installedLocalesIter.hasNext())
{
  Locale locale = (Locale) installedLocalesIter.next();
  list += locale.getDisplayName() + "\n";
}
```

Example

The following code returns a String object that contains a list of language packs that are installed in this SAP BusinessObjects Business Intelligence platform installation.

```
public String getInstalledLanguagePacks() throws SDKException
{
  ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
  ILanguageMgr languageMgr = sessionMgr.getLanguageMgr();
  List installedLocales = languageMgr.getInstalledLocales();
  Iterator installedLocalesIter = installedLocales.iterator();

  String list = new String("");
  while (installedLocalesIter.hasNext())
  {
    Locale locale = (Locale) installedLocalesIter.next();
    list += locale.getDisplayName() + "<p>";
  }
  return list;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.ILanguageMgr`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `java.util.Iterator`
- `java.util.List`
- `java.util.Locale`

5.12.4 To display all supported languages

1. Retrieve an instance of the `ISessionMgr` object.

```
 ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
```

2. Rretrieve an instance of the `ILanguageMgr` object.

```
ILanguageMgr languageMgr = sessionMgr.getLanguageMgr();
```

3. Use the `ILanguageMgr` object to retrieve the supported languages.

```
List supportedLocales = languageMgr.getSupportedLocales();
```

4. Iterate through the supported languages and get their display names.

```
String list = new String("");
Iterator it = supportedLocales.iterator();
while (it.hasNext())
{
  Locale locale = (Locale) it.next();
  list += locale.getDisplayName() + "<p>";
}
```

Example

The following code returns a String object that contains a list of the languages that are supported by this SAP BusinessObjects Business Intelligence platform installation.

```
public String displaySupportedLanguagesDescription() throws SDKException
{
     ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
     ILanguageMgr languageMgr = sessionMgr.getLanguageMgr();
     List supportedLocales = languageMgr.getSupportedLocales();

  String list = new String("");
     Iterator it = supportedLocales.iterator();
     while (it.hasNext())
     {
          Locale locale = (Locale) it.next();
          list += locale.getDisplayName() + "<p>";
     }
     return list;
}
```

This list includes the classes used by the sample code:

- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.CrystalEnterprise`
- `com.crystaldecisions.sdk.framework.ILanguageMgr`
- `com.crystaldecisions.sdk.framework.ISessionMgr`
- `java.util.Iterator`
- `java.util.List`
- `java.util.Locale`

5.13 Cryptographic key management

Cryptographic keys are used to encrypt InfoObjects that are stored in the BI platform. For more information about using cryptographic keys, see the *SAP Business Objects Business Intelligence Platform Administrator Guide*.

You can use this SDK to manage cryptographic keys. To perform most key management tasks, you must be authenticated as a member of the Cryptographic Officers group. Any user can view the state of a cryptographic key. However, only Cryptographic Officers users can create, modify, and delete cryptographic keys.

Users who are members of the Cryptographic Officers group can perform the following cryptographic key management tasks:

- Create a new key. The new key becomes the active key.
- Revoke a deactivated key. When a key is revoked, the InfoObjects that use it are re-encrypted with the active key.
- Delete revoked keys.

> ⓘ Note
>
> If you store encrypted data outside of the BI platform, you must manually decrypt it before you delete its encryption key.

Related Information

Cryptographic key states [page 282]

5.13.1 Cryptographic key states

Each Central Management Server (CMS) cluster has one active cryptographic key, and any number of deactivated and revoked keys.

The active key is used to encrypt and decrypt InfoObjects stored within a BI platform installation. To enhance security, you can periodically replace the active key with a new one. When this happens, the active key is deactivated. It is no longer used to encrypt data, but it continues to be used to decrypt objects that were encrypted with it.

If you want to delete a key from the repository, you must make sure that the InfoObjects that were encrypted with it are decrypted and re-encrypted with a new key. Otherwise, you will not be able to decrypt and access these InfoObjects after the cryptography key is deleted. You can automatically re-encrypt InfoObjects that are stored in a BI platform installation by using the `startRekey` method of the `ICryptographicKey` class. Once the re-keying process has been completed, the key may be deleted from the repository.

> ⓘ Note
>
> If you store encrypted data outside of a BI platform installation, you must manually decrypt it before you delete its cryptographic key.

You can determine the state of the cryptographic key by calling the `getKeyState` method of the `IClusterKey.KeyState` class.

State	Usage	Description
`ACTIVE`	Encryption Decryption	The key is used to encrypt (and decrypt) data.
`DEACTIVATED`	Decryption	The key is no longer used for encryption, but is still used to decrypt data that was encrypted with it when it was the active key.
`REKEY_IN_PROGRESS`	Decryption	A re-key operation is in progress. Objects that were encrypted with this key are in the process of being re-encrypted with the active cluster key.
`REKEY_SUSPENDED`	Decryption	A re-key operation was started but has been suspended.
`REVOKED`	None	The repository does not contain objects that are encrypted with this key. The key can be deleted. **Note** If you store encrypted objects outside of a BI platform installation, you must decrypt them manually before you delete the encryption key.

5.13.2 To create a new cryptographic key

To create a cryptographic key, you must be logged on to the BI platform using an account that is a member of the Cryptographic Officers group.

To enhance security, you can periodically replace the active cryptographic key with a new one. When you replace the active cryptographic key, it is automatically deactivated.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. Create a new `InfoObjects` collection to store the cryptographic key.

```
IInfoObjects infoObjects = infostore.newInfoObjectCollection();
```

3. Create a new cryptographic key by adding the plugin information to the `InfoObjects` collection.

```
ICryptographicKey cryptographicKey = (ICryptographicKey)
infoObjects.add(ICryptographicKey.KIND);
```

4. Set the name of the cryptographic key.

   This example sets the name of the cryptographic key to `CryptographicKey`.

```
cryptographicKey.setTitle("CryptographicKey");
```

5. Retrieve the `Cryptographic Keys` folder.

```
IInfoObjects folders = infostore.query("SELECT SI_ID FROM CI_SYSTEMOBJECTS
WHERE SI_KIND='" + IFolder.FOLDER_KIND +
  "' AND SI_CUID='" + CeSecurityCUID.RootFolder.CRYPTOGRAPHIC_KEYS + "'");
IInfoObject folder = (IInfoObject) folders.get(0);
```

6. Set the `Cryptographic Keys` folder to be the parent folder of the cryptographic key.

```
cryptographicKey.setParentID(folder.getID());
```

7. Commit the cryptographic key back to the Central Managment Server (CMS) to save the changes.

```
infostore.commit(infoObjects);
```

The active cryptographic key is deactivated and replaced with the new cryptographic key.

Example: Creating a new cryptographic key

```
void createKey(IEnterpriseSession enterpriseSession)throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");

  IInfoObjects infoObjects = infostore.newInfoObjectCollection();
  ICryptographicKey cryptographicKey =
(ICryptographicKey)infoObjects.add(ICryptographicKey.KIND);
  cryptographicKey.setTitle("CryptographicKey");

  IInfoObjects folders = infostore.query("SELECT SI_ID FROM CI_SYSTEMOBJECTS
WHERE SI_KIND='" + IFolder.FOLDER_KIND +
    "' AND SI_CUID='" + CeSecurityCUID.RootFolder.CRYPTOGRAPHIC_KEYS + "'");
  IInfoObject folder = (IInfoObject) folders.get(0);

  cryptographicKey.setParentID(folder.getID());
  infostore.commit(infoObjects);
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.cryptographickey.ICryptographicKey`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.CeSecurityCUID`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.plugin.desktop.folder.IFolder`

5.13.3 To re-encrypt data with a new key

To re-encrypt data with a new key, you must be logged on to the BI platform using an account that is a member of the Cryptographic Officers group. A key must be deactivated before the re-encryption process can begin.

You can re-encrypt InfoObjects with the active key by calling the `startRekey` method of the `ICryptographicKey` class. When the re-encryption process is complete, the key can be deleted from the repository.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

```
IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. Retrieve the cryptographic key from the InfoStore.

```
IInfoObjects infoObjects = infostore.query("Select SI_ID, SI_KEY_STATE from
CI_SYSTEMOBJECTS where SI_NAME='" + masterKeyName + "'");
ICryptographicKey key = (ICryptographicKey) infoObjects.get(0);
```

3. Verify that the key is in the `DEACTIVATED` state.

> **Note**
>
> If the key is still in the `ACTIVE` state, you can deactivate it by creating a new active cryptographic key.

```
ICryptographicKey.KeyState state = key.getKeyState();
if (state == ICryptographicKey.KeyState.DEACTIVATED)
  ...
```

4. Start the re-encryption process by calling the `startRekey` method of the `ICryptographicKey` class.

```
key.startRekey();
```

> **Note**
>
> You can suspend and restart the re-encryption process by using the `suspendRekey` and `resumeRekey` methods.

5. Save the key.

```
key.save();
```

InfoObjects encrypted with the key are re-encrypted with the active key, and the key state is changed to `KeyState.REVOKED`.

Example

```
void startRekey(IEnterpriseSession enterpriseSession, String masterKeyName)
throws SDKException
{
  IInfoStore infostore = (IInfoStore)enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_ID, SI_KEY_STATE from
CI_SYSTEMOBJECTS where SI_NAME='" + masterKeyName + "'");
  ICryptographicKey key = (ICryptographicKey) infoObjects.get(0);

  ICryptographicKey.KeyState state = key.getKeyState();
  if (state == ICryptographicKey.KeyState.DEACTIVATED)
  {
    key.startRekey();
    key.save();
  }
```

```
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.cryptographickey.ICryptographicKey`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`

5.13.4 To delete a cryptographic key

A cryptographic key must be in the `REVOKED` state before it can be deleted from the repository. Use the `startRekey` method of the `ICryptographicKey` class to revoke a key.

When a key has been revoked, it is no longer in use, and you can delete it from the repository.

> **Note**
>
> If you store encrypted data outside of the BI platform, you must manually decrypt it before you delete its cryptographic key.

1. Create a connection to the InfoStore service from a valid `IEnterpriseSession` session object.

   ```
   IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Get the cryptographic key from the InfoStore.

   ```
   IInfoObjects infoObjects = infostore.query("Select SI_ID, SI_KEY_STATE from
   CI_SYSTEMOBJECTS where SI_NAME='" + keyName + "'");
   ICryptographicKey key = (ICryptographicKey) infoObjects.get(0);
   ```

3. Verify that the key has been revoked.

   ```
   ICryptographicKey.KeyState state = key.getKeyState();
   if (state == ICryptographicKey.KeyState.REVOKED)
     ...
   ```

4. Call the `delete` method of the `IInfoObjects` class to delete the key.

   ```
   infoObjects.delete(key);
   ```

5. Commit the modified key collection back to the Central Management Server (CMS) to save the changes.

   ```
   infostore.commit(infoObjects);
   ```

Example: To delete an unused key

```
void deleteKey(IEnterpriseSession enterpriseSession, String keyName) throws
SDKException
```

```
{
  IInfoStore infostore = (IInfoStore) enterpriseSession.getService("InfoStore");
  IInfoObjects infoObjects = infostore.query("Select SI_ID, SI_KEY_STATE from
CI_SYSTEMOBJECTS where SI_NAME='" + keyName + "'");
  ICryptographicKey key = (ICryptographicKey) infoObjects.get(0);
  ICryptographicKey.KeyState state = key.getKeyState();
  if (state == ICryptographicKey.KeyState.REVOKED)
  {
    infoObjects.delete(key);
    infostore.commit(infoObjects);
  }
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.plugin.desktop.cryptographickey.ICryptographicKey`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`

Related Information

To re-encrypt data with a new key [page 284]

5.14 Business Intelligence Archive (BIAR)

The SAP BusinessObjects Business Intelligence platform BIAR Engine is a tool that lets administrators and delegated administrators back up business intelligence content like users, groups, security settings, and reports to a `BIAR` file, which can be used to restore the content at any time. It also lets administrators directly export content from one cluster to another without having to export the content to a `BIAR` file first.

The BIAR Engine can be used in the following ways:

- Using SAP BusinessObjects upgrade management tool.
  SAP BusinessObjects upgrade management tool imports and upgrades business intelligence from previous versions of SAP BusinessObjects Business Intelligence platform to the current version, as part of the process of upgrading your SAP BusinessObjects Business Intelligence platform deployment.

  > **Note**
  >
  > For more information, see the *SAP Business Objects Business Intelligence Platform Upgrade Guide*.

- Using Lifecycle management console for SAP BusinessObjects Business Intelligence platform 4.1.
  Lifecycle management console for SAP BusinessObjects Business Intelligence platform 4.1 promotes content from an SAP BusinessObjects Business Intelligence platform deployment to another SAP BusinessObjects Business Intelligence platform deployment at the same version, as part of a life cycle management process.

> **Note**
>
> For more information, see the *Lifecycle management console for SAP BusinessObjects Business Intelligence Platform 4.1 User Guide*.

- Using the BIAR engine command-line utility.
  The BIAR Command-Line Tool imports and exports `BIAR` files within a repository. The BIAR engine (`biarengine.jar`) can be found at the following default location of your SAP BusinessObjects Business Intelligence platform deployment: `C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\java\lib\`.

  > **Note**
  >
  > For more information, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

- Using this SDK.
  Using the SAP BusinessObjects Business Intelligence platform Java SDK, you can import InfoObjects from a `BIAR` file on the local file system into the Central Management Server (CMS), export InfoObjects from the CMS to a `BIAR` file on your local directory, and import InfoObjects directly from one CMS to another CMS.

> **Note**
>
> The BIAR Command Line Tool and this SDK support importing and exporting objects between versions of SAP BusinessObjects Business Intelligence platform at the same version. If you are importing objects to the current version of SAP BusinessObjects Business Intelligence platform from previous versions, you must use SAP BusinessObjects upgrade management tool.

> **Note**
>
> Your source and destination environments must be at the same Service Pack and FixPack level. Before importing or exporting objects, ensure that both environments are at the same update level.

Class used when working with the BIAR Engine

- `com.businessobjects.sdk.biar.BIARFactory`
  Used to create `IObjectManager` instances.
- `com.businessobjects.sdk.biar.ILiveToLivePipe`
  Used to transfer objects from one deployment to another without the need to create `BIAR` files.
- `com.businessobjects.sdk.biar.IExportOptions`
  Used to define options when exporting InfoObjects.
- `com.businessobjects.sdk.biar.IImportOptions`
  Used to define options when importing InfoObjects.
- `com.businessobjects.sdk.biar.om.IManagedObjectIterator`
  Used to iterate through InfoObjects contained in an `IObjectManager` instance.
- `com.businessobjects.sdk.biar.om.IObjectManager`
  Used to manage collections of InfoObjects

5.14.1 Import considerations

Performing test commits

A test commit allows you to determine what, if any, errors would occur if you commit a collection of InfoObjects to a CMS. The SDK instructs the CMS to commit the collection, collect any errors, and then automatically rollback the changes.

> **Note**
>
> Test commits only test adding collections of objects to the CMS. A test commit cannot be used to test copying files to the File Repository Servers.

You can perform a test commit by calling the `testCommit` method of an `IObjectManager` object. The `testCommit` method returns an `ICommitResult` object, which is a collection that contains any errors that occurred during the commit.

```
private ICommitResult performTestCommit(IObjectManager objectManager) {
  ICommitOptions commitOptions = BIARFactory.getCommitOptions();
  ICommitResult commitResult = objectManager.testCommit(commitOptions);
  return ICommitResult;
}
```

Importing objects and security

The BIAR Engine allows you to decide whether you want to import security settings when you import or export content.

To import or export content without also importing security settings, call the `setIncludeSecurity` method of the `IImportOptions` or `IExportOptions` class.

```
BIARFactory biarFactory;
IImportOptions importOptions = biarFactory.createImportOptions();
importOptions.setIncludeSecurity(false);
```

5.14.2 To export InfoObjects to a BIAR file

This example shows how to programmatically create and export a `BIAR` file from the database to your local directory.

1. Create a connection the the `InfoStore` service from a valid Enterprise session.

   ```
   IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
   ```

2. Retrieve the objects to be exported from the database.

   ```
   String query = "select * from ci_Infoobjects";
   IInfoObjects infoObjects = infoStore.query(query);
   ```

3. Create `BIARFactory` and `IObjectManager` objects.

```
BIARFactory biarFactory = BIARFactory.getFactory();
IObjectManager objectManager = biarFactory.createOM(enterpriseSession);
```

4. Add the objects to be exported to the `IObjectManager` object.

```
objectManager.insertAll(infoObjects.iterator());
```

5. Configure the export options.

```
IExportOptions exportOptions = biarFactory.createExportOptions();
exportOptions.setIncludeSecurity(true);
exportOptions.setFailUnresolvedIDs(true);
```

6. Call the `setCallback` method.

   A callback method is a means for event handling Events about exporting are handled in this SDK by the `IExportCallback` interface. The `IExportCallback` interface must be implemented by an object to be notified of export events. Therefore, in this example we use a `MyExportCallback` class to implement the `IExportCallback` interface method. You can later implement the `onSuccess` method to print a text string that provides user feedback.

```
exportOptions.setCallback(new MyExportCallback());
```

7. Export objects to a `BIAR` file by calling the `exportToArchive` method of the `IObjectManager` class. This method requires a string argument that represents the name and path of the `BIAR` file to which you want the content exported.

```
try
{
 objectManager.exportToArchive(biarFile, exportOptions);
}
catch(BIARException e)
{
 e.printStackTrace();
}
System.out.println(objectManager.size() + " objects processed.");
```

> **Note**
>
> For more information about exception handling, see Catch and handle SDK exceptions [page 69].

8. Dispose of the `IObjectManager` object.

```
objectManager.dispose();
}
```

When you run this code, the InfoObjects that you select are exported from the CMS system database to a `BIAR` file on your local directory. You must specify a local directory at run time. You can also use the `BIAR` file as a backup to restore settings at a later date.

Example

```
public void exportInfoObjects(IEnterpriseSession enterpriseSession, String
biarFile) throws SDKException, OMException
```

```
{
  IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");

  String query = "select * from ci_Infoobjects";
  IInfoObjects infoObjects = infoStore.query(query);
  BIARFactory biarFactory = BIARFactory.getFactory();
  IObjectManager objectManager = biarFactory.createOM(enterpriseSession);
  objectManager.insertAll(infoObjects.iterator());

  IExportOptions exportOptions = biarFactory.createExportOptions();
  exportOptions.setIncludeSecurity(true);
  exportOptions.setFailUnresolvedIDs(true);

  exportOptions.setCallback(new MyExportCallback());

  try
  {
    objectManager.exportToArchive(biarFile, exportOptions);
  }
  catch (BIARException e)
  {
    e.printStackTrace();
  }
  finally
  {
    System.out.println(objectManager.size()+ "objects processed.");
          objectManager.dispose();
  }

}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.biar.BIARException`
- `com.businessobjects.sdk.biar.BIARFactory`
- `com.businessobjects.sdk.biar.IExportCallback`
- `com.businessobjects.sdk.biar.IExportOptions`
- `com.businessobjects.sdk.biar.om.IObjectManager`
- `com.businessobjects.sdk.biar.om.OMException`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

Related Information

To import InfoObjects from a BIAR file [page 292]
To perform callback [page 296]

5.14.3 To import InfoObjects from a BIAR file

This task shows you how to import objects from a `BIAR` file on the local file system into the database. You must have a `BIAR` file on your local directory that you would like to import to the database.

1. Create a connection to the `InfoStore` service from a valid Enterprise session.

```
IInfoStore infoStore = (IInfoStore)enterpriseSession.getService("InfoStore");
```

2. Create `BIARFactory` and `IObjectManager` objects.

```
BIARFactory biarFactory = BIARFactory.getFactory();
IObjectManager objectManager = biarFactory.createOM(enterpriseSession);
```

3. Configure the import options.

```
IImportOptions importOptions = biarFactory.createImportOptions();
importOptions.setIncludeSecurity(true);
importOptions.setFailUnresolvedCUIDs(true);
```

4. Call the `setCallback` method.

```
importOptions.setCallback(new MyImportCallback());
```

> **Note**
>
> A callback method is a means for event handling. Events about importing are handled in this SDK by the `IImportCallback` interface. The `IImportCallback` interface must be implemented by an object to be notified of import events. Therefore, in this example we use a `MyImportCallback` class to implement the `IImportCallback` interface method.

5. Commit the changes to the database.

```
objectManager.commit();
System.out.println(objectManager.size() + " objects processed.");
```

6. Import the `BIAR` file.

```
try
{
 objectManager.importArchive(biarFile, importOptions);
}
catch(BIARException e)
{
 e.printlnStackTrace();
}
```

> **Note**
>
> If an InfoObject that you are trying to import into the database already exists, the existing object is merged with the new object. The merged object will contain the properties from both objects, with the properties from the source object taking precedence over those of the destination object in the event of any differences.

7. Display information about each object and then close the iterator.

```
IManagedObjectIterator managedObjectIter;
managedObjectIter = objectManager.readAll();
```

```
while(managedObjectIter.hasNext())
  {
    IInfoObject infoObject = managedObjectIter.next();
    System.out.println("SI_CUID: " + infoObject.getCUID()
      + ", SI_NAME: " + infoObject.getTitle() ", SI_KIND: " +
infoObject.getKind());
  }
managedObjectIter.close();
```

This step is optional but recommended for notification purposes.

8. Dispose the object manager.

```
objectManager.dispose();
}
```

The `BIAR` file from your local directory is imported to the database that you specified at run time. To verify this import, check your database. You can now use this imported `BIAR` file and export it to another location.

Example

```
public void importInfoObjects(IEnterpriseSession enterpriseSession, String
biarFile) throws SDKException, OMException
{
  BIARFactory biarFactory = BIARFactory.getFactory();
  IObjectManager objectManager = biarFactory.createOM(enterpriseSession);

  IImportOptions importOptions = biarFactory.createImportOptions();
  importOptions.setIncludeSecurity(true);
  importOptions.setFailUnresolvedCUIDs(true);

  importOptions.setCallback(new MyImportCallback());
  try
  {
    objectManager.importArchive(biarFile,importOptions);
                objectManager.commit();
    IManagedObjectIterator managedObjectIter;
    managedObjectIter = objectManager.readAll();

    while(managedObjectIter.hasNext())
    {
      IInfoObject infoObject = managedObjectIter.next();
      System.out.println("SI_CUID:" +infoObject.getCUID()+ ", SI_NAME:" +
infoObject.getTitle()+ ",SI_KIND:"+infoObject.getKind());
    }
  }
  catch (BIARException e)
  {
    e.printStackTrace();
  }
  finally
  {
    managedObjectIter.close();
       System.out.println(objectManager.size() + "objects processed.");
       objectManager.dispose();
  }
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.biar.BIARException`
- `com.businessobjects.sdk.biar.BIARFactory`
- `com.businessobjects.sdk.biar.IImportCallback`
- `com.businessobjects.sdk.biar.IImportOptions`
- `com.businessobjects.sdk.biar.om.IManagedObjectIterator`
- `com.businessobjects.sdk.biar.om.IObjectManager`
- `com.businessobjects.sdk.biar.om.OMException`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `java.util.Iterator`

Related Information

To export InfoObjects to a BIAR file [page 289]
To perform callback [page 296]

5.14.4 To perform a live-to-live transfer

This example shows how to programmatically export objects from a source environment and then import those objects into a destination environment, without having to use a `BIAR` file.

This procedure requires Enterprise sessions for both the source and destination CMSs.

1. Create a connection the the `InfoStore` service from a valid Enterprise session from the destination environment.

```
        IInfoStore infoStore = (IInfoStore)
sourceSession.getService("InfoStore");
```

2. Retrieve the objects to be exported from the database.

```
 String query = "select * from ci_Infoobjects";
 IInfoObjects infoObjects = infoStore.query(query);
```

3. Create `BIARFactory` and `IObjectManager` objects.

   Use the source Enterprise session when instantiating the `IObjectManager` object.

```
BIARFactory biarFactory = BIARFactory.getFactory();
IObjectManager objectManager = biarFactory.createOM(destinationSession);
```

4. Add the objects from the source CMS to the `IObjectManager` object.

```
objectManager.insertAll(infoObjects.iterator());
```

5. Configure the export options.

```
IExportOptions exportOptions = biarFactory.createExportOptions();
```

```
exportOptions.setIncludeSecurity(true);
exportOptions.setFailUnresolvedIDs(true);
exportOptions.setIncludeDependencies(true);
```

6. Configure the import options.

```
IImportOptions importOptions = biarFactory.createImportOptions();
importOptions.setIncludeSecurity(true);
importOptions.setFailUnresolvedCUIDs(true);
```

7. Call the `liveImport` method of the `IObjectManager` class. This method returns an `ILiveToLivePipe` object.

   You must pass the Enterprise Session from the source environment as a parameter.

```
ILiveToLivePipe pipe = objectManager.liveImport(sourceSession, exportOptions,
importOptions);
```

8. Call the `getExporter` method of the `ILiveToLivePipe` object to create an `IExporter` object. Use the IExporter object to export the objects from the source system to the destination system.

```
try{
     IExporter exporter = pipe.getExporter();
     try
     {
          exporter.exportAll(infoObjects);
          exporter.finish();
     }
     finally{
          exporter.close();
     }
     pipe.getResult().get();
}
finally
{
     dispose();
}
```

9. Commit and then dispose of the `IObjectManager`.

```
objectManager.commit();
objectManager.dispose();
```

Example

```
public String liveTransfer(IEnterpriseSession destinationSession,
IEnterpriseSession sourceSession) throws SDKException, OMException,
BIARException
{
  IInfoStore infoStore = (IInfoStore)sourceSession.getService("InfoStore");
  String query = "select * from ci_Infoobjects";
  IInfoObjects infoObjects = infoStore.query(query);

  BIARFactory biarFactory = BIARFactory.getFactory();
  IObjectManager objectManager = biarFactory.createOM(destinationSession);
  objectManager.insertAll(infoObjects.iterator());
  IExportOptions exportOptions = biarFactory.createExportOptions();
  exportOptions.setIncludeSecurity(true);
  exportOptions.setFailUnresolvedIDs(true);
  exportOptions.setIncludeDependencies(true);
```

```
  IImportOptions importOptions = biarFactory.createImportOptions();
  importOptions.setIncludeSecurity(true);
  importOptions.setFailUnresolvedCUIDs(true);

  ILiveToLivePipe pipe = objectManager.liveImport(sourceSession, exportOptions,
importOptions);

  try{
     IExporter exporter = pipe.getExporter();
     try
     {
          exporter.exportAll(infoObjects);
          exporter.finish();
     }
     finally{
          exporter.close();
     }
     pipe.getResult().get();
  }
  finally
  {
     dispose();
  }

  objectManager.commit();
  objectManager.dispose();
}
```

This list includes the classes used by the sample code:

- `com.businessobjects.sdk.biar.BIARException`
- `com.businessobjects.sdk.biar.BIARFactory`
- `com.businessobjects.sdk.biar.IExporter`
- `com.businessobjects.sdk.biar.IExportOptions`
- `com.businessobjects.sdk.biar.IImportCallback`
- `com.businessobjects.sdk.biar.IImportOptions`
- `com.businessobjects.sdk.biar.ILiveToLivePipe`
- `com.businessobjects.sdk.biar.om.IObjectManager`
- `com.businessobjects.sdk.biar.om.OMException`
- `com.crystaldecisions.sdk.exception.SDKException`
- `com.crystaldecisions.sdk.framework.IEnterpriseSession`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObject`
- `com.crystaldecisions.sdk.occa.infostore.IInfoObjects`
- `com.crystaldecisions.sdk.occa.infostore.IInfoStore`

5.14.5 To perform callback

To perform callback, you must be importing or exporting `BIAR` files to or from a database.

A callback allows an object to receive notification of events that occur during an import or export. To perform callback, you must implement the `IExportCallback` or `IImportCallback` interfaces respectively depending on whether you are performing an export or an import. This example shows how to perform an export callback by using the `MyExportCallback` class that implements the `IExportCallback` interface.

> **Note**
>
> In this example, the `MyExportCallback` class is called in the `exportInfoObjects` method by the `setCallback` method, which you can implement when you set `exportOptions` for the `BIAR` file.

1. Implement the `onSuccess` method.

```
public class MyExportCallback implements IExportCallback
{
  public void onSuccess(int id)
  {
    System.out.println("Object with ID " + id + " was exported
successfully.");
  }
...
```

2. Implement the `onFailure` method.

```
...
  public void onFailure(int id, BIARException cause)
  {
    System.out.println("Object with ID " + id + " failed to export: " +
cause.getMessage());
  }
}
```

> **Note**
>
> To perform callback for the import action, use the `onProgressUpdate` method instead of the `onSuccess` and `onFailure` methods. For more information about the `onProgressUpdate` method, see the *SAP BusinessObjects Business Intelligence Platform Java SDK*.

Example

```
public class MyExportCallback implements IExportCallback
{
  public void onSuccess(int id) {
    System.out.println("Object with ID " + id
      + " was exported successfully.");
  }
  public void onFailure(int id, BIARException cause) {
    System.out.println("Object with ID " + id + " failed to export: "
      + cause.getMessage());
  }
}
```

Related Information

To export InfoObjects to a BIAR file [page 289]
To import InfoObjects from a BIAR file [page 292]

6 References

6.1 Processing Extension API Reference

The Processing Extension API allows you to write your own processing extension. A processing extension is a plug-in DLL (on a Windows system) or shared library (on a UNIX system) that is able to manipulate a report when a request is made to view or schedule that report. Specifically, it gives you the opportunity to gather information about the report and, if you so choose, modify the report record selection formula and report parameters before the view or schedule request is processed by the Crystal Reports Cache Server, Job Server, or the Report Application Server (RAS).

A typical example of a processing extension is a report-processing plug-in that enforces row-level security. The code first determines the user who owns the processing job; then it looks up the user's data-access privileges in a third-party system. The code then generates and appends a record selection formula to the report in order to limit the data returned from the database. In this case, the processing extension serves as a way to implement customized row-level security.

6.1.1 Creating an extension

Your plug-in must support multi-threading; it is recommended that you use C/C++ to build your DLL or shared library.

- Processing extensions written in Java are not supported in this version.
- On Windows systems, dynamically loaded libraries are referred to as dynamic-link libraries (.dll file extension). On UNIX systems, dynamically loaded libraries are often referred to as shared libraries (.so file extension). You must include the .dll or .so file extension when you name your processing extensions. Also, file names cannot include the \ or / characters.

6.1.2 Loading an extension

The Central Management Console (CMC) and the SDK provide methods for enabling your processing extensions and for specifying which objects each extension should be applied to. By enabling processing extensions, you configure the appropriate BI platform server components to dynamically load your processing extensions at runtime.

- In the current release, processing extensions can be applied only to Crystal report objects. To apply an extension to a specific report, you can use the CMC, or you can retrieve `IProcessingExtension` by using `IReportGlobal` in the SDK.
- If a report that you schedule uses a processing extension, the instance that is created will need to use the same processing extension.

All Job Servers, Report Application Servers (RAS), and Crystal Reports Cache Servers have a list of the processing extensions that are installed on them. As well, each report has a list of processing extensions that apply to it. When either a view or schedule report request is issued, the BI platform will try to load the extension for that particular report. If the extension cannot be loaded - for example, if the needed extension is not installed on that particular server - then an error will be sent, and the operation will be aborted for security reasons.

If the extension is successfully loaded, it can be used to do the following:

- Request that the server retrieve information such as the user ID and report title.
- Create a new selection formula that will be appended to a selection formula that already exists in the report. The resulting selection formula is a combination (using the logical operator "AND") of the original selection formula and the new selection formula created by the processing extension.
- Change the prompt values for the report and its subreport.

Once the processing extension is used, the data and threading changes from shared to unshared.

Before loading processing extensions, you must make them available to each machine that will process the relevant schedule or view requests. You can register them on a machine by copying processing extensions onto each server or sharing processing extensions between multiple servers.

6.1.2.1 Copying processing extensions onto each server

The BI platform installation creates a default directory for your processing extensions on each Crystal Reports Cache Server, Job Server, and Report Application Server (RAS). It is recommended that you copy your processing extensions to the default directory on each server. For the location of the default processing extensions directory, see Directories [page 96].

> → Tip
>
> It is possible to share a processing extension file. For details, see Sharing processing extensions between multiple servers [page 300].

Depending upon the functionality that you have written into the extension, copy the library onto the following machines:

- If your processing extension intercepts only schedule requests, copy your library onto each machine that is running as a Crystal Reports Job Server.
- If your processing extension intercepts only view requests, copy your library onto each machine that is running as a Crystal Reports Cache Server.
- If your processing extension intercepts schedule and view requests, copy your library onto each machine that is running as a Crystal Reports Job Server and/or as a Crystal Reports Cache Server.

> ⓘ Note
>
> If the processing extension is required only for schedule/view requests made to a particular server group, you need only copy the library onto each Crystal Reports Cache Server/Job Server/RAS server in the group.

6.1.2.2 Sharing processing extensions between multiple servers

If you want to put all processing extensions in a single location, you can override the default processing extensions directory for each Crystal Reports Cache Server, Job Server or Report Application Server. First, copy your processing extensions to a shared directory on a network drive that is accessible to all of the servers. Map (or mount) the network drive from each machine with these servers.

> **Note**
>
> If you are running servers on both Windows and UNIX, you must copy a `.dll` and an `.so` version of every processing extension into the shared directory. In addition, the shared network drive must be visible to Windows and to UNIX machines (through Samba or some other file-sharing system).

Finally, change each server's command line to modify the default processing extensions directory. Do this by adding `"-report_ProcessExtPath <absolute path>"` to the command line. Replace `<absolute path>` with the path to the new folder, using whichever path convention is appropriate for the operating system that the server is running on (for example, `M:\code\extensions`, `/home/shared/code/extensions`, and so on).

The procedure for making this modification depends upon your operating system:

- On Windows, use the Central Configuration Manager to stop the Crystal Reports Cache Server/Job Server/RAS server. Then open the server's Properties to modify the command line. Start the server again when you have finished.
- On UNIX, run `ccm.sh` to the Crystal Reports Cache Server/Job Server/RAS server. Then edit `ccm.config` to modify the server's command line. Start the server again when you have finished. For reference, see the *SAP BusinessObjects Business Intelligence Platform Administrator Guide*.

6.1.3 Processing Extension API

This reference describes the API calls that you can use to create a processing extension. For details on how to create an extension, see Creating an extension [page 298].

The Processing Extension API reference contains information about these API calls.

API Contents	Description
Exported functions [page 301]	Loads, initializes, and terminates your extension.
IPROCESSEXTRequest Interface [page 304]	Obtains all other supported interfaces.
IReportInfo Interface [page 307]	Retrieves information about the report or subreport that is being processed. Read-Only.
IUserInfo Interface [page 310]	Retrieves information about the user for whom the report is being processed. Read-Only.

API Contents	Description
ITableSet Interface [page 311]	Retrieves information about the set of tables used in a particular report. Read-Only.
IPromptSet Interface [page 317]	Changes prompt information that has been previously set in the main report and its subreports.
ISelectionInfoFormula Interface [page 326]	Retrieves information about the current record selection formula and allows you to append additional criteria to it.
Macros [page 330]	Macros are the easiest way to retrieve an interface.

6.1.3.1 Exported functions

In order for the BI platform to properly load, initialize, and terminate your extension, you must export the following three functions using standard C calling conventions. They are called in this order:

- IInitializePlugin Function [page 302]
- IProcessRequest Function [page 303]
- ITerminatePlugin Function [page 304]

> **Note**
>
> If the programming language you are using is C++, these functions must be defined in your .cpp file with your own code inside them. They should also be exported using a .def file, or something similar if you are working on a UNIX platform. For an example, see `simpleclient.def` (Windows) or `simpleclient.def.unix` (UNIX). These files are located on the SAP BusinessObjects Business Intelligence platform CD in the following directory: `\samples\processext\simpleclient`.

Typical calling behavior

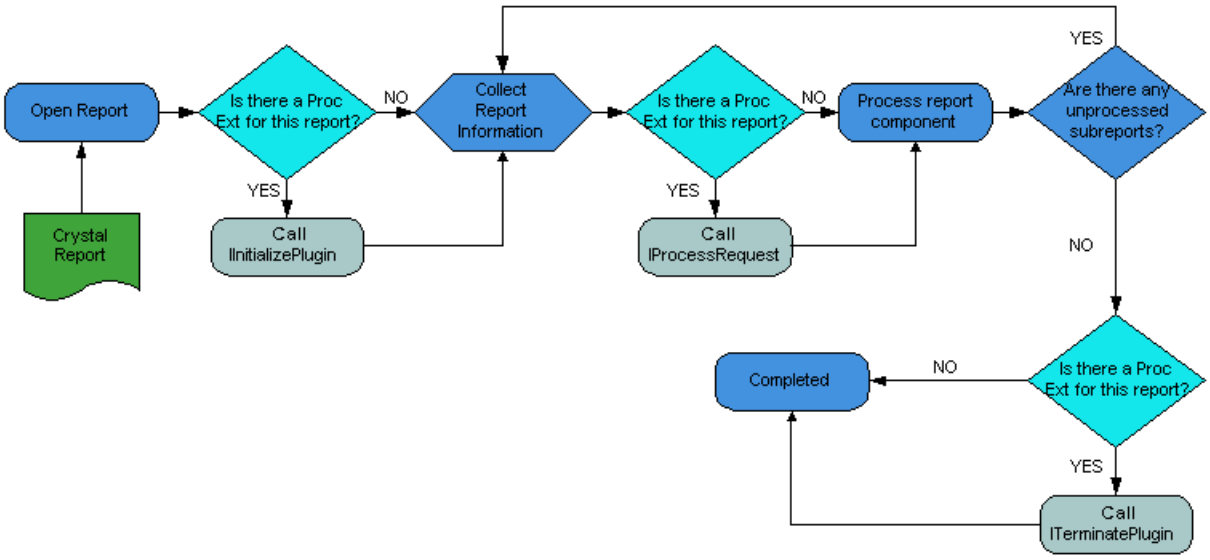
The IInitializePlugin function (which initializes the plugin) and ITerminatePlugin function (which releases the plug-in) are called once per request, whereas the IProcessRequest (which accesses and modifies the report) may be called several times, depending on the number of subreports that are being viewed or processed. For example, if a user views a report that uses a processing extension, and then drills down on two subreports, subreport1 and subreport2, the sequence of calls might look like this:

- IInitializePlugin
- IProcessRequest for main report
- IProcessRequest for subreport1
- IProcessRequest for subreport2
- ITerminatePlugin

> → Tip
>
> Many different requests may be sent to a processing extension concurrently, which means that the DLL or shared library must support multi-threading; that is, it must be thread-safe and re-entrant. Because a processing extension may need to handle more than one request, you will not know how long it will remain in memory.

Calling flow diagram

The processing extension is loaded on a per request basis. The following diagram illustrates how the processing extension is called for a schedule request that is being processed on the Job Server.



The calling behavior is similar for view requests that are processed on the Crystal Reports Cache Server. IProcessRequest is event-based, and is called when the user views the report or drills down on a subreport.

6.1.3.1.1 IInitializePlugin Function

Called when the extension is loaded.

Syntax

```
bool STD_CALL IProcessRequest(
    void* ptr,
    ConstRequestHandle i_reqHandle)
```

Parameters

ptr

Allows you to keep a handle on your own structure between the exported function calls. This value is passed to IProcessRequest and ITerminatePlugin.

Remarks

This function is called once per request. You can use the void pointer to keep control of your own structures between the three function calls. This is only for your own purposes. At this point, you do not have access to the report.

You will want to create the memory that you want to share across the function calls and set the function calls here.

6.1.3.1.2 IProcessRequest Function

Called to allow you to manipulate the report. This function gives you a handle with which you can do so.

Syntax

```
bool STD_CALL IProcessRequest(
    void* ptr,
    ConstRequestHandle i_reqHandle)
```

Parameters

ptr

A void pointer you can use to pass values between the three exported functions. This is the value assigned in the IInitializePlugin Function [page 302].

i_reqHandle

A handle you can use to manipulate the report. You need this handle to use all of the functions in the Processing Extension API [page 300].

Remarks

In this function, you can apply any logic of your own to manipulate the report using the API. For example, you can add code that will access user, report, or table information, and change prompt values and selection formulas.

6.1.3.1.3 ITerminatePlugin Function

Called when the DLL or shared library is about to be unloaded. The request will carry on as normal and be processed.

Syntax

```
bool STD_CALL ITerminatePlugin(
    void* ptr)
```

Parameters

ptr

Allows you to keep a handle on your own structure between the exported function calls. At this point, this value has been passed from IInitializePlugin and to IProcessRequest.

Remarks

All of your changes to the report should have been made at this point; this is simply an opportunity for cleanup. As a final step, you will want to free the data in memory that is associated with the ptr parameter.

6.1.3.2 IPROCESSEXTRequest Interface

This interface is used to obtain all other supported interfaces. It is the structure that is given to you as a pointer in the request i_reqHandle parameter in the IProcessRequest Function [page 303]. All functions require you to give them the original handle you received in IProcessRequest.

This interface also retrieves information about the type of request that is being processed. Knowing the type of request is useful in deciding how to apply a record selection formula to a report that is being scheduled and/or viewed. See SetSelectionFormula Function [page 329].

Variables and Functions

Variables and Functions	Description
hIRequest Variable [page 305]	The handle to the interface. You need to pass this handle to each function that the interface supports. Read-Only.
GetInterface Function [page 305]	Retrieves one of the interfaces documented in this API reference.
GetRequestType Function [page 306]	Retrieves the type of request that is being processed.

6.1.3.2.1 hIRequest Variable

The handle to the interface. You need to pass this handle to each function that the interface supports. Read-Only.

6.1.3.2.2 GetInterface Function

Retrieves one of the interfaces documented in this API reference. Use either one of the Macros [page 330] (recommended) or this function to gain access to the rest of the API.

Syntax

```
GetInterface(
    IPROCESSEXTREQ_HANDLE i_hIRequest,
    INTERFACE_ID i_id,
    void** o_pIRequest)
```

Parameters

i_hIRequest [In]

The handle you received in IProcessRequest. See hIRequest Variable [page 305].

i_id [In]

The ID of the interface that you are requesting. This ID is defined in the header file.

o_pIRequest [Out]

This gives you back the interface you requested. You must first create a pointer to the type of interface that you want. After you call this function, this pointer will point to the actual interface that you requested.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

Remarks

To get an interface, you can use one of the Macros [page 330]. This is the easiest way to retrieve an interface. For example, the following piece of code retrieves the UserInfo interface using the GET_USERINFO_INTERFACE macro:

```
IUserInfo* pUserInfoInterface = NULL;
GET_USERINFO_INTERFACE(i_reqHandle, pUserInfoInterface)
    unsigned short userNameSize=2;
    Char* userName = new Char[userNameSize];
    ret = pUserInfoInterface->GetUserName(pUserInfoInterface->hIUserInfo,
&userNameSize, userName) ;
```

You can also retrieve the UserInfo interface using the GetInterface function:

```
IUserInfo* pUserInfoInterface = NULL;
i_reqHandle->GetInterface(i_reqHandle, pUserInfoInterface);
     unsigned short userNameSize=2;
    Char* userName = new Char[userNameSize];
    ret = pUserInfoInterface->GetUserName(pUserInfoInterface->hIUserInfo,
&userNameSize, userName) ;
```

6.1.3.2.3 GetRequestType Function

Retrieves the type of request that is being processed.

Syntax

```
GetRequestType(
    IPROCESSEXTREQ_HANDLE i_hIRequest,
    RequestType** o_requestType)
```

Parameters

i_hIRequest [In]

The handle you received in IProcessRequest. See hIRequest Variable [page 305].

o_requestType [Out]

The type of request.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

Remarks

The returned request type may be one of the following.

Constant	Value
view	0
schedule	1
both	2
none	3

6.1.3.3 IReportInfo Interface

This is a read-only interface that retrieves information about the report or subreport that is being processed. To retrieve this interface, use the GetInterface Function [page 305] or the GET_REPORTINFO_INTERFACE Macro [page 330].

Subreports are processed individually from the main report, which means that the processing extension may receive numerous calls for the same report object ID. However, for each call, the report context may change. For example, if the user is viewing the main report, IProcessRequest is called with the report context of the main report. Any requests for the report ID, title, or tables will return the main report's information. If the user drills down on a subreport, IProcessRequest is called with the report context of the subreport, and the subreport's information is returned. Use the GetReportTitle Function [page 308] to determine whether the report being processed is the main report or a subreport.

Variables and Functions

Variables and Functions	Description
hIReportInfo Variable [page 308]	The handle to the interface. You need to pass this handle to each function that the interface supports. Read-Only.
GetReportTitle Function [page 308]	Retrieves the title (or name) of the report or subreport.
GetReportID Function [page 309]	Retrieves the ID of the report that is currently being processed.

6.1.3.3.1 hIReportInfo Variable

The handle to the interface. You need to pass this handle to each function that the interface supports. Read-Only.

6.1.3.3.2 GetReportTitle Function

Retrieves the title (or name) of the report or subreport.

Syntax

```
GetReportID(
    IREPORTINFO_HANDLE i_hIReportInfo,
    long* o_reportId)
```

Parameters

i_hIReportInfo [In]

The handle of the interface. See hIReportInfo Variable [page 308].

io_reportTitleSize [In, Out]

The size of your buffer.

o_reportTitle

The title of the report or subreport. If o_isSubReport is False, the title of the main report is retrieved; if o_isSubReport is True, the title of the subreport is retrieved.

o_isSubReport

Returns True if the report being processed is a subreport, and False if it is the main report. Whether or not the report is a subreport or a main report also determines which selection formula is retrieved by the GetOldSelectionFormula Function [page 328].

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

If the function call fails, check the value of the buffer size parameter. After a successful function call, this parameter contains the size of the buffer that you passed in. However, if the buffer size is too small, then the function returns PROCESSEXTERR_BUFFERSIZETOOSMALL, and the buffer size that is needed is specified in the parameter.

6.1.3.3.3 GetReportID Function

Retrieves the ID of the report that is currently being processed.

Syntax

```
GetReportID(
    IREPORTINFO_HANDLE i_hIReportInfo,
    long* o_reportId)
```

Parameters

i_hIReportInfo [In]

The handle of the interface. See hIReportInfo Variable [page 308].

o_reportId [Out]

The ID of the report or subreport.

Returns

Greater than zero if successful, zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

Remarks

This function retrieves the object ID of a report or subreport. If the intercepted request is a view request, it retrieves the ID of the report or instance that is being viewed. If the intercepted request is a schedule request, the GetReportID function returns the ID of the scheduled object, which is in fact the parent of the generated instance. For example, if you schedule an instance based on ReportA, GetReportID returns the object ID of ReportA. If you schedule an instance based on InstanceB, GetReportID returns the object ID of InstanceB.

6.1.3.4 IUserInfo Interface

This is a read-only interface that retrieves information about the user for whom the report is being processed. To retrieve this interface, use the GetInterface Function [page 305] or the GET_USERINFO_INTERFACE Macro [page 331].

Variables and Functions

Variables and Functions	Description
hIUserInfo Variable [page 310]	The handle to the IUserInfo interface. You need to pass this handle to all functions supported by IUserInfo. Read-Only.
GetUserName Function [page 310]	Retrieves the name of the user for whom the request is being processed.

6.1.3.4.1 hIUserInfo Variable

The handle to the IUserInfo interface. You need to pass this handle to all functions supported by IUserInfo. Read-Only.

6.1.3.4.2 GetUserName Function

Retrieves the name of the user for whom the request is being processed.

Syntax

```
GetUserName(
```

```
    IUSERINFO_HANDLE i_hIUserInfo ,
    unsigned short* io_userNameSize,
    Char* o_userName)
```

Parameters

i_hIUserInfo [In]

The handle to the interface. See hIUserInfo Variable [page 310].

io_userNameSize [In, Out]

The size of your buffer.

o_userName [Out]

The name of the user.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

If the function call fails, check the value of the buffer size parameter. After a successful function call, this parameter contains the size of the buffer that you passed in. However, if the buffer size is too small, then the function returns PROCESSEXTERR_BUFFERSIZETOOSMALL, and the buffer size that is needed is specified in the parameter.

Remarks

During a view request, GetUsername retrieves the name of the user who has logged on to the system and is viewing the report. During a schedule request, this function retrieves the name of the user who owns the instance that is being scheduled, and not the name of the submitter or the report owner.

6.1.3.5 ITableSet Interface

This is a read-only interface that retrieves information about the set of tables used in a particular report. Information about each table—for example the table name and alias name—is stored in an array; this array is considered to be the tableset. To retrieve this interface, use the GetInterface Function [page 305] or the GET_TABLESET_INTERFACE Macro [page 331].

Subreports are processed individually from the main report, which means that the processing extension may receive numerous calls for the same report object ID. However, for each call, the report context may change. For example, if the user is viewing the main report, IProcessRequest is called with the report context of the

main report. Any requests for the report ID, title, or tables will return the main report's information. If the user drills down on a subreport, IProcessRequest is called with the report context of the subreport, and the subreport's information is returned. Use the GetReportTitle Function [page 308] to determine whether the report being processed is the main report or a subreport.

Variables and Functions

Variables and Functions	Description
hITableSet Variable [page 312]	A handle to the ITableSet interface. You need to pass this handle to each function in the interface that you use. Read-Only.
NumOfTables Function [page 312]	Retrieves the number of tables currently being used by the report being processed.
GetTableName Function [page 313]	Retrieves the name of a table currently being used in the report being processed.
GetAliasName Function [page 314]	Retrieves the alias name of a table that is currently being used in the report being processed.
SearchByName Function [page 315]	Uses the table's name to search for a table in the report.
SearchByAlias Function [page 316]	Uses the alias name to search for a table in the report.

6.1.3.5.1 hITableSet Variable

A handle to the ITableSet interface. You need to pass this handle to each function in the interface that you use. Read-Only.

6.1.3.5.2 NumOfTables Function

Retrieves the number of tables currently being used by the report being processed.

Syntax

```
NumOfTables(
    ITABLESET_HANDLE i_hITableSet,
    int* o_iIndex)
```

Parameters

i_hITableSet [In]

The handle to the ITableSet interface. See hITableSet Variable [page 312].

o_iIndex [Out]

The size of the tableset array.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

6.1.3.5.3 GetTableName Function

Retrieves the name of a table currently being used in the report being processed.

Syntax

```
GetTableName(
    ITABLESET_HANDLE i_hITableSet,
    int i_iIndex,
    unsigned short* io_tableNameSize,
    Char* o_tableName)
```

Parameters

i_hITableSet [In]

The handle to the ITableSet interface. See hITableSet Variable [page 312].

i_iIndex [In]

The table's index number.

io_tableNameSize [In, Out]

The size of your buffer.

o_tableName [Out]

The name of the table.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

If the function call fails, check the value of the buffer size parameter. After a successful function call, this parameter contains the size of the buffer that you passed in. However, if the buffer size is too small, then the function returns PROCESSEXTERR_BUFFERSIZETOOSMALL, and the buffer size that is needed is specified in the parameter.

6.1.3.5.4 GetAliasName Function

Retrieves the alias name of a table that is currently being used in the report being processed.

Syntax

```
GetAliasName(
    ITABLESET_HANDLE i_hITableSet,
    int i_iIndex,
    unsigned short* io_aliasNameSize,
    Char* o_aliasName)
```

Parameters

i_hITableSet [In]

The handle to the ITableSet interface. See hITableSet Variable [page 312].

i_iIndex [In]

The table's index number.

io_aliasNameSize [In, Out]

The size of your buffer.

o_aliasName [Out]

The name of the alias.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

If the function call fails, check the value of the buffer size parameter. After a successful function call, this parameter contains the size of the buffer that you passed in. However, if the buffer size is too small, then the function returns PROCESSEXTERR_BUFFERSIZETOOSMALL, and the buffer size that is needed is specified in the parameter.

Remarks

An alias is a string that serves as a unique identifier for a table. The name of the alias does not change, even if you change the name or location of the database. The alias name is especially useful for ensuring that selection formulas can be executed properly. For example, if a formula uses a table that has been renamed or moved, the alias can be used to find the table's new name or location.

6.1.3.5.5 SearchByName Function

Uses the table's name to search for a table in the report.

Syntax

```
SearchByName(
    ITABLESET_HANDLE i_hITableSet,
    ConstChar* i_tableName,
    bool* o_bIsExists,
    int* o_iIndex)
```

Parameters

i_hITableSet [In]

The handle to the ITableSet interface. See hITableSet Variable [page 312].

i_tableName [In]

The name of the table.

o_bIsExists [Out]

Returns True if the table exists, and False otherwise.

o_iIndex [Out]

The table's index number.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

6.1.3.5.6 SearchByAlias Function

Uses the alias name to search for a table in the report.

Syntax

```
SearchByAlias(
    ITABLESET_HANDLE i_hITableSet,
    ConstChar* i_aliasName,
    bool* o_bIsExists,
    int* o_iIndex)
```

Parameters

>

i_hITableSet [In]

The handle to the ITableSet interface. See hITableSet Variable [page 312].

i_aliasName [In]

The table's alias name.

o_bIsExists [Out]

Returns True if the table exists, and False otherwise.

o_iIndex [Out]

The table's index number.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

6.1.3.6 IPromptSet Interface

This interface allows you to change prompt information that has been previously set in the main report and its subreports. Information about each prompt—for example, the prompt name and set of values—is stored in an array on the server; this array is considered to be the prompt set. The prompt set that is retrieved is a union of all report contexts; that is, it contains the prompts from the main report and each subreport.

You can visualize a prompt set in the following way.

<table>
<tr><td rowspan="6">PromptNum 1</td><td rowspan="2">ValueIndex 1</td><td>PromptValue 1</td></tr>
<tr><td>PromptValue 2</td></tr>
<tr><td rowspan="2">ValueIndex 2</td><td>PromptValue 1</td></tr>
<tr><td>PromptValue 2</td></tr>
<tr><td rowspan="2">ValueIndex 3</td><td>PromptValue 1</td></tr>
<tr><td>PromptValue 2</td></tr>
<tr><td rowspan="4">PromptNum 2</td><td rowspan="2">ValueIndex 1</td><td>PromptValue 1</td></tr>
<tr><td>PromptValue 2</td></tr>
<tr><td rowspan="2">ValueIndex 2</td><td>PromptValue 1</td></tr>
<tr><td>PromptValue 2</td></tr>
</table>

A report may have a number of prompts. Each prompt may have one or more prompt values, and each of these prompt values may be discrete or ranged. A discrete value is a single value, and is represented by PromptValue1; a ranged value has two values (an upper bound and a lower bound), which are represented by PromptValue 1 and PromptValue 2 respectively.

Typically, you will want to find the total number of prompts, retrieve the information for each, and then change their values. To retrieve this interface, use the GetInterface Function [page 305] or the GET_PROMPTSET_INTERFACE Macro [page 332].

- The functions in this interface ignore prompts whose values are not needed.
- You can only change existing prompt values. A value cannot be changed to NULL, nor can NULL be changed to a value.

Variables and Functions

Variables and Functions	Description
hIPromptSet Variable [page 318]	A handle to the IPromptSet interface. You need to pass this handle to each function in the interface that you use. Read-Only.
NumOfPrompts Function [page 318]	Retrieves the number of prompts currently being used in the report that is being processed.

Variables and Functions	Description
GetName Function [page 319]	Retrieves the name of a prompt currently being used in the report being processed.
GetSubReportName Function [page 320]	Retrieves the name of the subreport in which a particular prompt is being used.
GetType Function [page 321]	Retrieves the type of a particular prompt.
NumOfPromptValues Function [page 322]	Retrieves the number of available values for a particular prompt.
GetPromptValue Function [page 323]	Retrieves the value of a particular prompt.
SetPromptValue Function [page 324]	Overwrites one of the values for a particular prompt.

6.1.3.6.1 hIPromptSet Variable

A handle to the IPromptSet interface. You need to pass this handle to each function in the interface that you use. Read-Only.

6.1.3.6.2 NumOfPrompts Function

Retrieves the number of prompts currently being used in the report that is being processed.

Syntax

```
NumOfPrompts(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int* o_iIndex)
```

Parameters

i_hIPromptSet [In]

The handle to the IPromptSet interface. See hIPromptSet Variable [page 318].

o_iIndex [Out]

The size of the prompt set array.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

Remarks

This function retrieves the number of prompts that are shared between the main report and its subreports. Unlike other interfaces, the IPromptSet interface does not distinguish between reports and subreports; thus, when calling prompt-related functions, the o_isSubReport parameter from the GetReportTitle Function [page 308] is always False.

6.1.3.6.3 GetName Function

Retrieves the name of a prompt currently being used in the report being processed.

Syntax

```
GetName(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iIndex,
    unsigned short* io_nameSize,
    Char* o_name)
```

Parameters

i_hIPromptSet [In]

The handle to the IPromptSet interface. See hIPromptSet Variable [page 318].

i_iIndex [In]

The prompt's index number.

io_nameSize [In, Out]

The size of your buffer.

o_name [Out]

The name of the prompt.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

If the function call fails, check the value of the buffer size parameter. After a successful function call, this parameter contains the size of the buffer that you passed in. However, if the buffer size is too small, then the function returns PROCESSEXTERR_BUFFERSIZETOOSMALL, and the buffer size that is needed is specified in the parameter.

6.1.3.6.4 GetSubReportName Function

Retrieves the name of the subreport in which a particular prompt is being used.

Syntax

```
GetName(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iIndex,
    unsigned short* io_nameSize,
    Char* o_name)
```

Parameters

i_hIPromptSet [In]

The handle to the IPromptSet interface. See hIPromptSet Variable [page 318].

i_iIndex [In]

The prompt's index number.

io_subreportSize [In, Out]

The size of your buffer.

o_subreportName [Out]

The name of the subreport.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

If the function call fails, check the value of the buffer size parameter. After a successful function call, this parameter contains the size of the buffer that you passed in. However, if the buffer size is too small, then the function returns PROCESSEXTERR_BUFFERSIZETOOSMALL, and the buffer size that is needed is specified in the parameter.

6.1.3.6.5 GetType Function

Retrieves the type of a particular prompt.

Syntax

```
GetName(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iIndex,
    unsigned short* io_nameSize,
    Char* o_name)
```

Parameters

i_hIPromptSet [In]

The handle to the IPromptSet interface. See hIPromptSet Variable [page 318].

i_iIndex [In]

The prompt's index number.

o_type [Out]

The prompt type.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

Remarks

The prompt type may be one of the following.

Constant	Value
PROMPTTYPE_INVALID	0
PROMPTTYPE_NUMBER	1
PROMPTTYPE_CURRENCY	2
PROMPTTYPE_BOOL	3
PROMPTTYPE_DATE	4
PROMPTTYPE_TIME	5
PROMPTTYPE_DATETIME	6
PROMPTTYPE_STRING	7

6.1.3.6.6 NumOfPromptValues Function

Retrieves the number of available values for a particular prompt.

Syntax

```
NumOfPromptValues(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iPromptNum,
    int* o_iValueSize)
```

Parameters

i_hIPromptSet [In]

The handle to the IPromptSet interface. See hIPromptSet Variable [page 318].

i_iPromptNum [In]

The prompt's index number.

o_iValueSize [Out]

The number of values for the prompt.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

6.1.3.6.7 GetPromptValue Function

Retrieves the value of a particular prompt.

Syntax

```
GetPromptValue(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iPromptNum,
    int i_iValueIndex,
    PromptValuesType* o_valueType,
    unsigned short* io_promptValueSize1,
    Char* o_promptValue1,
    unsigned short* io_promptValueSize2,
    Char* o_promptValue2)
```

Parameters

i_hIPromptSet [In]

The handle to the IPromptSet interface. See hIPromptSet Variable [page 318].

i_iPromptNum [In]

The prompt's index number.

i_iValueIndex [In]

The prompt value's index number.

o_valueType [Out]

The prompt value type.

io_promptValueSize1 [In, Out]

The buffer size for the first prompt value.

o_promptValue1 [Out]

The first prompt value.

io_promptValueSize2 [In, Out]

The buffer size for the second prompt value.

o_promptValue2 [Out]

The second prompt value.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

If the function call fails, check the value of the buffer size parameter. After a successful function call, this parameter contains the size of the buffer that you passed in. However, if the buffer size is too small, then the function returns PROCESSEXTERR_BUFFERSIZETOOSMALL, and the buffer size that is needed is specified in the parameter.

Remarks

The prompt value type will be one of the following.

Constant	Value	Explanation
pvalue_single	0	A discrete value.
pvalue_range	1	A ranged value.

If the prompt type is a range, and you do not provide enough space for both values, you must call GetPromptValue twice: once to add a larger value for io_promptValueSize1, and once to add a larger value for io_promptValueSize2.

6.1.3.6.8 SetPromptValue Function

Overwrites one of the values for a particular prompt.

Syntax

```
SetPromptValue(
    IPROMPTSET_HANDLE i_hIPromptSet,
    int i_iPromptNum,
    int i_iValueIndex,
    PromptValuesType i_valueType,
    ConstChar* i_promptValueInfo1,
    ConstChar* i_promptValueInfo2)
```

Parameters

i_hIPromptSet [In]

The handle to the IPromptSet interface. See hIPromptSet Variable [page 318].

i_iPromptNum [In]

The prompt's index number.

i_iValueIndex [In]

The prompt value's index number.

i_valueType [In]

The prompt value type. This should be the same as the current prompt value type; it cannot be changed.

i_promptValueInfo1 [In]

The prompt value to be set in the first buffer.

i_promptValueInfo2 [In]

The prompt value to be set in the second buffer. See GetPromptValue Function [page 323].

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

Remarks

The prompt value type will be one of the following.

Constant	Value	Description
pvalue_single	0	A discrete value.
pvalue_range	1	A ranged value.

If the prompt value type is discrete, only a single prompt value is required as a parameter. If the prompt value type is ranged, two prompt values are required.

> **Note**
>
> You cannot change prompts for an instance unless you refresh the instance. If you want to change prompt values, apply the processing extension to the report before you schedule the report.

6.1.3.7 ISelectionInfoFormula Interface

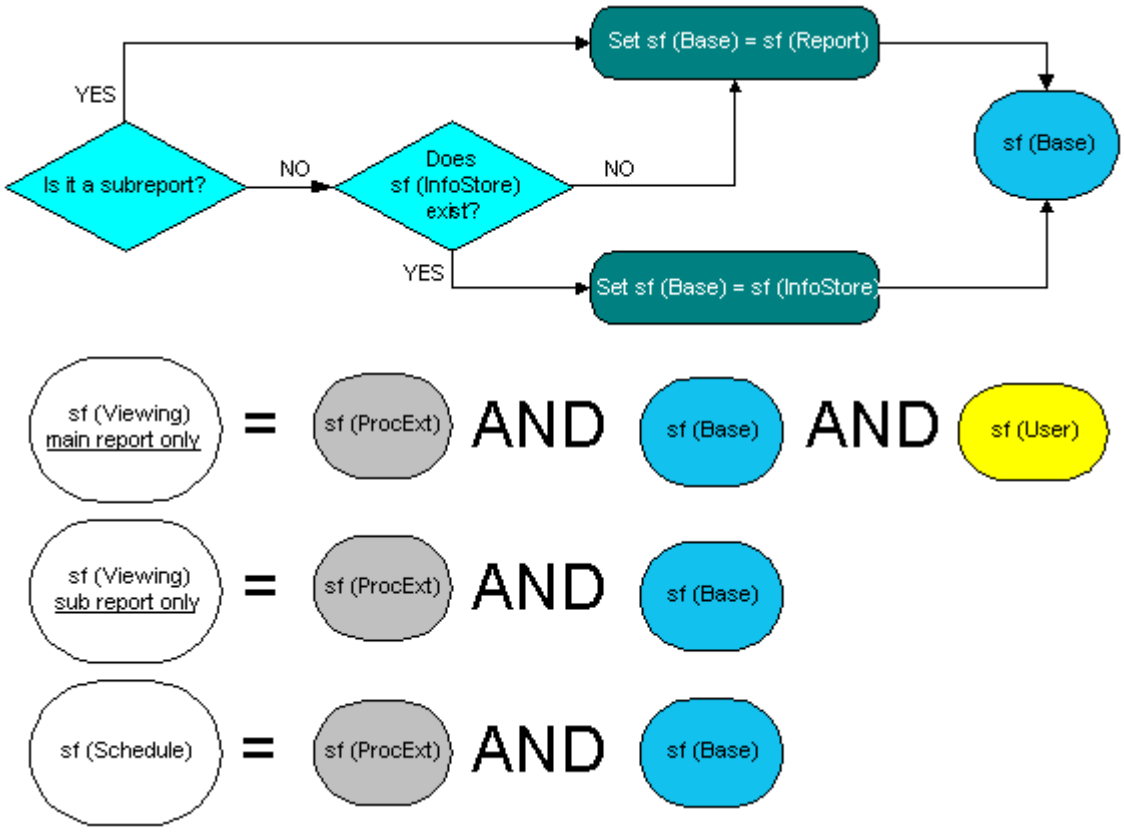
Retrieves information about the current record selection formula and allows you to append additional criteria to it. To retrieve this interface use the GetInterface Function [page 305] or the GET_SELECTIONFORMULA_INTERFACE Macro [page 332].

> **Note**
>
> This interface allows you to change only record selection formulas; it does not apply to group selection formulas.

Depending on whether the report or its subreports are being viewed or scheduled, the current selection formula may be a combination of filter strings that have been set the following ways:

- Saved with the existing Crystal report.
- Specified by the user through the client application.
- Specified by the administrator through the Central Management Console (CMC).

The processing extension can be used to extend the current selection formula. As the following diagram shows, it will "AND" the existing selection formula with the one set by the processing extension.



sf (Viewing):

The selection formula that is generated when a view request is made through the Crystal Reports Cache Server. The sf (User) selection formula is applied only when viewing the main report, and not when viewing subreports.

sf (Schedule):

The selection formula that is generated when a schedule request is made and run on the Job Server.

sf (InfoStore):

The selection formula that is stored in the CMS is accessible to an administrator via the Central Management Console (CMC). This is the selection formula for the main report; selection formulas are not stored for subreports. If a report contains a selection formula, this formula is copied from the report to the CMS when the report is added to the CMS.

sf (Report):

The selection formula that is retrieved from the existing report or subreport. Since the processing extension is called for the main report and each subreport, the selection formula changes depending on the current report context.

sf (User):

The selection formula that is specified by the user's client application. For example, it could be passed from the client to the server as part of the URL: `&sf=<selection formula>`.

sf (ProcExt):

The selection formula that is set by the user-defined processing extension that is associated with the report. If there are multiple processing extensions associated with a report, each one's selection formula is added together for the current report context.

> **Note**
>
> While processing extensions have no effect on the parameters that are contained in report instances, they do have an effect on the selection formulas.

Variables and Functions

Variables and Functions	Description
hISelectionFormula Variable [page 327]	A handle to the ISelectionInfoFormula interface. You need to pass this handle to each function in the interface that you use. Read-Only.
GetOldSelectionFormula Function [page 328]	Retrieves the selection formula currently being used by the report being processed.
SetSelectionFormula Function [page 329]	Sets the report's record selection formula.

6.1.3.7.1 hISelectionFormula Variable

A handle to the ISelectionInfoFormula interface. You need to pass this handle to each function in the interface that you use. Read-Only.

6.1.3.7.2 GetOldSelectionFormula Function

Retrieves the selection formula currently being used by the report being processed.

Syntax

```
GetOldSelectionFormula(
    ISELECTIONFORMULA_HANDLE i_hISelectionFormula,
    int* io_selectionFormulaSize,
    Char* o_selectionFormula)
```

Parameters

i_hISelectionFormula [In]

The handle to the ISelectionInfoFormula interface. See hISelectionFormula Variable [page 327].

io_selectionFormulaSize [In, Out]

The size of your buffer. The size is a combination of the selection formulas that have previously been specified in the report, by the user (if the report is being viewed on demand), or through the Central Management Console (CMC).

o_selectionFormula [Out]

The record selection formula that is currently being used in the report or subreport. This is considered to be the "old" selection formula.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

If the function call fails, check the value of the buffer size parameter. After a successful function call, this parameter contains the size of the buffer that you passed in. However, if the buffer size is too small, then the function returns PROCESSEXTERR_BUFFERSIZETOOSMALL, and the buffer size that is needed is specified in the parameter.

Remarks

What is retrieved by the GetOldSelectionFormula function depends on whether or not the report is the main report or a subreport. If the o_isSubReport parameter (see GetReportTitle Function [page 308]) is False, then

the function retrieves the selection formula that has already been set for the main report. If o_isSubReport is True, then it retrieves the old selection formula for the subreport.

A new selection formula can be specified with the SetSelectionFormula Function [page 329]. This function appends a new filter string to the old selection formula.

6.1.3.7.3 SetSelectionFormula Function

Sets the report's record selection formula.

Syntax

```
SetSelectionFormula(
    ISELECTIONFORMULA_HANDLE i_hISelectionFormula,
    ConstChar* i_selectionFormula)
```

Parameters

i_hISelectionFormula [In]

The handle to the ISelectionInfoFormula interface. See hISelectionFormula Variable [page 327].

i_selectionFormula [In]

The filter string that will be appended to the current selection formula. The server will concatenate the old selection formula with the new filter string that is passed through i_selectionFormula. It will then send this modified selection formula to the report.

Returns

Each function returns a RET_VALUE that is an error code of type Long. The return value is greater than zero if the call was successful, and zero or less if not. Use RET_OK(x) and RET_ERROR(x) to check for failure.

If the function call fails, check the value of the buffer size parameter. After a successful function call, this parameter contains the size of the buffer that you passed in. However, if the buffer size is too small, then the function returns PROCESSEXTERR_BUFFERSIZETOOSMALL, and the buffer size that is needed is specified in the parameter.

Remarks

You may want the record selection to be applied to all requests, or you may want it to be applied only when a report is being scheduled or viewed. Use the GetRequestType Function [page 306] to determine the type of request that is being made.

To ensure row-level security at all times, you must apply the processing extension to both view and schedule requests. For example, if you have an employee profile report that uses a selection formula such as sf = {Employee.Last Name}<"M", and you apply the processing extension when scheduling the report, then the returned data will be the last name up to, but not including, "M". However, if the user refreshes the report, and a processing extension is not applied at view time, then the complete set of data (the last names from "A" to "Z") will be retrieved from the database. In other words, if you allow users to refresh and view a scheduled report that uses a processing extension, then if you want to continue to restrict access to report data, you must remember to use the processing extension on view as well as on schedule.

6.1.3.8 Macros

To retrieve a specific interface, you can use either the IPROCESSEXTRequest interface's GetInterface Function [page 305], or you can use a macro. Using a macro is the easiest way to retrieve an interface. See GetInterface Function [page 305] for an example of both methods.

Macro	Description
GET_REPORTINFO_INTERFACE Macro [page 330]	Retrieves the IReportInfo Interface [page 307].
GET_USERINFO_INTERFACE Macro [page 331]	Retrieves the IUserInfo Interface [page 310].
GET_TABLESET_INTERFACE Macro [page 331]	Retrieves the IUserInfo Interface [page 310].
GET_PROMPTSET_INTERFACE Macro [page 332]	Retrieves the IPromptSet Interface [page 317].
GET_SELECTIONFORMULA_INTERFACE Macro [page 332]	Retrieves the ISelectionInfoFormula Interface [page 326].

6.1.3.8.1 GET_REPORTINFO_INTERFACE Macro

Retrieves the IReportInfo Interface [page 307].

Syntax

```
GET_REPORTINFO_INTERFACE(
    i_pIRequest,
    o_pIReportInfo);
```

Parameters

i_pIRequest [In]

The handle you received in IProcessRequest. See hIRequest Variable [page 305].

o_pIReportInfo [Out]

A pointer to the IReportInfo Interface [page 307].

6.1.3.8.2 GET_USERINFO_INTERFACE Macro

Retrieves the IUserInfo Interface [page 310].

Syntax

```
GET_USERINFO_INTERFACE(
    i_pIRequest,
    o_pIUserInfo);
```

Parameters

i_pIRequest [In]

The handle you received in IProcessRequest. See hIRequest Variable [page 305].

o_pIUserInfo [Out]

A pointer to the IUserInfo Interface [page 310].

6.1.3.8.3 GET_TABLESET_INTERFACE Macro

Retrieves the ITableSet Interface [page 311].

Syntax

```
GET_TABLESET_INTERFACE(
    i_pIRequest,
    o_pITableSet);
```

Parameters

i_pIRequest [In]

The handle you received in IProcessRequest. See hIRequest Variable [page 305].

o_pITableSet [Out]

A pointer to the ITableSet Interface [page 311].

6.1.3.8.4 GET_PROMPTSET_INTERFACE Macro

Retrieves the IPromptSet Interface [page 317].

Syntax

```
GET_PROMPTSET_INTERFACE(
    i_pIRequest,
    o_pIPromptSet);
```

Parameters

i_pIRequest [In]

The handle you received in IProcessRequest. See hIRequest Variable [page 305].

o_pIPrompt [Out]

A pointer to the IPromptSet Interface [page 317].

6.1.3.8.5 GET_SELECTIONFORMULA_INTERFACE Macro

Retrieves the ISelectionInfoFormula Interface [page 326].

Syntax

```
GET_SELECTIONFORMULA_INTERFACE(
    i_pIRequest,
    o_pIUserInfo);
```

Parameters

i_pIRequest [In]

The handle you received in IProcessRequest. See hIRequest Variable [page 305].

o_pIUserInfo [Out]

A pointer to the ISelectionInfoFormula Interface [page 326].

6.1.4 Examples

The following examples demonstrate various aspects of the Processing Extension API.

See Creating an extension [page 298] for further information.

6.1.4.1 Retrieving the request type

```
    // Request Demo
    // Accesses the request handle and retrieves the request type.
    std::cout<<"-------------- Request Demo --------------\n";

    if(! i_reqHandle->GetRequestType(i_reqHandle,&rtype))
    {
        std::cout<<"Error getting RequestType!\n";
        return false;
    }
    else
        std::cout<<"Client: request type="<<rtype<<std::endl;
```

6.1.4.2 Retrieving the report title and ID

```
// Report Info Demo
// Retrieve the report title and report ID. (The ID that identifies the report
inside CMS)
std::cout<<"-------------- Report Info Demo --------------\n";
IReportInfo* pReportInfoInterface = NULL;
if( RET_OK( GET_REPORTINFO_INTERFACE(i_reqHandle, pReportInfoInterface) ) )
{
    // Try to get the report info interface.
    if( pReportInfoInterface == NULL )
    {
        std::cout<<"Error: no ReportInfo interface!\n";
        return false;
    }

    // Retrieve the report ID.
    long reportid=0;
    if(! RET_OK(pReportInfoInterface->GetReportID(pReportInfoInterface-
>hIReportInfo,&reportid)  ))
```

```
    {
        std::cout<<"Error getting the Report ID!\n";
        return false;
    }
    else
        std::cout<<"Client: report id="<<reportid<<std::endl;

    // Retrieve the report title.

    // Because the size of the report title is unknown, try 2 first.
    unsigned short ReportTitleSize=2;
    Char* ReportTitle = new Char[ReportTitleSize];
    // The following is a Boolean flag to indicate whether the report interface
    // is that of a subreport.
    bool* isSubreport = false;
    ret = pReportInfoInterface->GetReportTitle(pReportInfoInterface-
>hIReportInfo, &ReportTitleSize,ReportTitle, isSubreport) ;
    // If 2 is too small, get the real size and call the GetReportTitle function
again.
    if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
    {
        std::cout<<"ReportTitle size too small, should be:"
<<ReportTitleSize<<"\n";
        // Re-create the variable with the actual size.
        delete ReportTitle;
        ReportTitle = new Char[ReportTitleSize];
        // Call the function again.
           if(! RET_OK( pReportInfoInterface-
>GetReportTitle(pReportInfoInterface->hIReportInfo,
&ReportTitleSize,ReportTitle, isSubreport) ))
           {
            std::cout<<"Error getting ReportTitle!"<<ret<<"\n";
           }
        else
           {
            char* myReportTitle = new char[ReportTitleSize];
                 std::cout<<"Client: ReportTitle ="<<w2ac(ReportTitle,
ReportTitleSize, myReportTitle)<<std::endl;

                // An example using the reportTitle:
                // Compare the reportTitle to the title hard-coded in the
tochangeSF
                // function. If a match is found, then change the selection
formula
                // later in the program (see the selection formula demo, below).
                if( tochangeSF(myReportTitle) )
                     changeSF=true;
                delete myReportTitle;
            }
        }
        // If 2 chars is large enough, check if the funtion returns correctly.
        // If it did not return correctly...
        else if (!RET_OK(ret))
        {
            std::cout<<"ERROR at getting ReportTitle!"<<ret<<"\n";
            return false;
        }
        // Otherwise, if it returned correctly...
        else
        {
            char* myReportTitle = new char[ReportTitleSize];
            std::cout<<"Client: ReportTitle ="<<w2ac(ReportTitle,
ReportTitleSize, myReportTitle)<<std::endl;
            if ( tochangeSF(myReportTitle) )
             changeSF=true;
            delete myReportTitle;
        }
        // Clean up.
```

```
        delete ReportTitle;
    }
    // If the reportinfo interface could not be retrieved...
    else
    {
        std::cout<<"Error: get reportinfo interface return false!\n";
        return false;
    }
```

6.1.4.3 Retrieving the user name

```
// UserInfo Demo
// Retrieves the user's name.
std::cout<<"-------------- UserInfo Demo --------------\n";
IUserInfo* pUserInfoInterface = NULL;
if( RET_OK( GET_USERINFO_INTERFACE(i_reqHandle, pUserInfoInterface) ) )
{
    // Test for the UserInfo interface.
    if( pUserInfoInterface == NULL )
    {
        std::cout<<"Error: no userInfo interface!\n";
        return false;
    }
    if( pUserInfoInterface != NULL )
    {
        // The interface is available, so retrieve the user name.
        // The length of the user name is unknown, so try a length of 2 first.
        unsigned short userNameSize=2;
        // A variable for the user name (2 chars in length).
        Char* userName = new Char[userNameSize];
        // Try to retrieve the user name.
        ret = pUserInfoInterface->GetUserName(pUserInfoInterface-
>hIUserInfo,&userNameSize,userName) ;
        // 2 is most likely too short, but the actual size of the user name
        // is returned by the function, so now the real size can be used.
        if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
        {
            std::cout<<"UserName size too small, should be:"
<<userNameSize<<"\n";
            // Re-create the variable.
            delete userName;
            userName = new Char[userNameSize];
            // Retrieve the user name.
            if(! RET_OK( pUserInfoInterface->GetUserName(pUserInfoInterface-
>hIUserInfo,&userNameSize,userName) ))
            {
                std::cout<<"Error getting user name!"<<ret<<"\n";
            }
            else
            {
                // A character buffer for the w2ac function. (See note in
function declaration, below.)
                char* myuserName = new char[userNameSize];
                std::cout<<"Client: userName ="<<w2ac(userName, userNameSize,
myuserName)<<std::endl;
                delete myuserName;
            }
        }
        // 2 chars was big enough. Check if the function succeeded.
        // If the function did not succeed...
        else if (!RET_OK(ret))
        {
            std::cout<<"ERROR getting user name!"<<ret<<"\n";
```

```
            return false;
        }
        // If the function succeeded...
        else
        {
            char* myuserName = new char[userNameSize];
            std::cout<<"Client: userName ="<<w2ac(userName, userNameSize,
myuserName)<<std::endl;
            delete myuserName;
        }
        // Clean up the variable.
        delete userName;
    }
}
// Could not retrieve the user info interface.
else
{
    std::cout<<"Error: get userinfo interface return false!\n";
    return false;
}
```

6.1.4.4 Retrieving the table information for a report

```
    // TableSet Demo:
    // Retrieves the number of tables used in this report
    // as well as the alias name and table name for each table.

    std::cout<<"-------------- TableSet Demo --------------\n";
    ITableSet* pTableSetInterface = NULL;
    if( RET_OK( GET_TABLESET_INTERFACE(i_reqHandle, pTableSetInterface) ) )
    {
        // Retrieve the TableSet interface.
        if( pTableSetInterface == NULL )
        {
            std::cout<<"Error: no TableSet interface!\n";
            return false;
        }
        // If the interface is retrieved...
        if( pTableSetInterface != NULL )
        {
            // Get the number of tables.
            int numOfTables;
            short userid=0;
            if(! RET_OK(pTableSetInterface->NumOfTables(pTableSetInterface-
>hITableSet,&numOfTables)  ))
            {
                std::cout<<"Error getting number of tables!\n";
                return false;
            }
            else
            std::cout<<"Client: num of tables="<<numOfTables<<std::endl;
            // Loop through the tables.
            for(int i=0; i<numOfTables; i++)
            {
                // Retrieve the alias name.
                // Since the length of the alias name is unknown, try a length
                // of 2 first.
                unsigned short aliasNameSize=2;
                Char* aliasName = new Char[aliasNameSize];
                ret = pTableSetInterface->GetAliasName(pTableSetInterface-
>hITableSet,i,&aliasNameSize,aliasName) ;
                // Check if the 2 char buffer is large enough.
                // If the buffer is not large enough...
```

```
                if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                {
                    std::cout<<"aliasName size is too small, should be:"
<<aliasNameSize<<"\n";
                    // Re-create the variable with the real size.
                    delete aliasName;
                    aliasName = new Char[aliasNameSize];
                    // Call GetAliasName again.
                    if(! RET_OK( pTableSetInterface-
>GetAliasName(pTableSetInterface->hITableSet,i,

&aliasNameSize,aliasName) ))
                    {
                        std::cout<<"Error getting aliasName!"<<ret<<"\n";
                    }
                    // Convert the alias name using the w2ac function. (See note
in w2ac function, below.)
                    else
                    {
                        char* myaliasName = new char[aliasNameSize];
                        std::cout<<"Client: aliasName ="<<w2ac(aliasName,
aliasNameSize, myaliasName)<<std::endl;
                        delete myaliasName;
                    }
                }
                // The buffer size is OK. Check if an error was returned.
                // If an error was returned...
                else if (!RET_OK(ret))
                {
                    std::cout<<"Error getting aliasName!"<<ret<<"\n";
                    return false;
                }
                // If no error was returned...
                else
                {
                    char* myaliasName = new char[aliasNameSize];
                    std::cout<<"Client: aliasName ="<<w2ac(aliasName,
aliasNameSize, myaliasName)<<std::endl;
                    delete myaliasName;
                }
                // Clean up.
                delete aliasName;


                // Retrieve the table name.
                // Since the length of the table name is unknown, try a length
                // of 2 first.
                unsigned short tableNameSize=2;
                Char* tableName = new Char[tableNameSize];
                ret = pTableSetInterface->GetTableName(pTableSetInterface-
>hITableSet,i,

&tableNameSize,tableName) ;
                // Chek if 2 chars is big enough.
                // If it is not big enough ...
                if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                {
                    std::cout<<"tableName size is too small, should be:"
<<tableNameSize<<"\n";
                    // Re-create the variable and try again with the
                    // real size.
                    delete tableName;
                    tableName = new Char[tableNameSize];
                    if(! RET_OK( pTableSetInterface-
>GetTableName(pTableSetInterface->hITableSet,i,

&tableNameSize,tableName) ))
                    {
```

```
                        std::cout<<"Error getting tableName!"<<ret<<"\n";
                    }
                    else
                    {
                        char* mytableName = new char[tableNameSize];
                        std::cout<<"Client: tableName ="<<w2ac(tableName,
tableNameSize, mytableName)<<std::endl;
                        delete mytableName;
                    }
                }
                // Check if the buffer size is OK.
                // If an error occurred...
                else if (!RET_OK(ret))
                {
                    std::cout<<"Error getting tableName!"<<ret<<"\n";
                    return false;
                }
                // If no error occurred...
                else
                {
                    char* mytableName = new char[tableNameSize];
                    std::cout<<"Client: tableName ="<<w2ac(tableName,
tableNameSize, mytableName)<<std::endl;
                    delete mytableName;
                }
                // Clean up.
                delete tableName;
            }
        }
    }
    // The tableset interface could not be retrieved.
    else
    {
        std::cout<<"Error: get table set interface return false!\n";
        return false;
    }
```

6.1.4.5 Retrieving prompt information

```
    // Prompt Demo:
    // Retrieve the number of prompts in the report.
    // For each prompt, get its subreport name, prompt type, number of values
for that prompt
    // and what those values are. Note that there will be no subreport name if
it is the main report.
    // Allow the oportunity for the program to translate and reset the values
for a particular prompt.
    std::cout<<"-------------- PromptSet Demo --------------\n";

    // Initiliazing the PromptSet interface.
    IPromptSet* pPromptSetInterface = NULL;
    // Retrieve the PromptSet interface.
    if( RET_OK( GET_PROMPTSET_INTERFACE(i_reqHandle, pPromptSetInterface) ) )
    {
        if( pPromptSetInterface == NULL )
        {
            std::cout<<"Error: No Prompt interface!\n";
            return false;
        }
        // If the promptset interface is retrieved, get the number of prompts.
        if( pPromptSetInterface != NULL )
        {
            int numOfPrompt;
```

```
            if(! RET_OK(pPromptSetInterface->NumOfPrompts(pPromptSetInterface-
>hIPromptSet,&numOfPrompt)  ))
            {
                std::cout<<"Error getting the number of prompts!\n";
                return false;
            }
            else
            std::cout<<"Client: num of Prompts="<<numOfPrompt<<std::endl;
            // Loop through the promptset.
            for(int i=0; i<numOfPrompt; i++)
            {
                // Get prompt name.
                // Since the length of the prompt name is unknown,
                // try 2 first.
                unsigned short nameSize=2;
                Char* name = new Char[nameSize];
                ret = pPromptSetInterface->GetName(pPromptSetInterface-
>hIPromptSet,i,&nameSize, name);
                // If 2 is too short...
                if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                {
                    std::cout<<"Name size is too small, should be:"
<<nameSize<<"\n";
                    // ...recreate the variable and try again with the
                    // real size.
                    delete name;
                    name = new Char[nameSize];
                    if(! RET_OK( pPromptSetInterface-
>GetName(pPromptSetInterface->hIPromptSet,i,
                                                                  &nameSize,
name)))
                    {
                        std::cout<<"Error getting name!"<<ret<<"\n";
                    }
                    else
                    {
                        char* myname = new char[nameSize];
                        std::cout<<"Client: name ="<<w2ac(name, nameSize,
myname)<<std::endl;
                        delete myname;
                    }
                }
                // If a buffer length of 2 is ok, check if there were errors.
                // If there were errors ...
                else if (!RET_OK(ret))
                {
                    std::cout<<"ERROR at getting name!"<<ret<<"\n";
                    return false;
                }
                //  If there were no errors...
                else
                {
                    char* myname = new char[nameSize];
                    std::cout<<"Client: name ="<<w2ac(name, nameSize,
myname)<<std::endl;
                    delete myname;
                }
                // Clean up.
                delete name;

                // Get the sub-reportname of the current prompt.
                // Since the length of the sub-reportname is unknown, try 2
first.
                unsigned short subrptnameSize=2;
                Char* subrptname = new Char[subrptnameSize];
                ret = pPromptSetInterface->GetSubReportName(pPromptSetInterface-
>hIPromptSet,i,
```

```
&subrptnameSize, subrptname);
                // If the buffer size is too small...
                if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                {
                    std::cout<<"Subreport name size is too small, should be:"
<<subrptnameSize<<"\n";
                    // ...recreate the variable and try again with the real size
                    delete subrptname;
                    subrptname = new Char[subrptnameSize];
                    if(! RET_OK( pPromptSetInterface-
>GetSubReportName(pPromptSetInterface->hIPromptSet,i,&subrptnameSize,
subrptname)))
                    {
                        std::cout<<"Error getting the name!"<<ret<<"\n";
                    }
                    else
                    {
                        char* mysubrptname = new char[subrptnameSize];
                        std::cout<<"Client: subrptname ="<<w2ac(subrptname,
subrptnameSize, mysubrptname)<<std::endl;
                        delete mysubrptname;
                    }
                }
                //  If the buffer size is OK...
                else if (!RET_OK(ret))
                {
                    std::cout<<"ERROR at getting subrptname!"<<ret<<"\n";
                    return false;
                }
                //  If the buffer size is not OK...
                else
                {
                    char* mysubrptname = new char[subrptnameSize];
                    std::cout<<"Client: subrptname ="<<w2ac(subrptname,
subrptnameSize, mysubrptname)<<std::endl;
                    delete mysubrptname;
                }
                // Cleanup.
                delete subrptname;

                // Get the number of values.
                int numOfValues;
                 if(! RET_OK(pPromptSetInterface-
>NumOfPromptValues(pPromptSetInterface->hIPromptSet, i, &numOfValues)  ))
                {
                    std::cout<<"Error getting the number of prompt Values!\n";
                    return false;
                }
                else
                    std::cout<<"Client: num of Prompt values
="<<numOfValues<<std::endl;
                // Loop through the prompt set.
                for(int j=0; j<numOfValues; j++)
                {
                    // Get the individual prompt values.
                    // Since the length of the prompt values is unknown, try 10
first.
                    unsigned short valueSize1=10;
                    Char* value1 = new Char[valueSize1];
                    unsigned short valueSize2=10;
                    Char* value2 = new Char[valueSize2];
                    PromptValuesType ptype;
                    ret = pPromptSetInterface-
>GetPromptValue(pPromptSetInterface->hIPromptSet,i, j, &ptype,
                                                                &valueSize1,
value1, &valueSize2, value2);
```

```
                    // The buffer is too small so...
                    if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                    {
                        std::cout<<"Value size is too small, should be:"
<<valueSize1<<","<<valueSize2<<"\n";
                        // ...recreate the variables and try again with the real
size.
                        delete value1;
                        value1 = new Char[valueSize1];
                        delete value2;
                        value2 = new Char[valueSize2];
                        if(! RET_OK( pPromptSetInterface-
>GetPromptValue(pPromptSetInterface->hIPromptSet,i,j, &ptype,
                                                          &valueSize1,
value1,&valueSize2, value2)))
                        {
                            std::cout<<"Error getting value."<<ret<<"\n";
                        }
                        else
                        {
                            char* myvalue1 = new char[valueSize1];

                            std::cout<<"Client: prompt value1 ="<<w2ac(value1,
valueSize1, myvalue1)<<std::endl;
                            delete myvalue1;
                            // If the prompt type is a range, there will
                            // be a second value.
                            if(ptype == pvalue_range)
                            {
                                char* myvalue2 = new char[valueSize2];
                                std::cout<<"Client: prompt value2
="<<w2ac(value2, valueSize2, myvalue2)<<std::endl;
                                delete myvalue2;
                            }

                        }
                    }
                    // If there are any errors...
                    else if (!RET_OK(ret))
                    {
                        std::cout<<"Error getting prompt value."<<ret<<"\n";
                        return false;
                    }
                    // If there are no errors...
                    else
                    {
                        char* myvalue1 = new char[valueSize1];

                        std::cout<<"Client: prompt value1 ="<<w2ac(value1,
valueSize1, myvalue1)<<std::endl;
                        delete myvalue1;
                        // If and only if the prompt type is range, then will
there
                        // be a second value.
                        if(ptype == pvalue_range)
                        {
                            char* myvalue2 = new char[valueSize2];
                            std::cout<<"Client: prompt value2 ="<<w2ac(value2,
valueSize2, myvalue2)<<std::endl;
                            delete myvalue2;
                        }
                    }
                    // Cleanup.
                    delete value1;
                    delete value2;

                    // Reset value for the individual prompt.
                    int vSize=3;
```

```
                    Char* v = new Char[vSize];
                    v[0] = '5';
                    v[1] = '5';
                    v[2] = '\0';
                    if(! RET_OK(pPromptSetInterface-
>SetPromptValue(pPromptSetInterface->hIPromptSet, i,j, ptype,v, v)  ))
                    {
                        std::cout<<"Error setting num of prompt values.\n";
                        return false;
                    }
                    else
                        std::cout<<"changed values!\n";
                    // Cleanup.
                    delete v;
                }
            }
        }
    }
    // Not able to retrieve the interface.
    else
    {
        std::cout<<"Error: Get prompt interface return false!\n";
        return false;
    }
```

6.1.4.6 Modifying a selection formula

```
    //  Selection Formula Demo
    //  In the report info demo above, the report is checked to see if its
    //  name matches the one in the tochangeSF() function below.
    //  If a match of report is found, modify the selection formula as follows:
    //  add the selection formula to the existing selection formula.
    //  That is, after calling
    //  SetSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
mySF),
    //  the selection formula becomes: existingSF "AND" newSF.
    //  Also demonstrates retrieving the existing selection formula
    //  (GetOldSelectionFormula) for reference.
    std::cout<<"-------------- Request Demo --------------\n";

    if(! i_reqHandle->GetRequestType(i_reqHandle,&rtype))
    {
        std::cout<<"Error getting RequestType!\n";
        return false;
    }
    else
        std::cout<<"Client: request type="<<rtype<<std::endl;

    // If this is the required report, get the SelectionFormula interface.
    if(changeSF )
    {
        ISelectionFormula* pSelectionFormulaInterface = NULL;
        // Check to see if the SelectionFormula interface was retrieved.
        if( RET_OK( GET_SELECTIONFORMULA_INTERFACE(i_reqHandle,
pSelectionFormulaInterface) ) )
        {
            if( pSelectionFormulaInterface == NULL )
            {
                std::cout<<"Error: no SelectionFormula interface!\n";
                return false;
            }
            std::cout<<"-------------- sf Demo --------------\n";
            // If the request type equals schedule...
```

```
            if(rtype == schedule)
            {

                if( pSelectionFormulaInterface != NULL )
                {
                    // Create a char* with a selection
                    // condition "{Employee.Last Name}>"H".
                    int strSize=25;
                    Char* str = new Char[strSize];

                    str[0]='{' ;
                    str[1]='E';
                    str[2]='m';
                    str[3]='p';
                    str[4]='l';
                    str[5]='o';
                    str[6]='y';
                    str[7]='e';
                    str[8]='e';
                    str[9]='.';
                    str[10]='L';
                    str[11]='a';
                    str[12]='s';
                    str[13]='t';
                    str[14]=' ';
                    str[15]='N';
                    str[16]='a';
                    str[17]='m';
                    str[18]='e';
                    str[19]='}';
                    str[20]='>';
                    str[21]='"';
                    str[22]='H';
                    str[23]='"';
                    str[24]='\0';

                    // SetSelectionFormula ***APPENDS*** the condition to the
existing SelectionFormula.
                    ret = pSelectionFormulaInterface-
>SetSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
                                                        str) ;
                    // Cleanup.
                    delete str;
                    std::cout<<"sf ret:"<<ret<<std::endl;
                    // Retrieve the existing selection formula.
                    // Since the length of the selection formula is unknown, try
2 first.
                    int oldsfSize=2;
                    Char* oldsf = new Char[oldsfSize];
                    ret = pSelectionFormulaInterface-
>GetOldSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
                                                                  &oldsfSize,
oldsf);

                    // If the buffer is too small...
                    if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                    {
                        std::cout<<"oldsf size too small, should be:"
<<oldsfSize<<"\n";
                        // Re-create the variable and delete oldsf.
                        delete oldsf;
                        oldsf = new Char[oldsfSize];
                        if(! RET_OK( pSelectionFormulaInterface-
>GetOldSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
                                                                 &oldsfSize,
oldsf)))
                        {
                            std::cout<<"Error getting oldsf!"<<ret<<"\n";
```

```
                        }
                        else
                        {
                            char* myoldsf = new char[oldsfSize];
                            std::cout<<"Client: oldsf ="<<w2ac(oldsf, oldsfSize,
myoldsf)<<std::endl;
                            delete myoldsf;
                        }
                    }
                    // Check if the buffer size is OK.
                    // If there were any errors...
                    else if (!RET_OK(ret))
                    {
                        std::cout<<"ERROR at getting oldsf!"<<ret<<"\n";
                        return false;
                    }
                    //  If there were no errors...
                    else
                    {
                        char* myoldsf = new char[oldsfSize];
                        std::cout<<"Client: oldsf ="<<w2ac(oldsf, oldsfSize,
myoldsf)<<std::endl;
                        delete myoldsf;
                    }
                    // Clean up.
                    delete oldsf;
                }
            }
            // If the request type is view...
            if(rtype == view)
            {
                // Set up a select criterion of:
                // {Employee.Last Name} < "M"
                int str2Size=25;
                Char* str2 = new Char[str2Size];

                str2[0]='{' ;
                str2[1]='E';
                str2[2]='m';
                str2[3]='p';
                str2[4]='l';
                str2[5]='o';
                str2[6]='y';
                str2[7]='e';
                str2[8]='e';
                str2[9]='.';
                str2[10]='L';
                str2[11]='a';
                str2[12]='s';
                str2[13]='t';
                str2[14]=' ';
                str2[15]='N';
                str2[16]='a';
                str2[17]='m';
                str2[18]='e';
                str2[19]='}';
                str2[20]='<';
                str2[21]='"';
                str2[22]='M';
                str2[23]='"';
                str2[24]='\0';

                // SetSelectionFormula ***APPENDS*** the condition to the
existing selection formula.
                ret = pSelectionFormulaInterface-
>SetSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
                                                                 str2) ;
```

```
                // Cleanup.
                delete str2;
                std::cout<<"sf2 ret:"<<ret<<std::endl;
                // Retrieve the existing selection formula.
                // Since the length of the selection formula is unknown,
                // try 2 first.
                int oldsf2Size=2;
                Char* oldsf2 = new Char[oldsf2Size];
                ret = pSelectionFormulaInterface-
>GetOldSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,

&oldsf2Size, oldsf2);

                // If the buffer is too small...
                if(ret == PROCESSEXTERR_BUFFERSIZETOOSMALL)
                {
                    std::cout<<"oldsf size too small, should be:"
<<oldsf2Size<<"\n";
                    // ... Re-create the variable with the real size.
                    delete oldsf2;
                    oldsf2 = new Char[oldsf2Size];
                    if(! RET_OK( pSelectionFormulaInterface-
>GetOldSelectionFormula(pSelectionFormulaInterface->hISelectionFormula,
                                                                    &oldsf2Size,
oldsf2)))
                    {
                        std::cout<<"Error getting oldsf2!"<<ret<<"\n";
                    }
                    else
                    {
                        char* myoldsf2 = new char[oldsf2Size];
                        std::cout<<"Client: oldsf2 ="<<w2ac(oldsf2, oldsf2Size,
myoldsf2)<<std::endl;
                        delete myoldsf2;
                    }
                }
                // If the buffer size is OK, check for any errors?
                // If there are errors...
                else if (!RET_OK(ret))
                {
                    std::cout<<"Error getting oldsf2!"<<ret<<"\n";
                    return false;
                }
                // If there are no errors...
                else
                {
                    char* myoldsf2 = new char[oldsf2Size];
                    std::cout<<"Client: oldsf2 ="<<w2ac(oldsf2, oldsf2Size,
myoldsf2)<<std::endl;
                    delete myoldsf2;
                }
                // Cleanup.
                delete oldsf2;
            }
        }
    }
```

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