IBM Cognos Software Development Kit
Version 10.2.1

Mashup Service Developer Guide

Note
Before using this information and the product it supports, read the information in "Notices" on page 261.

Product Information

This document applies to IBM Cognos Software Development Kit Version 10.2.1 and may also apply to subsequent releases.

Contents

Introduction

This document is intended for use with the IBM® Cognos® Mashup Service, which allows you to develop applications that expose IBM Cognos outputs, such as reports and analyses, as Web services (both SOAP and REST).

As a developer, you can use the Mashup Service to create applications that use a structured view of IBM Cognos outputs as input.

This document describes in detail how you can develop applications using the IBM Cognos Mashup Service. It also contains detailed reference information about the Mashup Service.

Audience

To use the Mashup Service Developer Guide effectively, you should be familiar with the following items:

- the IBM Cognos services and outputs you will be using
- Web services such as SOAP, WSIL, WSDL, and REST
- XML and XML schemas
- HTML and the JavaScript scripting language
- programming languages and integrated development environments (IDEs), such as the Java™ programming language and the Eclipse IDE, or the C# programming language and the Microsoft Visual Studio IDE.

Note that although it is possible to combine the use of the IBM Cognos Software Development Kit and the IBM Cognos Mashup Service, the two products are separate, and use of, or knowledge about, the Software Development Kit is not required in order to develop Mashup Service applications.

Finding information

For more information about using this product or for technical assistance, visit the IBM Cognos Customer Center (http://www.ibm.com/software/data/cognos/customercenter). This site provides information on support, professional services, and education.

You can also read PDF versions of the product release notes and installation guides directly from IBM Cognos product disks.

Forward-looking statements

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of features or functionality remain at the sole discretion of IBM.

Samples disclaimer

The Sample Outdoors Company, Great Outdoors Company, GO Sales, any variation of the Sample Outdoors or Great Outdoors names, and Planning Sample depict fictitious business operations with sample data used to develop sample applications for IBM and IBM customers. These fictitious records include sample data for sales transactions, product distribution, finance, and human resources. Any resemblance to actual names, addresses, contact numbers, or transaction values is coincidental. Other sample files may contain fictional data manually or machine generated, factual data compiled from academic or public sources, or data used with permission of the copyright holder, for use as sample data to develop sample applications. Product names referenced may be the trademarks of their respective owners. Unauthorized duplication is prohibited.

Accessibility features

This product does not currently support accessibility features that help users with a physical disability, such as restricted mobility or limited vision, to use this product.

Chapter 1. What's new?

This section contains a list of new, changed, and deprecated features for this and previous releases. It will help you plan your upgrade and application deployment strategies and the training requirements for your users.

To review an up-to-date list of environments supported by IBM Cognos products, including operating systems, patches, browsers, Web servers, directory servers, database servers, and application servers, visit the software environments page (http://www.ibm.com/support/docview.wss?uid=swg27037784).

For information about upgrading IBM Cognos Mashup Service applications created in previous versions of the product, see Chapter 11, "Upgrading Mashup Service applications," on page 91.

New features in version 10.2.1

New features have been added to the IBM Cognos Mashup Service and are described here.

Embedding images in HTML output

You can have HTML output from the IBM Cognos Mashup Service contain image data inline instead of containing URLs to images on the server. For more information, see "Embedding images in HTML output" on page 56

New sample programs

Sample programs have been added.

- The following JavaScript sample programs have been added.

  drillDownFromChart
  This sample program runs a report with a drillable chart and drills down when a specific area of the chart is clicked.

  drillThroughFromChart
  This sample program performs a drill through from one report to another report.

  htmlTreePrompt
  This sample program prompts for tree prompts and then runs a report with the selected prompts.

- The following Java sample program has been added.

  ExpandTreePrompt
  This sample program prompts for tree prompts and then runs a report with the selected prompts.

- The following C# sample program has been added.

  ExpandTreePrompt
  This sample program prompts for tree prompts and then runs a report with the selected prompts.

Deprecated features in version 10.2.1

The following resource type has been deprecated and will be removed in a future release of this product.

The `providerOutput` REST resource type has been deprecated. IBM Cognos Mashup Service applications should use the `outputFormat` REST resource type instead.

New features in version 10.2

New features have been added to the IBM Cognos Mashup Service and are described here.

Using a relative URL for the prompt page response

If the internal dispatcher URI of your IBM Cognos Business Intelligence server is hidden behind a firewall, you can receive the prompt page URL based on the external gateway URI instead. For more information, see "Retrieving a relative prompt page URL" on page 56

New features in version 10.1.1

New features have been added to the IBM Cognos Mashup Service and are described here.

Running reports and retrieving output in IBM Cognos Viewer formats

You can run reports and retrieve outputs in the formats used by IBM Cognos Viewer (such as PDF, CSV, Microsoft Excel).

You can use the outputFormats resource type (REST applications) or the getOutputFormats method (SOAP applications) to retrieve a list of supported output formats for a report. You can then use the outputFormat resource type (REST applications) or the getOutputFormat method (SOAP applications) to run the report and retrieve the output in a specified format.

See "Running reports and retrieving output in IBM Cognos Viewer formats" on page 40 for more information.

New features in version 10.1.0

New features have been added to the IBM Cognos Mashup Service and are described here.

Retrieving report output a page at a time

You can run a report and retrieve the first page of output with the `pagedReportData` (REST applications) and `getPagedReportData` (SOAP applications) methods. You can then use secondary method calls to retrieve additional pages of report output.

Report Caching

After running a report, you can request the same report output again without rerunning the report.

You may also change the output format and get the new output without rerunning the report.

Saving report versions

You can save report versions in the Content Store by using the `saveOutput` option (REST applications) or the `saveOutput` option (SOAP applications). The report is saved in the preferred format as set in IBM Cognos Connection. See "Saving report versions" on page 41 for more information.

Accessing report versions saved in IBM Cognos BI studios

You can save report versions saved in IBM Cognos Business Intelligence studios by using the `providerOutput` REST resource type. See "Accessing report outputs saved by IBM Cognos BI studios" on page 42 for more information.

Prompt description pages in LDX format

The LDX format has been enhanced to include prompt description pages and you can use the `reportPrompts` (REST applications) and `getReportPrompts` (SOAP applications) methods to get the prompt description pages for a report. You See "Sample prompt request page in LDX format" on page 72 for more information.

Increased layout fidelity

The LDX schema now includes layout block (blk), widget (widget), and layout table (tbl) elements that give Mashup Service applications more control over formatting.

Mashup Service applications can use the `includeLayout` (REST applications) and `includeLayout` (SOAP applications) options to control the use of the new layout elements.

New output formats

Three new output formats have been added. They are the DataSet, DataSetAtom, and the Image format. See "Output formats" on page 9 for more information.

Additional XPath support

When using Xpath expressions to filter report output, you can now use the following additional Xpath constructs:

- The XPath child axis is supported. The following examples will filter on all charts or lists:
  - `/document/pages/page/body/item[cht or lst]`
  - `/document/pages/page/body/item[child::cht or child::lst]`
- The `local-name()` function is now supported.

Deprecated features in version 10.1.0

The following method and resource type have been deprecated and will be removed in a future release of this product.

The `getPromptDescription` generic SOAP method and the `promptDescription` REST resource type have been deprecated. Mashup Service applications should use the `getReportPrompts` SOAP method and `reportPrompts` REST resource type instead.

New and changed features in version 8.4.1

The IBM Cognos Mashup Service Developer Guide is now available to all IBM Cognos Software Development Kit customers.

There have been additions and changes to the layout formats and to the schemas. These changes requires modifications to Mashup Service applications created in version 8.4.1 of the product. These modifications are described in "Upgrading to version 8.4.1" on page 92.

Layout format changes

A new element, the `pages` element, is a container for all `pageGroup` and `page` elements. The following LDX code:

```
<document>
  <pageGroup>
    <page>
      ...
    </page>
  </pageGroup>
</document>
```

would be rendered as follows in the current version:

```
<document>
  <pages>
    <pageGroup>
      <pages>
        <page>
      ...
        </page>
      </pages>
    </pageGroup>
  </pages>
</document>
```

RDS schema changes

The following elements have been added:

- `extension`
- `GetReportDataRequest`
- `includePageBreaks`
- `LDXOutput`

The following elements have been removed:

- `ContentOutput`
- `GetReportContentRequest`
- `GetReportFormattedRequest`
- `paged`

The content model of the following element has changed:

- `output`

Layout Data schema changes

The namespace has changed from `http://developer.cognos.com/schemas/rds/contentmodel/1` to `http://www.ibm.com/xmlns/prod/cognos/layoutData/200904`.

The following element has been added:

- `pages`

Chapter 2. Overview of the Mashup Service

The IBM Cognos Mashup Service gives you a simplified programmatic access to IBM Cognos content. This service exposes the application content built with IBM Cognos products (such as reports, analyses, and PowerPlay® reports) as Web services, both SOAP and REST. This allows you to integrate IBM Cognos content into new client environments like mashups, BPM/BPEL workflow processes, desktop widgets, alternate visualizations like third party charting engines and rich Internet applications.

The Mashup Service transforms all IBM Cognos content into a single format called Layout Data (LDX) format. This format allows you to customize the presentation of IBM Cognos content using a simple API. The LDX format captures the logical structure of the content, as well as some formatting information. For example, list grouping, crosstab dimensions, data values and styling information are represented in an LDX instance.

The Mashup Service can transform LDX instances into in a variety of formats, including HTML, HTMLFragment, and JSON, to facilitate the integration of IBM Cognos content into your applications.

The Mashup Service allows you to access existing IBM Cognos content but does not provide any authoring functionality.

The following diagram illustrates how the Mashup Service interfaces link to the underlying IBM Cognos services, or providers; how they move in LDX format through selections, such as XPATH or ObjectID; and how they can be represented, such as in XML, HTML or JSON.

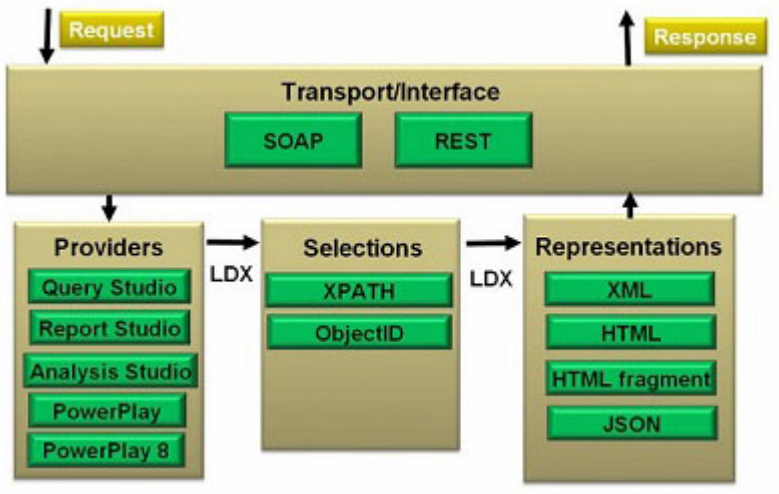

Figure 1. Mashup Service interfaces

Programming interfaces

The Mashup Service offers you the choice of two programming interfaces, REST and SOAP.

REST interface

The representational state transfer (REST) interface is a lightweight interface that use HTTP requests to communicate with the IBM Cognos Business Intelligence server. This interface requires the consuming application to understand the response and has a low overhead when dealing with large amounts of data.

Mashup Service REST requests use the following syntax:

```
http://webservername:portnumber/ibmcognos/cgi-bin/cognos.cgi/rds/
resource_type/source_type/source_id?option1=val1&option2=val2...
```

For example, the following request retrieves the output for the report with storeID `icb01bd1241024cc5bf2086fb10cb40d2` in LDX format:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData
/report/icb01bd1241024cc5bf2086fb10cb40d2
```

SOAP interface

The simple object access protocol (SOAP) interface provides an interface in which SOAP messages are used to pass requests and responses between the client application and the IBM Cognos BI server. This interface provides an object-oriented model that is easily integrated into Java™ or .NET applications. The SOAP interface has an overhead cost that makes it unsuitable for applications that work with large amounts of data. Use of the SOAP interface requires the use of a toolkit that can consume "document/literal" WSDL files and create methods and classes, such as the Eclipse IDE or Microsoft Visual Studio.

You can use the SOAP interface to create generic applications that work with any report or report-specific applications that use a simplified "per-report" WSDL file that makes writing an application to work with a specific report easier. To create a SOAP application that works with any report, import the generic WSDL file into your toolkit using the following URL:

```
http://webservername:portnumber/ibmcognos/cgi-bin/cognos.cgi/rds/wsdl
```

Identifying reports

You can specify reports to run in three different ways with the Mashup Service, by using the report path, storeID, or searchPath. The storeID identifier for the same report will differ in different IBM Cognos Business Intelligence server installations. However, the storeID remains unchanged if the report is moved within a content store.

You can manually find report identifiers using IBM Cognos Connection or programmatically inside an application. See “Finding reports” on page 35 for more information.

path

The path identifier represent a report by its simplified path. A path can have as its root:

- `Public%20Folders` for objects contained in "Public Folders"
- a tilde (~) for the "My Folders" of the current user

- `CAMID(user)` for the "My Folders" of another user

A sample path identifier is shown here.

```
Public%20Folders/Samples/Models/GO%20Sales%20(query)/Report%20Studio%20Report%20Samples
/Order%20Invoices%20-%20Donald%20Chow%2c%20Sales%20Person
```

storeID

The path identifier represent a report by its storeID.

A sample storeID identifier is shown here.

```
i0E130B9A0A21463582535CF2D47B45F8
```

searchPath

The path identifier represent a report by its search path.

A sample search path identifier is shown here.

```
/content/folder[@name='Samples']/folder[@name='Models']/package[@name='GO
Sales (query)']/folder[@name='Report Studio Report Samples']/report[@name='Order
Invoices - Donald Chow, Sales Person']
```

Output formats

You can choose to retrieve report information through the IBM Cognos Mashup Service in a number of different formats, giving you maximum flexibility to choose the type of output you need, depending on how you are going to process the data.

When writing applications that consume these formats, you should not assume that the outputs produced by the Mashup Service will not change over time. Although the outputs will always conform to the specifications described, the actual outputs returned may vary with new releases of this product.

Layout Data (LDX)

Layout Data (LDX) is an XML format, based on the Layout Data Schema Reference, that is an abstraction of rendered IBM Cognos content. LDX is a suitable output if you are writing a generic application for use with any IBM Cognos resource.

See Chapter 7, “Understanding the Layout Data format,” on page 59 for a detailed description of this format.

Simple format

Simple format is an XML format that is less complex than the LDX format. This format is suitable for use when you are writing a report-specific application.

See Chapter 8, “Understanding the Simple format,” on page 75 for a detailed description of this format.

HTML

Requesting a report output in HTML format returns a complete Web page containing the requested output. An inline stylesheet in the head element contains style information. Use this format when you want to display the output as a Web page without any additional processing.

The following features of a report rendered in IBM Cognos Viewer are not supported by the Mashup Service HTML output:

- drill up and down links
- tooltips

HTML fragment

Requesting a report output in HTML Fragment format returns a portion of a Web page containing the requested output. Style information is included within the elements in the fragment. Use this format when you want to add the output to a Web page without any additional processing.

The report output is returned as a `div` root element. This element contains child `div` elements that contain the report parts as HTML.

The limitation mentioned above regarding HTML output also apply to this form of output.

JavaScript Object Notation

JavaScript Object Notation (JSON) is a lightweight data interchange notation that is easy for programs to parse. It is suitable for use in programming environments where XML processing is not convenient.

The JSON output contains the same the information as the LDX output.

DataSet

DataSet format is a simplified XML format that contains only the report data without any formatting information.

See Chapter 9, "Understanding the DataSet format," on page 83 for a detailed description of this format.

DataSetAtom

DataSetAtom format consists of DataSet output in an ATOM wrapper.

Image

Image format returns the report output as a portable network graphic (.png) image.

Sample programs

The IBM Cognos Mashup Service includes sample programs that illustrate how to use the SOAP and REST interfaces to develop mashup applications. There are 3 sets of code samples:

- Java samples that illustrate the SOAP interface using the Java™ programming language.
- C# samples that illustrate the SOAP interface using the C# programming language.
- JavaScript samples that illustrate the REST interface.

These samples use the Mashup Service to get report outputs from the IBM Cognos Business Intelligence samples. You can use them as learning tools or as examples to help you develop your own applications. See Chapter 3, "Cognos Mashup Service samples," on page 13 for more information on the samples.

Chapter 3. Cognos Mashup Service samples

The IBM Cognos Mashup Service includes code samples that illustrate how to use the SOAP and REST interfaces to develop mashup applications. There are 3 sets of code samples:

- Java samples that illustrate the SOAP interface using the Java programming language. For more information on Java samples, see Java samples.
- C# samples that illustrate the SOAP interface using the C# programming language. For more information on C# samples, see page 85.
- JavaScript samples that illustrate the REST interface. For more information, see JavaScript samples on page 87.

These samples use the Mashup Service to get report outputs from the IBM Cognos Business Intelligence samples. You can use them as learning tools or as examples to help you develop your own applications.

In order to run these samples, you must have installed the Great Outdoors Company sample databases and imported the sample packages from the sample deployment archive.

Java samples

The IBM Cognos Mashup Service includes Java program samples that show you some types of applications you can design. The samples include source files so that you can test making changes to the sample code, and batch files or shell scripts for compiling and running the samples.

The source files contain comments that describe the main purpose of each sample. The batch files and shell scripts contain instructions that you must follow before you run them. Each program sample also has an HTML page that explains the purpose of the sample, and provides instructions on how to run the sample.

Java sample file locations

The program samples are installed in folders under the *installation_location*/sdk/cms_samples/java folder. The contents of each folder are described here.

Authentication
: This sample program lets you pass the user credentials to the IBM Cognos server, retrieve the given report output in LayoutDataXML format, and display and save report output.

AuthenticationPrompt
: This sample program runs a report for a given search path and saves the layoutDataXML output to a file.

common_class
: This folder contains the class libraries generated by the Web services. These class libraries are used by all of the Java sample programs.

ExecReports
: This sample program runs a report and outputs the first page of report output in HTML format. You can then use **First Page**, **Previous Page**, **Next Page**, and **Last Page** buttons to retrieve subsequent output pages.

ExpandTreePrompt
This sample program prompts for tree prompts for the **Tree prmopt sample** report and then runs the report.

PromptAnswers
This sample program runs the **Returns by Order Method - Prompted Chart** report using the Simple format and specifies values to satisfy the report prompts. It then outputs the value of the largest return quantity for Defective products for the Web order method that was specified by the prompt.

SearchPromptValue
This sample program illustrates the use of Search & Select prompts using the **Search Prompt Product** report.

SingleReportPart
This sample program runs a report and returns a single report part.

Running the Java samples

You can run the Java sample programs from a command line or in the Eclipse IDE

Each sample subdirectory contains the following files:

- A build.bat file that builds the Java sample on Windows operating systems.
- A run.bat file to run the Java samples on 32-bit Windows installations.
- A run64.bat file to run the Java samples on 64-bit Windows installations.
- A build.sh file that builds the Java sample on UNIX or Linux operating systems.
- A run.sh file to run the Java sample on UNIX or Linux operating systems.
- A <*sample_name*>_Explain.html file that describes the sample and lists any prerequisites for running it.
- One or more .java source files.
- .class files corresponding to each .java source file.

In addition, the Java directory contains the following files:

- A build-samples.bat file that allows you to build all the samples at once on Windows operating systems.
- A build-samples.sh file that allows you to build all the samples at once on UNIX or Linux operating systems.
- A JavaSamples.html file that lists all the samples and links to the individual description .html files.

You can run the Java sample programs from the command line or by using the Eclipse IDE, as shown here.

Running the Java samples on Windows operating systems

1. Ensure that a Java Development Kit, version 1.5 or higher, is installed.
2. Modify the .bat or .bat64 files so the JAVA_HOME variable points to the JDK location.
3. Run build-samples.bat to build all the samples or an individual build.bat to build a single sample.
4. Read the <*sample_name*>_Explain.html to get the instructions for running an individual sample.

   Note that some samples require anonymous access to the IBM Cognos server, while other samples can be used to test authenticated access.

5. Run run.bat (for 32-bit installations) or run64.bat (for 64-bit installations) for the sample you want to try.

Running the Java samples on UNIX or Linux operating systems

1. Ensure that a Java Development Kit, version 1.5 or higher, is installed.
2. Modify the .sh files so the JAVA_HOME variable points to the JDK location.
3. Run build-samples.sh to build all the samples or an individual build.sh to build a single sample.
4. Read the <*sample_name*>_Explain.html to get the instructions for running an individual sample.

   Note that some samples require anonymous access to the IBM Cognos server, while other samples can be used to test authenticated access.
5. Run run.sh for the sample you want to try.

Running the Java samples on the Eclipse IDE

1. Create a project in the Eclipse IDE with the *installation_location*/sdk/cms_samples/java folder as the source.
2. Add the .jar files referenced in the build.bat files to the build path.
3. Read the <*sample_name*>_Explain.html to get the instructions for running an individual sample.

   Note that some samples require anonymous access to the IBM Cognos server, while other samples can be used to test authenticated access.
4. Run the sample program from within the Eclipse IDE.

C# samples

The IBM Cognos Mashup Service includes C# program samples that show you some types of applications you can design. The samples include source files so that you can test making changes to the sample code, and batch files or shell scripts for compiling the samples.

The source files contain comments that describe the main purpose of each sample. The batch files and shell scripts contain instructions that you must follow before you run them. Each program sample also has an HTML page that explains the purpose of the sample, and provides instructions on how to run the sample.

C# sample file locations

The sample files are installed in folders under the *installation_location*/sdk/cms_samples/csharp folder. The contents of each folder are described here.

Authentication
: This sample program lets you pass the user credentials to the IBM Cognos server, retrieve the given report output in LayoutDataXML format, and display and save report output.

AuthenticationPrompt
: This sample program runs a report for a given search path and saves the layoutDataXML output to a file.

bin This folder contains executable versions of all of the C# sample programs.

CMSCommon
: This folder contains files common to all of the C# sample programs. It also contains the Web references generated from the Web services.

ExecuteReports
This sample program runs a report and outputs the first page of report output in HTML format. You can then use **First Page**, **Previous Page**, **Next Page**, and **Last Page** buttons to retrieve subsequent output pages.

ExpandTreePrompt
This sample program prompts for tree prompts for the **Tree prmopt sample** report and then runs the report.

PromptAnswers
This sample program runs a report using the report-specific interface with defined prompt values

SearchPromptValue
This sample program illustrates the use of Search & Select prompts using the **Search Prompt Product** report.

SingleReportPartFetch
This sample program runs a report and returns a single report part.

Running the C# samples

You can run the C# sample programs from a command line or in Microsoft Visual Studio or the Microsoft Visual C# IDE

Each sample subdirectory contains the following files:

- A build.bat file that builds the C# sample.
- A <*sample_name*>_Explain.html file that describes the sample and lists any prerequisites for running it.
- A <*sample_name*>_.csproj Microsoft Visual Studio project file.
- One or more source files.

In addition, the csharp directory contains the following files:

- A CMS_Samples.sln Microsoft Visual Studio solution file.
- A CSharpSamples.html file that lists all the samples and links to the individual description .html files.

To run the C# samples, you can run the executable versions of the sample programs from the bin folder or, if you want to examine the sample programs in more detail, perform the following.

Procedure

1. Ensure that the Microsoft Visual Studio or Microsoft Visual C# IDE, version 2005 or later, is installed.
2. Open CMS_Samples.sln in the Microsoft Visual Studio or Microsoft Visual C# IDE.
3. Read the <*sample_name*>_Explain.html to get the instructions for running an individual sample.

   Note that some samples require anonymous access to the IBM Cognos server, while other samples can be used to test authenticated access.
4. Run the sample in the Microsoft Visual Studio or Microsoft Visual C# IDE.

JavaScript samples

The IBM Cognos Mashup Service includes JavaScript program samples that show you some types of applications you can design. The samples include source files so that you can test making changes to the sample code.

The source files contain comments that describe the main purpose of each sample. Each program sample also has an HTML page that explains the purpose of the sample, and provides instructions on how to run the sample.

JavaScript sample file locations

The sample files are installed in subdirectories under the *installation_location*/webcontent/samples/sdk/cms directory. The contents of each folder are described here.

atom This sample program explores the Mashup Service atom feed for a report.

authentication
This sample program displays an HTML Fragment of a given report by passing the user credentials to the IBM Cognos Business Intelligence server.

cmsExplorer
This sample program traverses the Content Store and provides the URL to link to specific report parts.

common
This folder contains files common to all of the JavaScript sample programs.

drillDown
This sample program runs a report and drills down.

drillDownFromChart
This sample program runs a report with a drillable chart and drills down when a specific area of the chart is clicked.

drillThrough
This sample program performs a drill through using the Mashup Service.

drillThroughFromChart
This sample program performs a drill through from one report to another report.

execReportPart
This sample program displays an HTML Fragment of a report part and retrieves report outputs one page at a time.

getSavedReport
This sample program displays an HTML Fragment of a saved report.

htmlAuthenticationPrompt
This sample program retrieve an HTML fragment for a report part, and shows how to open the standard IBM Cognos BI logon/logoff pages in a separate window.

htmlPromptValue
This sample program displays an HTML Fragment of a given report and prompts the user using HTML prompting if required.

htmlTreePrompt
This sample program prompts for tree prompts and then runs the report with the selected prompts.

json This sample program runs a report in JSON format.

xpath This sample program retrieves a piece of the report using an XPath expression.

Running the JavaScript samples

You can run the JavaScript sample programs in a Web browser.

Each sample subdirectory contains the following files:

- A <*sample_name*>_Explain.html file that describes the sample and lists any prerequisites for running it. The file also porvides a link to run the sample.
- One or more HTML source files.

In addition, the cms directory contains a JavaScriptSamples.html file that lists all the samples and links to the individual description .html files. All the JavaScipt samples can be run from the links in this file. It can be accessed using the following URL:

```
http://webservername:portnumber/ibmcognos/samples/sdk/cms/
JavaScriptSamples.html
```

Chapter 4. Developing Mashup Service applications using the REST interface

You can develop IBM Cognos Mashup Service applications that use the representational state transfer (REST) interface. REST is a lightweight interface that uses HTTP requests to communicate with the IBM Cognos Business Intelligence server. This interface has a low overhead when dealing with large amounts of data.

The REST interface uses HTTP requests to communicate with the IBM Cognos BI server. The sample REST programs included with the Mashup Service use the XMLHttpRequest API to implement this communication. This API allows JavaScript programs to send HTTP requests directly to a server and to load the server responses into JavaScript objects. For simplicity, the REST code in this guide use HTML form GET action URLs to communicate with the BI server.

Creating a Mashup Service application includes the following steps common to all applications.

- Logging on and logging off
- Running Mashup Service methods
- Retrieving report data

Additional topics are covered in Chapter 6, "Performing additional tasks using the Mashup Service," on page 35.

Logging on and logging off

If your IBM Cognos Business Intelligence server requires authentication, you must log on before accessing report outputs. To log on, use either the standard IBM Cognos BI logon page or the authentication methods included with the Mashup Service.

Logging on using the standard IBM Cognos BI logon page

To log on using the standard IBM Cognos Business Intelligence logon page, submit the following URL:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi?
b_action=xts.run&m=portal/close.xts&h_CAM_action=logonAs
```

The BI server returns a logon page that closes automatically after you enter your logon credentials.

The htmlAuthenticationPrompt JavaScript sample uses this authentication method.

Logging on using the Mashup Service authentication methods

You can use the IBM Cognos Mashup Service authentication methods to logon.

Important: If you use a custom authentication provider to handle authentication, the provider must be able to use form fields to authenticate users. For more information, see the topics on authentication request flow scenarios and the JDBCSample sample program in the *IBM Cognos Custom Authentication Provider Developer Guide*.

Use the auth/logon resource type, submitting a credentials element with the xmlData option.

You can determine which credentials the server requires by submitting an empty credentials element:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/auth/logon?
xmlData=<credentials/>
```

The server response is a credentialPrompt element that lists actualValue elements for which the server has values and missingValue elements whose values must be supplied.

You can then submit an auth/logon request with a credentials element that contains values for the missingValue elements.

If your logon attempt is successful, the server sends a response containing an accountInfo element.

If your logon request contains incorrect data, or still has missing values, the server response is another credentialPrompt element.

The following JavaScript code snippet illustrates how you can code a logon request.

To see this code in context, view the following sample:

installation_location/webcontent/samples/sdk/cms/javascript/authentication/reportOutput.html

```
function doLogon()
{
  var myNameSpace=document.getElementById("nameSpace").value;
    var myUserName=document.getElementById("userName").value;
    var myPassword=document.getElementById("password").value;

    var xmlData =
      "xmlData=<credentials>"
      +"<credentialElements><name>CAMNamespace</name><label>Namespace:</label>"
      +"<value><actualValue>"+myNameSpace+"</actualValue></value>"
      +"</credentialElements><credentialElements><name>CAMUsername</name><label>User ID:</label>"
      +"<value><actualValue>"+myUserName+"</actualValue></value>"
      +"</credentialElements><credentialElements><name>CAMPassword</name><label>Password:</label>"
      +"<value><actualValue>"+myPassword+"</actualValue></value>"
      +"</credentialElements></credentials>";

  try
  {
    objXHR.open("POST", parent.settings.document.getElementById("serverURL").value +
"/rds/auth/logon", false);
    objXHR.send(xmlData);
    checkLoginStatus();

  }
  catch (e)
  {
    alert("Error occurs when doing logon.\r\n"+e);
  }

}
```

Logging off using the standard IBM Cognos BI user interface

To log off from the BI server, submit the following URL:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi?b_action=xts.run&m=portal/logoff.xts
```

Logging off using the Mashup Service authentication methods

You can use the Mashup Service authentication methods to logoff.

Use the auth/logoff resource type as shown in the example here.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/auth/logoff
```

Running Mashup Service methods

The REST interface syntax for initial requests is

http://*webservername:portnumber*/ibmcognos/cgi-bin/cognos.cgi/rds/*resource_type/source_type/source_id?option1=val1&option2=val2...*

Some Mashup Service tasks require several interactive steps to complete. Examples include retrieving report output one page at a time, collecting report prompts, and drilling up or down in a report. In these cases, the initial request has the syntax displayed above. Secondary requests have the following syntax:

http://*webservername:portnumber*/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput/conversationID/*conv_ID/secondary_request?option1=val1&option2=val2...*

For more information about the resource types, source types, options, and secondary requests, see Chapter 13, "REST interface reference," on page 115.

Running asynchronous versus synchronous requests

Reports can be run synchronously or asynchronously. Asynchronous execution is the preferred, default method, because it significantly improves scalability.

The value of the async option. determines how the report is run. Some operations, such as Drilling up and down in reports, are only supported with the asynchronous interface.

To run a report asynchronously without manual redirection by the Web client, use the following syntax:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/report
/i915943B5003541778F660265BC0CF286?async=AUTO
```

The Web server returns the http redirect response code 303 and a redirect URL. If the Web client follows redirects, this process continues automatically until the report output is displayed. This is the default behavior if the async option is omitted.

To run a report asynchronously with manual redirection by the Web client, use the following syntax:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/report
/i915943B5003541778F660265BC0CF286?async=MANUAL
```

The Web server returns the http response code 202 and a response that includes a redirect URL. If this redirect URL is followed, then eventually the report output is displayed. The exact format of the response depends on the Web server that is running the IBM Cognos Business Intelligence server.

To run a report synchronously, use the following syntax:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/report
/i915943B5003541778F660265BC0CF286?async=OFF
```

The response from the server is the report in the requested format.

Secondary operations

For some resource calls, secondary calls can be used to retrieve additional output after the initial call has completed.

For example, after using `pagedReportData` to get the first page of the report output, the `next` secondary method can be called to get the next page of report output. To make a secondary resource call, use the URL that accompanies the response to the initial call, append the secondary request name along with any options, and submit this URL to the BI server. For example, the response to a request for the first page of report output could include the following URL:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput
/conversationID/ia1204bcaaa004c64b74921108f07c227
```

Submitting the following URL will retrieve the next page of report output:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput
/conversationID/ia1204bcaaa004c64b74921108f07c227/next
```

Retrieving report data

You can retrieve report output or parts of it.

The `reportData` resource type retrieves the entire report output while the `pagedReportData` resource type retrieves report output one page at a time. For more information, see Obtaining paged report outputs.

Obtaining report outputs in different formats

You can choose output formats other than LDX.

Use the `fmt` option to specify output formats. See "Output formats" on page 9 for descriptions of the possible output formats.

When using the `Image` output format, use the `height` and `width` options to specify the size of the graphic returned.

Obtaining paged report outputs

Use the `pagedReportData` resource type to retrieve report outputs one page at a time. This output is similar to the paged HTML output from IBM Cognos Viewer.

The following example uses the sample report **Public Folders > Samples > Models > GO Data Warehouse (query) > Report Studio Report Samples > Bursted Sales Performance Report** which has 16 pages of output when viewed with IBM Cognos Viewer. Running the following report:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData
/report/i475989eebe9e44dba3409afe33d72d62
```

returns the data contained in the first page of the report, when viewed in IBM Cognos Viewer.

You can now retrieve additional pages of report output by submitting the `next` secondary request. See "Secondary operations" for more information. Repeat this process to retrieve subsequent pages of report output.

You can also use `previous` to retrieve the previous page of report output, `first` to retrieve the first page of output, and `last` to retrieve the last page of report output. If you are retrieving the report in LDX format, the `secondaryOperations` element in the report output indicates which secondary requests are available. HTML and HTMLFragment outputs also indicate available secondary requests in a `div` element.

Chapter 5. Developing Mashup Service applications using the SOAP interface

You can develop IBM Cognos Mashup Service applications that use a simple object access protocol (SOAP) interface in which SOAP messages are used to transmit requests and responses between client applications and the IBM Cognos Business Intelligence server. This interface provides an object-oriented model that is easily integrated into Java™ or .NET applications.

Creating a Mashup Service application includes the following steps common to all applications.

- Setting up the integrated development environment (IDE)
- Logging on and logging off
- Creating a report service instance
- Running Mashup Service methods
- Retrieving report data
- Handling exceptions

Additional topics are covered in Chapter 6, "Performing additional tasks using the Mashup Service," on page 35.

Generic versus report-specific applications

When developing applications using the SOAP interface, you can choose between a generic set of methods and classes that can be used with any IBM Cognos Business intelligence report, or a set of methods and classes that are specific to a particular report.

The generic methods are suitable when you use the LDX report format (see Chapter 7, "Understanding the Layout Data format," on page 59) while the report-specific methods are suitable when you use the Simple report format (see Chapter 8, "Understanding the Simple format," on page 75). One disadvantage of using the LDX report format is that the highly nested structure of LDX documents requires complex commands to access report data. However, the report-specific methods provide shortcuts to accessing report data, which results in less complex applications. For more information, see "Retrieving report data" on page 30.

Setting up the integrated development environment (IDE)

After creating a new project in the Eclipse IDE or in Microsoft Visual Studio, add a Web Service Client (Eclipse IDE) or a Web reference (Microsoft Visual Studio).

To create a generic application, access the generic service WSDL file with the following URL:

```
http://webservername:portnumber/ibmcognos/cgi-bin/cognos.cgi/rds/wsdl
```

To create a report-specific application, instead of importing the generic service WSDL file, access the report-specific service WSDL file with the following URL:

```
http://webservername:portnumber/ibmcognos/cgi-bin/cognos.cgi/rds
/wsdl/source_type/source_id
```

source_type is either `path`, `report`, or `searchPath`, and is used to identify the report to use. The IDE consumes the WSDL file and create the Mashup Service-specific methods and classes for your application.

Logging on and logging off

If your IBM Cognos Business Intelligence server requires authentication, import the authentication methods and classes using the following URL:

`http://webservername:portnumber/ibmcognos/cgi-bin/cognos.cgi/rds/auth/wsdl`

Use the `logon` method to log on to the Business Intelligence server using the Mashup Service authentication methods.

You can determine which credentials the server requires by submitting an empty `credentials` element.

The server response includes a `credentialPrompt` element that lists `actualValue` elements for which the server has values and `missingValue` elements whose values must be supplied.

You can then submit another logon request with a `credentials` element that contains values for the `missingValue` elements.

If your logon attempt is successful, the server response includes an `accountInfo` element that provides authentication details.

If your log on request contains incorrect data, or still has missing values, the server response is another `credentialPrompt` element.

C# example

To see this code in context, view the following sample program:

installation_location/sdk/cms_samples/csharp/Authentication/genericAuthentication.cs

```
AuthService authService = new AuthService();
authService.Url = url;

LogonRequestType authRequest = new LogonRequestType();
authRequest.credentials = new CredentialType();
authRequest.credentials.credentialElements = new CredentialElementType[3];

authRequest.credentials.credentialElements[0] = new CredentialElementType();
authRequest.credentials.credentialElements[0].name = "CAMNamespace";
authRequest.credentials.credentialElements[0].value = new ValueElementType();
authRequest.credentials.credentialElements[0].value.Item = nameSpace;

authRequest.credentials.credentialElements[1] = new CredentialElementType();
authRequest.credentials.credentialElements[1].name = "CAMUsername";
authRequest.credentials.credentialElements[1].value = new ValueElementType();
authRequest.credentials.credentialElements[1].value.Item = userName;

authRequest.credentials.credentialElements[2] = new CredentialElementType();
authRequest.credentials.credentialElements[2].name = "CAMPassword";
authRequest.credentials.credentialElements[2].value = new ValueElementType();
authRequest.credentials.credentialElements[2].value.Item = passWord;

LogonResponseType authResp = authService.logon(authRequest);
```

Java example

To see this code in context, view the following sample program:

installation_location/sdk/cms_samples/java/Authentication/genericAuthentication.java

```
String nameSpaceStr=strNameSpace;    //namespace
String userNameStr=strUserName;      //username
String userPasswordStr=strPassword;  //password
String reportIDStr=strReportID;      //reportID

AuthServiceLocator authlocator = new AuthServiceLocator();
AuthServicePort authService = authlocator.getAuthServicePort(new URL(serverURL));
CredentialType credentialType = new CredentialType();

CredentialElementType nameSpaceElement = new CredentialElementType();
ValueElementType nameSpaceValue = new ValueElementType();
nameSpaceValue.setActualValue(nameSpaceStr);
nameSpaceElement.setName("CAMNamespace");
nameSpaceElement.setValue(nameSpaceValue);

CredentialElementType userNameElement = new CredentialElementType();
ValueElementType userNameValue = new ValueElementType();
userNameValue.setActualValue(userNameStr);
userNameElement.setName("CAMUsername");
userNameElement.setValue(userNameValue);

CredentialElementType passWordElement = new CredentialElementType();
ValueElementType passWordValue = new ValueElementType();
passWordValue.setActualValue(userPasswordStr);
passWordElement.setName("CAMPassword");
passWordElement.setValue(passWordValue);

//Login IBM Cognos server using the CMS Authentication Service
credentialType.setCredentialElements(new CredentialElementType[]
    {nameSpaceElement,userNameElement,passWordElement});
LogonRequestType logonRequest = new LogonRequestType(credentialType, null);
LogonResponseType logonResponse = authService.logon(logonRequest);

//Copy the SOAP header from the Authentication Service to CMS
org.apache.axis.message.SOAPHeaderElement[] headers
          =((org.apache.axis.client.Stub) authService).getResponseHeaders();
```

When the application has completed, log off the user if your BI server requires authentication.

C# example

```
LogoffRequestType logoffRequest = new LogoffRequestType();
logoffResponseType logoffResp = authService.logoff(logoffRequest);
```

Java example

```
LogoffRequestType LogoffRequest = new LogoffRequestType();
LogoffResponseType logoff = authService.logoff(LogoffRequest);
```

Creating a report service instance

To create a report service instance that is used to run Mashup Service methods, create a `ReportDataServicePort` object (Java application) or a `ReportDataService` object (C# application).

C# example

```
 ReportDataService svc = new ReportDataService();
```

Java example

```
static String serverURL = "http://localhost/ibmcognos/cgi-bin/cognos.cgi";
...
ReportDataServiceLocator rdslocator = new ReportDataServiceLocator();
ReportDataServicePort rdsservice = rdslocator.getReportDataServiceBinding(new URL(serverURL));
```

If you are creating a report-specific application, create a *report_name* object. The following example is based on the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Returns by Order Method - Prompted Chart** used in the Prompt Answers sample programs.

C# example

```
 CMSCommon.CCS_1.Returns__by__Order__Method___Prompted__Chart_Service
service =
  new CMSCommon.CCS_1.Returns__by__Order__Method___Prompted__Chart_Service();
service.Url = url + suffix;
```

Java example

```
Returns__by__Order__Method___x002D__Prompted__Chart_ServiceLocator serviceLocator
= new Returns__by__Order__Method___x002D__Prompted__Chart_ServiceLocator();
Returns__by__Order__Method___x002D__Prompted__Chart mashupService=serviceLocator
.getReturns__by__Order__Method___x002d__Prompted__Chart_Service(new
URL(url+suffix));
```

If your application requires authenticated access to the server, copy the authentication credentials to the service object.

C# example

```
svc.biBusHeaderValue = new CMSCommon.CCS_Generic.biBusHeader();
svc.biBusHeaderValue.Any = authService.biBusHeaderValue.Any;
svc.Url = url;
```

Java example

```
((org.apache.axis.client.Stub) rdsservice).setHeader(headers[0]);
```

Running Mashup Service methods

After authenticating with the IBM Cognos Business Intelligence server and creating a IBM Cognos Mashup Service object, you can start issuing Mashup Service requests to the server.

All Mashup Service SOAP methods are asynchronous requests, except for the `logon`, `logoff`, `getCognosURL`, `getOutputFormat`, `getOutputFormats`, `getPromptAnswers`, `getPromptPage`, and `release` methods. When an asynchronous methods requests output from the Business Intelligence server, the response from the server includes a `session` element, which contains a `conversationID` element and a `status` element. The `conversationID` element contains an identifier that allow subsequent method calls to refer to the same asynchronous conversation. The `status` element consists of either the value `working` or `complete`. When the value is `complete`, the server response also includes the requested output.

For a generic application, all asynchronous methods have `GetOutputResponse` as the response class, and the `getOutput` method is used to poll the BI server until the requested output is available. The `session` element is copied from the `GetOutputResponse` element to the subsequent `GetOutputRequest` element.

C# example

```
while (response.session.status == SessionTypeStatus.working)
{
  GetOutputRequest waitRequest = new GetOutputRequest();
  waitRequest.session = response.session;
  response = svc.getOutput(waitRequest);
}
```

Java example

If the application uses authenticated access to the server, copy the response header to the next method call, as shown in the highlighted text in the following example.

```
while (response.getSession().getStatus() == SessionTypeStatus.working)
{
  GetOutputRequest oreq = new GetOutputRequest(response.getSession(), null);
  headers = ((org.apache.axis.client.Stub) rdsservice).getResponseHeaders();
  ((org.apache.axis.client.Stub) rdsservice).setHeader(headers[0]);
  response = rdsservice.getOutput(oreq);
}

result=response.getOutput().getFormatOutput();
```

For a report-specific application, the method used to poll the server is the same as the method used to initiate the asynchronous conversation. As in the case of a generic application, copy the `session` element from the response element to the next request element.

C# example

```
CMSCommon.CCS_1.GetReportResponseType response = service.getReport(request);

while (response.session.status == CMSCommon.CCS_1.StatusEnum.working)
{
  request.session = response.session;
  response = service.getReport(request);
}
```

Java example

This application does not use authenticated access to the BI server, and the response header is not copied to the next method call.

```
GetReportResponseType response = mashupService.getReport(request);

/*
 * This loop is necessary when running asynchronously
 */
while (response.getSession().getStatus()== StatusEnum.working) {
  request.setSession(response.getSession());
  response=mashupService.getReport(request);
}
```

Secondary operations

For some method calls, secondary operations can be used to retrieve additional output after the initial method call has completed.

For example, after using `getPagedReportData` to get the first page of report output, the `next` secondary method can be called to get the next page of report output. When calling a secondary method, the `session` element is copied from the initial method response to the secondary method request. The following examples illustrate the use of secondary operations.

C# example

```
GetPagedReportDataRequest req = new GetPagedReportDataRequest();
...
GetOutputResponse resp = svc.getPagedReportData(req);

resp = waitForOutput(resp);
...
NextRequest nextreq = new NextRequest();
nextreq.session = resp.session;
...
resp = svc.next(nextreq);
resp = waitForOutput(resp);

...
private GetOutputResponse waitForOutput(GetOutputResponse resp)
   throws RemoteException, CCSAuthenticationFault, CCSPromptFault, CCSGeneralFault
{
  while (resp.session.status == SessionTypeStatus.working)
  {
    GetOutputRequest waitReq = new GetOutputRequest();
    waitReq.session = resp.session;
    resp = svc.getOutput(waitReq);
  }
  return resp;
}
```

Java example

```
GetPagedReportDataRequest req = new GetPagedReportDataRequest();
...
GetOutputResponse resp = svc.getPagedReportData(req);

resp = waitForOutput(resp);
...
NextRequest nextreq = new NextRequest();
nextreq.setSession(resp.getSession());
...
resp = svc.next(nextreq);
resp = waitForOutput(resp);
```

```
...
private GetOutputResponse waitForOutput(GetOutputResponse resp)
     throws RemoteException, CCSAuthenticationFault, CCSPromptFault, CCSGeneralFault
{
  while (resp.getSession().getStatus() == SessionTypeStatus.working)
  {
    GetOutputRequest waitReq = new GetOutputRequest(resp.getSession(), null);
    headers = ((org.apache.axis.client.Stub) svc).getResponseHeaders();
    ((org.apache.axis.client.Stub) svc).setHeader(headers[0]);
    resp = svc.getOutput(waitReq);
  }
  return resp;
}
```

You can also use `previous` to retrieve the previous page of report output, `first` to retrieve the first page of output, and `last` to retrieve the last page of report output. If you are retrieving the report in LDX format, the `secondaryOperations` element in the report output indicates which secondary requests are available. HTML and HTMLFragment outputs also indicate available secondary requests in a `div` element.

Retrieving report data

Report outputs can be retrieved using the Mashup Service methods in a variety of ways.

Generic applications

There are two methods you can use to retrieve report data in a generic application:

- The `getPagedReportData` method retrieves the first page of report output. The secondary methods `next`, `previous`, `first`, and `last` can be used to retrieve additional pages of report output.
- The `getReportData` method retrieves the complete report output. The output is retrieved as a single page unless `includePageBreaks` is `true`, in which case the report output is separated into pages. See "Reports with multiple pages" on page 64 for more information.

To retrieve report output in a generic application, identify the report, using the `sourceID` and `sourceType` elements in the request. For more information, see "Identifying reports" on page 8.

When the asynchronous report data request completes, the report output is contained in the `output` child of `GetOutputResponse`. If the `format` option in the report data request is not used, the report output will be contained in the `LDXOutput` child of `output`, and this report data can be accessed using additional methods. If the `format` option is set, the report output will be contained in the `FormatOutput` child of `output` as a string object.

Report-specific applications

When you create an application using a report-specific WSDL file, a number of customized methods are available for retrieving report data.

These are the `get_<element>` and `getFormatted_<element>` methods. The `<element>` parts of the method names correspond to the Simple format element names that are derived from LDX `id` elements, see Chapter 8, "Understanding the Simple format," on page 75. For example, the Simple format structure of the "Employee Satisfaction 2006" report is shown in Chapter 8, "Understanding the Simple format," on page 75. The versions of the `get_<element>` method for this report are:

- `get_Page1`

- get_FirstPage_x005FReportTitle2121
- get_FirstPage_x005FSubtitle1121
- get_Combination__Chart___x002D__survey__topic__scores__by__department
- get_Combination__Chart___x002D__survey__scores__and__benchmark
- get_Crosstab1
- get_RunDate1
- get_PageNumber
- get_RunTime1

There are two sets of methods that report-specific applications can use to retrieve report output:

- The getReport and get_<element> methods retrieve report output in Simple format and the report output can be accessed using additional methods.
- The getFormattedReport and getFormatted_<element> methods retrieve report output in the format specified in the format object in the request. The response is contained in a string object.

Report-specific method limitations for some reports

The customized methods are not available for reports with certain structural elements.

In addition, the getReport method returns the report output in Layout Data (LDX) format, not in Simple format. The response is returned in the report-specific namespace, not in the generic LDX namespace.

Reports with the following structural elements are subject to this limitation:

- Lists that contain list, crosstab, repeater, or repeater table objects. This case includes lists that are split into sections.
- Crosstabs that contains dimensions with the same label but different data items.
- Lists that use report expressions or data item values as the column title.

Report output examples

The following examples illustrate different ways of accessing the same data item in a report. The report is **Employee Satisfaction 2006** and we are retrieving the value in the **Employee rankings and terminations by department** crosstab for the row **Human Resources** and column **Satisfactory Employee ranking**.

Generic application with pagedReportData method

This example retrieves the entire report, and then parts of it are selected.

The report is run using the getPagedReportData method with the default options. A snippet of the LDX output is shown here.

```
<document...
  <pages>
    <page>
      <id>Page1</id>
      ...
      <body>
        <item>
          <tbl>
            <trow>
            ...
            <trow>
              <tcell>
              ...
              <tcell>
                <item>
```

```
...
<item>
  <ctab>
    <id>Crosstab1</id>
    ...
    <table>
      <row>
        <cell>
        ...
        <cell>
        ...
          <item>
            <txt>
            ...
              <val>0.0294117647058824</val>
              ...
```

The following code snippets show how the highlighted value above can be retrieved.

C# example

```
LDXOutputType ldx = response.output.Item as LDXOutputType;
Document doc = ldx.Item as Document;
pagesType[] pgs = doc.pages as pagesType;
Page pge = pgs[0].Item as Page;
ReportElement body = pge.body as ReportElement;
ReportElementTbl[] tbl = body.Item as ReportElementTbl;
CrossTab ctab = tbl[0].trow[1].tcell[1].item[0].Item as CrossTab;
TextFrame txt = ctab.table[0].cell[1].item[0].Item as TextFrame;
String value = txt.val as String;
```

Java example

```
value = resp.getOutput().getLDXOutput().getDocument().getPages(0).getPage().getBody()
     .getTbl()[0].getTrow(1).getTcell(1).getItem(0).getCtab().getTable()[0]
     .getCell(1).getItem(0).getTxt().getVal();
```

Generic application with pagedReportData method using filter and includeLayout options

This example retrieves a portion of the report, and then parts of it are selected.

The report is run using the `getPagedReportData` method with a `filters` element to select the crosstab (`filterType` = `OBJECT_ID` and `filterValue` = `Crosstab1`.). A snippet of the LDX output is shown here.

```
<filterResultSet...
  <filterResult>
    <filterType>OBJECT_ID</filterType>
    <filterValue>Crosstab1</filterValue>
    <reportElement>
      <ctab>
        <id>Crosstab1</id>
        ...
        <table>
          <row>
            <cell>
            ...
            <cell>
            ...
              <item>
                <txt>
                ...
                  <val>0.0294117647058824</val>
                  ...
```

The following code snippets show how the highlighted value above can be retrieved.

C# example

```
LDXOutputType ldx = response.output.Item as LDXOutputType;
FilterResultSet frs = response.output.Item as FilterResultSet;
CrossTab ctab = frs.filterResult[0].reportElement[0].Item as CrossTab;
TextFrame txt = ctab.table[0].cell[1].item[0].Item as TextFrame;
String value = txt.val as String;
```

Java example

```
value = resp.getOutput().getLDXOutput().getFilterResultSet().getFilterResult(0)
       .getReportElement(0).getCtab().getTable()[0].getCell(1).getItem(0).getTxt().getVal();
```

Report-specific application with get_Crosstab1 method

The report is run using the `get_Crosstab1` method with the default options. A snippet of the Simple output is shown here.

```
<results xmlns="...">
  <Crosstab1>
  ...
    <table>
      <row>
        <cell>
        ...
        <cell>
        ...
          <item>
            <txt>
            ...
              <val>0.0294117647058824</val>
              ...
```

The following code snippets show how the highlighted value above can be retrieved.

C# example

```
CMSCommon.CCS_1.TextFrame tframe = (CMSCommon.CCS_1.TextFrame)response
     .results.Crosstab1.table[0].cell[1].item[0].Item;
String value = (String)itm.tframe;
```

Java example

```
TextFrame txt = (TextFrame)resp.results.Crosstab1.table[0].cell[1].item[0].Item;
String value = txt.val;
```

Handling exceptions

Your application will need to handle both Mashup Service-specific exceptions and general exceptions.

C# example

```
catch (System.Web.Services.Protocols.SoapException ex)
{
  if (ex.Detail != null)
  {
    if (ex.Detail.FirstChild.LocalName == "CCSPromptFault")
    {
      System.Console.WriteLine(ex.ToString());
      return "Please make sure the report ID is correct ...";
    }
    else if (ex.Detail.FirstChild.LocalName == "CCSAuthenticationFault")
    {
      System.Console.WriteLine(ex.ToString());
      return "Login Failed. Please try again.";
    }
    else if (ex.Detail.FirstChild.LocalName == "CCSGeneralFault")
    {
      System.Console.WriteLine(ex.ToString());
      return "Please make sure the report ID is correct ...";
    }
    else
    {
      return (ex.Message);
    }
  }
  else
  {
    System.Console.WriteLine("ERROR: " + ex.Message);
    return (ex.Message);
  }
}
```

Java example

```
catch (CCSGeneralFault e) {
  e.printStackTrace();
  return "Please make sure the report ID is correct ...";
}
catch (CCSPromptFault e) {
```

```
  e.printStackTrace();
  return "Please make sure the report ID is correct ...";
}
catch (CCSAuthenticationFault e) {
  e.printStackTrace();
  return "Login Failed. Please try again.";
}
catch (RemoteException e) {
  e.getMessage();
  return e.getMessage();
}
catch (ServiceException e) {
  e.printStackTrace();
  return e.getMessage();
}
catch (Exception e)
{
  e.printStackTrace();
}
```

Chapter 6. Performing additional tasks using the Mashup Service

You can use the Mashup Service to perform a number of tasks related to running reports. You can perform the following tasks:

- Finding reports
- Finding report parts
- Accessing parts of a report output
- Saving report versions
- Accessing saved report versions
- Running reports and retrieving output in IBM Cognos Viewer formats
- Accessing report outputs saved by IBM Cognos BI studios
- Running reports with prompts
- Drilling up and down in reports
- Drilling through to another report
- Using a URL to display a report in IBM Cognos Viewer

The following topics provide general descriptions on how to perform these tasks using the Mashup Service.

Finding reports

In order to retrieve report output using the Mashup Service, you first need to identify a report. Report identifiers can be either directly given in an application, either supplied by a user or hardcoded into the application, or they can be discovered programmatically.

Identifying reports programmatically

SOAP applications can discover reports in a content store by using methods from the IBM Cognos Software Development Kit. The Content Store Explorer sample program included with the IBM Cognos Software Development Kit demonstrates the use of these methods. See the *IBM Cognos Software Development Kit Developer Guide* for more information.

REST applications can use the Web Service Inspection Language (WSIL) to recursively explore a content store and reveal reports within it. Typing `http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/wsil` in a Web browser will generate the XML output shown here.

```
...
<inspection xmlns="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
    xmlns:wsilwsdl="http://schemas.xmlsoap.org/ws/2001/10/inspection/wsdl/">
  <link referencedNamespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
    location="http://localhost:80/ibmcognos/cgi-bin/cognos.cgi/rds/wsil
    /path/Public%20Folders/sales_and_marketing">
    <abstract>sales_and_marketing</abstract>
  </link>
  <link referencedNamespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
    location="http://localhost:80/ibmcognos/cgi-bin/cognos.cgi/rds/wsil
   /path/Public%20Folders/Samples">
    <abstract>Samples</abstract>
  </link>
</inspection>
```

This XML output lists the subfolders of the Public Folder" folder of the IBM Cognos installation. Copying the Web address for the Samples folder into the Web browser address bar will generate the following output.

```
...
  <link referencedNamespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
    location="http://localhost:80/ibmcognos/cgi-bin/cognos.cgi/rds/wsil
    /path/Public%20Folders/Samples/Metrics">
    <abstract>Metrics</abstract>
  </link>
  <link referencedNamespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
      location="http://localhost:80/ibmcognos/cgi-bin/cognos.cgi/rds/wsil
      /path/Public%20Folders/Samples/Models">
    <abstract>Models</abstract>
  </link>
  ...
```

This procedure can be repeated until the contents of a subfolder are not other subfolders, but are reports, as shown here.

```
...
  <service>
    <name>2005 Quarterly Sales Forecast</name>
    <abstract>IBM Cognos Content service</abstract>
    <description referencedNamespace="http://schemas.xmlsoap.org/wsdl/"
      location="http://localhost:80/ibmcognos/cgi-bin/cognos.cgi
    /rds/wsdl/path/Public%20Folders/Samples/Models/GO%20Sales%20%28analysis%29
    /Report%20Studio%20Report%20Samples/2005%20Quarterly%20Sales%20Forecast"/>
  </service>
  <service>
    <name>2005 Sales Summary</name>
    <abstract>IBM Cognos Content service</abstract>
    <description referencedNamespace="http://schemas.xmlsoap.org/wsdl/"
    location="http://localhost:80/ibmcognos/cgi-bin/cognos.cgi
    /rds/wsdl/path/Public%20Folders/Samples/Models/GO%20Sales%20%28analysis%29
    /Report%20Studio%20Report%20Samples/2005%20Sales%20Summary"/>
  </service>
  ...
```

The paths of these reports can now be used to retrieve report outputs from these reports.

The URLs used previously are examples of the wsil and wsdl resource types.

The cmsExplorer JavaScript sample program uses the methods described to navigate a content store and run selected reports.

Finding report parts

Using the Mashup Service you can work with individual report parts, as well as with complete reports. Report parts are named parts of a report specification. Names are automatically generated by Report Studio for major pieces of a report (lists, crosstabs, charts, etc.) and users can name report parts in Report Studio.

Use the ATOM feed for a report to get a list of report parts. The list includes the associated URLs of the report parts and this can be used during application development to determine which report parts should be extracted. The ATOM feed can also be used at runtime to offer a list of report parts to an application user, who can then select the desired output.

To get the ATOM feed for the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Employee Satisfaction 2006** submit the following URL to the IBM Cognos Business Intelligence server.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/atom/path
/Public%20Folders/Samples/Models/GO%20Data%20Warehouse%20%28analysis%29
/Report%20Studio%20Report%20Samples/Employee%20Satisfaction%202006
```

The ATOM output for a report or report part contains information about the report or report part, followed by one or more `atom:entry` elements that contain information about, and links to, the report parts that make up the report or report part. The ATOM output contains standard ATOM elements as well as elements that are unique to the Mashup Service. See atom for a description of the Mashup Service-specific elements.

An annotated version of the ATOM output is shown here.

```
<atom:feed...
  <atom:title>Employee Satisfaction 2006</atom:title>
  <atom:updated>2010-01-21T20:44:24Z</atom:updated>
  <atom:author>
    <atom:name>Anonymous</atom:name>
  </atom:author>
  <atom:link rel="self" type="application/atom+xml" href="..."/>
  <atom:link rel="alternate" type="text/xml" href="..."/>
  <atom:link rel="alternate" type="application/x-pagedldx+xml" href="..."/>
  ...
  <d:location>Public Folders > Samples > Models > GO Data Warehouse (analysis)
     > Report Studio Report Samples > Employee Satisfaction 2006</d:location>
  <d:description>This report shows employee satisfaction survey results by department,
    compared to targets and industry standards. It also shows employee
    rankings and terminations.</d:description>
  <d:thumbnailURL>.../d:thumbnailURL>
  <atom:entry>
    ...
    <atom:title>Page1</atom:title>
    ...
    <atom:link rel="alternate" type="application/atom+xml"
         href="http://localhost:80/ibmcognos/cgi-bin/cognos.cgi
            /rds/atom/path/Public%20Folders/Samples/Models
            /GO%20Data%20Warehouse%20%28analysis%29
            /Report%20Studio%20Report%20Samples
           /Employee%20Satisfaction%202006?selection=Page1&eltype=page"/>
    <atom:link rel="alternate" type="application/x-ldx+xml"
        href="http://localhost:80/ibmcognos/cgi-bin/cognos.cgi/rds/reportData
           /path/Public%20Folders/Samples/Models/GO%20Data%20Warehouse%20%28analysis%29
           /Report%20Studio%20Report%20Samples
          /Employee%20Satisfaction%202006?selection=Page1&eltype=page"/>
    <atom:link rel="alternate" type="application/x-pagedldx+xml"
         href="http://localhost:80/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData
            /report/i3fccdd8960f94960aed086b252cc9ca5?selection=Page1&version=LATEST"/>
    <cm:location>Public Folders > Samples > Models > GO Data Warehouse (analysis)
          > Report Studio Report Samples > Employee Satisfaction 2006 > Page1</cm:location>
    <d:type>page</d:type>
    <d:storeID>i3fccdd8960f94960aed086b252cc9ca5</d:storeID>
    <d:partID>Page1</d:partID>
  </atom:entry>
</atom:feed>
```

The ATOM feed contains a lot of information about the report, including its name, summary description, author, location, and the URLs to run the report using the pagedReportData and reportData REST resources. The `atom:entry` elements provide information about the report parts contained in this report. In particular, the `<atom:link rel="alternate" type="application/atom+xml" href="..."/>` elements can be used to recursively obtain the ATOM feed for each report part. This procedure can be repeated until the ATOM feed provides `atom:entry` elements for the base report parts of the report. An example is shown here.

```
<atom:entry>
  ...
  <atom:link rel="thumbnail" type="image/gif" href="http://localhost:80/ibmcognos/cgi-bin
        /cognos.cgi/rds/thumbnail/report/i3fccdd8960f94960aed086b252cc9ca5?selection
           =Combination__Chart___x002d__survey__topic__scores__by__department"/>
  <atom:link rel="alternate" type="application/x-ldx+xml" href="http://localhost:80
            /ibmcognos/cgi-bin/cognos.cgi/rds/reportData/path
            /Public%20Folders/Samples/Models/GO%20Data%20Warehouse%20%28analysis%29
            /Report%20Studio%20Report%20Samples/Employee%20Satisfaction%202006
            ?selection=Combination__Chart___x002d__survey__topic__scores__by__department"/>
  <atom:link rel="alternate" type="application/x-pagedldx+xml"
         href="http://localhost:80/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData
           /report/i3fccdd8960f94960aed086b252cc9ca5?selection
           =Combination__Chart___x002d__survey__topic__scores__by__department&version=LATEST"/>
```

```
  <cm:objectClass>reportpart</cm:objectClass>
  <cm:location>Public Folders > Samples > Models > GO Data Warehouse (analysis)
      > Report Studio Report Samples > Employee Satisfaction 2006
      > Combination Chart - survey topic scores by department</cm:location>
  <d:type>chart</d:type>
  <d:storeID>i3fccdd8960f94960aed086b252cc9ca5</d:storeID>
  <d:partID>Combination Chart - survey topic scores by department</d:partID>
</atom:entry>
```

The atom JavaScript sample program illustrates the use of ATOM feeds to identify report parts and view their Mashup Service HTML output.

Accessing parts of a report output

You can filter the report output that is returned from the Business Intelligence server in a number of ways. Filtering the amount of report output that is returned can improve performance considerably.

Except as noted in the following topics, filtering can be used for any output format. See "Filter output structure" on page 64 for a description of filtered LDX output and "Filter output structure in Simple format" on page 76 for a description of filtered Simple output.

Accessing named report parts

You can restrict the report output to specific named report parts. This option is available for all output formats.

Although report part names must be unique within a report, it is possible that a report page name will be the same as a report part name. In this case, use the `excludePage` option (REST applications) or the `excludePage` element (SOAP applications) to retrieve the report part instead of the report page.

REST example

In a REST application use the `selection` option to return only named report parts. To return more than one part in a single request, separate report part names with a semi-colon (;). An example is shown here:

```
selection=List1;List3
```

SOAP example

In a SOAP application, include `filters` objects in the report output request. The value of the `filterType` child object is `OBJECT_ID` and the value of the `filterValue` child object is the report part name. Include one `filters` object for each report part to be returned.

Using XPath expressions to filter report output

You can use XPath expressions to filter report output. You can only use XPath expressions to filter LDX report output. For example, the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Employee Satisfaction 2006** can be filtered on the XPath expression `/document/pages/page[id='Page1']/body/item/tbl/trow[2]/tcell/item/ctab[id='Crosstab1']/table`

A portion of the layout data output is shown here.

```
<filterResultSet xmlns="http://developer.cognos.com/schemas/rds/contentmodel/1">
  <filterResult>
    <filterType>XPATH</filterType>
    <filterValue>/document/pages/page[id='Page1']/body/item/tbl/trow[2]/tcell
       /item/ctab[id='Crosstab1']/table</filterValue>
    <reportElement>
      <table>
      ...
      </table>
    </reportElement>
  </filterResult>
</filterResultSet>
```

Only a subset of the full XPath 2.0 specification is supported. In particular, the only axis available is the `child` axis (either specified or implied) and the only predicate functions available are `local-name()` and `pos()`. The XPath location is evaluated relative to the LDX representation of the report, before any selection filters have been applied.

The supported syntax for the XPath query parameter includes:

- Absolute rooted paths (`/document/page`) and descendent path (`//page`)
- The XPath child axis is supported. The following examples will filter on all charts or lists:
  - `/document/pages/page/body/item[cht or lst]`
  - `/document/pages/page/body/item[child::cht or child::lst]`
- `*` as a wildcard matching any element, such as (`//page/*/item`
- Predicates based on a descendent axis, using `=`, `<`, or `>` comparisons, such as`//page[id=Page1]`
- Predicates based on index and position, such as`//page[2]` or `//page[pos()2]`)
- Compound predicates with a single binary operator (either `or` or `and`), such as `//page[pos()>10 and pos()<20]`, `//item[txt/ref=R19, txt/ref=R21]`, or `//styleGroup[hAlign=CENTER][vAlign=TOP]`

Unsupported Xpath syntax includes:

- Nested predicates, such as `/a[b[(c=3)]`
- Lookback predicates, such as `/a[../b=3]`

Note: The output returned will be an LDX `filterResultSet` object with the value of `filterType` being `XPATH` and `filterValue` containing the XPath expression. The response will not necessarily validate against the Layout Data schema.

REST example

In a REST application use the `xpath` option to filter on an XPath expression. An example is shown here:

```
xpath=/document/pages/page[id='Page1']/body/item/tbl/trow[2]/tcell/item
/ctab[id='Crosstab1']/table
```

SOAP example

In a SOAP application, include `filters` objects in the report output request. The value of the `filterType` child object is `XPATH` and the value of the `filterValue` child object is the XPath expression. Set the `format` to be `layoutDataXML`. The response will be contained in the `FormatOutput` object in the response.

Restricting the number of rows of output

You can restrict the report output to a maximum number of rows by using the `rowLimit` option (REST applications) or the `rowLimit` element (SOAP applications).

Running reports and retrieving output in IBM Cognos Viewer formats

You can run reports and retrieve outputs in the formats used by IBM Cognos Viewer (such as PDF, CSV, Microsoft Excel).

You can use the outputFormats resource type (REST applications) or the getOutputFormats method (SOAP applications) to retrieve a list of supported output formats for a report. The list is contained in a GetOutputFormatsResponse element. You can then use the outputFormat resource type (REST applications) or the getOutputFormat method (SOAP applications) to obtain the report output in a specified format.

REST example

The following URL requests a list of the supported output formats for the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Employee Satisfaction 2006**.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/outputFormats/report/
i1AD695527913442D92D07DDB3425740F
```

The response from the IBM Cognos BI Server is shown here:

```
<rds:GetOutputFormatsResponse>
  <rds:supportedFormats>
   <rds:outputFormatName>CSV</rds:outputFormatName>
   <rds:outputFormatName>MHT</rds:outputFormatName>
   <rds:outputFormatName>PDF</rds:outputFormatName>
   <rds:outputFormatName>spreadsheetML</rds:outputFormatName>
   <rds:outputFormatName>XLWA</rds:outputFormatName>
   <rds:outputFormatName>XML</rds:outputFormatName>
  </rds:supportedFormats>
</rds:GetOutputFormatsResponse>
```

To run the report and retrieve the output for the **Crosstab1** report part in PDF format, submit the following URL to the BI Server:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/outputFormat/report/
i1AD695527913442D92D07DDB3425740F/PDF?selection=Crosstab1
```

SOAP example

The following code snippets shows how to perform the same actions as in the above REST example.

C# sample

```
rds.ReportDataService svc = new rds.ReportDataService();
rds.GetOutputFormatsRequest formatsreq = new rds.GetOutputFormatsRequest();

formatsreq.sourceID = "i1AD695527913442D92D07DDB3425740F";
formatsreq.sourceType = rds.SourceTypeEnum.report;
rds.GetOutputFormatsResponse formats = svc.getOutputFormats(formatsreq);
foreach (String format in formats.supportedFormats.outputFormatName)
{
    System.Console.WriteLine("Supporting: " + format);
}
```

```
rds.GetOutputFormatRequest formatreq = new rds.GetOutputFormatRequest();

rds.Filter report_part = new rds.Filter();
report_part.filterType = rds.FilterTypeEnum.OBJECT_ID;

report_part.filterValue = "Crosstab1";

formatreq.sourceID = "i1AD695527913442D92D07DDB3425740F";
formatreq.sourceType = rds.SourceTypeEnum.report;
formatreq.outputFormatName = "PDF";
formatreq.filters = new rds.Filter[] { report_part };

rds.GetOutputFormatResponse formatresp = svc.getOutputFormat(formatreq);

System.Console.WriteLine("URL: " + formatresp.outputFormatURL);
```

Java sample

```
ReportDataServiceLocator rdslocator = new ReportDataServiceLocator();
ReportDataServicePort rdsservice =
        rdslocator.getReportDataServiceBinding(new URL(url));

GetOutputFormatsRequest formatsreq = new GetOutputFormatsRequest();
formatsreq.setSourceID("i1AD695527913442D92D07DDB3425740F");
formatsreq.setSourceType(SourceTypeEnum.report);

GetOutputFormatsResponse formatsresp =
          rdsservice.getOutputFormats(formatsreq);

for (String str : formatsresp.getSupportedFormats().getOutputFormatName())
            { System.out.println("Supporting: " + str);
}

GetOutputFormatRequest req = new GetOutputFormatRequest();
req.setSourceID("i1AD695527913442D92D07DDB3425740F");
req.setSourceType(SourceTypeEnum.report);
req.setOutputFormatName("PDF");

Filter[] filters = {new Filter("Crosstab1", FilterTypeEnum.OBJECT_ID, null)};
req.setFilters(filters);

GetOutputFormatResponse resp = rdsservice.getOutputFormat(req);

System.out.println("URL: " + resp.getOutputFormatURL());
```

The response from the BI server from either of the above programs is shown here:

```
Supporting: CSV
Supporting: MHT
Supporting: PDF
Supporting: spreadsheetML
Supporting: XLWA
Supporting: XML
URL: http://localhost:80/ibmcognos/cgi-bin/cognos.cgi/rds/
outputformat/report/i1AD695527913442D92D07DDB3425740F/PDF
?selection=Crosstab1
```

Saving report versions

Use the `saveOutput` option (REST applications) or the `saveOutput` option (SOAP applications) when running a Mashup Service report to save a version of the report output in the Content Store. In order to save report versions in the Content Store, the report properties must be set to save report output versions. The option **Enable enhanced user features in saved output versions** does not need to be checked on the *Advanced options* report properties page.

Accessing saved report versions

You can retrieve reports that have been saved using the IBM Cognos Mashup Service as well as reports saved in IBM Cognos Viewer.

Use the `version` and `versionID` options (REST application) and the `versionType` and `versionName` elements (SOAP application) to retrieve a saved version of a report.

Tip: The value of the `versionID` option or `versionName` element for a saved report can only be determined by using the IBM Cognos Software Development Kit. The ContentStoreExplorer sample program included with the Software Development Kit can be used to determine the version names of stored reports. See the *IBM Cognos Software Development Kit Developer Guide* for more information.

Accessing report outputs saved by IBM Cognos BI studios

You can retrieve reports outputs saved by Analysis Studio, Query Studio, and Report Studio.

Use the `outputFormat` resource type in a REST application to retrieve reports outputs. The report output can only be retrieved in the format it was saved in, and it cannot be filtered. However, the `burstID` and `burstKey` options can be specified

- If the `fmt` option is specified, the report output will be retrieved in the specified format if there is a version saved in that format. Otherwise, the IBM Cognos Business Intelligence server will return an error page of type 404.
- A particular `version` of a saved report cannot be retrieved.

Running reports with prompts

To run reports with prompts, you need to determine prompt values and then submit them to the IBM Cognos Business Intelligence server when running the report.

Collecting prompts

You can use the standard IBM Cognos user interface to collect prompt answers, or you can collect the answers yourself.

Using the IBM Cognos prompt page interface

You can use the standard IBM Cognos prompt page interface to display prompts to the users

Submit a `promptPage` request (REST) or a `getPromptPage` request (SOAP) to the BI server. The server response includes a url than can be used to display the prompt page. A sample prompt request page is shown here.

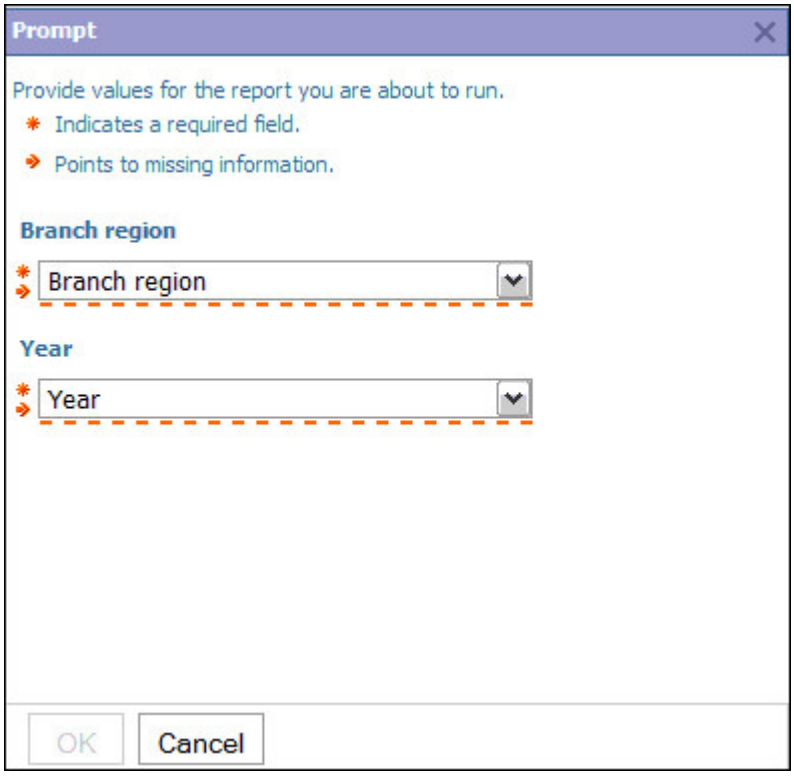

Figure 2. Prompt request page

After the user submits prompt answers using this interface the browser window closes. You can then retrieve the prompt answers by submitting a `promptAnswers` request (REST) or a `getPromptAnswers` request (SOAP) to the BI server.

You can then run the report, submitting the prompt answers along with the report request. See "Running a report with prompts" on page 54 for more information.

Examples using the REST and SOAP interfaces are shown here. The examples are based on the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Global Bonus Report**.

REST example

Submit the following URL to the server using the `promptPage` resource type. In this case we are using the ID of the report:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/promptPage/report
/ia195960a5e77488cb4583d74b56c78d6
```

The response looks like the following:

```
<rds:GetPromptPageResponse>
  <rds:promptID>ieb88ba5380774c85b13491fc90acdd32</rds:promptID>
  <rds:url>
   http://localhost:80/ibmcognos/cgi-bin/cognos.cgi?b_action=xts.run
   &m=ccs/ccs_prompt.xts
   &ui.object=storeID("ia195960a5e77488cb4583d74b56c78d6")
   &promptID=ieb88ba5380774c85b13491fc90acdd32
  </rds:url>
</rds:GetPromptPageResponse>
```

Typing the `url` element into a Web browser address bar returns the standard IBM Cognos prompt page.

After the user submit prompt answers using this interface the browser window closes. You can retrieve the prompt answers by using the following URL with the promptAnswers resource type, in which the conversationID field contains the value of the promptID field in the initial response from the server:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/promptAnswers
    /conversationID/ieb88ba5380774c85b13491fc90acdd32
```

If you selected the Asia Pacific branch region and the year 2006 as the prompt values, the server returns the following response.

```
<rds:promptAnswers>
  <rds:promptValues>
    <rds:name>pYear</rds:name>
    <rds:values>
      <rds:item>
        <rds:SimplePValue>
          <rds:inclusive>true</rds:inclusive>
          <rds:useValue>
            [Employee summary].[Time].[Time].[Year]->[Time].[2006]
          </rds:useValue>
          <rds:displayValue>2006</rds:displayValue>
        </rds:SimplePValue>
      </rds:item>
    </rds:values>
  </rds:promptValues>
  <rds:promptValues>
    <rds:name>pRegion</rds:name>
    <rds:values>
      <rds:item>
        <rds:SimplePValue>
          <rds:inclusive>true</rds:inclusive>
          <rds:useValue>
            [Employee summary].[Employee by region].[Employee by region].
            [Branch region]->[Employee by region].[740]
          </rds:useValue>
          <rds:displayValue>Asia Pacific</rds:displayValue>
        </rds:SimplePValue>
      </rds:item>
    </rds:values>
  </rds:promptValues>
</rds:promptAnswers>
```

SOAP example

Submit a getPromptPage request to the BI server, specifying the sourceType and sourceID of the report.

The response from the server includes promptID and url objects. Display the url to the user in a Web browser.

After the user enters prompt values, the Web page closes. Submit a getPromptAnswers request to the server, including in the request the promptID that was in the response from the getPromptPage request. The server response contains promptValues elements that include the chosen values for the prompts.

Note: The generic and report-specific versions of getPromptPage and getPromptAnswers differ slightly. The generic versions use promptID to link the two methods, while the report-specific versions use session for this purpose.

Collecting prompts with an alternate interface

You can collect prompt answers using a custom interface.

Use the reportPrompts resource type (REST application) or the getReportPrompts method call (SOAP application).

The response is an LDX document that describes the possible prompt values for the report. (See Prompt request page in LDX format for more information.)

This method of retrieving possible prompt values is not available for report-specific SOAP applications.

Collecting prompts from a tree prompt page

Some reports have a tree prompt page. An example of such a report is the sample report **Public Folders > Samples > Models > GO Data Warehouse (query) > SDK Report Samples > Tree prompt sample**.

If you use the IBM Cognos prompt user interface, the prompt page has a hierarchical tree structure that you can use to drill down to a product category or product.

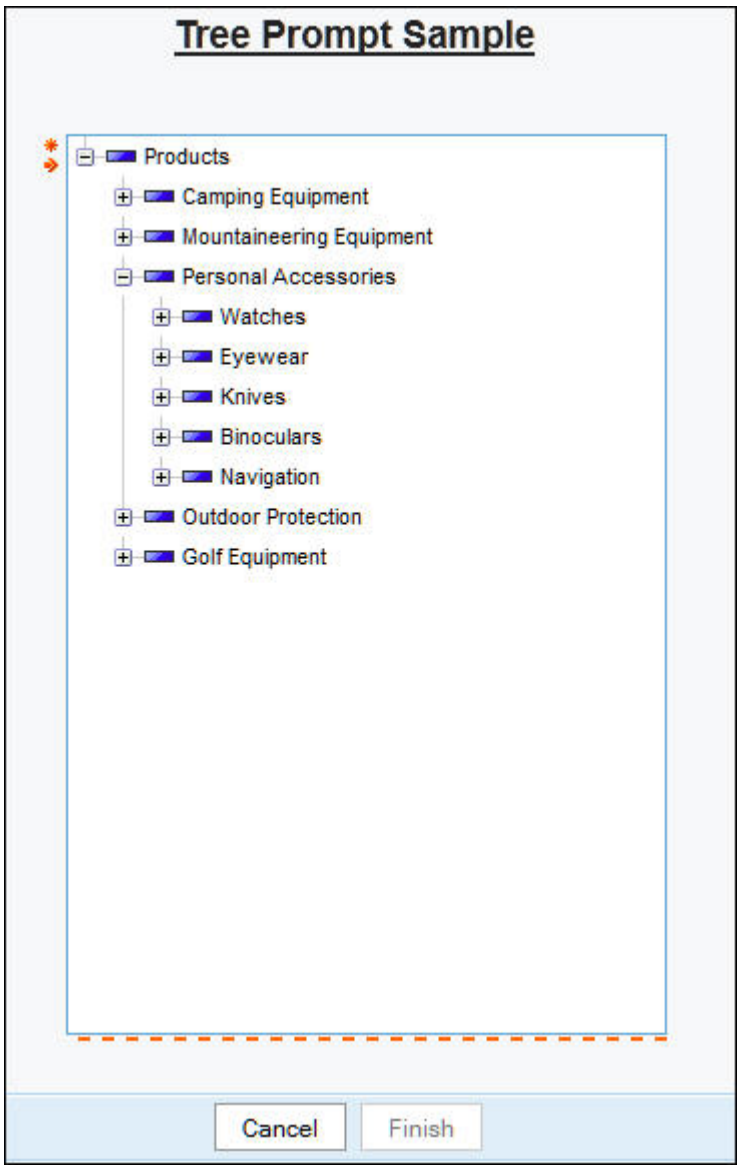

Figure 3. Tree prompt

After you submit prompt values, they can be retrieved in the same way as any other prompted report.

If you collect prompts yourself, retrieve the prompt information that is associated with the first level hierarchy (**Products**). For more information, see"Collecting prompts with an alternate interface" on page 44.

To drill down to lower levels in the prompt tree, use the `treePromptNode` secondary operation, to specify the `pname` and `use` values from the response. The response is an RDS page that contains a `treePromptNode` element. For example:

```
<rds:treePromptNode xmlns:rds="http://developer.cognos.com/schemas/rds/types/2">
  <rds:options>
   <rds:useValue>[Sales].[Products].[Products].[Product line]->[all].[991]
</rds:useValue>
   <rds:displayValue>Camping Equipment</rds:displayValue>
  </rds:options>
  <rds:options>
   <rds:useValue>[Sales].[Products].[Products].[Product line]->[all].[992]
</rds:useValue>
   <rds:displayValue>Mountaineering Equipment</rds:displayValue>
  </rds:options>
  <rds:options>
   <rds:useValue>[Sales].[Products].[Products].[Product line]->[all].[993]
</rds:useValue>
   <rds:displayValue>Personal Accessories</rds:displayValue>
  </rds:options>
  <rds:options>
   <rds:useValue>[Sales].[Products].[Products].[Product line]->[all].[994]
</rds:useValue>
   <rds:displayValue>Outdoor Protection</rds:displayValue>
  </rds:options>
  <rds:options>
   <rds:useValue>[Sales].[Products].[Products].[Product line]->[all].[995]
</rds:useValue>
   <rds:displayValue>Golf Equipment</rds:displayValue>
  </rds:options>
</rds:treePromptNode>
```

You can submit a prompted report to run by submitting the `pname` and `useValue` values, or you can submit extra `treePromptNode` requests by specifying the original `pname` value and a `useValue` value from the response. If you attempt to drill down below the lowest level, the response is an empty `treePromptNode` element.

REST example

This example is based on the **Tree prompt sample** report available in the sample database. Submit the following URL to the IBM Cognos Business Intelligence server:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportPrompts/report
/iAB3DAAF3A4A9446E9445C6C0C1693EC2
```

The response is an LDX page that contain the **Products** category. Note the URL that is associated with the response page: Submit the following secondary request:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput
/conversationID/iD41D1EE47B1148EB97B57C7CB4AFEBBB
/treePromptNode?p_ProductPara=[Sales].[Products].[Products].[Products]->[all]
```

The response is the `treePromptNode` element. Select the **Personal Accessories** category to drill down further, by using the following request:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput
/conversationID/iD41D1EE47B1148EB97B57C7CB4AFEBBB
/treePromptNode?p_ProductPara=[Sales].[Products].[Products].[Product line]->[all].[993]
```

You can use the response to run the report on the **Binoculars** category as described in "Running a report with prompts" on page 54 by using the following parameter:

```
p_ProductPara=[Sales].[Products].[Products].[Product type]->[all].[993].[963]
```

SOAP example

For examples of using tree prompts in the C# and Java programming languages, see the `ExpandTreePrompt` samples programs.

Collecting Select & Search prompt values

Some reports collect prompt values that are based on search criteria that are specified by the report user. An example of such a report is **Public Folders > Samples > Models > GO Data Warehouse (query > SDK Report Samples > Search Prompt Product**.

If you use the IBM Cognos prompt user interface, you see the following prompt page.

Figure 4. Select & Search prompts request

After you type keywords into the search box, the selected keywords are submitted to the IBM Cognos BI server and the **Choice** list box is populated with all possible values that match one or more of the keywords. The values that you choose from the list box are submitted to the server when you click **Finish** and can then be retrieved as with any other prompted report.

If you are collecting prompts with an alternative interface, the response from the `reportPrompts` resource type (REST application) or the `getReportPrompts` method call (SOAP application) is an LDX file that contains the following snippet.

```
<item>
  <p_srch>
   <id>_P288725932</id>
   <ref>R5</ref>
   <style>S5</style>
   <pname>product</pname>
   <rows>5000</rows>
   <mtchany>false</mtchany>
   <mtchall>false</mtchall>
   <showopt>false</showopt>
   <cname>Product</cname>
  </p_srch>
</item>
```

Use the `reprompt` secondary request to submit search values and other parameters. For example, by using the REST interface you can submit

```
<gateway>/rds/sessionOutput/conversationID/<conversation_ID>
/reprompt?srchval=edge&swsID=_P288725932&pname=product
&nocase=true&mtchAny=true
```

to use `edge` as the search keyword. A portion of the response is shown here.

```
<item>
  <p_srch>
   <id>_P288725932</id>
   <ref>R6</ref>
   <style>S6</style>
   <pname>product</pname>
   <rows>5000</rows>
   <mtchany>true</mtchany>
   <mtchall>false</mtchall>
   <showopt>false</showopt>
   <srchval>edge</srchval>
   <cname>Product</cname>
   <selOptions>
    <sval>
     <use>Bear Edge</use>
     <disp>Bear Edge</disp>
    </sval>
    <sval>
     <use>Bear Survival Edge</use>
     <disp>Bear Survival Edge</disp>
    </sval>
    <sval>
     <use>Double Edge</use>
     <disp>Double Edge</disp>
    </sval>
    <sval>
     <use>Edge Extreme</use>
     <disp>Edge Extreme</disp>
    </sval>
    <sval>
     <use>Single Edge</use>
     <disp>Single Edge</disp>
    </sval>
   </selOptions>
  </p_srch>
</item>
```

You can reprompt again with a different search string to collect different prompts or use one of returned values and run the report.

REST example

Submit the following URL to the Cognos Business Intelligence server:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportPrompts
/report/i4C970FA60C364BB0AB24A832F034A886
```

The response is an LDX page containing information about the prompt, in this case, the comparison country. Note the URL associated with the response page:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput
/conversationID/i689320782E8D45B1A4A18A9D4144504B
```

In order to get the search values that match the keyword `edge`, submit the following URL, using the `reprompt` secondary request.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput/conversationID/
i689320782E8D45B1A4A18A9D4144504B/reprompt?
srchval=edge&swsID=_P288725932&pname=product&nocase=true&mtchAny=true
```

The response is an LDX page containing the search terms that match the keyword `edge`. To select the search result `Bear Edge`, submit the following URL:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput/conversationID/
i689320782E8D45B1A4A18A9D4144504B/forward?
p_product=Bear Edge
```

The response from the Cognos Business Intelligence server is a `promptAnswers` response that you can use to submit to run the report for the `Bear Edge` product category.

SOAP example

For an example of using select & search prompts in SOAP applications, see the SearchPromptValue Java and C# sample programs.

Collecting prompts from multiple prompt pages

Some reports have multiple prompt pages. An example of such a report is the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Employee Training by Year**.

If you use the IBM Cognos prompt user interface, the prompt page has **Back** and **Next** buttons, enabling you to provide responses to the prompts.

A single prompt with navigation buttons is shown here.

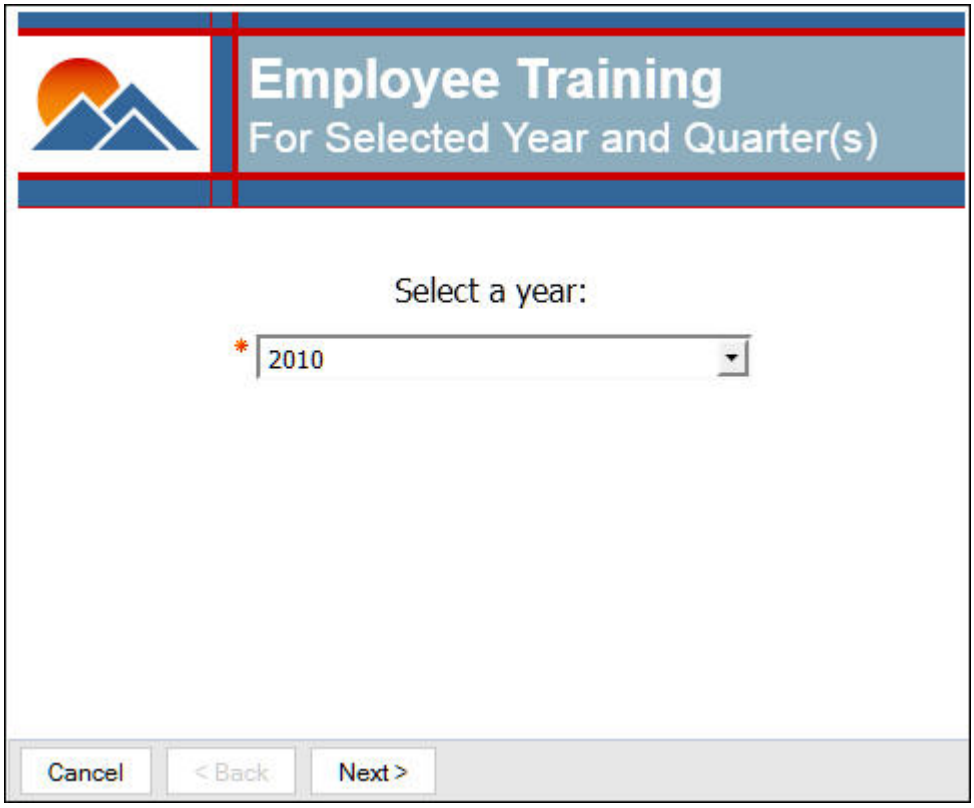

Figure 5. Single prompt with navigation buttons

After you submit prompt values, they can be retrieved as with any other prompted report.

If you are collecting prompts yourself, retrieve the prompt information associated with the first prompt page using the procedure described in "Collecting prompts with an alternate interface" on page 44.

To retrieve subsequent pages of prompt information, use the `forward` secondary operation, including the responses to prompts on the current page in the request. When the `forward` secondary operation for the last page has been submitted, the response from the server will include all prompt values.

Examples using the REST and SOAP interfaces are shown here.

REST example

Submit the following URL to the BI server:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportPrompts
/report/i20b89f17f37c4818a2e807614ccbb11f
```

The response is an LDX page containing information about the first prompt, in this case, the year. Note the URL associated with the response page:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput
/conversationID/i2f24a760fae047958356a7a0a9e99c1f
```

In order to get the second prompt page request, for the quarter, submit the following URL, using the `forward` secondary request along with the selected response to the prompt for the year (in this case, 2010).

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput/conversationID/
i2f24a760fae047958356a7a0a9e99c1f/forward?
p_P_Year=[Employee training].[Time].[Time].[Year]->[Time].[2010]
```

The response is an LDX page containing the prompt request for the quarter of 2010 to be selected. When you have selected the quarter, submit the following URL to the server.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput/conversationID/
i2f24a760fae047958356a7a0a9e99c1f/forward?
p_P_Quarter=[Employee training].[Time].[Time].[Quarter]->[Time].[2010].[20103]
```

Since there are only two prompt pages, the response from the IBM Cognos BI server is a `promptAnswers` response as in the case with non-cascading prompt pages.

SOAP example

Submit a `getReportPrompts` request to the BI server. When the asynchronous conversation completes, the output from the server includes a `PromptAnswerOutput` object, which contains the `promptValues` objects, that contain the prompt information for the first prompt page.

After you have determined the prompt answers for the first prompt page, submit them to the BI server in a `forward` secondary request. the server response will include the `promptValues` for the second page of prompts. The server will return a `PromptAnswerOutput` object for the second prompt page.

When there are no more prompt pages to process, the response from the `forward` request will include `promptValues` for all the prompts in the report, which can now be run.

Notes

- Although this sample report uses cascading prompts, the procedures are no different if the report has multiple, independent prompts.
- If default values are available for prompts, you can use the `finish` secondary request to skip any subsequent prompt pages. This is equivalent to pressing the **Finish** button on a prompt page.
- You can use the `back` secondary request to return to the previous prompt page. This is equivalent to pressing the **Back** button on a prompt page.
- You can use the `reprompt` secondary request to submit one or more prompt responses and have the current prompt page refreshed instead of moving to the next prompt page. Use this request if the prompt page contains cascading prompts and you need to submit a prompt response before getting the next prompt request.

Collecting cascading prompts from a single prompt page

Some reports collect source and cascading prompts from a single prompt page. An example of such a report is **Public Folders > Samples_PowerCube > Cubes > Sales and Marketing (cube) > Report Studio Report Samples > Selected Retailer Country**.

If you use the IBM Cognos prompt user interface you will get the following prompt page.

Figure 6. Cascading prompts request

After you select the **Select Baseline Retailer Country** value from the drop-down box, the selected value is automatically submitted to the IBM Cognos Business Intelligence server and the **Select Retailer Countries To Compare To** list box is populated with all possible values. The values you choose are submitted to the server when you click the **Finish** button and can then be retrieved as with any other prompted report.

Part of the LDX document is shown here:

```
<document xmlns="http://www.ibm.com/xmlns/prod/cognos/layoutData/200904">
...
  <item>
    <p_value>
      <id>_P2884238914</id>
      <ref>R11</ref>
      <style>S11</style>
      <pname>comparison country</pname>
      <rows>5000</rows>
      <selectUI>DROP_DOWN</selectUI>
      <auto>true</auto>
      <cname>Retailer country</cname>
      <autocascade>true</autocascade>
      <selOptions>
        <sval>
          <use>[sales_and_marketing].[Retailers].[Retailers].[Retailer country]
            ->:[PC].[@MEMBER].[90001]</use>
          <disp>United States</disp>
        </sval>
        ...
        <sval>
          <use>[sales_and_marketing].[Retailers].[Retailers].[Retailer country]
           ->:[PC].[@MEMBER].[90021]</use>
          <disp>Spain</disp>
        </sval>
      </selOptions>
    </p_value>
  </item>
  ...
  <item>
    <p_value>
      <id>_P3964241042</id>
      <ref>R13</ref>
      <style>S13</style>
      <pname>compared countries</pname>
      <multi>true</multi>
      <cascadeon>comparison country</cascadeon>
      <rows>5000</rows>
      <disabled>true</disabled>
      <selectUI>LIST_BOX</selectUI>
```

```
      <cname>Retailer country</cname>
      <selOptions/>
    </p_value>
  </item>
```

The first prompt item, comparison country, has the `autocascade` element set to `true`. This means that when this prompt is submitted, the prompt page should be refreshed with possible values for the compared countries prompt. The compared countries prompt has `disabled` set to `true`. This specifies that this prompt is disabled until the first prompt has been selected. When a value for the **Retailer country** is submitted to the BI server, a prompt page for the compared countries can be retrieved, parts of which are shown here.

```
<p_value>
  <id>_P2831558871</id>
  <ref>R4</ref>
  <style>S4</style>
  <pname>compared countries</pname>
  <multi>true</multi>
  <rows>5000</rows>
  <selectUI>LIST_BOX</selectUI>
  <cname>Retailer country</cname>
  <selOptions>
    <sval>
      <use>[sales_and_marketing].[Retailers].[Retailers].[Retailer country]
       ->:[PC].[@MEMBER].[90009]</use>
      <disp>Australia</disp>
    </sval>
    <sval>
      <use>[sales_and_marketing].[Retailers].[Retailers].[Retailer country]
       ->:[PC].[@MEMBER].[90019]</use>
      <disp>Austria</disp>
    </sval>
    ...
    <sval>
      <use>[sales_and_marketing].[Retailers].[Retailers].[Retailer country]
       ->:[PC].[@MEMBER].[90001]</use>
      <disp>United States</disp>
    </sval>
  </selOptions>
</p_value>
```

A value for the comparison country prompt can then be submitted using the `forward` secondary request, setting the prompt name to **comparison country** and the prompt value to **[sales_and_marketing].[Retailers].[Retailers].[Retailer country]->:[PC].[@MEMBER].[90002]**.

The response is an LDX document that contains the possible values for the compared countries prompt which can then be submitted to run the report.

REST example

Submit the following URL to the Cognos Business Intelligence server:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportPrompts
/report/i3D3D474CA3804E5CB0D56D7752950303
```

The response is an LDX page containing information about the first prompt, in this case, the comparison country. Note the URL associated with the response page:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput
/conversationID/iCE05F3B0D14C4469B122F26DEBAFCCB5
```

In order to get the second prompt page request, for the comparison country, submit the following URL, using the `forward` secondary request along with the selected response to the prompt for the year (in this case, the United States).

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput/conversationID/
iCE05F3B0D14C4469B122F26DEBAFCCB5/forward?
p_comparison country=[sales_and_marketing].[Retailers].[Retailers].[Retailer country]
->:[PC].[@MEMBER].[90001]
```

The response is an LDX page containing the prompt request for the compared country to be selected. When you have selected the country, in this case Canada, submit the following URL to the server.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/sessionOutput/conversationID/
iCE05F3B0D14C4469B122F26DEBAFCCB5/forward?
p_compared countries=[sales_and_marketing].[Retailers].[Retailers].[Retailer country]
->:[PC].[@MEMBER].[90002]
```

Since there are only two prompts required, the response from the Cognos Business Intelligence server is a `promptAnswers` response that you can use to submit to run the report.

Running a report with prompts

After collecting the prompt answers, you can run the report by submitting the prompt answers with the request.

REST example

The report can be run by including the `promptAnswers` element in an `xmlData` option as shown here.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/report/
ia195960a5e77488cb4583d74b56c78d6?xmlData=
<promptAnswers><promptValues><name>pYear</name><values><item><SimplePValue>
<inclusive>true</inclusive><useValue>[Employee summary].[Time]
    .[Time].[Year]-%26gt;[Time].[2006]</useValue>
<displayValue>2006</displayValue></SimplePValue></item></values></promptValues>
<promptValues><name>pRegion</name><values><item><SimplePValue><inclusive>true</inclusive>
<useValue>[Employee summary].[Employee by region].[Employee by region]
    .[Branch region]-%26gt;[Employee by region].[740]</useValue>
<displayValue>Asia Pacific</displayValue></SimplePValue></item></values>
   </promptValues></promptAnswers>
```

Alternatively, you can use a simplified expression using the `p_parameter` option.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/report/
ia195960a5e77488cb4583d74b56c78d6?p_pYear=
[Employee summary].[Time].[Time].[Year]->[Time].[2006]&p_pRegion=
[Employee summary].[Employee by region].[Employee by region]
   .[Branch region]->[Employee by region].[740]
```

In this case the value of each prompt is the value of the corresponding `useValue` element of the `promptAnswers` response.

SOAP example

To run a report with prompts, include `promptValues` objects (generic applications) or a `PromptAnswersType` object (report-specific application) in the report request.

Drilling up and down in reports

You can drill up or down in an existing report session if the underlying report supports drill operations. You can get drill information from LDX output (for both SOAP and REST applications) and from HTML output (for REST applications only).

The sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Customer Returns and Satisfaction** supports drilling up and drilling down. See "Sample drill-up and drill-down report in LDX format" on page 70 for a description of the LDX elements that support drilling up and drilling down.

REST example for LDX output

In a REST application use the `drill` secondary resource type to perform a drill up or down operation. An example is shown here based on the above report:

```
drill?contextId=50&direction=DOWN
```

SOAP example for LDX output

In a generic SOAP application, use the `drill` secondary method to perform a drill up or drill down, populating the `direction` and `contextID` of the `DrillRequest` as above.

REST example for HTML output

By default, reports returned in HTML format do not include drill-up or drill-down information. Use the drillurls option to include drill-up or drill-down information in the HTML output. For the report above, drill-up or drill-down information will be included for **1 for 1 Sports shop** as shown here.

```
<td class="S66">
  <span class="S65" drills='{"down":"./drill?contextId=50&direction=DOWN",
 "up":" ./drill?contextId=50&direction=UP"}'>1 for 1 Sports shop</span>
</td>
```

In a REST application use the `drill` secondary resource type to perform a drill up or down operation. An example is shown here based on the above report snippet:

```
drill?contextId=50&direction=DOWN
```

Drilling through to another report

You can drill through from one report to another using the Mashup Service REST interface. You can get drill-through information from LDX and HTML outputs.

This example is based on a drill-through link from the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Employee Satisfaction 2006** to the sample report **Public Folders > Samples > Models > GO Data Warehouse (query) > Report Studio Report Samples > Compensation (hidden)**. See thew topic about using drill-through access in the *IBM Cognos Business Intelligence Report Studio User Guide* for more information on this example. See "Sample drill-through definitions in LDX format" on page 71 for a description of the LDX elements that support drilling up and drilling down.

By default, reports returned in HTML format do not include drill-through information. Use the drillurls option to include drill-through information in the HTML output.

The equivalent HTML snippet for the drill-through definition for this report is as follows.

```
<td ctx="110" class="S51">
  <span class="S50" drills="{"drillthroughs":"
  /rds/reportData/searchPath/content/folder[@name='Samples']
  /folder[@name='Models']  /package[@name='GO Data Warehouse (query)']
  /folder[@name='Report Studio Report Samples']
  /report[@name='Compensation (hidden)']
  ?p_pPosition=[Employee survey].[Position-department].
```

```
 [Position-department].[Position-department name (level 3)]->
 [Position-department].[100].[200].[300]"}">
 Human Resources</span>
</td>
```

The drillDefinitions element contains a drill element that contains the search path (targetPath) of the report to drill through to and parameter name (name) to use in the prompt request.

To drill through to the **Compensation (hidden)** report, run a report using the targetPath value as the search path and including a prompt with a prompt name of pPosition and a value of Human Resources.

The drillThrough JavaScript sample program illustrates the use of drill through using the above reports.

Embedding images in HTML output

You can have HTML output from the IBM Cognos Mashup Service contain image data inline instead of containing URLs to images on the server.

By default, HTML output from the Cognos Mashup Service contains links to the IBM Cognos Business Intelligence server for any images contained in the HTML output. If embedImages is true, any images in the HTML output will be contained inside the HTML content as data, so that a link to the server is not required in order to render the report.

The encoding scheme for the embedImages option adheres to the IETF RFC 2397 “data” URL scheme standard.

Using a URL to display a report in IBM Cognos Viewer

You can use the IBM Cognos Mashup Service to obtain a URL that lets you run a report and display its output in IBM Cognos Viewer.

To retrieve the URL that runs a report and deploys it in IBM Cognos Viewer, use the cognosurl REST resource type or the getCognosURL SOAP method. the response from the Business Intelligence server includes a URL that can be used to display the report in IBM Cognos Viewer.

Retrieving a relative prompt page URL

If the internal dispatcher URI of your IBM Cognos Business Intelligence server is hidden behind a firewall, you can receive the prompt page URL based on the external gateway URI instead.

Use the useRelativeURL option (REST applications) or the useRelativeURL option (SOAP applications) to receive the prompt page URL based on the external gateway URI.

A request to run a report that requires prompts returns a response that contains a URL to a page that you use to request prompts from a user. The prompt request URL is based on the internal dispatcher URI of your Cognos Business Intelligence server. If this address is hidden behind a gateway, you can retrieve a relative URL instead that you can use to form a URL based on the external gateway URI.

For example, the response to the following URL:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/report/
i3D3D474CA3804E5CB0D56D7752950303?useRelativeURL=true
```

includes a url as follows:

```
<rds:url>/ibmcognos/cgi-bin/cognos.cgi?b_action=xts.run&m=ccs/ccs_prompt.xts
&promptID=i8D98FF4D826441FD944F6DB26A8A5BB6&ui.object=
storeID("i3D3D474CA3804E5CB0D56D7752950303")</rds:url>
```

It is up to your application to form a correct URL from this relative URL.

Chapter 7. Understanding the Layout Data format

A Layout Data (LDX) document instance is an XML document that is an abstraction of rendered IBM Cognos content. This content is referred to in this document as a resource and can be a:

- Report Studio report
- Query Studio query
- Analysis Studio analysis
- PowerPlay report
- PowerPlay Studio report

The Mashup Service provides these IBM Cognos resources in the LDX format, a consistent format that can then be rendered in other applications.

The LDX instance includes all of the data and formatting from the resource, including list grouping, crosstab dimensions and styling information.

Please note that the LDX instance that represents a specific IBM Cognos resource may change, but will always comply with the `layoutDataXMLV2.xsd` schema. Do not assume that LDX output will always be exactly the same from one run to the next.

The LDX schema file in an IBM Cognos Business Intelligence installation can be found at *installation_location*/templates/ccs/wsdl/layoutDataXMLV2.xsd. You can open this file in an XML schema editor to examine its structure. For reference information, see Chapter 16, "Layout Data (LDX) schema reference," on page 179.

You can easily obtain the LDX representation of a report by using the REST interface to run the report. The LDX document is rendered in a Web browser and you can view the XML in the browser or save the document and examine it using an XML editor. See "Retrieving report data" on page 22 for more information.

The LDX samples used blow are based on the Great Outdoors Company sample databases and reports included in IBM Cognos BI.

Basic structure of a layout data document

Each layout data document contains a representation of all the data and layout information from the source resource output.

The root element in a layout data document is either `document`, if the layout data document represents the entire source resource, or `filterResultSet` if the layout data document represents a filtered subset of the source resource.

An LDX document contains the following elements:

- `secondaryOperations` elements that indicate which secondary operations are allowed. See "Secondary operations" on page 60 for more information.
- A `versionBase` element if the report is a saved report.

- `locationReference` elements at the beginning of a layout data document that specify locations in the source resource. These locations are referenced by layout elements in the layout data document. See "Location references" for more information.
- The report output is contained in
  - `pages` elements if the root element is `document`. There will be one `pages` element for each page in the source resource. See "Report output structure" on page 61 for more information.
  - `filterResult` elements if the root element is `filterResultSet`. There will be one `pages` element for each selected report element. See "Filter output structure" on page 64 for more information.
- `drillDefinitions` elements if there are drill throughs defined in the report. See "Sample drill-through definitions in LDX format" on page 71 for more information.
- `styleGroup` elements that contain style information for the report content. See "Style information" on page 61 for more information.

Secondary operations

Many IBM Cognos Mashup Service applications require multiple interactions with the IBM Cognos Business Intelligence server for a single report. These interactions include:

- Retrieving report output one page at a time.
- Collecting report prompts.
- Drilling up or down in a report.

The contents of the `value` child element of the `secondaryOperations` element indicate which secondary operations are available for the report.

See the chapters on developing mashup applications for information about how to send secondary requests to the BI server.

Location references

You can specify locations in the source based on the report specification.

`locationReference` elements are referenced by elements further down in the layout data document. An example is shown here.

```
<locationReference>
  <ref>R40</ref>
  <loc>./layouts/layout/reportPages/page/pageBody/contents/list</loc>
</locationReference>
```

In this example, the `ref` element is referenced in the body of the report in the following way:

```
<lst>
  <id>List1</id>
  <ref>R40</ref>
```

The `ref` element provides the report specification location of the `lst` element in the report. See the *IBM Cognos Software Development Kit Developer Guide* for more information about report specifications.

Style information

Styles are defined within each layout data document in a series of `styleGroup` elements. Each `styleGroup` element represents one style used in the document and has a `name` element to specify a name. This name is referenced in a layout element using the `style` child element.

For example, the following cell,

```
<cell>
  <ref>R9</ref>
  <style>S9</style>
  <item>
    <txt>
      <ref>R8</ref>
      <ctx>12:11</ctx>
      <style>S8</style>
      <val>Lanterns</val>
      <valTyp>text</valTyp>
      <fmtVal>Lanterns</fmtVal>
    </txt>
  </item>
</cell>
```

references the following style:

```
<styleGroup>
    <name>S9</name>
    <font>
      <family>Tahoma</family>
      <size>
        <val>8.000000</val>
        <units>PT</units>
      </size>
    </font>
    <textStyle>
      <strictLineBreaking>true</strictLineBreaking>
    </textStyle>
    <boxStyle>
      ...
    </boxStyle>
    <fgColor>
      <Red>0</Red>
      <Green>0</Green>
      <Blue>0</Blue>
    </fgColor>
    <vAlign>TOP</vAlign>
  </styleGroup>
```

Report output structure

This sample illustrates the main parts of a layout data document.

This example is based on the sample report "Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Employee Satisfaction 2012."

The REST URL to run this report and obtain the LDX output is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData
/path/Public%2520Folders/Samples/Models
/GO%2520Data%2520Warehouse%2520%2528analysis%2529
/Report%2520Studio%2520Report%2520Samples/Employee%2520Satisfaction%25202012
```

The HTML output of the report is shown here.

Figure 7. Employee Satisfaction 2012 report

This report output displays the following items:

- A header containing a graphic and the text **Employee Satisfaction by Department 2012**.
- A body containing two charts, a crosstab, and a sentence with a single data element.
- A footer containing the report run date and time, and the page number.

An abbreviated version of the `pages` element is shown here.

```
<pages>
  <page>
    <id>Page1</id>
    <header>
                    <item>
                      <txt>
                        <id>FirstPage_ReportTitle2121</id>
                        ...
                      </item>
                    </blk>
                  </item>
                  <item>
                    <blk>
                      <ref>R11</ref>
                      <style>S11</style>
                      <item>
                        <txt>
                          <id>FirstPage_Subtitle1121</id>
                          ...
    </header>
    <body>
      <item>
        <tbl>
          <ref>R68</ref>
          <style>S70</style>
          <trow>
            <style>S30</style>
            <tcell>
              <ref>R27</ref>
              <style>S27</style>
```

```
              <item>
                <cht>
                  <id>Combination Chart - survey topic scores by department</id>
                  ...
                </cht>
              </item>
              <item>
                <blk>
                ...
                    <sngl>
                      <id>Singleton1</id>
                      ...
                      <item>
                        <txt>
                          <ref>R23</ref>
                          <style>S23</style>
                          <val>-0.121043027052657</val>
                          <valTyp>number</valTyp>
                          <fmtVal>-12.1%</fmtVal>
                          <fmtPatrn>#,##0.0%</fmtPatrn>
                          <exclPatrn>\#\,\#\#0\.0%</exclPatrn>
                        </txt>
                      </item>
                    </sngl>
                </item>
                ...
              </blk>
            </item>
          </tcell>
<tcell>
            <ref>R29</ref>
            <style>S29</style>
            <item>
              <cht>
                <id>Combination Chart - survey scores and benchmark</id>
                ...
              </cht>
            </item>
          </tcell>
</trow>
<trow>
          <style>S69</style>
          <tcell>
            <ref>R66</ref>
            <style>S68</style>
            <cspan>2</cspan>
            <item>
              <blk>
                <ref>R32</ref>
                <style>S32</style>
                <item>
                  <txt>
                    <ref>R31</ref>
                    <style>S31</style>
                    <val>Employee rankings and terminations by department</val>
                    <valTyp>text</valTyp>
                  </txt>
                </item>
              </blk>
            </item>
            <item>
              <ctab>
                <id>Crosstab1</id>
                ...
              </ctab>
            </item>
          </tcell>
        </trow>
      </tbl>
    </item>
  </body>
  <footer>
  ...
  </footer>
 </page>
</pages>
```

A `tbl` element is used to layout the principal report parts. The first `tcell` contains a `cht` with the **Survey topic scores by department** combination chart. In the same

cell following the chart is a `blk` element containing **Customer Service average score is -12.1% compared to the company average**. A `sngl` element contains the calculated value in the sentence. The second `tcell` contains the **Survey topic scores, targets and industry standard** combination chart.

The last `tcell` spans two cells and contains a `blk` element containing the crosstab title followed by a `ctab` element that contains the crosstab. See "Sample crosstab in LDX format" on page 68 for more information on crosstabs.

Reports with multiple pages

Many reports span several pages when viewed in IBM Cognos Viewer. You can retrieve reports a page at a time, or you can retrieve the entire report, using the Mashup Service. When retrieving an entire report, you can choose to receive the report in one page or split into separate pages corresponding to the pagination displayed by IBM Cognos Viewer.

For example, the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Budget vs. Actual** spans three pages when viewed in Cognos Viewer. This report can be retrieved in one page using the following URL.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/path
/Public%20Folders/Samples/Models
/GO%20Data%20Warehouse%20%28analysis%29/Report%20Studio%20Report%20Samples
/Budget%20vs.%20Actual?includeLayout=true
```

The report output will contain a single `pages` that contains the entire report. If the report is retrieved using the following URL, it will contain three `pages` elements, each containing the same data as each page of the Cognos Viewer output.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/path
/Public%20Folders/Samples/Models
/GO%20Data%20Warehouse%20%28analysis%29/Report%20Studio%20Report%20Samples
/Budget%20vs.%20Actual?includePageBreaks=true&includeLayout=true
```

Filter output structure

You can request a filtered output containing parts of a report output.

Using the same report as in "Report output structure" on page 61, you can request a filtered output containing one of the combination charts and the crosstab.

The REST URL to run this report and obtain the LDX output is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData/path
/Public%2520Folders/Samples/Models
/GO%2520Data%2520Warehouse%2520%2528analysis%2529
/Report%2520Studio%2520Report%2520Samples/Employee%2520Satisfaction%25202006
?selection=Combination
Chart - survey scores and benchmark;Crosstab1
```

The following sample XML is the above filtered report in LDX format. This sample XML does not include the entire layout data document. Removed sections are represented by ellipses (...).

```
<filterResultSet...
  <filterResult>
    <filterType>OBJECT_ID</filterType>
    <filterValue>Combination Chart - survey scores and benchmark</filterValue>
    <reportElement>
      <cht>
        <id>Combination Chart - survey scores and benchmark</id>
        ...
```

```
    </cht>
  </reportElement>
</filterResult>
<filterResult>
  <filterType>OBJECT_ID</filterType>
  <filterValue>Crosstab1</filterValue>
  <reportElement>
    <ctab>
      <id>Crosstab1</id>
      ...
    </ctab>
  </reportElement>
</filterResult>
...
```

Since this is a filtered report, the root element is a `filterResultSet`element. This element contains two `filterResult` elements, one for each requested report part. The value of the `filterType` elements is `OBJECT_ID`, indicating that we filtered on a named report part, which is contained in the value of `filterValue` elements.

Each `reportElement` element contains a requested chart or crosstab.

Sample grouped list in LDX format

This sample illustrates the main parts of a layout data document for a grouped list report.

This example is based on the sample report "Public Folders > Samples > Models > GO Data Warehouse (query) > SDK Report Samples > Product Quantity and Price."

The REST URL to run this report is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData/path
/Public%2520Folders/Samples/Models/GO%2520Data%2520Warehouse%2520%2528analysis%2529
/Report%2520Studio%2520Report%2520Samples/Employee%2520Satisfaction%25202006
```

The HTML output of the grouped list report is shown here.

Product type	Product	Quantity	Unit sale price
Binoculars	Opera Vision	82,016	$109.44
	Ranger Vision	251,865	$164.91
	Seeker 35	296,455	$99.17
	Seeker 50	159,701	$126.31
	Seeker Extreme	112,199	$167.12
	Seeker Mini	172,851	$75.68
Binoculars			
Climbing Accessories	Firefly Charger	302,114	$51.82
	Firefly Climbing Lamp	213,370	$37.49
	Firefly Rechargeable Battery	1,332,666	$7.35
	Granite Belay	259,975	$66.23
	Granite Carabiner	3,146,194	$3.81
	Granite Chalk Bag	202,090	$17.85
	Granite Pulley	393,842	$36.92
Climbing Accessories			
Cooking Gear	TrailChef Canteen	965,723	$11.96
	TrailChef Cook Set	813,780	$44.72
	TrailChef Cup	1,812,123	$3.36
	TrailChef Deluxe Cook Set	442,136	$111.07
	TrailChef Double Flame	245,559	$138.59
	TrailChef Kettle	2,336,950	$11.66
	TrailChef Kitchen Kit	866,669	$22.76

Figure 8. HTML output of the grouped list report

The following sample XML is the above grouped list report in LDX format. This sample XML does not include the entire layout data document. Removed sections are represented by ellipses (...).

```
<document...
  ...
  <pages>
    <page>
      <id>Page1</id>
      <header>
      ...
      </header>
      <body>
        <item>
          <lst>
            <id>List1</id>
            <ref>R40</ref>
            <style>S40</style>
            <colTitle>
              <ref>R20</ref>
              <style>S20</style>
              <item>
                <txt>
                ...
                  <val>Product type</val>
                  ...
                </txt>
              </item>
            </colTitle>
            ...
            <group>
```

```
 <grp>
 <di>Product type</di>
<dv>Binoculars</dv>
                <row>
                  <cell>
                    <ref>R29</ref>
                    <style>S29</style>
                    <rspan>6</rspan>
                    <item>
                      <txt>
                      ...
                        <val>Binoculars</val>
                        ...
                      </txt>
                    </item>
                  </cell>
                  <cell>
                    <ref>R31</ref>
                    <style>S31</style>
                    <item>
                      <txt>
                      ...
                        <val>Opera Vision</val>
                        ...
                      </txt>
                    </item>
                  </cell>
                  <cell>
                    <ref>R33</ref>
                    <style>S33</style>
                    <item>
                      <txt>
                      ...
                        <val>82016</val>
                        ...
                      </txt>
                  </cell>
                  <cell>
                    <ref>R35</ref>
                    <style>S35</style>
                    <item>
                      <txt>
                      ...
                        <val>109.436407</val>
                        ...
                      </txt>
                    </item>
                  </cell>
                </row>
                <row>
                  <cell>
                    <ref>R31</ref>
                    <style>S31</style>
                    <item>
                      <txt>
                      ...
                        <val>Ranger Vision</val>
                        ...
                      </txt>
                    </item>
                  </cell>
                  <cell>
                  ...
                <footer>
                  <row>
                    <cell>
                      <ref>R38</ref>
                      <style>S38</style>
                      <cspan>4</cspan>
                      <item>
                        <txt>
                        ...
                          <val>Binoculars</val>
                          ...
                        </txt>
                      </item>
                    </cell>
                  </row>
                </footer>
                <depth>1</depth>
```

```
            </grp>
            <grp>
              <di>Product type</di>
              <dv>Climbing Accessories</dv>
              <row>
              ...
            </grp>
            <depth>0</depth>
          </group>
        </lst>
      </item>
    </body>
    <footer>
    ...
    </footer>
  </page>
</pages>
..
```

The above sample XML includes one page of rendered content, represented by the `page` element. The list is located on the body of the page and is represented by the `lst` element. Within the list there are four columns, each with a `colTitle` element for the titles.

Because this is a grouped list, each value from the grouped data item in the report output appears in the layout data document, represented by a `grp` element. The `di` child element specifies the data item name, and the `dv` child element specifies the data item value.

Sample crosstab in LDX format

This sample illustrates the main parts of an layout data document for a crosstab report.

This example is based on the sample report "Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Returns by Order Method."

This report is run using a filter to return just the crosstab. The REST URL to run this report is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData/path
/Public%20Folders/Samples/Models/GO%20Data%20Warehouse%20%28analysis%29
/Report%20Studio%20Report%20Samples
/Returns%20by%20Order%20Method?selection=Crosstab1
```

Part of the HTML output of the crosstab report is shown here.

Return quantity	Defective product	Incomplete product	Unsatisfactory product	Wrong product shipped	2004	2005	2006	2007
Camping Equipment	36,046	57,043	61,549	52,199	33,817	50,769	57,013	65,238
Golf Equipment	6,221	5,817	10,068	13,105	6,287	6,731	14,047	8,146
Mountaineering Equipment	15,492	8,176	27,012	30,765		20,635	34,530	26,280
Outdoor Protection	79,351	374	230,481	7,210	217,622	62,349	30,670	6,775
Personal Accessories	46,290	26,808	34,934	98,875	43,834	52,190	63,056	47,827

Figure 9. Crosstab report sample

The following sample XML is the above crosstab report in LDX format. This sample XML does not include the entire layout data document. Removed sections are represented by ellipses (...).

```
<filterResultSet...
  <filterResult>
    <filterType>OBJECT_ID</filterType>
    <filterValue>Crosstab1</filterValue>
    <reportElement>
      <ctab>
        <id>Crosstab1</id>
        <ref>R13</ref>
        <style>S13</style>
        <corner>
          <ref>R4</ref>
          <ctx>1</ctx>
          <style>S4</style>
          <item>
            <txt>
              <ref>R1</ref>
              <style>S1</style>
              <val>Return quantity</val>
              <valTyp>text</valTyp>
            </txt>
          </item>
        </corner>
        <column>
          <name>
            <ref>R5</ref>
            <ctx>2</ctx>
            <style>S5</style>
            <item>
              <txt>
                <ref>R2</ref>
                <ctx>2</ctx>
                <style>S2</style>
                <val>Defective product</val>
                <valTyp>text</valTyp>
              </txt>
            </item>
          </name>
          <start>0</start>
          <size>1</size>
        </column>
        <column>
          <name>
            <ref>R5</ref>
            <ctx>3</ctx>
            <style>S5</style>
            <item>
              <txt>
                <ref>R2</ref>
                <ctx>3</ctx>
                <style>S2</style>
                <val>Incomplete product</val>
                <valTyp>text</valTyp>
              </txt>
            </item>
          </name>
          <start>1</start>
          <size>1</size>
        </column>
        ...
        <row>
          <name>
            <ref>R9</ref>
            <ctx>10</ctx>
            <style>S9</style>
            <item>
              <txt>
                <ref>R8</ref>
                <ctx>10</ctx>
                <style>S8</style>
                <val>Camping Equipment</val>
                <valTyp>text</valTyp>
              </txt>
            </item>
          </name>
          <start>0</start>
          <size>1</size>
        </row>
        ...
        <table>
          <row>
```

```
<cell>
  <ref>R11</ref>
  <ctx>11::10::2</ctx>
  <style>S11</style>
  <item>
    <txt>
      <ref>R10</ref>
      <style>S10</style>
      <val>36046</val>
      <valTyp>number</valTyp>
      <fmtVal>36,046</fmtVal>
      <fmtPatrn>#,##0</fmtPatrn>
      <exclPatrn>\#\,\#\#0</exclPatrn>
    </txt>
  </item>
</cell>
...
```

The crosstab is represented by the `ctab` element. The text that appears in the corner of the crosstab, **Return quantity**, is represented by the `corner` element. The header values that appear at the top of each column are represented by `txt` elements inside a series of `column` elements. The header values for the row dimension appear in a series of `row` elements.

The `start` element in each row and column specifies the row or column to which the header corresponds. For example, the value in the first column header is 2004, and the `start` value is `0`. This means that **Defective product** is the header for the first column in the data table. The `start` value for the second column header is 1, and the `start` values for each column header increase incrementally from left to right. The `size` element specifies the row or column span. In this example, each column and row header span a single row or column, therefore the value of the `size` element is always 1.

The measure values that appear in the centre of the crosstab are represented by a `table` element.

Sample drill-up and drill-down report in LDX format

In some reports, you can drill up or drill down to move through a hierarchy of information.

The ability to drill up or drill down is indicated by the presence of `drillAction` elements.

This example uses the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Customer Returns and Satisfaction**.

This portion of the report in IBM Cognos Viewer displays a drillable report element.

Figure 10. Drillable report element

The following sample XML is the above report output in LDX format. This sample XML does not include the entire layout data document. Removed sections are represented by ellipses (...).

```
...
  <di>Retailer name</di>
<dv>1 for 1 Sports shop</dv>
<grp>
  <di>Retailer type</di>
  <dv>Outdoors Shop</dv>
  <grp>
    <di>Retailer topic score</di>
    <dv>0.7179375</dv>
    <row>
      <cell>
        <ref>R66</ref>
        <style>S66</style>
        <item>
          <txt>
            <ref>R65</ref>
            <ctx>50</ctx>
            <style>S65</style>
            <drillAction>
              <direction>UP</direction>
            </drillAction>
            <drillAction>
              <direction>DOWN</direction>
            </drillAction>
            <val>1 for 1 Sports shop</val>
            <valTyp>text</valTyp>
            <fmtVal>1 for 1 Sports shop</fmtVal>
          </txt>
        </item>
      </cell>
      ...
```

The `drillAction` element indicates that we can drill UP or DOWN on **1 for 1 Sports shop**.

Sample drill-through definitions in LDX format

Using drill-through access, you can move from one report to another within a session while maintaining your focus on the same piece of data.

Drill throughs are defined in two places in each layout data document. The first place is the drill-through definition, the second is the drill-through instance.

This example is based on a drill through in the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Employee Satisfaction 2006**. See the topic about using drill-through access in the *IBM Cognos Business Intelligence Report Studio User Guide* for more information on this report.

The drill-through definitions are defined in `drill` elements that are children of the `drillDefinitions` element. Each definition specifies the general information for a drill through, including whether or not it is prompted, the list of parameters available, and whether or not to show the target resource in a new window. Each `drill` element has a `drillRef` child element that specifies the identifier for the drill-through definition. For example:

```
<drillDefinitions>
  <drill>
    <drillRef>0</drillRef>
    <label>DrillToHiddenRep</label>
    <showInNewWindow>false</showInNewWindow>
    <sendFilterContext>false</sendFilterContext>
    <prompt>no</prompt>
    <outputFormat/>
    <method>execute</method>
    <targetPath>/content/folder[@name='Samples']/folder[@name='Models']
```

```
     /package[@name='GO Data Warehouse (query)']
     /folder[@name='Report Studio Report Samples']
     /report[@name='Compensation (hidden)']</targetPath>
    <parameters>
      <parameter>
        <name>pPosition</name>
        <type>xsdString</type>
      </parameter>
    </parameters>
    <modelPaths>
      <objectPath>storeID("i81b8b50c4a284aeeaa6253b3b4a126f7")</objectPath>
      <objectPath>storeID("if35fc523ed39464dbe3c11fd7f91e308")/model[last()]</objectPath>
      <objectPath>/content/folder[@name='Samples']/folder[@name='Models']
        /package[@name='GO Data Warehouse (analysis)']
        /model[@name='model']</objectPath>
      <objectPath>/content/folder[@name='Samples']/folder[@name='Models']
        /package[@name='GO Data Warehouse (analysis)']
        /model[last()]</objectPath>
      <locale>en-us</locale>
    </modelPaths>
  </drill>
</drillDefinitions>
```

A drill-through instance is represented by a `drill` element on a layout element. The drill-through instance references a drill through definition using the `drillRef` element. For example, the following drill-through instance references the above drill-through definition:

```
<item>
  <txt>
    <ref>R50</ref>
    <ctx>110</ctx>
    <style>S50</style>
    <drills>
      <drill>
        <drillRef>0</drillRef>
        <parm>
          <name>pPosition</name>
          <value>Human Resources</value>
          <mun>[Employee survey].[Position-department].[Position-department]
           .[Position-department name (level 3)]->[Position-department]
            .[100].[200].[300]</mun>
        </parm>
      </drill>
    </drills>
    <val>Human Resources</val>
    <valTyp>text</valTyp>
  </txt>
</item>
```

Sample prompt request page in LDX format

This sample illustrates the main parts of a layout data document for a prompt request page. This sample is based on the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Employee Training by Year**.

This prompt page is shown here when displayed using the standard IBM Cognos prompt interface.

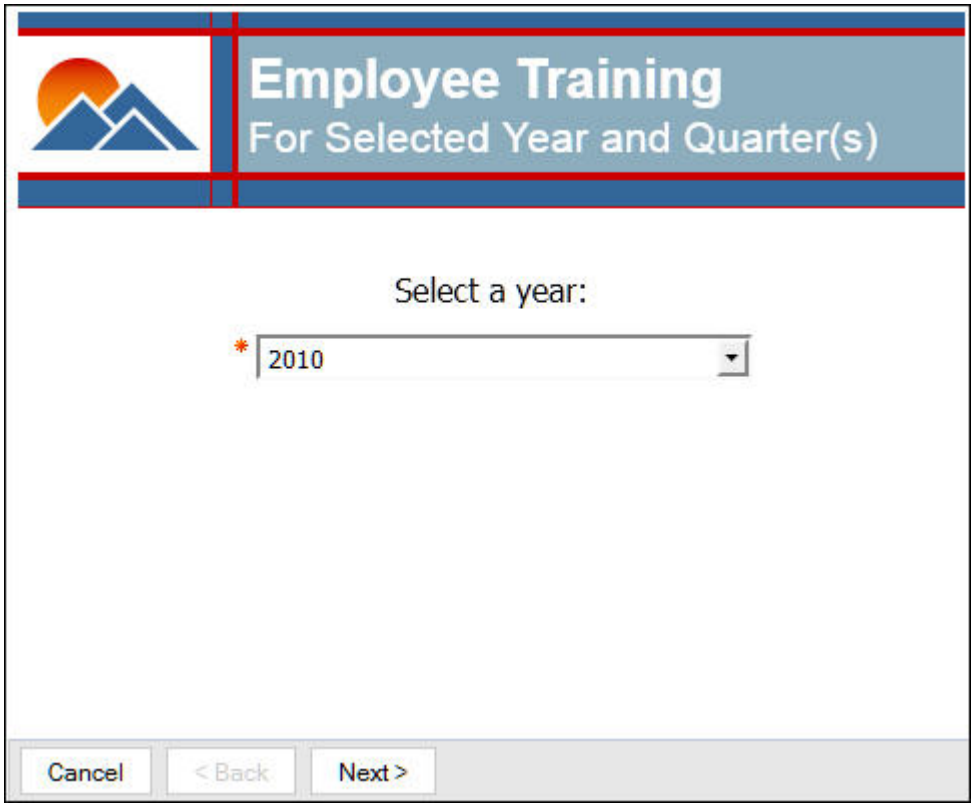

Figure 11. Report prompt page

The REST URL to obtain the prompt request page in LDX format is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportPrompts/path/
Public%2520Folders/Samples/Models/GO%2520Data%2520Warehouse%2520%2528analysis%2529
/Report%2520Studio%2520Report%2520Samples/Employee%2520Training%2520by%2520Year
```

The following sample XML is the returned prompt page in LDX format. This sample XML does not include the entire layout data document. Removed sections are represented by ellipses (...).

```
<document xmlns="http://www.ibm.com/xmlns/prod/cognos/layoutData/200904">
  ...
  <pages>
    <page>
      <canFinish>false</canFinish>
      <canNext>true</canNext>
      <canBack>false</canBack>
      ...
      <body>
        ...
        <item>
          <blk>
            ...
            <item>
              <p_value>
                <id>_P3369165171</id>
                <ref>R27</ref>
                <style>S27</style>
                <pname>P_Year</pname>
                <rows>5000</rows>
                <selectUI>DROP_DOWN</selectUI>
                <cname>Year</cname>
                <selOptions>
                  <sval>
                    <use>[Employee training].[Time].[Time].[Year]->[Time].[2004]</use>
                    <disp>2004</disp>
                  </sval>
                  <sval>
```

```
                    <use>[Employee training].[Time].[Time].[Year]->[Time].[2005]</use>
                    <disp>2005</disp>
                  </sval>
                  <sval>
                    <use>[Employee training].[Time].[Time].[Year]->[Time].[2006]</use>
                    <disp>2006</disp>
                  </sval>
                  <sval>
                    <use>[Employee training].[Time].[Time].[Year]->[Time].[2007]</use>
                    <disp>2007</disp>
                  </sval>
                </selOptions>
              </p_value>
            </item>
          </blk>
        </item>
      </body>
      <footer>
        <item>
          <p_btn>
            <ref>R31</ref>
            <style>S31</style>
            <bType>CANCEL</bType>
          </p_btn>
        </item>
        <item>
          <p_btn>
            <ref>R32</ref>
            <style>S32</style>
            <bType>BACK</bType>
          </p_btn>
        </item>
        <item>
          <p_btn>
            <ref>R33</ref>
            <style>S33</style>
            <bType>FORWARD</bType>
          </p_btn>
        </item>
      </footer>
    </page>
  </pages>
...
```

The canBack. canFinish, and canNext elements specify which secondary operations can be used with this page. The p_value element specifies that this is a **Value Prompt** control. Most of the child elements of this control correspond directly to the prompt properties exposed in Report Studio. The p_btn elements in the document correspond to the buttons displayed on the prompt page.

For more information about prompt pages, see “Running reports with prompts” on page 42.

Chapter 8. Understanding the Simple format

You can render an IBM Cognos resource in Simple format when you are developing a client application that works with specific reports or other resources and you need a format that is smaller and simpler than the layout data (LDX) format.

If you are writing a generic application for use with any IBM Cognos resource, use the LDX format. For more information, see Chapter 7, "Understanding the Layout Data format," on page 59.

An IBM Cognos resource rendered in Simple format is similar to a resource rendered in layout data (LDX) format, but the element names are derived from object IDs and other information from the source resource.

Source resource objects that have `id` elements will use the value as the element name in Simple format. For example, the following element in LDX format:

```
<txt>
  <id>Text1</id>
  <ref>R5</ref>
  <style>S5</style>
  <val>2007</val>
  <valTyp>text</valTyp>
</txt>
```

becomes the following element in Simple format:

```
<Text1>
  <style>S5</style>
  <val>2007</val>
  <valTyp>text</valTyp>
</Text1>
```

Source resource objects that do not have `id` elements are not included in the Simple format instances.

Some resource objects have additional transformations applied in Simple format. For more information and examples, see "List in Simple format" on page 77, "Grouped list in Simple format" on page 78, "Multiple items in a cell in Simple format" on page 79, and "Crosstab in Simple format" on page 80.

Simple format XML encoding

Because special characters are not supported in XML element tags, all special characters in Simple format are encoded as follows:

- If an ID starts with a special character or a number, then the element name will become the ID preceded by the letter `e`. Element names in XML must start with a letter.
- A space will be encoded as two underscore characters: __
- Special characters will be encoded as _x<code>, where <code> is the 4-digit hex code that corresponds to the code for that character.

Report output structure in Simple format

You can rerieve report output in Simple format by including the `fmt=Simple` option in the URL.

The sample report from "Report output structure" on page 61 can be retrieved in Simple format with the following URL:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData/path
/Public%2520Folders/Samples/Models/GO%2520Data%2520Warehouse%2520%2528analysis%2529
/Report%2520Studio%2520Report%2520Samples
/Employee%2520Satisfaction%25202006?fmt=Simple
```

The following sample XML is the above report output in Simple format. This sample XML does not include the entire layout data document. Removed sections are represented by ellipses (...).

```
<report xmlns="http://developer.cognos.com/schemas/raas/Employee__Satisfaction__2006">
  <Page1>
    <header>
      <FirstPage_x005fReportTitle2121>
      ...
      <FirstPage_x005fSubtitle1121>
      ...
    </header>
    <body>
      <Combination__Chart___x002d__survey__topic__scores__by__department>
      ...
      <Combination__Chart___x002d__survey__scores__and__benchmark>
      ...
      <Crosstab1>
        <style>S67</style>
        <Employee__ranking>
        ...
    </body>
    <footer>
    ...
      <RunDate1>
      ...
      <PageNumber>
      ...
      <RunTime1>
      ...
    </footer>
  </Page1>
  <styleGroup>
  ...
```

The root element of the document is a `report` element. The main element names in Simple format are the values of the `id` elements of the corresponding report parts. For example, the `cht` with an `id` value of **Combination Chart - survey topic scores by department** is rendered as a `Combination__Chart___x002d__survey__topic__scores__by__department` element in Simple format. The `blk` elements that do not have `id` child elements are omitted from the Simple report output.

The Simple report output contains `StyleGroup` elements that share the structure of the `styleGroup` elements in LDX report output.

Filter output structure in Simple format

You can retrieve filtered output in Simple format.

Using the same report as in "Report output structure in Simple format," you can request a filtered output containing one of the combination charts and the crosstab.

The REST URL to run this report and obtain the Simple format output is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData/path/
Public%2520Folders/Samples/Models/GO%2520Data%2520Warehouse%2520%2528analysis%2529/
Report%2520Studio%2520Report%2520Samples/Employee%2520Satisfaction%25202006
?selection=Combination Chart - survey scores and benchmark;Crosstab1&fmt=Simple
```

The following sample XML is the above filtered report in Simple format. This sample XML does not include the entire layout data document. Removed sections are represented by ellipses (...).

```
<results xmlns="http://developer.cognos.com/schemas/raas/Employee__Satisfaction__2006">
  <Combination__Chart___x002d__survey__scores__and__benchmark>
  ...
  </Combination__Chart___x002d__survey__scores__and__benchmark>
  <Crosstab1>
  ...
  </Crosstab1>
  <styleGroup>
  ...
</results>
```

Since this is a filtered report, the root element is a `results` element. This element contains a `Combination__Chart___x002d__survey__scores__and__benchmark` element and a `Crosstab1` element, as specified in the request.

List in Simple format

For a list in Simple format, the column titles are used as the element names for the detail rows. If no titles are available in the report then columns are identified as `Column1`, `Column2`, etc.

Here is an example based on the sample report **Public Folders > Samples > Models > GO Data Warehouse (query) > SDK Report Samples > Order Product List**.

The REST URL to run this report is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData/path/
Public%2520Folders/Samples/Models/GO%2520Data%2520Warehouse%2520%2528query%2529/
SDK%2520Report%2520Samples/Order%2520Product%2520List?fmt=Simple
```

The following sample XML is the above list output in Simple format. This sample XML does not include the entire document. Removed sections are represented by ellipses (...).

```
<List1>
  <style>S31</style>
  <columnTitle>
    <ref>R22</ref>
    <style>S22</style>
    <item>
      <txt>
      ...
        <val>Order number</val>
        <valTyp>text</valTyp>
      </txt>
    </item>
  </columnTitle>
  <columnTitle>
    <ref>R24</ref>
    <style>S24</style>
    <item>
      <txt>
      ...
        <val>Product</val>
        <valTyp>text</valTyp>
      </txt>
    </item>
```

```
</columnTitle>
<row>
  <Order__number>
    <style>S26</style>
    <val>100003</val>
    ...
  </Order__number>
  <Product>
    <style>S28</style>
    <val>Polar Extreme</val>
    ...
  </Product>
</row>
<row>
  <Order__number>
    <style>S26</style>
    <val>100009</val>
    ...
  </Order__number>
  <Product>
    <style>S28</style>
    <val>BugShield Natural</val>
    ...
  </Product>
</row>
...
```

Grouped list in Simple format

For a grouped list in LDX format, the grouping levels are identified using the column titles. In a grouped list in Simple format:

- The `grp` element is replaced with the column title.
- The `dv` element value for the group is a nested element with the same name as the column title again.
- The cell value for a group only appears on the first row of that group.

Here is an example based on the sample report used in "Sample grouped list in LDX format" on page 65

The REST URL to run this report is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData/searchPath/
content/folder[@name='Samples']/folder[@name='Models']/package[@name='GO
Data Warehouse (query)']/folder[@name='SDK Report Samples']/report[@name='Product
Quantity and Price']?fmt=Simple
```

The following sample XML is the above grouped list output in Simple format. This sample XML does not include the entire document. Removed sections are represented by ellipses (...).

```
<List1>
  <style>S40</style>
  <columnTitle>
    <ref>R20</ref>
    <style>S20</style>
    <item>
      <txt>
      ...
        <val>Product type</val>
        ...
      </txt>
    </item>
  </columnTitle>
  ...
  <Product__type>
    <Product__type>Binoculars</Product__type>
    <row>
      <Product__type>
        <style>S28</style>
        <val>Binoculars</val>
        <valTyp>text</valTyp>
```

```
      <fmtVal>Binoculars</fmtVal>
    </Product__type>
    <Product>
      <style>S30</style>
      <val>Opera Vision</val>
      <valTyp>text</valTyp>
      <fmtVal>Opera Vision</fmtVal>
    </Product>
    <Quantity>
      <style>S32</style>
      <val>82016</val>
      <valTyp>number</valTyp>
      <fmtVal>82,016</fmtVal>
      <fmtPatrn>#,##0</fmtPatrn>
      <exclPatrn>\#\,\#\#0</exclPatrn>
    </Quantity>
    <Unit__sale__price>
      <style>S34</style>
      <val>109.436407</val>
      <valTyp>number</valTyp>
      <fmtVal>$109.44</fmtVal>
      <fmtPatrn>¤#,##0.00;(¤#,##0.00)</fmtPatrn>
      <exclPatrn>\[$$-409\]\#\,\#\#0\.00\;\\\(\[$$-409\]\#\,\#\#0\.00\\\)</exclPatrn>
    </Unit__sale__price>
  </row>
  <row>
  ...
```

Multiple items in a cell in Simple format

If there is only one item in a cell, then the cell is flattened so that the element name becomes the column associated with the cell.

If there are multiple items in a cell, then the items are identified based on their type and a number based on how many of that type of item have preceded it in that cell.

To illustrate this transformation, here is a sample cell in LDX format followed by the same cell in Simple format.

Multiple items in a cell in LDX format

```
<cell>
  <ref>R18</ref>
  <style>S18</style>
  <item>
    <txt>
      <ref>R16</ref>
      <ctx>6:4:5</ctx>
      <style>S16</style>
      <val>332986338.06</val>
      <valTyp>number</valTyp>
      <fmtVal>332,986,338.06</fmtVal>
      <fmtPatrn>#,##0.##</fmtPatrn>
      <exclPatrn>\#\,\#\#0\.\#\#</exclPatrn>
    </txt>
  </item>
  <item>
    <img>
      <ref>R17</ref>
      <style>S17</style>
    </img>
  </item>
</cell>
```

Multiple items in a cell in Simple format

```
<Revenue>
  <TextFrame1>
    <style>S16</style>
    <val>332986338.06</val>
    <valTyp>number</valTyp>
    <fmtVal>332,986,338.06</fmtVal>
    <fmtPatrn>#,##0.##</fmtPatrn>
    <exclPatrn>\#\,\#\#0\.\#\#</exclPatrn>
  </TextFrame1>
  <Image1>
    <style>S17</style>
  </Image1>
</Revenue>
```

Crosstab in Simple format

A table of crosstab cells is the same in Simple format as it is in LDX format. The crosstab corner is also the same in both formats. The element names for columns and rows, however, are transformed to use the data item name.

Here is an example based on the sample report used in "Sample crosstab in LDX format" on page 68

The REST URL to run this report is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/pagedReportData/path/
Public%20Folders/Samples/Models/GO%20Data%20Warehouse%20%28analysis%29/
Report%20Studio%20Report%20Samples/Returns%20by%20Order%20Method
?selection=Crosstab1&fmt=Simple
```

The following sample XML is the above crosstab output in Simple format. This sample XML does not include the entire document. Removed sections are represented by ellipses (...).

```
<results xmlns="http://developer.cognos.com/schemas/raas/Returns__by__Order__Method">
  <Crosstab1>
    <style>S13</style>
    <corner>
      <ref>R1</ref>
      <style>S1</style>
      <val>Return quantity</val>
      <valTyp>text</valTyp>
    </corner>
    <Return__reason>
      <name>
        <ref>R5</ref>
        <ctx>2</ctx>
        <style>S5</style>
        <item>
          <txt>
            <ref>R2</ref>
            <ctx>2</ctx>
            <style>S2</style>
            <val>Defective product</val>
            <valTyp>text</valTyp>
          </txt>
        </item>
      </name>
      <start>0</start>
      <size>1</size>
    </Return__reason>
    <Return__reason>
      <name>
        <ref>R5</ref>
        <ctx>3</ctx>
        <style>S5</style>
        <item>
          <txt>
            <ref>R2</ref>
```

```
        <ctx>3</ctx>
        <style>S2</style>
        <val>Incomplete product</val>
        <valTyp>text</valTyp>
      </txt>
    </item>
  </name>
  <start>1</start>
  <size>1</size>
</Return__reason>
...
<Product__line>
  <name>
    <ref>R9</ref>
    <ctx>10</ctx>
    <style>S9</style>
    <item>
      <txt>
        <ref>R8</ref>
        <ctx>10</ctx>
        <style>S8</style>
        <val>Camping Equipment</val>
        <valTyp>text</valTyp>
      </txt>
    </item>
  </name>
  <start>0</start>
  <size>1</size>
</Product__line>
<Product__line>
  <name>
    <ref>R9</ref>
    <ctx>19</ctx>
    <style>S9</style>
    <item>
      <txt>
        <ref>R8</ref>
        <ctx>19</ctx>
        <style>S8</style>
        <val>Mountaineering Equipment</val>
        <valTyp>text</valTyp>
      </txt>
    </item>
  </name>
  <start>1</start>
  <size>1</size>
</Product__line>
...
<table>
  <row>
    <cell>
      <ref>R11</ref>
      <ctx>11::10::2</ctx>
      <style>S11</style>
      <item>
        <txt>
          <ref>R10</ref>
          <style>S10</style>
          <val>36046</val>
          <valTyp>number</valTyp>
          <fmtVal>36,046</fmtVal>
          <fmtPatrn>#,##0</fmtPatrn>
          <exclPatrn>\#\,\#\#0</exclPatrn>
        </txt>
      </item>
    </cell>
```

Chapter 9. Understanding the DataSet format

You can render an IBM Cognos resource in the DataSet format when you are only interested in the data contained in the report output, and not in any formatting data.

DataSet format output consists of a `dataSet` root element containing one or more `dataTable` elements. Each `dataTable` element corresponds to a report part, or to a page of report part output if the report is requested with page breaks.

Requests for DataSet formatted output will usually be for a single report part without page breaks.

Sample grouped list in DataSet format

This sample illustrates the main parts of a DataSet document for a grouped list report.

Here is an example based on the sample report used in "Sample grouped list in LDX format" on page 65

The REST URL to run this report is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/path
/Public%2520Folders/Samples/Models/GO%2520Data%2520Warehouse%2520%2528analysis%2529
/Report%2520Studio%2520Report%2520Samples
/Employee%2520Satisfaction%25202006?fmt=DataSet
```

The following sample XML is the above grouped list report in DataSet format. This sample XML does not include the entire layout data document. Removed sections are represented by ellipses (...).

```
<dataSet>
  <dataTable>
    <id>List1</id>
    <row>
      <Product__type>Binoculars</Product__type>
      <Product>Opera Vision</Product>
      <Quantity>82016</Quantity>
      <Unit__sale__price>109.436407</Unit__sale__price>
    </row>
    ...
    <row>
      <Product__type>Binoculars</Product__type>
      <Product>Seeker Mini</Product>
      <Quantity>172851</Quantity>
      <Unit__sale__price>75.681431</Unit__sale__price>
    </row>
    <row>
      <Product__type>Binoculars</Product__type>
    </row>
    <row>
      <Product__type>Climbing Accessories</Product__type>
      <Product>Firefly Charger</Product>
      <Quantity>302114</Quantity>
      <Unit__sale__price>51.817049</Unit__sale__price>
    </row>
    ...
  </dataTable>
</dataSet>
```

The `id` child element of the `dataTable` contains the name of the report part from Report Studio. each row in the table is represented by a `row` element. Element

names inside the row element are based on the column title names in the report, and the content of these elements is the cell value from the table.

Sample crosstab in DataSet format

This sample illustrates the main parts of a DataSet document for a crosstab report.

Here is an example based on the sample report used in "Sample crosstab in LDX format" on page 68

This report is run using a filter to return just the crosstab. The REST URL to run this report is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/path
/Public%20Folders/Samples/Models/GO%20Data%20Warehouse%20%28analysis%29
/Report%20Studio%20Report%20Samples/Returns%20by%20Order%20Method
?selection=Crosstab1&fmt=DataSet
```

The following sample XML is the above crosstab report in LDX format. This sample XML does not include the entire layout data document. Removed sections are represented by ellipses (...).

```
<dataSet>
  <dataTable>
    <id>Crosstab1</id>
    <row>
      <Product__line>Camping Equipment</Product__line>
      <Defective__product>36046</Defective__product>
      <Incomplete__product>57043</Incomplete__product>
      <Wrong__product__shipped>52199</Wrong__product__shipped>
      <Unsatisfactory__product>61549</Unsatisfactory__product>
      <e2004>33817</e2004>
      <e2005>50769</e2005>
      <e2006>57013</e2006>
      <e2007>65238</e2007>
    </row>
    ...
    <row>
      <Product__line>Golf Equipment</Product__line>
      <Defective__product>6221</Defective__product>
      <Incomplete__product>5817</Incomplete__product>
      <Wrong__product__shipped>13105</Wrong__product__shipped>
      <Unsatisfactory__product>10068</Unsatisfactory__product>
      <e2004>6287</e2004>
      <e2005>6731</e2005>
      <e2006>14047</e2006>
      <e2007>8146</e2007>
    </row>
  </dataTable>
</dataSet>
```

As with the grouped list example, element names in the row elements are derived from the column title names in the original report.

Sample chart in DataSet format

This sample illustrates the main parts of a DataSet document for a chart report.

This example is based on the sample report **Public Folders > Samples > Models > GO Data Warehouse (analysis) > Report Studio Report Samples > Employee Satisfaction 2012**.

The REST URL to run this report and obtain the LDX output is shown here:

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/path/
Public%2520Folders/Samples/Models/GO%2520Data%2520Warehouse%2520%2528analysis%2529/
Report%2520Studio%2520Report%2520Samples/Employee%2520Satisfaction%25202012
?selection=Combination Chart - survey scores and benchmark&fmt=DataSet
```

The HTML output of the report is shown here.

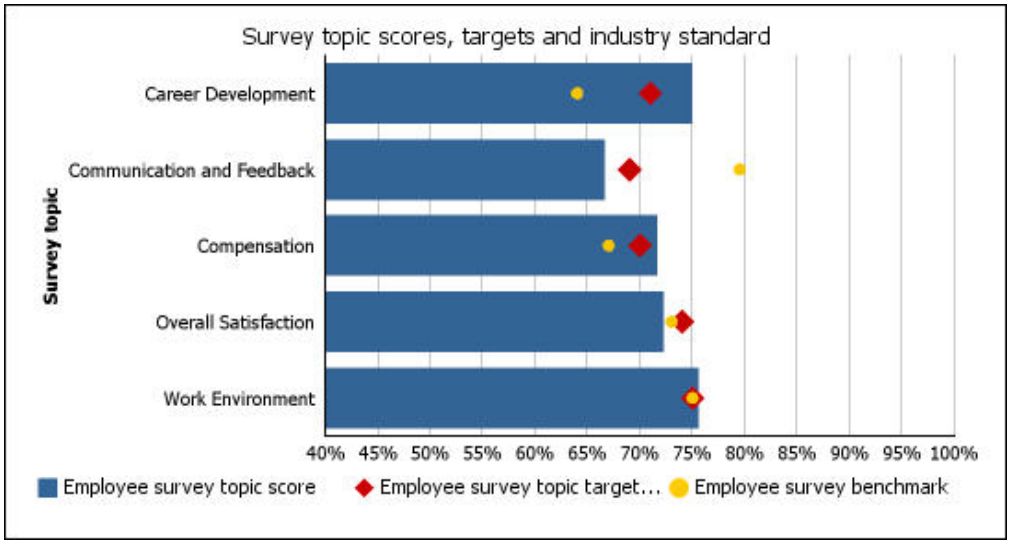

Figure 12. Chart from Employee Satisfaction 2012 report

An abbreviated version of the pages element is shown here.

```
<dataSet>
  <dataTable>
    <id>Combination Chart - survey scores and benchmark</id>
    <row>
      <Survey__topic>Overall Satisfaction</Survey__topic>
      <Actual__score>Employee survey topic score</Actual__score>
      <Topic__target__score>0.722459</Topic__target__score>
      <Industry__Standard/>
      <dim5/>
      <dim6/>
      <dim7/>
    </row>
    ...
    <row>
      <Survey__topic>Overall Satisfaction</Survey__topic>
      <Actual__score/>
      <Topic__target__score/>
      <Industry__Standard>Employee survey topic target score</Industry__Standard>
      <dim5>0.740000</dim5>
      <dim6/>
      <dim7/>
    </row>
    ...
    <row>
      <Survey__topic>Overall Satisfaction</Survey__topic>
      <Actual__score/>
      <Topic__target__score/>
      <Industry__Standard/>
      <dim5/>
      <dim6>Employee survey benchmark</dim6>
      <dim7>0.730000</dim7>
    </row>
    ...
  </dataTable>
</dataSet>
```

This chart contains three data points for each of five rows, giving a total of 15 data points. The DataSet report output contains 15 `row` elements, one for each data point. The preceding code snippet displays the three data points for the first row, **Overall Satisfaction**.

Chapter 10. Troubleshooting Mashup Service applications

This section provides information about potential problems you may encounter when using the IBM Cognos Mashup Service and provides solutions and workarounds.

For troubleshooting information that is not specific to the Mashup Service, see the Troubleshooting section of the *IBM Cognos Business Intelligence Troubleshooting Guide*. You can also refer to component-specific documents.

In IBM Cognos Connection, the service that supports the Mashup Service is called the report data service. You can set logging levels for this service for use when debugging Mashup Service problems. See the *IBM Cognos Business Intelligence Administration and Security Guide* for instruction on setting up logging.

Attempting to access a saved report version causes the report to be run

If your report is run when you are attempting to retrieve a saved report version, ensure you have modified the report properties in IBM Cognos Connection.

See "Saving report versions" on page 41 for more information.

SOAP application loops indefinitely while waiting for output

If your SOAP application loops indefinitely while waiting for output, ensure you are copying the response session element to the request session element inside the loop that waits for report output.

See "Running Mashup Service methods" on page 28 for details.

SOAP application cannot get response from server

If your REST application requires authentication, you must copy the headers from the authentication object to the report data service object.

See "Creating a report service instance" on page 27 for details.

Web server responses vary for "async=MANUAL" REST option

If your REST application uses the `async=MANUAL` option and also examines the body of this http response, you should be aware that the contents of the body will differ depending on which Web server is being used by IBM Cognos.

XPath limitations in REST requests

If your REST application uses the `xpath` option you should be aware that only a subset of XPath can be used.

For more information, see "Using XPath expressions to filter report output" on page 38.

Cookies are required for REST authentication

If your REST application needs to authenticates users to the IBM Cognos server, you must have cookies enabled in your Web browser.

A page not found error is returned for a Mashup Service request

Cognos Mashup Service requests may return a page not found error.

There are two possible solutions for this error as explained here:

- Ensure that the Gateway URI in IBM Cognos Configuration is set to the name of the server and not to localhost.
- If you are using the `path` or `searchPath` source types, try using the `report` source type. If this works, the `path` or `searchPath` you tried may have been incorrect.

Updated content is not returned when using the Windows Internet Explorer browser for development work

In the **Temporary Internet Files and History Settings** dialog box, check for newer versions of Web pages every time the browser visits the Web page.

REST requests do not work when the path or searchPath contains non-Latin-1 characters

Some Web servers do not correctly handle URLs containing URL encodings of characters that are not in the Latin-1 character set.

This can occur if your IBM Cognos Business Intelligence installation is in a language that has characters that are not in the Latin-1 character set and you are using the path or searchPath source types.

In this case, instead of using the standard URL-encoding (%xx), encode non-Latin-1 characters in the path or searchPath as _xCCCC, where CCCC is the hexadecimal UTF-8 code point for the character.

Notes

- SOAP applications are not affected by this issue.
- If you have the sequence _x in the URL, encode it as _x005Fx.
- The sequence __ (two underscores) is interpreted as a space character. If you have __ in the URL, encode it as _x005F_x005F.
- REST options appearing after the question-mark symbol (?) in a URL are not affected since they are not decoded by the Web server. They should be URL-encoded using the standard URI-encoding procedure.

Asynchronous REST requests do not work when the Web server uses the Apache HTTPClient

Asynchronous REST requests use redirects to the same URL while waiting for the report request to complete. If your server uses the Apache HTTPClient, ensure that the value of the HTTP client parameter `http.protocol.allow-circular-redirects` is `true`.

Adding a service reference in Microsoft Visual Studio 2008 or later fails

Using **Add Service Reference** in Microsoft Visual Studio 2008 or later fails. The service reference assumes a SOAP 1.2 interface but the Mashup Service uses a SOAP 1.1 interface.

To add a Mashup Service reference, in the **Add Service Reference** dialog box, click **Advanced**. In the **Service Reference Settings** dialog box, click **Add Web Reference**. You can now add the Mashup Service Web reference.

Add Web Reference is available directly in Microsoft Visual Studio 2012 and the procedure in the preceding paragraph is not needed.

Missing report-specific SOAP methods for some reports

The following methods are missing from the report-specific WSDL for some reports.

- `drill_<element>`
- `getFormatted_<element>`
- `get_<element>`

In addition, the response to a `getReport` method request consists of the report output in Layout Data (LDX) format, not in Simple format.

Note: The response is returned in the report-specific namespace, not in the generic LDX namespace.

Reports with certain structural elements are subject to this limitation. For more information, see "Report-specific method limitations for some reports" on page 31.

Retrieving multiple report outputs in a single-signon authentication environment fails

When attempting to retrieve multiple report outputs in a single-signon authentication environment, one or more of the retrievals fails with the error code RDS-ERR-1000.

This problem occurs because session cookies are being overwritten by multiple asynchronous report requests.

Resolve this issue with one of the following workarounds.

- Wait until a report output is retrieved before requesting a subsequent report.
- Request each report from a different HTML `iframe` element.
- Use a URL to run each report and display it in Cognos Viewer. For more information, see "Using a URL to display a report in IBM Cognos Viewer" on page 56.

Cognos Mashup Service session expires before timeout limit for authentication provider

When attempting to run a report using the IBM Cognos Mashup Service, the error message "RDS-ERR-1020 The currently provided credentials are invalid. Please provide the logon credentials." is displayed even though the inactivity timeout for the authentication provider has not been reached.

This problem occurs because when you log on a CAM passport is created for your Web browser session. This passport is checked first when you send a Cognos Mashup Service and, if it has expired, a new logon page is returned by the dispatcher.

You can resolve this issue in one of two ways.

- Set the inactivity timeout in IBM Cognos Configuration to a value greater than the inactivity timeout of your authentication provider. The timeout values is set in the **Security** > **Authentication** window in Cognos Configuration.
- Log off from the Cognos Mashup Service before sending additional requests. This action will clear the CAM passport.

Chapter 11. Upgrading Mashup Service applications

To take advantage of new features in the IBM Cognos Mashup Service, upgrade your Mashup Service applications to comply with the latest version. Some features of previous releases are deprecated in a current release, and will not be available in future releases. You can make minor changes so that your existing applications can function with a current release, however, we recommend that you fully upgrade your applications to the latest version if possible.

Before you can upgrade your Mashup Service applications, you must upgrade your server software from the previous version and install the IBM Cognos Software Development Kit.

Upgrading to version 10.2.1

If you have Mashup Service applications created in IBM Cognos Business Intelligence versions 10.2, 10.1.1, 10.1.0 or 8.4.1, you do not need to make any changes to run them in version 10.2.1.

However, you should review the new features described in "New features in version 10.2.1" on page 1 to see if their use could enhance your applications.

If you are upgrading from IBM Cognos Business Intelligence version 8.4.1, you should also review "New features in version 10.1.0" on page 2

If you are upgrading Mashup Service applications from IBM Cognos Business Intelligence version 8.4.0 you will need to make the changes described in "Upgrading to version 8.4.1" on page 92.

Upgrading to version 10.2

If you have Mashup Service applications created in IBM Cognos Business Intelligence versions 10.1.1, 10.1.0 or 8.4.1, you do not need to make any changes to run them in version 10.2.

However, you should review the new features described in "New features in version 10.2" on page 2 to see if their use could enhance your applications.

If you are upgrading from IBM Cognos Business Intelligence version 8.4.1, you should also review "New features in version 10.1.0" on page 2

If you are upgrading Mashup Service applications from IBM Cognos Business Intelligence version 8.4.0 you will need to make the changes described in "Upgrading to version 8.4.1" on page 92.

Upgrading to version 10.1.1

If you have Mashup Service applications created in IBM Cognos Business Intelligence versions 10.1.0 or 8.4.1, you do not need to make any changes to run them in version 10.1.1.

However, you should review the new features described in "New features in version 10.1.1" on page 2 to see if their use could enhance your applications.

If you are upgrading from Cognos BI version 8.4.1, you should also review "New features in version 10.1.0" on page 2

If you are upgrading Mashup Service applications from Cognos 8 BI version 8.4.0 you will need to make the changes described in "Upgrading to version 8.4.1."

Upgrading to version 10.1.0

If you have Mashup Service applications created in IBM Cognos Business Intelligence version 8.4.1, you do not need to make any changes to run them in version 10.1.0.

However, you should review the new features described in "New features in version 10.1.0" on page 2 to see if their use could enhance your applications.

If you are upgrading Mashup Service applications from Cognos BI version 8.4.0 you will need to make the changes described in "Upgrading to version 8.4.1."

If you use the deprecated `promptDescription` (REST applications) or the `getPromptDescription` (SOAP applications) methods, you should consider replacing them with the `reportPrompts` (REST applications) or the `getReportPrompts` (SOAP applications) methods. The deprecated methods will be removed in a later version of this product. See "Running reports with prompts" on page 42 for information on how to use the new methods.

Upgrading to version 8.4.1

There have been changes made to the IBM Cognos Mashup Service in IBM Cognos Business Intelligence version 8.4.1. There have been changes to the layout formats, the schemas, and to the SOAP and REST programming interfaces. These changes are described in the following topics. Applications created with the previous version of the Mashup Service will run correctly on an Cognos BI version 8.4.1 server but you can also upgrade your Mashup Service applications using the procedures shown here.

SOAP programming changes

The SOAP programming changes for Java and C# Mashup Service applications are shown here. There are no changes to report-specific applications, only to generic applications.

Applications created with the previous version of the Mashup Service will work correctly, even if they use methods and classes that have been removed from the current version of the Mashup Service. However, you can also update these applications to use the current classes and methods as described here.

You will need to update the Web services you created using the WSDL from the IBM Cognos BI server version 8.4 with the WSDL from the IBM Cognos BI server version 8.4.1. If you also created a Web service from the authentication WSDL, you do not need to recreate it, since the authentication WSDL has not changed. After resecting the Web service, you can make the class and method changes described here.

A number of classes and methods have been renamed in the current version of the Mashup Service. These changes affect the service definitions and the method calls.

Java service definitions

Replace

```
RDSLocator rdslocator = new RDSLocator();
RDSService rdsservice = rdslocator.getRDS(new URL(serverURL));
```

with

```
ReportDataServiceLocator rdslocator = new ReportDataServiceLocator();
ReportDataServicePort rdsservice = rdslocator.getReportDataServiceBinding(new URL(serverURL));
```

Replace

```
RDSServiceProxy proxy = new RDSServiceProxy();
RDSService mashupService = proxy.getRDSService();
```

with

```
ReportDataServicePortProxy proxy = new ReportDataServicePortProxy();
ReportDataServicePort mashupService = proxy.getReportDataServicePort();
```

Java method calls

Calls to `GetReportContent` and `GetReportFormatted` are replaced by calls to `getReportData`.

For example, the following code

```
GetReportContentRequest req = new GetReportContentRequest();
...
GetOutputResponse resp = svc.getReportContent(req);
```

is replaced with

```
GetReportDataRequest req = new GetReportDataRequest();
...
GetOutputResponse resp = svc.getReportData(req);
```

When retrieving report output, replace calls to `getContentOutput` with calls to `getLDXOutput`. In addition, you may need to add calls to `getPages`.

For example, the following code

```
resp.getOutput().getContentOutput().getDocument().getPage().getBody()
```

is replaced with

```
resp.getOutput().getLDXOutput().getDocument().getPages()[0].getPage().getBody()
```

C# service definitions

Replace

```
RDS svc = new RDS();
```

with

```
ReportDataService svc = new ReportDataService();
```

C# method calls

Calls to `GetReportContent` and `GetReportFormatted` are replaced by calls to `getReportData`.

For example, the following code

```
GetReportFormattedRequest request = new GetReportFormattedRequest();
...
GetOutputResponse response = svc.getReportFormatted(request);
```

is replaced with

```
GetReportDataRequest request = new GetReportDataRequest();
...
GetOutputResponse response = svc.getReportData(request);
```

When retrieving report output, replace calls to `ContentOutput` with calls to `LDXOutput`. In addition, you may need to add calls to `Pages`.

For example, the following code

```
resp.Output.ContentOutput.Document.Page.Body
```

is replaced with

```
resp.Output.LDXOutput.Document.Pages[0].Page.Body
```

REST programming changes

The REST programming changes for Mashup Service applications are shown here.

The `output` resource type has been replaced by the `reportData` resource type.

The `paged` option has been replaced by the `includePageBreaks` option.

If you use the `xpath` option, you will need to modify XPath expressions that use `pageGroup` or `page` elements to include the new `pages` element. For example, the following XPath expression

```
/document/pageGroup[1]/page/body/...
```

is replaced with

```
/document/pages[1]/pageGroup/pages/page/body/...
```

Chapter 12. SOAP methods reference

You can use the methods described in this chapter to retrieve and manipulate report outputs.

SOAP faults and exceptions

SOAP faults encountered during execution of an application are converted into one of three different exceptions as shown in this table.

Table 1. SOAP exceptions

Exception	Description
CCSAuthenticationFault	Exception raised for authentication errors.
CCSPromptFault	Exception raised if unanswered prompts prevent a report from running.
CCSGeneralFault	Exception raised for other errors.

Authentication methods

Use these methods to log on to, and off from, a server that requires authentication.

logon

Supplies the credentials for authenticated access to the IBM Cognos Business Intelligence server.

See "Logging on and logging off" on page 26 for more information.

Method signatures

C# programming language

```
public LogonResponseType logon(LogonRequestType logonRequest)
```

Java programming language

```
public com.cognos.developer.schemas.ccs.auth.types._1.LogonResponseType
   logon(com.cognos.developer.schemas.ccs.auth.types._1.LogonRequestType parameter)
```

Output parameter

"logonResponse" on page 138

Input parameter

"logonRequest" on page 138

logoff

Logs the user off.

Method signatures

C# programming language

```
public logoffResponseType logoff(LogoffRequestType logoffRequest)
```

Java programming language

```
public com.cognos.developer.schemas.ccs.auth.types._1.LogoffResponseType
   logoff(com.cognos.developer.schemas.ccs.auth.types._1.LogoffRequestType parameter)
```

Output parameter

"logoffResponse" on page 138

Input parameter

"logoffRequest" on page 137

Generic methods

Use these methods when developing a generic C# or Java application.

getCognosURL

Retrieves the URL of a report to display in IBM Cognos Viewer.

See "Using a URL to display a report in IBM Cognos Viewer" on page 56 for more information.

Secondary methods

none

Method signatures

C# programming language

```
public GetCognosURLResponse getCognosURL( GetCognosURLRequest request))
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetCognosURLResponse
   getCognosURL(com.cognos.developer.schemas.rds.types._2.GetCognosURLRequest request)
```

Output parameter

"GetCognosURLResponse" on page 152

Input parameter

"GetCognosURLRequest" on page 152

getOutput

Polls the IBM Cognos Business Intelligence server until an asynchronous method completes.This method is then used to retrieve the report output.

See "Running Mashup Service methods" on page 28 for more information.

Secondary methods

Any secondary method that is valid for the original asynchronous method.

Method signatures

C# programming language

```
public GetOutputResponse getOutput( GetOutputRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   getOutput(com.cognos.developer.schemas.rds.types._2.GetOutputRequest request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

"GetOutputRequest" on page 153

getOutputFormat

Retrieves a URL that can be used to retrieve a report in an IBM Cognos Viewer format.

See "Running reports and retrieving output in IBM Cognos Viewer formats" on page 40 for more information.

Secondary methods

None.

Method signatures

C# programming language

```
public getOutputFormatResponse getOutputFormat( getOutputFormatRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputFormatResponse
   GetOutputFormat(public com.cognos.developer.schemas.rds.types._2.GetOutputFormatRequest request)
```

Output parameter

getOutputFormatResponse

Input parameter

getOutputFormatRequest

getOutputFormats

Retrieves the list of supported provider output formats for the specified report.

See "Running reports and retrieving output in IBM Cognos Viewer formats" on page 40 for more information.

Secondary methods

None.

Method signatures

C# programming language

```
public getOutputFormatsResponse getOutputFormats( getOutputFormatsRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputFormatsResponse
   GetOutputFormats(public com.cognos.developer.schemas.rds.types._2.GetOutputFormatsRequest request)
```

Output parameter

getOutputFormatsResponse

Input parameter

GetOutputFormatsRequest

getPagedReportData

Retrieves the first page of the output of a content store object, such as a report. You can then use secondary requests to retrieve additional pages of the output.

Secondary methods

`drill`, `first`, `last`, `next`, `previous`, `release`

Method signatures

C# programming language

```
public GetOutputResponse getPagedReportData( GetPagedReportDataRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   getPagedReportData(com.cognos.developer.schemas.rds.types._2.GetPagedReportDataRequest request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

"GetPagedReportDataRequest" on page 153

getPromptAnswers

Retrieves the prompt answers chosen by a user in a prompt page.

This method is used following a getPromptPage request.

Secondary methods

None.

Method signatures

C# programming language

```
public GetPromptAnswersResponse getPromptAnswers( GetPromptAnswersRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetPromptAnswersResponse
   getPromptAnswers(com.cognos.developer.schemas.rds.types._2.GetPromptAnswersRequest request)
```

Output parameter

"GetPromptAnswersResponse" on page 154

Input parameter

"GetPromptAnswersRequest" on page 154

getPromptDescription

Gets the prompts associated with a prompt page.

This method is deprecated and will be removed in a future version of this product. Use the getReportPrompts resource type to retrieve prompt descriptions.

Secondary methods

`getTreePromptNode`, `release`

Method signatures

C# programming language

```
public GetPromptDescriptionResponse getPromptDescription( GetPromptDescriptionRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetPromptDescriptionResponse
   getPromptDescription(com.cognos.developer.schemas
   .rds.types._2.GetPromptDescriptionRequest request)
```

Output parameter

"GetPromptDescriptionResponse" on page 154

Input parameter

"GetPromptDescriptionRequest" on page 154

getPromptPage

Collects prompt answers using the IBM Cognos prompt user interface.

Secondary methods

None.

Method signatures

C# programming language

```
public GetPromptPageResponse getPromptPage( GetPromptPageRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetPromptPageResponse
    getPromptPage(com.cognos.developer.schemas.rds.types._2.GetPromptPageRequest request)
```

Output parameter

"GetPromptPageResponse" on page 155

Input parameter

"GetPromptPageRequest" on page 155

getReportData

Retrieves the content of a report.

Secondary methods

`drill`, `release`

Method signatures

C# programming language

```
public GetOutputResponse getReportData( GetReportDataRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   getReportData(com.cognos.developer.schemas.rds.types._2.GetReportDataRequest request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

"GetReportDataRequest" on page 155

getReportPrompts

Retrieves a prompt page in LDX format.

Secondary methods

`back`,`finish`, `forward`, `getTreePromptNode`, `release`, `reprompt`

Method signatures

C# programming language

```
public GetOutputResponse getReportPrompts( GetReportPromptsRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   getReportPrompts(com.cognos.developer.schemas.rds.types._2.GetReportPromptsRequest request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

"GetReportPromptsRequest" on page 155

Secondary methods

The secondary methods available for generic methods are described here.

See "Secondary operations" on page 29 for more information.

back

Returns to the previous prompt page.

Method signatures

C# programming language

```
public GetOutputResponse back( BackRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   back(com.cognos.developer.schemas.rds.types._2.BackRequest request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

"BackRequest" on page 143

drill

Drills up or down in an existing report session.

Method signatures

C# programming language

```
public GetOutputResponse drill( DrillRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   drill(com.cognos.developer.schemas.rds.types._2.DrillRequest request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

"DrillRequest" on page 148

finish

Skip subsequent prompt pages. You can use this method if subsequent prompts have default values.

Method signatures

C# programming language

```
public GetOutputResponse finish( ForwardRequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   finish(com.cognos.developer.schemas.rds.types._2.ForwardRequestType request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

ForwardRequestType

first

Retrieves the first page of report output.

Method signatures

C# programming language

```
public GetOutputResponse first( FirstRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   first(com.cognos.developer.schemas.rds.types._2.FirstRequestType request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

"FirstRequest" on page 151

forward

Retrieves the next prompt page.

Method signatures

C# programming language

```
public GetOutputResponse forward( ForwardRequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   forward(com.cognos.developer.schemas.rds.types._2.ForwardRequestType request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

ForwardRequestType

getTreePromptNode

Retrieves the next child level for a specified node.

Method signatures

C# programming language

```
public GetTreePromptNodeResponse getTreePromptNode(GetTreePromptNodeRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetTreePromptNodeResponse
   getTreePromptNode(com.cognos.developer.schemas.rds.types._2.GetTreePromptNodeRequest parameters)
```

Output parameter

"GetTreePromptNodeResponse" on page 156

Input parameter

"GetTreePromptNodeRequest" on page 156

last

Retrieves the last page of report output.

Method signatures

C# programming language

```
public GetOutputResponse last( LastRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   last(com.cognos.developer.schemas.rds.types._2.LastRequest request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

"LastRequest" on page 158

next

Retrieves the next page of report output.

Method signatures

C# programming language

```
public GetOutputResponse next( NextRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   next(com.cognos.developer.schemas.rds.types._2.NextRequest request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

"NextRequest" on page 161

previous

Retrieves the previous page of report output.

Method signatures

C# programming language

```
public GetOutputResponse previous( PreviousRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   previous(com.cognos.developer.schemas.rds.types._2.PreviousRequest request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

"PreviousRequest" on page 164

release

Removes inactive requests from the service cache earlier than they would be removed automatically by the system. Removing inactive requests makes more resources available for other requests, which can improve performance.

Method signatures

C# programming language

```
public ReleaseResponse release( ReleaseRequest request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.ReleaseResponse
   release(com.cognos.developer.schemas.rds.types._2.ReleaseRequest request)
```

Output parameter

"ReleaseResponse" on page 167

Input parameter

"ReleaseRequest" on page 167

reprompt

Use this secondary method on a prompt page that has multiple prompts to submit one or more prompt values and have the current prompt page refreshed, instead of moving to the next prompt page. Use this request if the page contained cascading prompts and you needed to submit a prompt response before receiving the subsequent prompt request.

Method signatures

C# programming language

```
public GetOutputResponse reprompt( ForwardRequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.rds.types._2.GetOutputResponse
   reprompt(com.cognos.developer.schemas.rds.types._2.ForwardRequestType request)
```

Output parameter

"GetOutputResponse" on page 153

Input parameter

ForwardRequestType

Report-specific methods

Use these methods when developing a report-specific C# or Java application.

getCognosURL

Gets the URL of a report to display in IBM Cognos Viewer.

Secondary methods

none

Method signatures

C# programming language

```
public GetCognosURLResponseType getCognosURL( object request))
```

Java programming language

```
public com.cognos.developer.schemas.raas.<report_name>.GetCognosURLResponseType
   getCognosURL( java.lang.Object request)
```

Output parameter

GetCognosURLResponseType

Input parameter

Not applicable.

getFormattedReport

Retrieves the content of a report in a specified format.

Secondary methods

drillFormatted, release

Method signatures

C# programming language

```
public GetFormattedReportResponseType getFormattedReport( GetFormattedReportRequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.raas.<report_name>.GetFormattedReportResponseType
   getFormattedReport(com.cognos.developer.schemas.raas
   .<report_name>.GetFormattedReportRequestType request)
```

Output parameter

"GetFormattedReportResponseType" on page 110

Input parameter

"GetFormattedReportRequestType" on page 110

getFormatted_*<element>*

Retrieves a specific report element in a specified format.

Note: This method is not available for certain reports. For more information, see "Report-specific method limitations for some reports" on page 31

Secondary methods

drill_<element>, release

Note: The element name in the drill secondary request must match the element name in the getFormatted request.

Method signatures

C# programming language

```
public GetFormattedReportResponseType
   getFormatted_<element>( GetFormatted_<element>RequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.raas.<report_name>.GetFormattedReportResponseType
   GetFormatted_<element>RequestType request)
```

Output parameter

"Get_<element>ResponseType" on page 111

Input parameter

"GetFormatted_<element>RequestType" on page 110

getPromptAnswers

Retrieves the prompt answers chosen by a user in a prompt page.

This method is used following a getPromptPage request.

Secondary methods

none

Method signatures

C# programming language

```
public PromptAnswersResponseType getPromptAnswers( PromptAnswersRequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.raas.<report_name>.PromptAnswersResponseType
   getPromptAnswers(com.cognos.developer.schemas
   .raas.<report_name>.PromptAnswersRequestType request)
```

Output parameter

"PromptAnswersResponseType" on page 111

Input parameter

"PromptAnswersRequestType" on page 111

getPromptPage

Collects prompt answers using the IBM Cognos prompt user interface.

Secondary methods

none

Method signatures

C# programming language

```
public PromptPageResponseType getPromptPage( PromptPageRequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.raas.<report_name>.PromptPageResponseType
   getPromptPage(com.cognos.developer.schemas.raas.<report_name>.PromptPageRequestType request)
```

Output parameter

"PromptPageResponseType" on page 112

Input parameter

"PromptPageRequestType" on page 112

getReport

Retrieves the content of a report.

This method retrieves the report output in LDX format for some reports.

For more information, see "Report-specific method limitations for some reports" on page 31

Secondary methods

drillFormatted, release

Method signatures

C# programming language

```
public GetReportResponseType getReport( GetReportRequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.raas.<report_name>.GetReportResponseType
   getReport(com.cognos.developer.schemas.raas.<report_name>.GetReportRequestType request)
```

Output parameter

"GetReportResponseType" on page 111

Input parameter

"GetReportRequestType" on page 110

get_*<element>*

Retrieves the requested report element only.

This method is not available for certain reports. For more information, see "Report-specific method limitations for some reports" on page 31

Secondary methods

drill_<element>, release

Note: The element name in the drill secondary request must match the element name in the getFormatted request.

Method signatures

C# programming language

```
public Get_<element>ResponseType get_<element>( Get_<element>RequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.raas.<report_name>.Get_<element>ResponseType
   getOutput(com.cognos.developer.schemas.raas.<report_name>.Get_<element>RequestType request)
```

Output parameter

"Get_<element>ResultsType" on page 111

Input parameter

"Get_<element>RequestType" on page 111

Secondary methods

The secondary methods available for report-specific methods are described here.

See "Secondary operations" on page 29 for more information.

drillFormatted

Drills up or down in an existing formatted report session.

Method signatures

C# programming language

```
public GetFormattedReportResponseType drillFormatted( DrillRequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.raas.<report_name>.GetFormattedReportResponseType
   drillFormatted(com.cognos.developer.schemas.raas.<report_name>.DrillRequestType request)
```

Output parameter

"GetFormattedReportResponseType" on page 110

Input parameter

DrillRequestType

drill_*<element>*

Drills up or down in a specific element in an existing report session.

This method is not available for certain reports. For more information, see "Report-specific method limitations for some reports" on page 31

Method signatures

C# programming language

```
public Get_<element>ResponseType drill( DrillRequestType request)
```

Java programming language

```
public com.cognos.developer.schemas.raas.<report_name>.Get_<element>ResponseType
   drill(com.cognos.developer.schemas.raas.<report_name>.DrillRequestType request)
```

Output parameter

"Get_<element>ResponseType" on page 111

Input parameter

DrillRequestType

release

Removes inactive requests from the service cache earlier than they would be removed automatically by the system. Removing inactive requests makes more resources available for other requests, which can improve performance.

Method signatures

C# programming language

```
public object release( ReleaseRequestType request)
```

Java programming language

```
public java.lang.Objectrelease(com.cognos.developer.schemas.raas
   .<report_name>.ReleaseRequestType request)
```

Output parameter

Not applicable.

Input parameter

ReleaseRequestType

Request and response elements not included in the RDS schema

Some request and response elements are not included with the schemas documented in this guide.

The following table maps elements that have a direct equivalent element in the RDS Schema Reference.

Table 2. Report-specific method element mappings

Report-specific method element	RDS schema element
DrillRequestType	DrillRequest
ForwardRequestType	ForwardRequest
GetCognosURLResponseType	GetCognosURLResponse
ReleaseRequestType	ReleaseRequest

XML elements reference

The following XML elements are not part of the schemas documented elsewhere in this guide.

GetFormattedReportRequestType

Retrieves the formatted report content.

Content Model

session then format then PromptAnswersType *(any number)*

GetFormattedReportResponseType

Contains the formatted report content.

Content Model

session then results *(optional)* then extension *(optional)*

GetFormatted_<element>RequestType

Retrieves the formatted report content for a specific report part.

Content Model

version then session then format then PromptAnswersType

GetReportRequestType

Retrieves the report content.

Content Model

version then session *(optional)* then PromptAnswersType *(any number)*

GetReportResponseType
Contains the report content.

Content Model

session then report *(optional)* then extension *(optional)*

Get_<element>RequestType
Retrieves the report content for a specific report part.

Content Model

version then session *(optional)* then PromptAnswersType *(any number)*

Get_<element>ResponseType
Contains the report content for a specific report part.

Content Model

session then Get_<element>ResultsType *(optional)*

Get_<element>ResultsType
Contains the report output for a named report part.

Content Model

The content of this element varies depending on which report part is being retrieved. You can determine the content from the report-specific WSDL file or the tools in your integrated development environment.

PromptAnswersRequestType
Retrieves the prompt answers associated with the session identifier.

This command is used after the getPromptPage command.

Content Model

session then extension *(optional)*

PromptAnswersResponse
Contains the prompt answers.

Content Model

PromptValue *(any number)* then extension *(optional)*

PromptAnswersResponseType
Contains the prompt answers.

Content Model

PromptValue *(any number)* then extension *(optional)*

PromptAnswersType
Retrieves the prompt answers associated with the session identifier.

This command is used after the getPromptPage command.

Content Model

PromptValue then extension *(optional)*

PromptPageRequestType
Retrieves a url from the IBM Cognosserver to fulfill prompt answers.

It also retrieves a session identifier to use in the getPromptAnswers method.

Content Model

extension *(optional)*

PromptPageResponseType
Contains the prompt page response.

This command follows a getPromptPage command.

Content Model

session then url then extension *(optional)*

PromptValue
Represents one or more prompt values.

Content Model

name then PValueArrayItem *(any number)*

PValueArrayItem
Contains a prompt value.

Content Model

SimplePValue or RangePValue or TreePValue or extension *(optional)*

report
Contains the report output.

Content Model

The content of this element varies depending on which report part is being retrieved. You can determine the content from the report-specific WSDL file or the tools in your integrated development environment.

results
Contains the report output.

Content Model

Content type is string.

TreePValue

Specifies a tree prompt value.

Content Model

inclusive then TreePValue then SimplePValue

Chapter 13. REST interface reference

The REST interface syntax for initial requests is

```
http://webservername:portnumber/ibmcognos/cgi-bin/cognos.cgi/rds
/resource_type/source_type/source_id?option1=val1&option2=val2...
```

Some Mashup Service tasks require several interactive steps to complete. Examples are retrieving report output one page at a time, collecting report prompts, and drilling up or down in a report. In these cases, the initial request has the syntax that is displayed above. Secondary requests have the syntax shown here.

```
http://webservername:portnumber/ibmcognos/cgi-bin/cognos.cgi/rds
/sessionOutput/conversationID/conv_ID/secondary_request?option1=val1
&option2=val2...
```

The resource types, source types, options, and secondary requests are described in the following topics.

Most of the XML structures that are inputs to, or responses from, the REST commands are part of the Authentication Schema Reference, Layout Data Schema Reference, or the RDS Schema Reference. However, there are some XML structures that are unique to the URL interface and are not part of these three schemas. These are documented in "XML elements reference" on page 131.

Resource types

Resource types are the commands that you use to communicate with the IBM Cognos Business Intelligence server.

auth/logon

Supplies the credentials for authenticated access to an IBM Cognos BI server.

On the initial call, an empty `credentials` elements is passed using the `xmlData` option.

Source types

None.

Options

`xmlData`

Secondary requests

None.

Output

The output from the server consists of a `credentials` element in which `actualValue` elements contain credential elements whose value has been determined and `missingValue` elements contain credential elements whose values need to be supplied. The logon request is sent again, with the missing values

supplied. If all missing values are entered and the use can be authenticated, the server responds with an `accountInfo` element that contains the user's account information.

auth/logoff

Logs the user off.

Source types

None.

Options

None.

Secondary requests

None.

Output

The output from the server consists of a `noerror` element.

auth/wsdl

Retrieves the WSDL file for the authentication service. This is used by SOAP applications that must authenticate with the IBM Cognos BI server.

Source types

None.

Options

None.

Secondary requests

None.

Output

The output from the server consists of the WSDL file for the authentication service.

atom

Retrieves the description of a report as an ATOM feed.

Source types

`path`, `report`, `searchPath`

Options

`eltype`, `selection`

Secondary requests

None.

Output

The ATOM output for a report or report part contains information about the report or report part, followed by one or more `atom:entry` elements that contain information about, and links to, the report parts that make up the report or report part. The ATOM output contains standard ATOM elements as well as elements that are unique to the Mashup Service.

Top level (report) feed

The top level feed for a report contains standard ATOM elements as well as the following Mashup Service specific elements.

<atom:link rel="self" type="application/atom+xml" href="..."/>
A URL to the ATOM feed for the report.

<atom:link rel="alternate" type="text/xml" href="..."/>
A URL to run the report using the reportData resource type.

<atom:link rel="alternate" type="application/x-ldx+xml" href="..."/>
A URL to run the report using the reportData resource type.

<atom:link rel="alternate" type="application/x-pagedldx+xml" href="..."/>
A URL to run the report using the pagedReportData resource type.

d:owner
The display name of the owner of the report.

d:ownerEmail
The email address of the owner, if available.

d:contact
The display name of the contact for the report.

d:contactEmail
The email address of the contact, if available.

d:location
The path to the report.

d:description
The description of the report.

d:thumbnailURL
The URL of a thumbnail view of the report.

Report part feed

The feed for a report part can contain the following additional Mashup Service specific elements.

d:type The element type of this report part.

d:storeID
The Content Manager storeID of the report that contains this part.

d:partID
The name of the part.

cm:location
The content manager location of the part.

cognosurl

Retrieves the URL that displays the report output using IBM Cognos Viewer.

Source types

path, report, searchPath

Options

None.

Secondary requests

None.

Output

The output from the server consists of a GetCognosURLResponse element.

outputFormat

Runs a report and retrieves the output in a specified format.

You can determine the possible output formats by first running the outputFormats resource type.

Note: The format of this REST task differs from other REST commands. The format is

http://*webservername:portnumber*/ibmcognos/cgi-bin/cognos.cgi/rds/outputFormat/*source_type*/*source_id*/*output_format*

where *output_format* is the desired output format.

Source types

path, report, searchPath

Options

async, burstID, burstKey, p_*parameter*, saveOutput, selection, version, versionID, xmlData

Secondary requests

None.

Output

The output from the server consists of the report output in the specified format.

outputFormats

Retrieves a list of supported formats for the report.

The possible values for the output formats are described in the `outputFormatEnum` enumeration set documented in the Software Development Kit *Developer Guide*

Source types

`path`, `report`, `searchPath`

Options

None.

Secondary requests

None.

Output

The output from the server consists of a GetOutputFormatsResponse element.

pagedReportData

Retrieves the first page of the output of a content store object, such as a report. You can then use secondary requests to retrieve additional pages of the output.

Source types

`path`, `report`, `searchPath`

Options

`drill_through_parameter`, `async`, `burstID`, `burstKey`, `drillthroughurls`, `drillurls`, `embedImages`, `excludePage`, `fmt`, `height`, `includeLayout`, `p_`*`parameter`*, `rowLimit`, `saveOutput`, `selection`, `useRelativeURL`, `version`, `versionID`, `width`, `xmlData`, `xpath`

Secondary requests

`drill`, `first`, `last`, `next`, `previous`, `release`

Output

The output from the server consists of the first page of the requested report, or report elements, in the requested format.

The report is run synchronously or asynchronously depending on the value of the `async` option.

promptAnswers

Retrieves the prompt answers chosen by a user in a prompt page.

Source types

`conversationID`

Use the `promptID` returned in the response from a `promptPage` request to populate the source type.

Options

None.

Secondary requests

None.

Output

The output from the server consists of a `promptAnswers` element.

promptDescription

Retrieves the descriptions of the prompts for a report.

This resource type is deprecated and will be removed in a future version of this product. Use the reportPrompts resource type to retrieve prompt descriptions.

Source types

`path`, `report`, `searchPath`

Options

`xmlData`

Secondary requests

None.

Output

The output from the server consists of a `GetPromptDescriptionResponse` element.

promptPage

Displays the standard IBM Cognos prompt page for a report.

Source types

`path`, `report`, `searchPath`

Options

useRelativeURL

Secondary requests

None.

Output

The output from the server consists of a `GetPromptPageResponse` element. The `url` element in the response contains the URL of the IBM Cognos prompt page. The `promptID` element can be passed as a `conversationID` in a `promptAnswers` URL to retrieve the prompt answers selected by a user.

providerOutput

Returns a report output that has been saved in Content Manager.

This resource type is deprecated and will be removed in a future version of this product. Use the `outputFormat` resource type to retrieve report outputs.

Source types

`path`, `report`, `searchPath`

Options

`burstID`, `burstKey`, `fmt`

Secondary requests

None.

Output

The output from the server consists of the requested saved report.

reportData

Retrieves the output of a content store object, such as a report.

Source types

`path`, `report`, `searchPath`

Options

`drill_through_parameter`, `async`, `burstID`, `burstKey`, `drillthroughurls`, `drillurls`, `embedImages`, `excludePage`, `fmt`, `height`, `includeLayout`, `includePageBreaks`, `p_`*`parameter`*, `rowLimit`, `saveOutput`, `selection`, `useRelativeURL`, `version`, `versionID`, `width`, `xmlData`, `xpath`

Secondary requests

`drill`, `release`

Output

The output from the server consists of the requested report, or report elements, in the requested format.

The report is run synchronously or asynchronously depending on the value of the `async` option.

reportPrompts

Retrieves a prompt page in LDX format.

Source types

`path`, `report`, `searchPath`

Options

None.

Secondary requests

`finish`, `forward`, `reprompt`, `treePromptNode`

Output

The output from the server consists of a `GetPromptPageResponse` element. The `url` element in the response contains the URL of the IBM Cognos prompt page. The `promptID` element can be passed as a `conversationID` in a `promptAnswers` URL to retrieve the prompt answers selected by a user.

thumbnail

Retrieves a graphical preview of the report output, run without data. If a preview cannot be generated within 5 seconds, then a default image is returned.

Source types

`path`, `report`, `searchPath`

Options

None.

Secondary requests

None.

Output

The output from the server is a thumbnail image.

wsdl

Retrieves the SOAP-based Web service description of a report.

Source types

`path`, `report`, `searchPath`

Options

None.

Secondary requests

None.

Output

The output from the server is a WSDL file that is the Web service description of the report.

wsil

Retrieves the WSIL description of the Web services and other WSIL containers within a folder.

Source types

`path`, `report`, `searchPath`

Options

None.

Secondary requests

None.

Output

The output from the server is a WSIL file that is the WSIL description of the Web services and other WSIL containers within the folder.

Source types

Source types are used to specify which report or asynchronous conversation the REST command is referring to.

conversationID

The source is a resource that is identified by an ID.

If the resource type is not `promptAnswers`, this source is a conversation ID that is stored in Content Manager for asynchronous and secondary requests.

A sample URL using a `conversationID` is shown here.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData
   /conversationID/i0EDF76C5EC064255A1C4F27FB6AE147A
```

If the resource type is `promptAnswers`, this is the value of the `promptID` element contained in the response to a `promptPage` resource type.

path

The source is a resource that is referenced by its simplified path. The path can have as its root:

- `Public%20Folders` for objects contained in **Public Folders**.
- ~ for the **My Folders** of the current user
- `CAMID(user)` for the **My Folders** of another user

A sample URL using a path is shown here.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/path
    /Public%20Folders/Samples/Models/GO%20Sales%20(query)
     /Report%20Studio%20Report%20Samples/Order%20Invoices%20-%20Donald%20Chow%2c%20Sales%20Person
```

report

The source is a resource that is referenced by its `storeID`. Note that the `storeID` of the same report will differ in different IBM Cognos installations.

A sample URL using a storeID is shown here.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData
   /report/i0E130B9A0A21463582535CF2D47B45F8
```

searchPath

The source is a resource that is referenced by its search path.

A sample URL using a search path is shown here.

```
http://localhost/ibmcognos/cgi-bin/cognos.cgi/rds/reportData/searchPath
    /content/folder[@name='Samples']/folder[@name='Models']/package[@name='GO Sales (query)']
    /folder[@name='Report Studio Report Samples']
    /report[@name='Order Invoices - Donald Chow, Sales Person']
```

Options

Options enable you to provide further detail about a REST command. The options allowed vary depend on the resource type.

drill_through_parameter

Specifies the name and value of a drill through parameter used to run a drill through report. The name of the parameter is the name of the drill through parameter. See Drilling through to another report.

async

Specifies whether the report will be run synchronously or asynchronously. The possible values are shown here. The default value is AUTO.

AUTO The report is run asynchronously. The server returns the http redirect response code (303) and a redirect URL. If the Web client is set to follow redirects, then the client will poll automatically.

MANUAL The report is run asynchronously. The server returns the http response code (202) and a polling URL in the location http header. The format of the response body varies depending on the Web server being used.

OFF The report is run synchronously. The server will not respond to the request until the output is available.

burstID

If the value of version is VERSION_NAME, this option can be used to provide a burstId value.

burstKey

If the value of version is VERSION_NAME, this option can be used to provide a burstKey value.

contextId

Specifies the ctx item to drill up or down on.

direction

Specifies the drill direction in a report. Its value is one of the values of the direction element.

drillthroughurls

Specifies whether HTML output contains drill-through information. The default value is `false`.

`true` HTML output contains drill-through information

`false` HTML output does not contain drill-through information

For more information, see "Drilling through to another report" on page 55.

drillurls

Specifies whether HTML output contains drill-up and drill-down information. The default value is `false`.

`true` HTML output contains drill-up and drill-down information

`false` HTML output does not contain drill-up and drill-down information

For more information, see "Drilling up and down in reports" on page 54.

eltype

Specifies which element of a report part to return in an ATOM feed.

embedImages

Specifies whether images that are included in HTML output are expressed as links to images on the IBM Cognos Business Intelligence server or are embedded as data in the HTML output. The default value is `false`.

`true` HTML output contains images as data.

`false` HTML output contains links to images on the Cognos Business Intelligence server.

This option requires that `fmt` be set to `HTML` or `HTMLFragment`.

excludePage

Specifies that when the `selection` option is being used to select a report part, report pages are not considered. The default value is `false`.

Although report parts in a report must have unique names, it is possible for a report page to have the same name as a report part, such as a list. Setting this option to `true` will ensure that a report part is selected rather than a report page with the same name.

fmt

Specifies the format of the returned output.

If the resource type is not providerOutput (deprecated), the possible values are shown here. These values are case-sensitive. The default value is `layoutDataXML`.

See "Output formats" on page 9 for more information about output formats.

`layoutDataXML`
Output is XML based on the LDX schema. See `Layout Data Schema Reference`.

HTML Output is a complete HTML file with style information in an inline stylesheet.

HTMLFragment
Output is a fragment of an HTML file with inline style information.

JSON Output is in JavaScript Object Notation format.

Simple Output is in Simple format. See `Using the Simple Format`.

Image Output is a portable network graphic (.png) image of the report output.

DataSet
Output is XML in the DataSet format. See `Using the DataSet format`.

DataSetAtom
Output is XML in the Atom version of the DataSet format. See `Using the DataSet format`.

If the resource type is providerOutput (deprecated), the possible values are those in the `outputFormatEnum` enumeration set documented in the *IBM Cognos Software Development Kit Developer Guide*. The default value is the format of the default saved report.

height

Specifies the height of the image returned when using the Image output format (`fmt`). The default value provides an output that is approximately the same size as when the report is viewed in IBM Cognos Viewer.

includeLayout

Specifies whether the output should include all formatting elements or only a subset of them. If the value of this option is `false`, then the following layout elements are not included in the report output, although report elements inside these elements are returned:

- `blk`
- `sngl`
- `tbl`
- `p_ elements`

The default value is `false` when using the `reportData` resource type and `true` when using the `pagedReportData` resource type.

Note: This option has no effect when the value of fmt is `Simple`, `DataSet`, or `DataSetAtom`.

includePageBreaks

Specifies whether the output should be returned as one page or should be separated into pages as in Report Viewer. The default value is `false`.

true The output is separated into pages as in Report Viewer.

false The output is returned as a single page.

mtchAll

Specifies whether the search results returned match all of the search words entered. The default value is `false`.

true The results returned match all of the search words entered. When mtchAny is also `true`, then the search can contain all the keywords in any order.

false The results returned match any of the search words entered.

mtchAny

Specifies whether the search results contain or start with the keywords. The default value is `false`.

true The results returned contain the key words.

false The search results returned start with the keywords.

nocase

Specifies whether the search is case sensitive or case insensitive. The default value is `true`.

true The search is case insensitive.

false The search is case sensitive.

pname

Specifies the report parameter to search on.

p_*parameter*

Specifies the name and values of a prompt parameter. The values used are the `useValue` elements of a `GetPromptAnswersResponse` response element. For example, `p_P_Year=[Promotion].[Time].[Time].[Year]->[Time].[2005]`. (See `Running a report with prompts`.)

The following table shows the forms of this option that may also be used.

Table 3. Prompt parameter samples

Form	Example
p_*parameter*=<*start*>:<*end*>	p_OrderNumber=1156:1156
p_*parameter*=<MIN>:<*end*>	p_SalesTarget=<MIN>:500
p_*parameter*=<*start*>:<MAX>	p_SalesTarget=500:<MAX>
p_*parameter*=<NULL>	p_Country=<NULL>
p_*parameter*=<NONE>	p_Country=<NONE>
p_*parameter*=<![CDATA[<*value*>]]>	p_Time=<![CDATA[12:04:34]]>

p_*parameter*=<NONE> is used to skip an optional parameter. It is distinct from p_*parameter*=<NULL>, which send a NULL value, or p_*parameter*="", which sends a blank value for the prompt.

rowLimit

Specifies the number of rows of data to be returned. For example, `rowLimit=20`.

saveOutput

Specifies that a copy of the report output is to be saved in the Content Store. For example, `saveOutput=true`.

selection

Specifies that only specific report elements is to be returned. For example, `selection=List1`.

Note: For resource types other than outputFormat, you can specify a semicolon-seperated list to return multiple report elements. For example, `selection=List1;List3` will return both `List1` and `List3`.

srchVal

Specifies the keywords to search on.

swsID

Specifies the ID of the prompt.

useRelativeURL

When 1, the URL returned from the IBM Cognos Business intelligence server will be based on the external gateway URI and not on the internal dispatcher URI.

version

Specifies the version of the report to retrieve. Its value is one of the values of the `versionType` element.

versionID

If the value of `version` is `VERSION_NAME`, this option must be used to provide a `versionName` value.

width

Specifies the width of the image returned when using the Image output format (`fmt`). The default value provides an output that is approximately the same size as when the report is viewed in IBM Cognos Viewer.

xmlData

Specifies other request parameters in XML format.

xpath

Specifies a location of the report output as an XPath 2.0 expression. The XPath expression is relative to the layoutDataXML representation of the report output.

See "Using XPath expressions to filter report output" on page 38 for more information.

Secondary requests

Secondary requests are used in REST operations that require several interactive steps to complete, such as:

- Retrieving report output one page at a time.
- Collecting report prompts.
- Drilling up or down in a report.

In order to make secondary requests, the primary request must be an asynchronous request (see the `async` option).

Secondary operation requests are created by taking the URL that is returned by the IBM Cognos Business Intelligence server, and appending to it the secondary request and any required options. The resulting URL is then submitted to the server. For an example, see Collecting prompts from multiple prompt pages.

Any secondary request can also be configured as a resource type. For example,

```
http://webservername:portnumber/ibmcognos/cgi-bin/cognos.cgi/rds
/sessionOutput/conversationID/conv_ID/secondary_request
?option1=val1&option2=val2...
```

is equivalent to

```
http://webservername:portnumber/ibmcognos/cgi-bin/cognos.cgi/rds
/secondary_request/conversationID/conv_ID
?option1=val1&option2=val2...
```

However, when coding mashup applications using the REST interface, it will usually be more convenient to use the first version of the secondary operation syntax.

back

Returns to the previous prompt page.

See Collecting cascading prompts.

Options

None.

drill

Drills up or down in a report.

Options

`contextId`, `direction`

finish

Skips subsequent prompt pages. You can use this request if subsequent prompts have default values.

Options

`p_parameter`

first

Retrieves the first page of report output.

Options

None.

forward

Retrieves the next prompt page.

See Collecting cascading prompts.

Options

p_*parameter*

last

Retrieves the last page of report output.

Options

None.

next

Retrieves the next page of report output.

Options

None.

previous

Retrieves the previous page of report output.

Options

None.

release

Terminates an asynchronous conversation and frees up the server resources associated with this conversation.

Options

None.

reprompt

Use this secondary request on a prompt page that has multiple prompts to submit one or more prompt values and have the current prompt page refreshed, instead of moving to the next prompt page. Use this request if the page contains cascading prompts and you need to submit a prompt response before receiving the subsequent prompt request.

Options

mtchAll, mtchAny, nocase, pname, p_*parameter*, srchVal, swsID

treePromptNode

Use this secondary request on a tree prompt page. Use this request to move down to the next level in the tree hierarchy.

Options

, p_*parameter*

XML elements reference

The following XML elements are not part of the schemas documented elsewhere in this guide.

error

Contains the response to a URL request that fails.

Content Model

message then promptID *(optional)* then url *(optional)*

noerror

Contains the response to a successful log off request.

Content Model

empty

promptAnswers

Contains prompt answers.

This element contains the response to the `promptAnswers` URL request. This element is also submitted in the `xmlData` option when requesting multiple prompt pages with the `promptDescription` resource type or running a report that has prompts.

Content Model

conversationID *(optional)* then promptValues *(any number)*

Chapter 14. Authentication schema reference

For each layout data specification element, this section provides

- the name and description of the element
- sample code that demonstrates how to use the element, or a cross-reference to a topic that contains sample code
- information about attributes that apply to the element, including each attribute's name, description, optionality, legal values, and default value, if applicable
- content model information, consisting of a list of valid child elements presented as an element model group
- a list of valid parent elements

accountID

Specifies the IBM Cognos account ID from Cognos Access Manager (CAM).

For example:

```
<auth:accountInfo xmlns:auth="http://developer.cognos.com/schemas/ccs/auth/types/1">
  <auth:accountID>
    CAMID("ent:u=S-1-5-21-1764567485-459800859-2736415091-4187")
  </auth:accountID>
  <auth:displayName>campbelk</auth:displayName>
</auth:accountInfo>
```

Content model

Content type is string.

Parent elements

accountInfo

accountInfo

Contains the account information returned in a `logonResponse`.

For sample XML, see `accountID`.

Content model

accountID then displayName then extension *(optional)*

Parent elements

result

actualValue

Specifies a known value for the credential piece specified by the `name`.

For sample XML, see `credentialPrompt` or `credentials`.

Content model

Content type is string.

Parent elements

value

credentialElements

Contains a piece of the credential.

Content model

name then label *(optional)* then value then extension *(optional)*

Parent elements

credentialPrompt , credentials

credentialPrompt

Contains list of credentials required for authentication, including both missing and actual values.

For example:

```
<auth:credentialPrompt
  xmlns:auth="http://developer.cognos.com/schemas/ccs/auth/types/1">
  <auth:credentialElements>
    <auth:name>CAMNamespace</auth:name>
    <auth:value>
      <auth:actualValue>ent</auth:actualValue>
    </auth:value>
  </auth:credentialElements>
  <auth:credentialElements>
    <auth:name>CAMNamespaceDisplayName</auth:name>
    <auth:label>Namespace:</auth:label>
    <auth:value>
      <auth:actualValue>ent</auth:actualValue>
    </auth:value>
  </auth:credentialElements>
  <auth:credentialElements>
    <auth:name>CAMUsername</auth:name>
    <auth:label>User ID:</auth:label>
    <auth:value>
      <auth:missingValue>
        <auth:valueType>text</auth:valueType>
      </auth:missingValue>
    </auth:value>
  </auth:credentialElements>
  <auth:credentialElements>
    <auth:name>CAMPassword</auth:name>
    <auth:label>Password:</auth:label>
    <auth:value>
      <auth:missingValue>
        <auth:valueType>textnoecho</auth:valueType>
      </auth:missingValue>
    </auth:value>
  </auth:credentialElements>
</auth:credentialPrompt><auth:credentialPrompt
   xmlns:auth="http://developer.cognos.com/schemas/ccs/auth/types/1">
```

```
  <auth:credentialElements>
    <auth:name>CAMNamespace</auth:name>
    <auth:value>
      <auth:actualValue>ent</auth:actualValue>
    </auth:value>
  </auth:credentialElements>
  <auth:credentialElements>
    <auth:name>CAMNamespaceDisplayName</auth:name>
    <auth:label>Namespace:</auth:label>
    <auth:value>
      <auth:actualValue>ent</auth:actualValue>
    </auth:value>
  </auth:credentialElements>
  <auth:credentialElements>
    <auth:name>CAMUsername</auth:name>
    <auth:label>User ID:</auth:label>
    <auth:value>
      <auth:missingValue>
        <auth:valueType>text</auth:valueType>
      </auth:missingValue>
    </auth:value>
  </auth:credentialElements>
  <auth:credentialElements>
    <auth:name>CAMPassword</auth:name>
    <auth:label>Password:</auth:label>
    <auth:value>
      <auth:missingValue>
        <auth:valueType>textnoecho</auth:valueType>
      </auth:missingValue>
    </auth:value>
  </auth:credentialElements>
</auth:credentialPrompt>
```

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

credentialElements *(any number)* then extension *(optional)*

Parent elements

result

credentials

Contains list of credentials to pass to the server for authentication.

For example:

```
<auth:credentials xmlns:auth="http://developer.cognos.com/schemas/ccs/auth/types/1">
  <auth:credentialElements>
    <auth:name>CAMNamespace</auth:name>
    <auth:value>
      <auth:actualValue>ent</auth:actualValue>
    </auth:value>
  </auth:credentialElements>
  <auth:credentialElements>
    <auth:name>CAMNamespaceDisplayName</auth:name>
    <auth:value>
```

```
      <auth:actualValue>ent</auth:actualValue>
    </auth:value>
  </auth:credentialElements>
  <auth:credentialElements>
    <auth:name>CAMUsername</auth:name>
      <auth:value>
      <auth:actualValue>myuserid</auth:actualValue>
    </auth:value>
  </auth:credentialElements>
  <auth:credentialElements>
    <auth:name>CAMPassword</auth:name>
    <auth:value>
      <auth:actualValue>mypasswd</auth:actualValue>
    </auth:value>
  </auth:credentialElements>
</auth:credentials>
```

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

credentialElements *(any number)* then extension *(optional)*

Parent elements

logonRequest

displayName

Specifies the display name for the account, if available.

For sample XML, see `accountID`.

Content model

Content type is string.

Parent elements

accountInfo

enumeration

Contains a list of values that can be used for the credential. This list is provided by the server.

Content model

label then value

Parent elements

missingValue

extension

Reserved.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

item *(any number)*

Parent elements

accountInfo , credentialElements , credentialPrompt , credentials , logoffRequest , logoffResponse , logonRequest , logonResponse , missingValue , result , value

item

Reserved.

Content model

Content type is anyType.

Parent elements

extension

label

Specifies the label for the credential piece or the enumeration value.

Content model

Content type is string.

Parent elements

credentialElements , enumeration

logoffRequest

Requests a logoff.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

extension *(optional)*

logoffResponse

Contains the response to the `logoffRequest` command.

Content model

responseCode then responseMessage *(optional)* then extension *(optional)*

logonRequest

Requests a logon.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

credentials then extension *(optional)*

logonResponse

Contains response to the `logonRequest` command.

Content model

responseCode then responseMessage *(optional)* then result then extension *(optional)*

missingValue

Specifies a missing value required by the server for authentication.

For sample XML, see `credentialPrompt`.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

valueType then enumeration *(any number)* then extension *(optional)*

Parent elements

value

name

Specifies the name of the credential piece. This element contains an ID used by Cognos Access Manager (CAM).

For sample XML, see `credentialPrompt` or `credentials`.

Content model

Content type is string.

Parent elements

credentialElements

noResult

Specifies that no result is returned due to an error in response to the `logonRequest` command. This element is returned when the `responseCode` element contains the value `ERROR`.

Content model

Empty element.

Parent elements

result

responseCode

Specifies whether an error occurred during the request. This element contains the response code to the `logonRequest` and `logoffRequest` commands.

Content model

Content type is string.

The possible values of this element are restricted to the following.

ERROR

There was an error in the request.

NO_ERROR

There was no error in the request.

Parent elements

logoffResponse , logonResponse

responseMessage

Contains the response message returned for the `logonRequest` or the `logoffRequest` commands.

Content model

Content type is string.

Parent elements

logoffResponse , logonResponse

result

Contains the result for the `logonRequest` command.

Content model

credentialPrompt or accountInfo or noResult or extension

Parent elements

logonResponse

value

Specifies the value for the enumeration.

Content model

actualValue or missingValue or extension

Parent elements

credentialElements

value

Container for a missing or actual value in the credential piece.

Content model

Content type is string.

Parent elements

enumeration

valueType

Specifies the type of UI control to prompt for the missing value, for example: `text`, `textnoecho`, or `picklist`. This type is determined by Cognos Access Manager (CAM).

Content model

Content type is string.

Parent elements

missingValue

Chapter 15. RDS schema reference

For each RDS specification element, this section provides

- the name and description of the element
- information about attributes that apply to the element, including each attribute's name, description, optionality, legal values, and default value, if applicable
- content model information, consisting of a list of valid child elements presented as an element model group
- a list of valid parent elements

autoSubmit

Specifies whether to automatically submit the prompt value to the server if the selected value changes.

When `true`, the client should immediately submit the selected value to the server when the value changes and the `reprompt` element should be set to `true`.

Content model

Content type is boolean.

Parent elements

PDateTimeBox , PListBox , PSearchAndSelect , PTextBox , PTreePrompt

BackRequest

A secondary request to return to the previous prompt page.

Content model

session then extension *(optional)*

burstId

Specifies the `burstID`.

For more information on `burstID`, see the *IBM Cognos Software Development Kit Developer Guide*.

Content model

Content type is string.

Parent elements

burstInfo

burstInfo

Specifies the burst information required to retrieve a bursted version of a report.

To identify the burst, you must specify the `burstKey` or the `burstId`.

Content model

burstKey *(optional)* then burstId *(optional)* then extension *(optional)*

Parent elements

GetOutputFormatRequest , GetPagedReportDataRequest , GetReportDataRequest

burstKey

Specifies the `burstKey`.

For more information on `burstKey`, see the *IBM Cognos Software Development Kit Developer Guide*.

Content model

Content type is string.

Parent elements

burstInfo

calendarType

Specifies the type of calendar used for the prompt.

Content model

Content type is string.

The possible values of this element are restricted to the following.

GREGORIAN

The calendar type is the standard international calendar.

IMPERIAL

The calendar type is the Japanese Imperial calendar.

Parent elements

PDateTimeBox

canExpand

Specifies whether the tree prompt can be expanded.

Content model

Content type is boolean.

Parent elements

PTreePrompt

canFinish

Indicates whether the prompt page can be the last prompt page.

When `true`, the user can click the **Finish** button and run the report.

Content model

Content type is boolean.

Parent elements

prompts

caseInsensitive

Specifies whether case is considered when processing prompt values.

When `true`, the search is not case sensitive. When `false`, the search is case sensitive. The default value is `true`.

Content model

Content type is boolean.

Parent elements

PSearchAndSelect , searchPValue

CCSAuthenticationFault

Contains a fault that occurs when authentication is required before the request can be completed.

Content model

message then extension *(optional)*

CCSGeneralFault

Contains a fault that occurs when an error prevents the request from completing.

Content model

message then trace *(optional)*

CCSPromptFault

Contains a fault that occurs when prompt answers are required before the request can complete.

Content model

message then promptID then url then extension *(optional)*

columnName

Specifies the display name of the column that contains the prompt values.

Content model

Content type is string.

Parent elements

PDateTimeBox , PListBox , PSearchAndSelect , PTextBox , PTreePrompt

connection

Defines the connection to the data source.

Content model

name then searchPath then selected

Parent elements

PDataSource

contextID

Specifies the context information related to the object on which the drill is executed.

The value in this element matches the value in the `ctx` element in the layout data document.

Content model

Content type is string.

Parent elements

DrillRequest

conversationID

Specifies the `storeID` of the conversation object stored in the Content Manager.

For more information on `storeID`, see the *IBM Cognos Software Development Kit Developer Guide*.

Content model

Content type is string.

Parent elements

GetPromptDescriptionRequest , GetPromptDescriptionResponse , GetTreePromptNodeRequest , session

direction

Specifies if the drill is up or down.

Content model

Content type is string.

The possible values of this element are restricted to the following.

UP

The drill direction is up, to a higher level of granularity.

DOWN

The drill direction is down, to a lower level of granularity.

Parent elements

DrillRequest

displayMilliseconds

Specifies whether the prompt control displays milliseconds.

Content model

Content type is boolean.

Parent elements

PDateTimeBox

displaySeconds

Specifies whether the prompt control displays seconds.

Content model

Content type is boolean.

Parent elements

PDateTimeBox

displayValue

Specifies the display value that appears in the prompt control.

Content model

Content type is string.

Parent elements

end , options , SimplePValue , start

DrillRequest

Makes a secondary request to drill into a resource.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

session then contextID then direction

end

Specifies the end value of the range.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

inclusive then useValue then displayValue *(optional)*

Parent elements

RangePValue

excludePage

Specifies that when the filterType element is being used to select a report part by name, report pages are not considered. The default value is `false`.

Although report parts in a report must have unique names, it is possible for a report page to have the same name as a report part, such as a list. Setting this option to `true` will ensure that a report part is selected rather than a report page with the same name.

Content model

Content type is boolean.

Parent elements

GetPagedReportDataRequest , GetReportDataRequest

extension

Placeholder for future extensions.

Content model

Content type is anyType.

Parent elements

BackRequest , burstInfo , CCSAuthenticationFault , CCSPromptFault , filters , FinishRequest , FirstRequest , ForwardRequest , GetCognosURLRequest , GetCognosURLResponse , GetOutputFormatRequest , GetOutputFormatResponse , GetOutputFormatsRequest , GetOutputFormatsResponse , GetOutputRequest , GetOutputResponse , GetPagedReportDataRequest , GetPromptAnswersRequest , GetPromptAnswersResponse , GetPromptDescriptionRequest , GetPromptDescriptionResponse , GetPromptPageRequest , GetPromptPageResponse , GetReportDataRequest , GetReportPromptsRequest , GetTreePromptNodeRequest , GetTreePromptNodeResponse , item , item , LastRequest , NextRequest , output , PDateTimeBox , PListBox , PreviousRequest , PromptAnswerOutput , promptAnswers , PSearchAndSelect , PTextBox , PTreePrompt , RangePValue , ReleaseRequest , ReleaseResponse , RepromptRequest , searchPValue , searchValue , session , supportedFormats , treeNode , treePromptNode , version

filters

Defines filters applied to the layout data document. Filters allow users to select report parts to return, instead of the complete report.

See “Accessing parts of a report output” on page 38 for more information.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

filterValue then filterType then extension *(optional)*

Parent elements

GetOutputFormatRequest , GetPagedReportDataRequest , GetReportDataRequest

filterType

Specifies the type of filter applied.

Content model

Content type is string.

The possible values of this element are restricted to the following.

OBJECT_ID

The resource is filtered on a named object.

CONTEXT_SPEC

Reserved.

XPATH

The resource is filtered on a XPath expression.

Parent elements

filters

filterValue

Specifies the value used in the filter.

Content model

Content type is string.

Parent elements

filters

FinishRequest

A secondary request to skip subsequent prompt pages. You can use this request if subsequent prompts have default values.

Content model

session then promptValues *(any number)* then searchValue *(optional)* then extension *(optional)*

firstDate

Specifies the first date that can be selected in the date and time prompt control.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

Content type is date.

Parent elements

PDateTimeBox

FirstRequest

A secondary request to retrieve the first page of report output.

Content model

session then extension *(optional)*

format

Specifies the format of the retrieved report.

See "Output formats" on page 9 for more information.

layoutDataXML
Output is XML based on the LDX schema. See `Layout Data Schema Reference`.

HTML Output is a complete HTML file with style information in an inline stylesheet.

HTMLFragment
Output is a fragment of an HTML file with inline style information.

JSON Output is in JavaScript Object Notation format.

Simple Output is in Simple format. See `Using the Simple Format`.

Image Output is a portable network graphic (.png) image of the report output.

DataSet
Output is XML in the DataSet format. See `Using the DataSet format`.

DataSetAtom
Output is XML in the Atom version of the DataSet format. See `Using the DataSet format`.

Content model

Content type is string.

Parent elements

GetPagedReportDataRequest , GetReportDataRequest

FormatOutput

Contains the output from the `getPagedReportData` and `getReportData` commands when the `format` option is included in the request.

Content model

Content type is string.

Parent elements

output

ForwardRequest

A secondary request to obtain the next prompt page

Content model

session then promptValues *(any number)* then searchValue *(optional)* then extension *(optional)*

GetCognosURLRequest

Retrieves a URL to display the resource in IBM Cognos Viewer.

Content model

sourceID then sourceType then extension *(optional)*

GetCognosURLResponse

Contains the response to the `GetCognosURLRequest` command.

For example:

```
<rds:GetCognosURLResponse
  xmlns:rds="http://developer.cognos.com/schemas/rds/types/2">
  <rds:url>
  http://localhost:80/install/cgi-bin/cognos.cgi?b_action=
  cognosViewer&amp;amp;ui.action=run
    &amp;amp;ui.object=storeID("iA230F89540954FE79F8F1C87D8625835")
  </rds:url>
</rds:GetCognosURLResponse>
```

Content model

url then extension *(optional)*

GetOutputFormatRequest

Requests the information needed to run a report and retrieve the output in a specified format.

See "Running reports and retrieving output in IBM Cognos Viewer formats" on page 40 for more information.

Content model

sourceID then sourceType then outputFormatName then filters *(any number)* then version *(optional)* then burstInfo *(optional)* then saveOutput *(optional)* then promptValues *(any number)* then extension *(optional)*

GetOutputFormatResponse

Contains the information necessary to run a report and the retrieve the report output in a specified format.

Content model

outputFormatURL then xmlData *(optional)* then extension *(optional)*

GetOutputFormatsRequest

Requests a list of supported formats for a report.

See "Running reports and retrieving output in IBM Cognos Viewer formats" on page 40 for more information.

Content model

sourceID then sourceType then extension *(optional)*

GetOutputFormatsResponse

Contains the list of supported formats for a specific report

Content model

supportedFormats then extension *(optional)*

GetOutputRequest

Polls the server for a response for asynchronous methods.

Content model

session then extension *(optional)*

GetOutputResponse

Contains the response to asynchronous generic commands.

See "Running Mashup Service methods" on page 28 for more information.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

session then output *(optional)* then extension *(optional)*

GetPagedReportDataRequest

Runs a report interactively (if supported), retrieving the output page by page.

Content model

sourceID then sourceType then rowLimit *(optional)* then version *(optional)* then burstInfo *(optional)* then excludePage *(optional)* then filters *(any number)* then format *(optional)* then saveOutput *(optional)* then includeLayout *(optional)* then promptValues *(any number)* then extension *(optional)* then useRelativeURL *(optional)*

GetPromptAnswersRequest

Retrieves the prompt answers associated with the prompt ID.

This command is used after the `GetPromptPageRequest` command.

Content model

promptID then extension *(optional)*

GetPromptAnswersResponse

Contains the response to the `GetPromptAnswersRequest` command.

Content model

promptValues *(any number)* then extension *(optional)*

GetPromptDescriptionRequest

Retrieves a description of the prompts using the `getPromptDescription` command.

The `getPromptDescription` is deprecated and will be removed in a future version of this product. Use the `getReportPrompts` request instead.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

sourceID then sourceType then conversationID *(optional)* then reprompt *(optional)* then promptValues *(any number)* then searchPValue *(optional)* then extension *(optional)*

GetPromptDescriptionResponse

Contains the response to the `getPromptDescription` command.

For example:

```
<rds:GetPromptDescriptionResponse
  xmlns:rds="http://developer.cognos.com/schemas/rds/types/2">
  <rds:prompts>
    <canFinish>true</canFinish>
    <hasNextPage>false</hasNextPage>
    <rds:item>
      <rds:PSearchAndSelect>
        <rds:name>Product name</rds:name>
        <rds:multiSelect>true</rds:multiSelect>
        <rds:required>true</rds:required>
        <rds:id>_P1728309400</rds:id>
        <rds:parameter>Product name</rds:parameter>
        <rds:matchAll>false</rds:matchAll>
        <rds:matchAnywhere>false</rds:matchAnywhere>
        <rds:columnName>Product name</rds:columnName>
```

```
      </rds:PSearchAndSelect>
    </rds:item>
  </rds:prompts>
  <rds:conversationID>i6CFAC01FA8E74D4A8F0A526E798A78DE</rds:conversationID>
</rds:GetPromptDescriptionResponse>
```

The `getPromptDescription` is deprecated and will be removed in a future version of this product. Use the `getReportPrompts` instead.

Content model

prompts then conversationID then extension *(optional)*

GetPromptPageRequest

Retrieves a URL from the IBM Cognos Business Intelligence server to fulfill prompt answers, as well as a `promptID` to use in the `getPromptAnswers` command.

Content model

sourceID then sourceType then promptValues *(any number)* then extension *(optional)* then useRelativeURL *(optional)*

GetPromptPageResponse

Contains the response to the `getPromptPage` command.

For example:

```
<rds:GetPromptPageResponse
  xmlns:rds="http://developer.cognos.com/schemas/rds/types/2">
  <rds:promptID>iC2FA4D960B9E4D3DA41481DC12697691</rds:promptID>
  <rds:url>http://localhost:80/install/cgi-bin/cognos.cgi?b_action=xts.run
  &amp;m=ccs/ccs_prompt.xts
  &amp;ui.object=storeID("iFEEA9784505F408FA735A839790994B5")
  &amp;promptID=iC2FA4D960B9E4D3DA41481DC12697691</rds:url>
</rds:GetPromptPageResponse>
```

Content model

promptID then url then extension

GetReportDataRequest

Retrieves the report content using the `getReportData` command.

Content model

sourceID then sourceType then rowLimit *(optional)* then version *(optional)* then burstInfo *(optional)* then excludePage *(optional)* then filters *(any number)* then format *(optional)* then includeLayout *(optional)* then saveOutput *(optional)* then includePageBreaks *(optional)* then promptValues *(any number)* then extension *(optional)* then useRelativeURL *(optional)*

GetReportPromptsRequest

Retrieves a description of the prompts using the `getReportPrompts` command.

Content model

sourceID then sourceType then includeLayout *(optional)* then extension *(optional)*

GetTreePromptNodeRequest

Retrieves the next level of children for a specified node.

This command is used with the `GetPromptDescriptionRequest` command for `PTreePrompt` requests. The `nodeValue` child element specifies the node.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

conversationID *(optional)* then session then nodeValue then extension *(optional)*

GetTreePromptNodeResponse

Contains the response to the `GetTreePromptNodeRequest` request.

Content model

treePromptNode then extension *(optional)*

hasNextPage

Specifies that the prompt control has a **Next** page.

Content model

Content type is boolean.

Parent elements

prompts

id

Specifies an identifier for the search and select prompt control. This identifier is used to perform a search for prompt values.

Content model

Content type is string.

Parent elements

PSearchAndSelect , searchPValue

includeLayout

If true, include the following layout elements in the output: `blk`, `tbl`, and `sngl`. For `getReportData` requests, the default is `false`. For `getPagedReportData` and `getReportPrompts` requests, the default is `true`.

Content model

Content type is boolean.

Parent elements

GetPagedReportDataRequest , GetReportDataRequest , GetReportPromptsRequest

includePageBreaks

Specifies whether to retrieve the physical pages of output, or to retrieve the conceptual pages from the original resource.

When `true`, the document returned contains each physical page of output. When `false`, the document returned contains each type of page from the original resource. See “Reports with multiple pages” on page 64 for more information.

The default value is `false`.

Content model

Content type is boolean.

Parent elements

GetReportDataRequest

inclusive

Specifies that the value range for the prompt is inclusive.

Content model

Content type is boolean.

Parent elements

end , RangePValue , SimplePValue , start

item

Contains a prompt value.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

SimplePValue or RangePValue or sval or rval or extension

Parent elements

values

item

Contains a prompt value.

Content model

PListBox or PTextBox or PTreePrompt or PDateTimeBox or PDataSource or PSearchAndSelect or extension *(optional)*

Parent elements

prompts

lastDate

Specifies the last date value that can be selected in the prompt control.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

Content type is date.

Parent elements

PDateTimeBox

LastRequest

A secondary request to retrieve the last page of report output.

Content model

session then extension *(optional)*

LDXOutput

Contains the report output in LayoutDataXML format.

Content model

document or filterResultSet

Parent elements

output

matchAll

Specifies whether the search results returned match all of the search words entered.

When `true`, the results returned match all of the search words entered. When `matchAnywhere` is also `true`, then the search can contain all the keywords in any order.

When `false`, the results returned match any of the search words entered.

Content model

Content type is boolean.

Parent elements

PSearchAndSelect , searchPValue

matchAnywhere

Specifies whether the search results contain or start with the keywords.

When `true`, the results returned contain the key words. When `false`, the search results returned start with the keywords.

Content model

Content type is boolean.

Parent elements

PSearchAndSelect , searchPValue

message

Contains a descriptive message about the error that occurred.

Content model

Content type is string.

Parent elements

CCSAuthenticationFault , CCSGeneralFault , CCSPromptFault

mtchAll

Specifies whether the search results returned match all of the search words entered.

When `true`, the results returned match all of the search words entered. When `mtchAny` is also `true`, then the search can contain all the keywords in any order.

When `false`, the results returned match any of the search words entered.

Content model

Content type is boolean.

Parent elements

searchValue

mtchAny

Specifies whether the search results contain or start with the keywords.

When `true`, the results returned contain the key words. When `false`, the search results returned start with the keywords.

Content model

Content type is boolean.

Parent elements

searchValue

multiSelect

Specifies whether the prompt is a multi-value prompt.

When `true`, this is a multi-value prompt.

Content model

Content type is boolean.

Parent elements

PDateTimeBox , PListBox , PSearchAndSelect , PTextBox , PTreePrompt

name

Specifies the name of the prompt parameter or value.

Content model

Content type is string.

Parent elements

connection , nodeValue , PDataSource , PDateTimeBox , PListBox , promptValues , PSearchAndSelect , PTextBox , PTreePrompt , signon

NextRequest

A secondary request to retrieve the next page of report output.

Content model

session then extension *(optional)*

nocase

Specifies whether the search is case sensitive or case insensitive.

Content model

Content type is boolean.

Parent elements

searchValue

nodeValue

Specifies the prompt values to retrieve from the children of the node.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

name then values

Parent elements

GetTreePromptNodeRequest

numericOnly

Specifies whether the prompt requires numeric values.

Content model

Content type is boolean.

Parent elements

PTextBox

options

Specifies the list of possible values in the prompt control.

Content model

useValue then displayValue *(optional)*

Parent elements

PListBox , PSearchAndSelect , selections , selectionsAncestry , treeNode , treePromptNode

output

Contains the output from an asynchronous method.

Content model

LDXOutput or PromptAnswerOutput or FormatOutput or extension *(optional)*

Parent elements

GetOutputResponse

outputFormatName

The name of a format supported by the specified report.

The possible values for the output formats are described in the `outputFormatEnum` enumeration set documented in the *IBM Cognos Software Development Kit Developer Guide*

Content model

Content type is string.

Parent elements

GetOutputFormatRequest , supportedFormats

outputFormatURL

Contains a URL that, when submitted to the IBM Cognos BI server, runs a report and retrieves the output in a specified format.

Content model

Content type is string.

Parent elements

GetOutputFormatResponse

parameter

Specifies the name of the parameter associated with the prompt.

Content model

Content type is string.

Parent elements

PSearchAndSelect

parameterName

Specifies the name of the parameter associated with the search.

Content model

Content type is string.

Parent elements

searchPValue

PDataSource

Represents a data source signon prompt.

Content model

name then connection then signon *(any number)*

Parent elements

item

PDateTimeBox

Represents any of the date and time prompt controls, including date, date and time, time, and interval. This prompt control is usually rendered with a calendar control.

Content model

name then multiSelect then range then required then valueType then autoSubmit then calendarType *(optional)* then displaySeconds *(optional)* then displayMilliseconds *(optional)* then firstDate *(optional)* then lastDate *(optional)* then columnName *(optional)* then selections *(optional)* then extension *(optional)*

Parent elements

item

PListBox

Represents a value prompt control, usually represented as either a list box for selecting multiple values or as a drop-down list for selecting a single value.

Content model

name then multiSelect then range then required then autoSubmit then columnName *(optional)* then selections then options *(any number)* then extension *(optional)*

Parent elements

item

pname

Specifies the report parameter to search on.

Content model

Content type is string.

Parent elements

searchValue

PreviousRequest

A secondary request to retrieve the previous page of report output.

Content model

session then extension *(optional)*

PromptAnswerOutput

Contains the prompt answers submitted by a user on an IBM Cognos prompt page.

See "Using the IBM Cognos prompt page interface" on page 42 for more information.

Content model

promptValues *(any number)* then extension *(optional)*

Parent elements

output

promptAnswers

Reserved.

Content model

promptValues *(any number)* then extension *(optional)*

promptID

Specifies an identifier where the prompt answers will be stored.

See “Using the IBM Cognos prompt page interface” on page 42 for more information.

Content model

Content type is string.

Parent elements

CCSPromptFault , GetPromptAnswersRequest , GetPromptPageResponse

prompts

Contains the retrieved prompt control descriptions.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

canFinish then hasNextPage then item *(any number)* then promptValues *(optional)*

Parent elements

GetPromptDescriptionResponse

promptValues

Represents one or more prompt values.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

name then values

Parent elements

FinishRequest , ForwardRequest , GetOutputFormatRequest , GetPagedReportDataRequest , GetPromptAnswersResponse , GetPromptDescriptionRequest , GetPromptPageRequest , GetReportDataRequest , PromptAnswerOutput , promptAnswers , prompts , RepromptRequest

PSearchAndSelect

Represents a search and select prompt control.

This type of prompt retrieves values based on search criteria that a user specifies.

Content model

name then multiSelect then range then required then id then parameter then caseInsensitive then matchAll then matchAnywhere then autoSubmit then columnName *(optional)* then selections *(optional)* then options *(any number)* then extension *(optional)*

Parent elements

item

PTextBox

Represents a text box prompt control.

This type of prompt retrieves data based on a value that a user types.

Content model

name then multiSelect then range then required then numericOnly then autoSubmit then columnName *(optional)* then extension *(optional)*

Parent elements

item

PTreePrompt

Represents a tree prompt control.

This type of prompt retrieves data based on values that users select from a list. Values are organized hierarchically.

Content model

name then multiSelect then required then treeUI then canExpand then autoSubmit then columnName *(optional)* then selections *(optional)* then selectionsAncestry *(optional)* then treeNode then extension *(optional)*

Parent elements

item

range

Specifies whether this is a range prompt.

When `true`, this is a range prompt control. The client will render two text box or two list box UI elements for this type of prompt control.

Content model

Content type is boolean.

Parent elements

PDateTimeBox , PListBox , PSearchAndSelect , PTextBox

RangePValue

Specifies a range prompt value. The range may include a start and/or an end value, depending on the type of range prompt.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

inclusive then start *(optional)* then end *(optional)* then extension *(optional)*

Parent elements

item

ReleaseRequest

Releases a session so that no further requests can be made.

This request cancels any currently running requests, freeing resources associated with the request. If you do not make this request, the resources remain unavailable until the session times out.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

session then extension *(optional)*

ReleaseResponse

Contains the response to the `ReleaseRequest` command.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

extension *(optional)*

reprompt

Specifies whether the user is prompted again instead of going to the next page.

When `true`, the user is prompted again. Set this element to `true` when the `autoSubmit` element is `true`.

Content model

Content type is boolean.

Parent elements

GetPromptDescriptionRequest

RepromptRequest

Use this secondary request on a prompt page that has multiple prompts to submit one or more prompt values and have the current prompt page refreshed, instead of moving to the next prompt page. Use this request if the page contained cascading prompts and you needed to submit a prompt response before receiving the subsequent prompt request.

Content model

session then promptValues *(any number)* then searchValue *(optional)* then extension *(optional)*

required

Specifies whether a value must be supplied for this prompt to run the report.

When `true`, a value must be specified.

Content model

Content type is boolean.

Parent elements

PDateTimeBox , PListBox , PSearchAndSelect , PTextBox , PTreePrompt

rowLimit

Specifies the maximum number of rows to retrieve.

The default value is `0`, which returns all rows.

Content model

Content type is int.

Parent elements

GetPagedReportDataRequest , GetReportDataRequest

saveOutput

Specifies that a copy of the report output is to be saved in the Content Store.

The default value is `false`.

Content model

Content type is boolean.

Parent elements

GetOutputFormatRequest , GetPagedReportDataRequest , GetReportDataRequest

searchPath

Specifies a search path to the version object.

Content model

Content type is string.

Parent elements

connection , signon , version

searchPValue

Specifies the value entered in the search prompt control.

Content model

value then parameterName then id then caseInsensitive *(optional)* then matchAll *(optional)* then matchAnywhere *(optional)* then extension *(optional)*

Parent elements

GetPromptDescriptionRequest

searchValue

Specifies a Search & Select prompt value.

Content model

srchval then swsID then pname then nocase *(optional)* then mtchAny *(optional)* then mtchAll *(optional)* then extension *(optional)*

Parent elements

FinishRequest , ForwardRequest , RepromptRequest

selected

Specifies the selected signon or connection.

Content model

Content type is boolean.

Parent elements

connection , signon

selections

Contains a list of default prompt answers that user has saved.

Content model

options *(any number)*

Parent elements

PDateTimeBox , PListBox , PSearchAndSelect , PTreePrompt

selectionsAncestry

Defines the ancestry of the saved prompt answers defined in the `selections` element.

Content model

options *(any number)*

Parent elements

PTreePrompt

session

Specifies the session identifier for asynchronous and secondary requests.

See "Running Mashup Service methods" on page 28 for more information.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

conversationID then status then extension *(optional)*

Parent elements

BackRequest , DrillRequest , FinishRequest , FirstRequest , ForwardRequest , GetOutputRequest , GetOutputResponse , GetTreePromptNodeRequest , LastRequest , NextRequest , PreviousRequest , ReleaseRequest , RepromptRequest

signon

Represents the signon to the data source.

Content model

name *(optional)* then searchPath *(optional)* then selected

Parent elements

PDataSource

SimplePValue

Specifies a simple prompt value.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

inclusive then useValue then displayValue *(optional)*

Parent elements

item

sourceID

Specifies the identifier of the resource.

Content model

Content type is string.

Parent elements

GetCognosURLRequest , GetOutputFormatRequest , GetOutputFormatsRequest , GetPagedReportDataRequest , GetPromptDescriptionRequest , GetPromptPageRequest , GetReportDataRequest , GetReportPromptsRequest

sourceType

Specifies the type of resource.

Content model

Content type is string.

The possible values of this element are restricted to the following.

metrics

Reserved.

conversationID

The source is a conversation resource that is stored in content manager for async and secondary requests.

path

The source is a resource that is referenced by its simplified path.

report

The source is a resource that is referenced by its storeID.

searchPath

The source is a resource that is referenced by its search path.

Parent elements

GetCognosURLRequest , GetOutputFormatRequest , GetOutputFormatsRequest , GetPagedReportDataRequest , GetPromptDescriptionRequest , GetPromptPageRequest , GetReportDataRequest , GetReportPromptsRequest

srchval

Specifies the keywords to search on.

Content model

Content type is string.

Parent elements

searchValue

start

Specifies the start value for a range prompt.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

inclusive then useValue then displayValue *(optional)*

Parent elements

RangePValue

status

Specifies the async status.

Content model

Content type is string.

The possible values of this element are restricted to the following.

working

The asynch status is working.

complete

The asynch status is complete. The command output can now be retrieved in the output element.

Parent elements

session

supportedFormats

Contains the list of output formats supported by the specified report.

Content model

outputFormatName *(any number)* then extension *(optional)*

Parent elements

GetOutputFormatsResponse

swsID

Specifies the ID of the prompt.

Content model

Content type is string.

Parent elements

searchValue

trace

Contains additional trace data, if available.

Content model

Content type is string.

Parent elements

CCSGeneralFault

treeNode

Represents a root node in a prompt control.

Content model

options *(any number)* then extension *(optional)*

Parent elements

PTreePrompt

treePromptNode

Contains the children of the requested node.

Content model

options *(any number)* then extension *(optional)*

Parent elements

GetTreePromptNodeResponse

treeUI

Specifies the kind of tree used for the tree prompt control.

Content model

Content type is string.

Parent elements

PTreePrompt

url

Specifies the URL.

Content model

Content type is string.

Parent elements

CCSPromptFault , GetCognosURLResponse , GetPromptPageResponse

useRelativeURL

When `true`, the URL returned from the IBM Cognos Business intelligence server will be based on the external gateway URI and not on the internal dispatcher URI.

Content model

Content type is boolean.

Parent elements

GetPagedReportDataRequest , GetPromptPageRequest , GetReportDataRequest

useValue

Specifies the value used for the prompt control.

Content model

Content type is string.

Parent elements

end , options , SimplePValue , start

value

Specifies the keywords to search on.

Content model

Content type is string.

Parent elements

searchPValue

values

Specifies the selected prompt value(s).

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

item *(any number)*

Parent elements

nodeValue , promptValues

valueType

Specifies the type of prompt.

Content model

Content type is string.

The possible values of this element are restricted to the following.

DATE

The prompt is a date prompt.

This type of prompt retrieves data based on a date that the user selects.

TIME

The prompt is a time prompt.

This type of prompt retrieves data based on a time that the user selects.

DATETIME

The prompt is a date and time prompt.

This type of prompt retrieves data based on a date and time that the user selects.

INTERVAL

The prompt is an interval prompt.

This type of prompt retrieves data based on a time interval that the user specifies.

Parent elements

PDateTimeBox

version

Specifies the version of the report requested.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

versionType then versionName *(optional)* then searchPath *(optional)* then extension *(optional)*

Parent elements

GetOutputFormatRequest , GetPagedReportDataRequest , GetReportDataRequest

versionName

Specifies the reportVersion of the report being run. For example, for the report stored in Content Manager that has the following search path

```
/content/folder[@name='Samples']/folder[@name='Models']
/package[@name='GO Data Warehouse (analysis)']/folder[@name='Report Studio Report Samples']
/report[@name='Customer Returns and Satisfaction']
/reportVersion[@name='2008-10-08T15:33:46.781Z']
```

the `versionName` is `2008-10-08T15:33:46.781Z`

Content model

Content type is string.

Parent elements

version

versionType

Specifies the version type.

Content model

Content type is string.

The possible values of this element are restricted to the following.

NEW

The report is run to obtain content based on most recent data, instead of a stored version.

LATEST

Retrieves the latest stored version. If no stored version exists, the report is run and retrieved.

VERSION_NAME

Retrieve stored version with the specified name.

NO_DATA

The report is run without retrieving data. Artificial data is used instead of actual data from the data source.

LIMITED_DATA

The report is run with limited data based on design mode filters defined in Framework Manager.

Parent elements

version

xmlData

Contains data that must be submitted as form data in the `xmlData` REST option when retrieving report ouput.

For example, if the report is being run with prompt values, this element will contain a string giving prompt values in the form of a `promptAnswers` element. See Running a report with prompts for an example.

Content model

Content type is string.

Parent elements

GetOutputFormatResponse

Chapter 16. Layout Data (LDX) schema reference

For each layout data schema element, this section provides

- the name and description of the element
- information about attributes that apply to the element, including each attribute's name, description, optionality, legal values, and default value, if applicable
- content model information, consisting of a list of valid child elements presented as an element model group
- a list of valid parent elements

actionURL

Reserved.

Content model

Empty element.

Parent elements

drillAction

Alpha

Represents the alpha value of the color which is a value between 0.0 (transparent) and 1.0 (opaque). The default value is 1.0.

Content model

Content type is double.

Parent elements

bgColor , color , fgColor

alternateText

If available, the alternate text to display for this image

Content model

Content type is string.

Parent elements

area , cht , img

ancestors

Specifies a list of ancestors of the currently selected node.

Content model

sval *(any number)* or rval *(any number)* or extension *(optional)*

Parent elements

p_tree

annURL

Reserved.

Content model

Content type is string.

Parent elements

blk , bmrk , cell , cht , colTitle , corner , ctab , hlink , html , img , lcr , lst , name , p_btn , p_date , p_dsrc , p_dtime , p_intrvl , p_srch , p_time , p_tree , p_txtbox , p_value , rept , reptbl , rtxt , sngl , table , tbl , tcell , toc , txt , widget

area

Defines an area in a chart. The `coord` child element defines the coordinates of the area. The area can be used to define drill throughs and tooltips on a chart.

Content model

type then coord *(one or more)* then alternateText *(optional)* then drills *(optional)* then drillAction *(any number)* then label *(optional)* then ctx *(optional)* then member *(any number)* then measure *(any number)*

Parent elements

regions

attachment

Specifies whether the background image scrolls with the page. The default value is SCROLL.

Content model

Content type is string.

The possible values of this element are restricted to the following.

FIXED

The background image does not scroll with the page.

SCROLL

The background image scrolls with the page.

INHERIT

The scrolling behavior is inherited from the parent container.

Parent elements

bgImageProperties

auto

Specifies whether the application should submit the prompt page automatically, as soon as a value is changed.

For more information, see **Auto-Submit** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is boolean.

Parent elements

p_value

autocascade

This is `true` if the prompt satisfies the following conditions:

- The prompt is the source of a cascading prompt and the value of auto is `true`.
- The cascading prompt appears on the same page as this prompt.

In this case, when a prompt value is changed, the prompt should be submitted, and the values of the cascading prompt populated based on the value chosen in the source prompt.

See "Collecting cascading prompts from a single prompt page" on page 51 for an example of the use of this element.

Content model

Content type is boolean.

Parent elements

p_value

bgColor

Defines a background color.

Content model

Red then Green then Blue then Alpha *(optional)* then extension *(optional)*

Parent elements

styleGroup

bgImageProperties

Specifies how the background image associated with this style should be displayed.

Content model

position *(optional)* then attachment *(optional)* then repeat *(optional)* then extension *(optional)*

Parent elements

styleGroup

bgImageURL

Specifies a URL that points to a background image.

Content model

Content type is string.

Parent elements

styleGroup

biDirectional

Specifies that the formatting of the text is bi-directional.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

Content type is string.

The possible values of this element are restricted to the following.

NORMAL

The element does not open an additional level of embedding with respect to the bidirectional algorithm.

EMBED

If the element is inline-level, this value opens an additional level of embedding with respect to the bidirectional algorithm. The direction of this embedding level is given by the direction property.

OVERRIDE

For inline-level elements, this creates an override. For block-level, table-cell, table-caption, or inline-block elements this creates an override for inline-level descendants not within another block-level, table-cell, table-caption, or inline-block element.

Parent elements

textStyle

blk

Specifies a container into which you can insert other objects.

For more information, see **block** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then item *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

Blue

Specifies the blue RGB color value.

Content model

Content type is int.

Parent elements

bgColor , color , fgColor

bmrk

Defines a bookmark, or target point, of a link.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then label then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

body

Contains the contents of a page body.

Content model

style *(any number)* then item *(any number)* then extension *(optional)*

Parent elements

page

bold

Specifies that the font style is bold.

Content model

Content type is boolean.

Parent elements

fontStyle

bookmark

Specifies the bookmark that is the target point of the drill through.

Content model

Content type is string.

Parent elements

drill

bookmark

Specifies the `bmrk` element that is the target point of the entry in the table of contents.

Content model

Content type is string.

Parent elements

entry

border

Defines the border of a box.

Content model

top *(optional)* then left *(optional)* then right *(optional)* then bottom *(optional)*

Parent elements

boxStyle

bottom

Defines size of margin or padding on the bottom of a box.

Content model

val then units then extension *(optional)*

Parent elements

margin , padding

bottom

Defines color, line style and width of the bottom of a border.

Content model

color *(optional)* then lineStyle *(optional)* then width *(optional)*

Parent elements

border

boxStyle

Defines box styles, including height, width, margin, padding, and border.

Content model

height *(optional)* then width *(optional)* then margin *(optional)* then padding *(optional)* then border *(optional)* then extension *(optional)*

Parent elements

styleGroup

bType

Specifies the type of prompt button.

Content model

Content type is string.

The possible values of this element are restricted to the following.

FORWARD

A forward button.

BACK

A back button.

REPROMPT

A reprompt button.

FINISH

A finish button.

CANCEL

A cancel button.

Parent elements

p_btn

canBack

Specifies whether the **Back** button on a prompt page should be enabled.

Content model

Content type is boolean.

Parent elements

page

canExpand

Specifies whether the prompt tree control can be expanded

Content model

Content type is boolean.

Parent elements

p_tree

canFinish

Specifies whether the **Finish** button on a prompt page should be enabled.

Content model

Content type is boolean.

Parent elements

page

canNext

Specifies whether the **Next** button on a prompt page should be enabled.

Content model

Content type is boolean.

Parent elements

page

cascadeon

Specifies the pname of the parameter whose value is used to filter the values displayed in this control.

For more information, see **Cascade Source** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_srch , p_tree , p_value

cell

Defines a cell in a row of data.

Content model

id *(optional)* then ref *(optional)* then ctx *(optional)* then style *(any number)* then rspan *(optional)* then cspan *(optional)* then drillAction *(any number)* then item *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement , row

cgsData

Contains the data used to render the chart.

Content model

table *(any number)*

Parent elements

cgsWidget

cgsDataInfo

Reserved.

Content model

any *(any number)*

Parent elements

cgsWidget

cgsPropCanvas

Reserved.

Content model

`any` *(any number)*

Parent elements

details

cgsProperties

Reserved.

Content model

`any` *(any number)*

Parent elements

cgsWidget

cgsWidget

Defines chart information used to render the chart.

Content model

cgsData *(optional)* then cgsDataInfo *(optional)* then cgsProperties *(optional)* then extension *(optional)*

Parent elements

details

child

Defines the children of the parent tree node

Content model

use then disp *(optional)* then nullUse *(optional)* then nullDisp *(optional)*

Parent elements

node

choicesDeselectAllText

Specifies the text for the link that follows the results box that deselects all the items in the box. The default link text is Deselect All.

For more information, see **Results Deselect All Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_date , p_dtime , p_intrvl , p_srch , p_time , p_tree , p_txtbox , p_value

choicesSelectAllText

Specifies the text for the link that follows the results box that selects all the items in the box. The default link text is Select All.

For more information, see **Results Select All Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_date , p_dtime , p_intrvl , p_srch , p_time , p_txtbox , p_value

choicesText

Specifies the title that appears before the choices box when multiple selections are enabled. The default title text is Choices.

For more information, see **Choices Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_date , p_dtime , p_intrvl , p_srch , p_time , p_txtbox , p_value

choiceText

Specifies the title that appears above the choices box when only one selection is enabled. The default title text is Choice.

Content model

Content type is string.

Parent elements

p_srch , p_value

cht

Defines a chart.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then url then regions *(optional)* then details *(optional)* then alternateText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

clndr

Specifies the type of calendar to use.

Content model

Content type is string.

The possible values of this element are restricted to the following.

GREGORIAN

The standard international calendar.

IMPERIAL

The Japanese imperial calendar.

Parent elements

p_date , p_dtime

cmode

Specifies the type of clock user interface.

Content model

Content type is string.

The possible values of this element are restricted to the following.

STATIC

An input text boxes user interface.

LIVE

A clock control user interface.

Parent elements

p_time

cname

Specifies the title that appears before the list of choices in a value prompt. The default title text is the name of the level above the data items that are listed as choices; for example, Regions.

For more information, see **Header Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

extension , p_date , p_dtime , p_intrvl , p_srch , p_time , p_txtbox , p_value

color

Specifies the color of the border in RGB value format.

The `Red`, `Blue`, and `Green` child elements define the RGB values for the color.

Content model

Red then Green then Blue then Alpha *(optional)* then extension *(optional)*

Parent elements

bottom , left , right , top

colTitle

Defines the title that appears at the top of a column in a list.

Content model

id *(optional)* then ref *(optional)* then ctx *(optional)* then style *(any number)* then rspan *(optional)* then cspan *(optional)* then drillAction *(any number)* then item *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

group , grp , lst , table

column

Defines the top level group of column dimension values in a crosstab.

Content model

name then start then size then indent *(optional)* then nestedDimension *(any number)* then extension *(optional)*

Parent elements

ctab

connection

Specifies data source connection information.

Content model

name then searchPath then selected then signon *(any number)*

Parent elements

p_dsrc

contents

Contains the report element that the user can click to reach the hyperlink target.

Content model

style *(any number)* then item *(any number)* then extension *(optional)*

Parent elements

entry , hlink

coord

Specifies the coordinates for the set of points that define the area in the chart image.

Content model

x then y

Parent elements

area

corner

Defines a crosstab corner. A crosstab corner is the top left corner of the crosstab, above the row labels and to the left of the column labels.

Content model

id *(optional)* then ref *(optional)* then ctx *(optional)* then style *(any number)* then rspan *(optional)* then cspan *(optional)* then drillAction *(any number)* then item *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

ctab

cspan

Specifies the number of columns that the cell spans.

Content model

Content type is int.

Parent elements

cell , colTitle , corner , name , tcell , td , th

ctab

Defines a crosstab. Data in a crosstab appears in a grid, or in rows, columns, and cells.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then corner *(optional)* then column *(any number)* then row *(any number)* then table *(optional)* then summaryText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

ctx

Specifies context query data information for the cell or text item. This information is used for drill-up and drill-down operations.

The content of this element is automatically generated by Report Studio.

Content model

Content type is string.

Parent elements

area , cell , colTitle , corner , measure , member , name , txt

dataSourceName

Specifies a data source name.

Content model

Content type is string.

Parent elements

p_dsrc

dateUI

Specifies the type of date user interface.

Content model

Content type is string.

The possible values of this element are restricted to the following.

CALENDAR

A calendar control.

EDITBOX

An edit text box.

Parent elements

p_date , p_dtime

daysText

Specifies the title that appears above the days box in interval prompts. The default title text is Days.

For more information, see **Days Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_intrvl

depth

Specifies the level of the group.

Content model

Content type is int.

Parent elements

group , grp

deselectText

Specifies the text for the link that deselects the items when the selection is optional.The default link text is Deselect.

For more information, see **Deslect Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_srch , p_value

details

Contains the rendering details for a chart.

Content model

cgsWidget *(any number)* then cgsPropCanvas *(optional)* then extension *(optional)*

Parent elements

cht

di

Specifies the data item that corresponds to the page group or grouping level.

Content model

Content type is string.

Parent elements

group , grp , pageGroup

di

Specifies the data item referenced by the `refDataItem` element in the related report specification.

For more information about report specifications, see the *IBM Cognos Software Development Kit Developer Guide*.

Content model

Content type is string.

Parent elements

locationReference

direction

Indicates the direction for drill requests.

Content model

Content type is string.

The possible values of this element are restricted to the following.

UP

The drill direction is up.

DOWN

The drill direction is down.

Parent elements

drillAction

direction

Specifies the direction of the text in the layout.

Content model

Content type is string.

The possible values of this element are restricted to the following.

LEFT_TO_RIGHT

The text direction is from left to right.

RIGHT_TO_LEFT

The text direction is from right to left.

INHERIT

The text direction behavior is inherited from the parent container.

Parent elements

textStyle

disabled

This is `true` if the prompt satisfies the following conditions:

- The prompt is a cascading prompt.
- The source prompt appears on the same page as this prompt.
- The source prompt is required and has not yet been submitted.

In this case, the prompt should be rendered as being disabled (typically it would be greyed out) since it cannot be be submitted until the source prompt has been submitted.

See "Collecting cascading prompts from a single prompt page" on page 51 for an example of the use of this element.

Content model

Content type is boolean.

Parent elements

p_srch , p_tree , p_value

disp

Specifies the values rendered to the report user when the prompt is used. These values can be different than the ones that are actually used by the report.

For more information, see **Display Value** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

child , max , min , sval

display

Specifies how an object that references the `styleGroup` should be displayed.

none Do not render the object.

inline Render the object inline.

block Render the object inside a container.

Content model

Content type is string.

Parent elements

styleGroup

displayValue

Specifies the value that is displayed for this drill-through parameter.

Content model

Content type is string.

Parent elements

parm

div

Defines a division or section in an `item`.

Content model

style *(any number)* then item *(any number)* then extension *(optional)*

Parent elements

item

document

Defines the root element for a resource, for example, a report.

Content model

secondaryOperations *(any number)* then schemaSubversion *(optional)* then versionBase *(optional)* then id *(optional)* then style *(any number)* then locationReference *(any number)* then pages *(any number)* then drillDefinitions *(optional)* then styleGroup *(any number)* then extension *(optional)*

drill

Defines a drill-through instance.

The drill-through definition is specified on the corresponding `drill` element, defined as a child of the `drillDefinitions` element. The drill-through definition specifies information like the parameters, output format, and prompt information.

The `drillRef` child element references the `drill` element that contains the drill-through definition.

Content model

drillRef then parm *(any number)* then bookmark *(optional)* then URLParameters *(optional)*

Parent elements

drills

drill

Contains a drill-through definition.

This drill-through definition can be re-used throughout the layout data document. A drill-through instance is defined using the `drill` element. The `drillRef` child element contains the name referenced by each drill-through instance.

Content model

drillRef then label then showInNewWindow then sendFilterContext then prompt then outputFormat then method then targetPath *(optional)* then parameters *(optional)* then modelPaths *(optional)* then url *(optional)*

Parent elements

drillDefinitions

drillAction

Contains the possible drill actions on an item.

Content model

direction then actionURL *(optional)*

Parent elements

area , cell , colTitle , corner , img , name , txt

drillDefinitions

Contains the list of drill-through definitions used throughout the layout data document.

These definitions are referenced by the `drill` elements defined in other parts of the document, using the `drillRef` child element.

Content model

drill *(any number)*

Parent elements

document , filterResultSet

drillRef

Specifies the `drill` element that defines the drill-through definition.

Content model

Content type is string.

Parent elements

drill , drill

drills

Contains the drill-through instances defined on the area, image or text item.

Content model

drill *(one or more)*

Parent elements

area , img , txt

dv

Specifies the value of the data item that determines the grouping level for the page or list.

Content model

Content type is string.

Parent elements

group , grp , pageGroup

entry

Defines an entry in the table of contents.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then bookmark *(optional)* then contents *(optional)* then extension *(optional)*

Parent elements

toc

exclPatrn

If this element exists, it contains the format used to display `val`, expressed as a Microsoft Excel format. For example, `\#\,\#\#0`.

Content model

Content type is string.

Parent elements

txt

extension

Reserved.

Content model

id *(optional)* then extension *(optional)*

Parent elements

div , span , table

extension

Reserved.

Content model

cname *(optional)* then extension *(optional)*

Parent elements

blk

extension

Placeholder for future extensions.

Content model

`any` *(any number)*

Parent elements

ancestors , bgColor , bgImageProperties , bmrk , body , bottom , boxStyle , cell , cgsWidget , cht , color , colTitle , column , contents , corner , ctab , details , document , entry , extension , extension , fgColor , filterResult , filterResultSet , font , fontStyle , footer , group , grp , header , height , hlink , html , img , indent , item , item , kashidaSpace , lcr , left , listItem , locationReference , lst , name , nestedDimension , node , p_btn , p_date , p_dsrc , p_dtime , p_intrvl , p_srch , p_time , p_tree , p_txtbox , p_value , page , pageGroup , pages , reportElement , rept , reptbl , right , row , row , rtList , rtxt , secondaryOperations , selChoices , selOptions , size , sngl , styleGroup , table , tbl , tcell , td , th , toc , top , tr , txt , widget , width

family

Specifies the font family.

Content model

Content type is string.

Parent elements

font

faultcode

Specifies the fault code that generated the data source prompt.

Content model

Content type is string.

Parent elements

p_dsrc

faultstring

Specifies the error (such as incorrect logon) that caused the data source prompt to be generated.

Content model

Content type is string.

Parent elements

p_dsrc

fdate

Specifies the earliest date to render in the control, and the earliest date that can be selected.

For more information, see **First Date** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is date.

Parent elements

p_date

fdate

Specifies the earliest date and time to render in the control, and the earliest date and time that can be selected.

Content model

Content type is dateTime.

Parent elements

p_dtime

fgColor

Defines the foreground color in RGB format.

Content model

Red then Green then Blue then Alpha *(optional)* then extension *(optional)*

Parent elements

styleGroup

filterResult

Contains the layout data specification for a filtered portion of the original resource.

Content model

filterType then filterValue then reportElement *(any number)* then extension *(optional)*

Parent elements

filterResultSet

filterResultSet

Contains one or more layout data specifications for filtered portions of the original resource.

Content model

secondaryOperations *(any number)* then versionBase *(optional)* then locationReference *(any number)* then filterResult *(any number)* then drillDefinitions *(optional)* then styleGroup *(any number)* then extension *(optional)*

filterType

Specifies the type of filter used to filter the resource.

Content model

Content type is string.

The possible values of this element are restricted to the following.

OBJECT_ID

The resource is filtered on a report part or parts.

CONTEXT_SPEC

Reserved.

XPATH

The resource is filtered by an XPath expression.

Parent elements

filterResult

filterValue

Specifies the value used to filter the resource.

Content model

Content type is string.

Parent elements

filterResult

fmtLoc

Specifies the 2-character locale code used for formatting. For example, `CA` for Canada.

Content model

Content type is string.

Parent elements

txt

fmtPatrn

Specifies the International Component for Unicode (ICU) format used to display `val`, for example`#,##0`.

Content model

Content type is string.

Parent elements

txt

fmtScale

Defines the scale value used to render the value that appears in the output.

Content model

Content type is int.

Parent elements

txt

fmtVal

Specifies the formatted value displayed for `val`.

Content model

Content type is string.

Parent elements

txt

font

Defines the font family, size, and style.

Content model

family *(optional)* then size *(optional)* then fontStyle *(optional)* then extension *(optional)*

Parent elements

styleGroup

fontStyle

Specifies the font style.

Content model

bold *(optional)* then italics *(optional)* then underline *(optional)* then overline *(optional)* then strikethrough *(optional)* then extension *(optional)*

Parent elements

font

footer

Defines the page footer.

Content model

style *(any number)* then item *(any number)* then extension *(optional)*

Parent elements

group , grp , lst , page , table

footer

Defines the list group footer.

Content model

row *(one or more)*

fromText

Specifies the label that appears beside the beginning of a range. The default label text is From.

For more information, see **From Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_date , p_dtime , p_intrvl , p_time , p_txtbox , p_value

Green

Specifies the green RGB color value.

Content model

Content type is int.

Parent elements

bgColor , color , fgColor

group

Defines the grouping structure for a list.

Content model

di *(optional)* then dv *(optional)* then header *(optional)* then colTitle *(any number)* then (row *(any number)* or grp *(any number)*) then footer *(optional)* then depth then extension *(optional)*

Parent elements

lst , table

grp

Defines a level of grouping in a list.

Content model

di *(optional)* then dv *(optional)* then header *(optional)* then colTitle *(any number)* then (row *(any number)* or grp *(any number)*) then footer *(optional)* then depth then extension *(optional)*

Parent elements

group , grp

hAlign

Specifies the horizontal alignment for the style.

Content model

Content type is string.

The possible values of this element are restricted to the following.

LEFT

The horizontal alignment is left.

CENTER

The horizontal alignment is center.

RIGHT

The horizontal alignment is right.

JUSTIFY

The horizontal alignment is justify.

Parent elements

styleGroup

hdr

Specifies whether the cell is a table header. Use to make reports accessible for people who use screen readers. When set to Yes, screen readers and speech browsers programmatically create relationships between the table header and table cells.

For more information, see **Table Header** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is boolean.

Parent elements

tcell

header

Defines the page header.

Content model

style *(any number)* then item *(any number)* then extension *(optional)*

Parent elements

page

header

Defines the list group header.

Content model

row *(one or more)*

Parent elements

group , grp , lst , table

headerAfterOverall

Specifies whether the list page header is to be rendered after the overall header.

For more information, see **Display After Overall Header** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is boolean.

Parent elements

lst , table

height

Specifies the height of the box.

Content model

val then units then extension *(optional)*

Parent elements

boxStyle

hidden

Specifies whether this item should be hidden.

Content model

Content type is boolean.

Parent elements

styleGroup

highestValueText

Specifies the label that appears beside the highest value option when ranges are enabled. The default label text is Latest date, Latest time, or Highest interval.

For more information, see **Highest Value Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_date , p_dtime , p_intrvl , p_time , p_txtbox , p_value

hlink

Defines a hyperlink.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then contents then target then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

horizontalLayout

Specifies whether a table is laid out horizontally or vertically.

When `true`, the repeater table cells are laid out from left to right, then top to bottom. When `false`, the repeater table cells are laid out from top to bottom, then left to right.

Content model

Content type is boolean.

Parent elements

reptbl

horizontalSize

Specifies the number of repeater table element cells in each row.

Content model

Content type is int.

Parent elements

reptbl

hoursText

Specifies the title that appears above the hours box in interval prompts. The default title text is Hrs.

For more information, see **Hours Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_intrvl

html

Contains custom HTML from the source resource.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then val then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

htxt

Specifies whether this prompt should be rendered as a masked text field

Content model

Content type is boolean.

Parent elements

p_txtbox

id

Specifies a unique element identifier.

The content of this element corresponds to the `name` element in the source report specification.

Content model

Content type is string.

Parent elements

blk , bmrk , cell , cht , colTitle , corner , ctab , document , entry , extension , hlink , html , img , lcr , lst , name , p_btn , p_date , p_dtime , p_intrvl , p_srch , p_time , p_tree , p_txtbox , p_value , page , rept , reptbl , rtxt , sngl , table , tbl , tcell , toc , txt , widget

img

Defines an image.

Content model

id *(optional)* then ref *(optional)* then style *(optional)* then drills *(optional)* then drillAction *(any number)* then url then isCMMMap *(optional)* then alternateText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , item , reportElement

indent

Specifies the indentation of a dimension in a crosstab.

Content model

val then units then extension *(optional)*

Parent elements

column , nestedDimension , row

insertText

Specifies the label that appears on the button that is used to add items to the selected items box in all multiple selection prompts. The default label text is Insert.

For more information, see **Insert Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_date , p_dtime , p_intrvl , p_srch , p_time , p_txtbox , p_value

isCMMMap

Specifies that the image represents an IBM Cognos Metrics Manager map.

Content model

Content type is boolean.

Parent elements

img

italics

Specifies the font style is italic.

Content model

Content type is boolean.

Parent elements

fontStyle

item

Contains report parts.

Content model

txt or lst or cell or ctab or cht or img or hlink or html or rtxt or rept or reptbl or bmrk or toc or lcr or tbl or blk or sngl or widget or p_txtbox or p_value or p_date or p_time or p_dtime or p_intrvl or p_dsrc or p_srch or p_tree or p_btn or extension *(optional)*

Parent elements

blk , body , cell , colTitle , contents , corner , footer , header , lcr , name , rept , sngl , tcell

item

Contains rich text items.

Content model

txt or div or span or rtList or table or img or extension *(optional)*

Parent elements

div , listItem , rtxt , span , td , th

justification

Specifies text justification.

Content model

Content type is string.

The possible values of this element are restricted to the following.

DISTRIBUTE

The justification is distribute.

DISTRIBUTE LINES

Same as distribute, but also justifies last line in a paragraph.

INTERCLUSTER

The justification is intercluster.

INTERIDEOGRAPH

The justification is interideograph.

INTERWORD

The justification is interword.

KASHIDA

The justification is kashida.

NEWSPAPER

The justification is newspaper.

Parent elements

textStyle

kashidaSpace

Defines kashida size.

Content model

val then units then extension *(optional)*

Parent elements

textStyle

keywordsText

Specifies the title that appears above the keyword search box in select & search prompts. The default title text is Keywords.

For more information, see **Keywords Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_srch

label

Specifies the label or tooltip for the area, bookmark, drill, or measure.

Content model

Content type is string.

Parent elements

area , bmrk , drill , measure

lcr

Defines a layout component reference from the source report specification.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then reportPath *(optional)* then item *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

ldate

Specifies the latest date rendered in the control, and the last date that can be selected.

For more information, see **Last Date** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is date.

Parent elements

p_date

ldate

Specifies the latest date and time rendered in the control, and the last date and time that can be selected.

Content model

Content type is dateTime.

Parent elements

p_dtime

left

Defines size of margin or padding for the left side of the box.

Content model

val then units then extension *(optional)*

Parent elements

margin , padding

left

Defines color, line style, and width of left border.

Content model

color *(optional)* then lineStyle *(optional)* then width *(optional)*

Parent elements

border

lineStyle

Specifies the line style for the border.

Content model

Content type is string.

The possible values of this element are restricted to the following.

NONE

There is no line.

SOLID

The line style is solid.

DOUBLE

The line style is double.

DOTTED

The line style is dotted.

DASHED

The line style is dashed.

GROOVE

The line style is groove.

RIDGE

The line style is ridge.

INSET

The line style is inset.

OUTSET

The line style is outset.

Parent elements

bottom , left , right , top

listItem

Defines an item in a list of rich text items.

Content model

item *(any number)* then extension *(optional)*

Parent elements

rtList

loc

Specifies the location of the report element in the source specification.

Content model

Content type is string.

Parent elements

locationReference

locale

Specifies the locale of the model of the drill target.

Content model

Content type is string.

Parent elements

modelPaths

locationReference

Specifies the location of an element in the source report specification.

Content model

ref then di *(optional)* then loc then extension *(optional)*

Parent elements

document , filterResultSet

logonFailureCount

Specifies the number of failed logon attempts.

Content model

Content type is int.

Parent elements

p_dsrc

lowestValueText

Specifies the label that appears beside the lowest value option when ranges are enabled. The default label text is Earliest date, Earliest time, or Lowest interval.

For more information, see **Lowest Value Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_date , p_dtime , p_intrvl , p_time , p_txtbox , p_value

lst

Defines a list.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then colTitle *(any number)* then header *(optional)* then headerAfterOverall *(optional)* then group *(optional)* then footer *(optional)* then summaryText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

margin

Defines the size of the box margin.

Content model

top *(optional)* then left *(optional)* then right *(optional)* then bottom *(optional)*

Parent elements

boxStyle

max

Specifies the maximum possible value for the prompt range. If not specified, then the maximum value is unbounded.

Content model

use then disp *(optional)* then nullUse *(optional)* then nullDisp *(optional)*

Parent elements

rval

maximumValueCount

Specifies the maximum number of nodes that can be displayed in the tree prompt user interface.

Content model

Content type is int.

Parent elements

p_tree

measure

Specifies the chart measure.

Content model

ctx *(optional)* then label *(optional)*

Parent elements

area

member

Specifies the member used for the drill-up or drill-down operation.

Content model

ctx *(optional)*

Parent elements

area

memberDisplayCountDefault

Specifies the maximum number of member nodes that should be displayed in the tree prompt user interface.

Content model

Content type is int.

Parent elements

p_tree

memberDisplayCountLimit

Specifies the maximum number of member nodes that can be displayed in the tree prompt user interface.

Content model

Content type is int.

Parent elements

p_tree

method

Specifies the drill-through method.

Content model

Content type is string.

Parent elements

drill

milisecs

Specifies whether to display the milliseconds. The default value is `true`.

Content model

Content type is boolean.

Parent elements

p_intrvl , p_time

millisecondsText

Specifies the title that appears above the milliseconds box. The default title text is ms.

For more information, see **Milliseconds Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_intrvl

min

Specifies the minimum possible value for the prompt range. If not specified, then the minimum value is unbounded.

Content model

use then disp *(optional)* then nullUse *(optional)* then nullDisp *(optional)*

Parent elements

rval

minutesText

Specifies the title that appears above the minutes box in interval prompts. The default title text is Mins.

For more information, see **Minutes Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_intrvl

mline

Specifies whether this prompt should be displayed as a multi-line text box

Content model

Content type is boolean.

Parent elements

p_txtbox

modelPaths

Contains the search path and locale of the drill-through target.

Content model

objectPath *(any number)* then locale

Parent elements

drill

moreData

Specifies that there is more data that can be retrieved for the tree.

Content model

Content type is boolean.

Parent elements

p_tree

mtchall

Specifies whether the search results returned match all of the search words entered. The default value is `false`.

Value	Description
`true`	The results returned match all of the search words entered. When mtchany is also `true`, then the search can contain all the keywords in any order.
false	The results returned match any of the search words entered.

Content model

Content type is boolean.

Parent elements

p_srch

mtchany

Specifies whether the search results contain or start with the keywords. The default value is `false`.

Value	Description
`true`	The results returned contain the key words.
`false`	The search results returned start with the keywords.

Content model

Content type is boolean.

Parent elements

p_srch

multi

Specifies whether the control allows the selection of multiple values.

For more information, see **Multi-Select** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is boolean.

Parent elements

p_date , p_dtime , p_intrvl , p_srch , p_time , p_tree , p_txtbox , p_value

mun

Specifies the member unique name (MUN) of the parameter value.

See the topic about using drill-through access in the *IBM Cognos Business Intelligence Report Studio User Guide* for more information on member unique names.

Content model

Content type is string.

Parent elements

parm

name

Specifies the name in the name/value pair of the parameter.

Content model

Content type is string.

Parent elements

connection , parm , signon

name

Specifies the label used to identify the row, column, or nested dimension

Content model

id *(optional)* then ref *(optional)* then ctx *(optional)* then style *(any number)* then rspan *(optional)* then cspan *(optional)* then drillAction *(any number)* then item *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

column , nestedDimension , row

name

Specifies the name of the style.

Content model

Content type is string.

Parent elements

styleGroup

name

Specifies the name of the parameter.

This element is referenced by the `name` element that is a child of the `parm` element on the drill-through instance.

Content model

Content type is string.

Parent elements

parameter

nestedDimension

Defines a crosstab dimension.

Content model

name then start then size then indent *(optional)* then nestedDimension *(any number)* then extension *(optional)*

Parent elements

column , nestedDimension , row

noadorn

Specifies whether to hide the asterisk (*) on required prompts and arrow (->) on type-in prompts that are in an error state.

Content model

Content type is boolean.

Parent elements

p_date , p_dtime , p_intrvl , p_srch , p_time , p_tree , p_txtbox , p_value

nocase

Specifies whether the search is case insensitive. The default value is `true`.

Content model

Content type is boolean.

Parent elements

p_srch

node

Represents a parent node for a tree prompt.

Parent Tree node

Content model

child *(any number)* then extension *(optional)*

Parent elements

p_tree

noresults

Specifies whether the search returned no results

Content model

Content type is boolean.

Parent elements

p_srch

nullDisp

Specifies whether a null display value should be used.

Content model

Content type is boolean.

Parent elements

child , max , min , sval

nullUse

Specifies whether a null use value should be used.

Content model

Content type is boolean.

Parent elements

child , max , min , sval

num

Specifies whether the value supplied must be numeric

Content model

Content type is boolean.

Parent elements

p_txtbox

objectPath

Specifies the search path for the model of the drill-through target.

Content model

Content type is string.

Parent elements

modelPaths

ordered

Specifies whether this is an ordered list.

Content model

Content type is boolean.

Parent elements

rtList

outputFormat

Specifies the output format of the target resource.

Content model

Content type is string.

Parent elements

drill

overline

Specifies the font style is overline.

Content model

Content type is boolean.

Parent elements

fontStyle

p_btn

Specifies a prompt button control.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then bType *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

p_date

Specifies a **Date Prompt** control.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then pname then req *(optional)* then noadorn *(optional)* then range *(optional)* then multi *(optional)* then dateUI *(optional)* then clndr *(optional)* then fdate *(optional)* then ldate *(optional)* then cname *(optional)* then selChoices *(optional)* then choicesText *(optional)* then fromText *(optional)* then toText *(optional)* then lowestValueText *(optional)* then

highestValueText *(optional)* then choicesSelectAllText *(optional)* then choicesDeselectAllText *(optional)* then removeText *(optional)* then insertText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

p_dsrc

Specifies a **Data Source Prompt** control.

Content model

pname then faultcode *(optional)* then faultstring *(optional)* then dataSourceName *(optional)* then logonFailureCount *(optional)* then persistPrompt *(optional)* then connection *(one or more)* then signon *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

p_dtime

Specifies a **Date & Time Prompt** control.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then pname then req *(optional)* then noadorn *(optional)* then range *(optional)* then multi *(optional)* then dateUI *(optional)* then clndr *(optional)* then fdate *(optional)* then ldate *(optional)* then cname *(optional)* then selChoices *(optional)* then choicesText *(optional)* then fromText *(optional)* then toText *(optional)* then lowestValueText *(optional)* then highestValueText *(optional)* then choicesSelectAllText *(optional)* then choicesDeselectAllText *(optional)* then removeText *(optional)* then insertText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

p_intrvl

Specifies an **Interval Prompt** control.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then pname then req *(optional)* then noadorn *(optional)* then range *(optional)* then multi *(optional)* then secnds *(optional)* then milisecs *(optional)* then cname *(optional)* then selChoices *(optional)* then choicesText *(optional)* then fromText *(optional)* then toText *(optional)* then lowestValueText *(optional)* then highestValueText *(optional)* then choicesSelectAllText *(optional)* then choicesDeselectAllText *(optional)* then removeText *(optional)* then insertText *(optional)* then daysText *(optional)* then

hoursText *(optional)* then minutesText *(optional)* then secondsText *(optional)* then millisecondsText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

p_srch

Specifies a **Select & Search Prompt** control.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then pname then req *(optional)* then noadorn *(optional)* then range *(optional)* then multi *(optional)* then cascadeon *(optional)* then prepopulate *(optional)* then rows *(optional)* then disabled *(optional)* then noresults *(optional)* then mtchany *(optional)* then mtchall *(optional)* then showopt *(optional)* then srchval *(optional)* then nocase *(optional)* then cname *(optional)* then selOptions *(optional)* then selChoices *(optional)* then keywordsText *(optional)* then searchInstructionsText *(optional)* then choicesText *(optional)* then choiceText *(optional)* then resultsText *(optional)* then choicesSelectAllText *(optional)* then choicesDeselectAllText *(optional)* then resultsSelectAllText *(optional)* then resultsDeselectAllText *(optional)* then deselectText *(optional)* then removeText *(optional)* then insertText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

p_time

Specifies a **Time Prompt** control.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then pname then req *(optional)* then noadorn *(optional)* then range *(optional)* then multi *(optional)* then timeUI *(optional)* then cmode *(optional)* then secnds *(optional)* then milisecs *(optional)* then cname *(optional)* then selChoices *(optional)* then choicesText *(optional)* then fromText *(optional)* then toText *(optional)* then lowestValueText *(optional)* then highestValueText *(optional)* then choicesSelectAllText *(optional)* then choicesDeselectAllText *(optional)* then removeText *(optional)* then insertText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

p_tree

Specifies a **Tree Prompt** control.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then pname then req *(optional)* then noadorn *(optional)* then multi *(optional)* then cascadeon *(optional)* then

prepopulate *(optional)* then rows *(optional)* then disabled *(optional)* then treeUI *(optional)* then prepopulatelevels *(optional)* then canExpand *(optional)* then node *(optional)* then moreData *(optional)* then memberDisplayCountDefault *(optional)* then memberDisplayCountLimit *(optional)* then maximumValueCount *(optional)* then skipValueCount *(optional)* then selOptions *(optional)* then selChoices *(optional)* then ancestors *(optional)* then choicesDeselectAllText *(optional)* then resultsDeselectAllText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

p_txtbox

Specifies a **Text Box Prompt** control.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then pname then req *(optional)* then noadorn *(optional)* then range *(optional)* then multi *(optional)* then num *(optional)* then mline *(optional)* then htxt *(optional)* then thSep *(optional)* then cname *(optional)* then selChoices *(optional)* then choicesText *(optional)* then fromText *(optional)* then toText *(optional)* then lowestValueText *(optional)* then highestValueText *(optional)* then choicesSelectAllText *(optional)* then choicesDeselectAllText *(optional)* then removeText *(optional)* then insertText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

p_value

Specifies a **Value Prompt** control.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then pname *(optional)* then req *(optional)* then noadorn *(optional)* then range *(optional)* then multi *(optional)* then cascadeon *(optional)* then prepopulate *(optional)* then rows *(optional)* then disabled *(optional)* then selectUI *(optional)* then auto *(optional)* then cname *(optional)* then autocascade *(optional)* then selChoices *(optional)* then selOptions *(optional)* then choicesText *(optional)* then choiceText *(optional)* then resultsText *(optional)* then fromText *(optional)* then toText *(optional)* then lowestValueText *(optional)* then highestValueText *(optional)* then choicesSelectAllText *(optional)* then choicesDeselectAllText *(optional)* then resultsSelectAllText *(optional)* then resultsDeselectAllText *(optional)* then deselectText *(optional)* then removeText *(optional)* then insertText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

padding

Defines box padding.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

top *(optional)* then left *(optional)* then right *(optional)* then bottom *(optional)*

Parent elements

boxStyle

page

Defines a page of the resource.

Content model

canFinish *(optional)* then canNext *(optional)* then canBack *(optional)* then id *(optional)* then ref *(optional)* then style *(any number)* then header *(optional)* then body *(optional)* then footer *(optional)* then extension *(optional)*

Parent elements

pages

pageGroup

Defines a grouping level for groups of pages. Page breaks for the group can be defined, based on a data item.

Content model

di *(optional)* then dv *(optional)* then pages *(one or more)* then extension *(optional)*

Parent elements

pages

pages

Defines a container element for `page` and `pageGroup` elements.

Content model

page or pageGroup or extension *(optional)*

Parent elements

document , pageGroup

parameter

Defines a parameter on the drill-through definition.

The `name` child element identifies the parameter. This name is referenced by a drill-through instance through the `parm` element.

Content model

name then type

Parent elements

parameters

parameters

Contains the parameters used in the drill-through definition.

Content model

parameter *(one or more)*

Parent elements

drill

parm

Specifies the parameter used for the drill through operation.

The `name` child element references a `parameter` defined in the drill through definition.

Content model

name then value then displayValue *(optional)* then mun *(optional)*

Parent elements

drill

persistPrompt

Specifies whether data source credentials should be saved in Content Manager.

Content model

Content type is string.

Parent elements

p_dsrc

pname

Specifies the parameter that is satisfied by values chosen in the prompt control.

For more information, see **Parameter** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_date , p_dsrc , p_dtime , p_intrvl , p_srch , p_time , p_tree , p_txtbox , p_value

position

Specifies the position of the background image. Valid values of the string are the same as for the Cascading Style Sheets position property.

Content model

Content type is string.

Parent elements

bgImageProperties

prepopulate

Specifies whether to pre-populate the control with values, but only if the parent of this prompt control is optional. This only applies to prompt controls that have a parent in a cascade.

For more information, see **Pre-populate** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is boolean.

Parent elements

p_srch , p_tree , p_value

prepopulatelevels

Specifies the number of levels to pre-populate the prompt with. The default value is 1, which will pre-populate the prompt with only the root members.

For more information, see **Pre-populate Levels** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is int.

Parent elements

p_tree

prompt

Specifies whether the report is automatically prompted when the drill-through definition is executed.

Content model

Content type is string.

Parent elements

drill

range

Specifies whether this control accepts ranges.

For more information, see **Range** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is boolean.

Parent elements

p_date , p_dtime , p_intrvl , p_srch , p_time , p_txtbox , p_value

Red

Specifies the red RGB color value.

Content model

Content type is int.

Parent elements

bgColor , color , fgColor

ref

Specifies an identifier for the location reference.

Content model

Content type is string.

Parent elements

blk , bmrk , cell , cht , colTitle , corner , ctab , entry , hlink , html , img , lcr , locationReference , lst , name , p_btn , p_date , p_dtime , p_intrvl , p_srch , p_time

, p_tree , p_txtbox , p_value , page , rept , reptbl , rtxt , sngl , table , tbl , tcell , toc , txt , widget

regions

Contains the regions defined in the chart. Each region is defined by an `area` child element.

Content model

area *(any number)*

Parent elements

cht

removeText

Specifies the label that appears on the button that is used to remove items from the selected items box in all multiple selection prompts. The default label text is Remove.

For more information, see **Remove Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_date , p_dtime , p_intrvl , p_srch , p_time , p_txtbox , p_value

repeat

Specifies how the background image should be repeated

Content model

Content type is string.

The possible values of this element are restricted to the following.

REPEAT

Repeat image in both horizontal and vertical.

REPEAT-X

Repeat image horizontally only.

REPEAT-Y

Repeat image vertically only.

NO REPEAT

Do not repeat image.

INHERIT

The repeating behavior is inherited from the parent container.

Parent elements

bgImageProperties

reportElement

Contains the report elements returned from the filtered report.

Content model

txt or lst or cell or ctab or cht or img or hlink or html or rtxt or rept or reptbl or bmrk or toc or lcr or tbl or blk or sngl or widget or p_txtbox or p_value or p_date or p_time or p_dtime or p_intrvl or p_dsrc or p_srch or p_tree or p_btn or extension *(optional)*

Parent elements

filterResult

reportPath

Specifies the search path to the report that contains the referenced component.

Content model

Content type is string.

Parent elements

lcr

rept

Defines a repeater. A repeater renders query data with no layout structure.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then item *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

reptbl

Defines a repeater table. This element renders query data in a table. Data in a repeater table can be grouped.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then horizontalLayout *(optional)* then horizontalSize *(optional)* then verticalSize *(optional)* then row *(any number)* then summaryText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

req

Specifies whether the prompt is required or optional. If `true`, the prompt must have a value entered before the report can be run.

For more information, see **Required** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is boolean.

Parent elements

p_date , p_dtime , p_intrvl , p_srch , p_time , p_tree , p_txtbox , p_value

resultsDeselectAllText

Specifies the text for the link that follows the results box that deselects all the items in the box. The default link text is **Deselect All**.

For more information, see **Results Deselect All Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_srch , p_tree , p_value

resultsSelectAllText

Specifies the text for the link that follows the results box that selects all the items in the box. The default link text is Select All.

For more information, see **Results Select All Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_srch , p_value

resultsText

Specifies the title that precedes the results box in select & search prompts. The default title text is Results.

For more information, see **Results Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_srch , p_value

right

Specifies the size of the right margin or box padding.

Content model

val then units then extension *(optional)*

Parent elements

margin , padding

right

Defines the right side of the border.

Content model

color *(optional)* then lineStyle *(optional)* then width *(optional)*

Parent elements

border

row

Defines the top level group of row dimension values in a crosstab.

Content model

name then start then size then indent *(optional)* then nestedDimension *(any number)* then extension *(optional)*

Parent elements

ctab

row

Defines a row in a table, footer, header, group, or repeater table.

Attributes

Adding Other Attributes

`anyAttribute` indicates that any attribute within the specified namespace(s) is permitted. Applicable namespace(s) and processing considerations are specified by the namespace and processContents parameters, respectively.

Content model

cell *(any number)* then extension *(optional)*

Parent elements

footer , group , grp , header , reptbl , table

rows

Specifies the maximum number of rows to show at one time. (See **Rows** in the Report Studio *User Guide*.)

Content model

Content type is int.

Parent elements

p_srch , p_tree , p_value

rspan

Specifies the number of rows that the cell spans.

Content model

Content type is int.

Parent elements

cell , colTitle , corner , name , tcell , td , th

rtList

Defines a list of rich text items.

Content model

style *(any number)* then ordered then listItem *(any number)* then extension *(optional)*

Parent elements

item

rtxt

Defines a rich text item.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then item *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

rval

Specifies a prompt answer whose value lies in a range of values.

Content model

min *(optional)* then max *(optional)*

Parent elements

ancestors , selChoices , selOptions

schemaSubversion

Reserved.

Content model

Content type is string.

Parent elements

document

searchInstructionsText

Specifies the instructions that appear before the keyword search box in select & search prompts. The default text is as follows: Type one or more keywords separated by spaces.

For more information, see **Search Instructions Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_srch

searchPath

Specifies the search path to a signon or data source entry in a data source prompt.

Content model

Content type is string.

Parent elements

connection , signon

secnds

Specifies whether to display the seconds.

Content model

Content type is boolean.

Parent elements

p_intrvl , p_time

secondaryOperations

Specifies a list of available secondary operations.

Content model

value then extension *(optional)*

Parent elements

document , filterResultSet

secondsText

Specifies the title that appears before the seconds box in interval prompts. The default title text is s.

For more information, see **Seconds Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_intrvl

selChoices

Specifies the collection of default selections for a prompt control.

For more information, see **Default Selections** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

sval *(any number)* or rval *(any number)* or extension *(optional)*

Parent elements

p_date , p_dtime , p_intrvl , p_srch , p_time , p_tree , p_txtbox , p_value

selected

Specifies whether this connection or signon should be displayed as selected in the user interface for a data source prompt

Content model

Content type is boolean.

Parent elements

connection , signon

selectUI

Specifies which interface the prompt control renders.

For more information, see **Select UI** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

The possible values of this element are restricted to the following.

DROP_DOWN

A drop down box.

LIST_BOX

A list box.

RADIO

A radio button group.

CHECK_BOX

A check box.

Parent elements

p_value

selOptions

Specifies the collection of all possible selections for a prompt control.

Content model

sval *(any number)* or rval *(any number)* or extension *(optional)*

Parent elements

p_srch , p_tree , p_value

sendFilterContext

Specifies whether to send the dynamic filter context.

Content model

Content type is boolean.

Parent elements

drill

showInNewWindow

Specifies whether the target resource appears in a new window when the drill-through definition Is executed.

Content model

Content type is boolean.

Parent elements

drill

showopt

Specifies whether the options to control the search (`mtchall`, `mtchany`, and `nocase`) should be expanded and shown to the user.

Content model

Content type is boolean.

Parent elements

p_srch

signon

Specifies a data source signon.

Content model

name *(optional)* then searchPath *(optional)* then selected

Parent elements

connection , p_dsrc

size

Specifies the number of rows or columns that the dimension spans.

Content model

Content type is int.

Parent elements

column , nestedDimension , row

size

Specifies the size of the font.

Content model

val then units then extension *(optional)*

Parent elements

font

skipValueCount

Specifies how many rows of data to skip when fetching data to populate a tree prompt.

Content model

Content type is int.

Parent elements

p_tree

sngl

Represents a singleton in the report.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then item *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

span

Defines a span container.

Content model

style *(any number)* then item *(any number)* then extension *(optional)*

Parent elements

item

srchval

Contains the keywords that are passed to the search.

Content model

Content type is string.

Parent elements

p_srch

start

Specifies the first row or column of the dimension in the data table.

Content model

Content type is int.

Parent elements

column , nestedDimension , row

strictLineBreaking

Specifies the text style is strict line breaking. This style is used for Japanese text.

Content model

Content type is boolean.

Parent elements

textStyle

strikethrough

Specifies the text style is strikethrough.

Content model

Content type is boolean.

Parent elements

fontStyle

style

Specifies the style to apply.

The string specified in this element references the `name` element of the style defined in the parent `styleGroup` element.

Content model

Content type is string.

Parent elements

blk , bmrk , body , cell , cht , colTitle , contents , corner , ctab , div , document , entry , footer , header , hlink , html , img , lcr , lst , name , p_btn , p_date , p_dtime , p_intrvl , p_srch , p_time , p_tree , p_txtbox , p_value , page , rept , reptbl , rtList , rtxt , sngl , span , table , table , tbl , tcell , td , th , toc , tr , trow , txt , widget

styleGroup

Defines a style used in the layout data document.

Content model

name then font *(optional)* then textStyle *(optional)* then boxStyle *(optional)* then fgColor *(optional)* then bgColor *(optional)* then bgImageURL *(optional)* then bgImageProperties *(optional)* then hAlign *(optional)* then vAlign *(optional)* then hidden *(optional)* then display *(optional)* then extension *(optional)*

Parent elements

document , filterResultSet

summaryText

Specifies summary text for table-like objects. Use to make your reports accessible for people who use screen readers. The summary text is never displayed in visual Web browsers. Summary text is used only for screen readers and speech browsers.

For more information, see **Summary Text** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

ctab , lst , reptbl , table , tbl

sval

Specifies a prompt answer whose value is a single value.

Content model

use then disp *(optional)* then nullUse *(optional)* then nullDisp *(optional)*

Parent elements

ancestors , selChoices , selOptions

table

Specifies the data in a crosstab.

Content model

row *(any number)*

Parent elements

ctab

table

Represents an HTML table in a rich text item.

Content model

style *(any number)* then tr *(any number)* then extension *(optional)*

Parent elements

item

table

Specifies the data in a chart.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then colTitle *(any number)* then header *(optional)* then headerAfterOverall *(optional)* then group *(optional)* then footer *(optional)* then summaryText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

cgsData

target

Specifies the URL for the hyperlink target.

Content model

Content type is string.

Parent elements

hlink

targetPath

Specifies the search path to the target resource.

Content model

Content type is string.

Parent elements

drill

tbl

Represents a layout table. Used to position elements for formatting in a report.

Content model

id *(optional)* then ref *(optional)* then style *(optional)* then trow *(any number)* then summaryText *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

tcell

Represents a layout table cell.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then cspan *(optional)* then rspan *(optional)* then hdr *(optional)* then item *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

trow

td

Represents a rich text table cell. Equivalent to the HTML td element.

Content model

style *(any number)* then rspan *(optional)* then cspan *(optional)* then item *(any number)* then extension *(optional)*

Parent elements

tr

textStyle

Defines a text style.

Content model

wrapping *(optional)* then direction *(optional)* then writingMode *(optional)* then biDirectional *(optional)* then justification *(optional)* then kashidaSpace *(optional)* then wordBreak *(optional)* then wordBreakStyle *(optional)* then strictLineBreaking *(optional)*

Parent elements

styleGroup

th

Represents a rich text table header cell. Equivalent to the HTML th element.

Content model

style *(any number)* then rspan *(optional)* then cspan *(optional)* then item *(any number)* then extension *(optional)*

Parent elements

tr

thSep

Specifies whether to delimit digit groups with the thousands separator.

For more information, see **Use Thousands Separator** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is boolean.

Parent elements

p_txtbox

timeUI

Specifies which interface the prompt control renders.

For more information, see **Select UI** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

The possible values of this element are restricted to the following.

CLOCK

A clock control.

EDITBOX

An edit text box.

Parent elements

p_time

toc

Defines a table of contents.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then entry *(any number)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

top

Defines the size of box margin or box padding.

Content model

val then units then extension *(optional)*

Parent elements

margin , padding

top

Defines the color, line style, and width of top border.

Content model

color *(optional)* then lineStyle *(optional)* then width *(optional)*

Parent elements

border

toText

Specifies the label that appears beside the end of a range. The default label text is To.

For more information, see **Select UI** in the *IBM Cognos Business Intelligence Report Studio User Guide*.

Content model

Content type is string.

Parent elements

p_date , p_dtime , p_intrvl , p_time , p_txtbox , p_value

tr

Represents a rich text table row. Equivalent to the HTML tr element.

Content model

style *(any number)* then (th or td) *(any number)* then extension *(optional)*

Parent elements

table

treeUI

Specifies which interface the prompt control renders.

Content model

Content type is string.

The possible values of this element are restricted to the following.

NORMAL

The standard tree prompt user interface.

COMPRESSED

Reserved.

DROPDOWN

Reserved.

CASCADING

Reserved.

Parent elements

p_tree

trow

Specifies a layout table row.

Content model

style *(any number)* then tcell *(any number)*

Parent elements

tbl

txt

Defines a text frame.

Content model

id *(optional)* then ref *(optional)* then ctx *(optional)* then style *(any number)* then drills *(optional)* then drillAction *(any number)* then val then valErrorState *(optional)* then valTyp then fmtVal *(optional)* then fmtPatrn *(optional)* then exclPatrn *(optional)* then fmtLoc *(optional)* then fmtScale *(optional)* then annURL *(optional)* then extension *(optional)*

Parent elements

item , item , reportElement

type

Specifies the type of area, for example, the legend.

Content model

Content type is string.

Parent elements

area

type

Specifies the type of parameter.

Content model

Content type is string.

Parent elements

parameter

underline

Specifies the font style is underline.

Content model

Content type is boolean.

Parent elements

fontStyle

units

Specifies the unit of measurement for the size.

Content model

Content type is string.

The possible values of this element are restricted to the following.

PX

The unit of measurement is pixels.

PERCENT

The unit of measurement is percent.

CM

The unit of measurement is centimeters.

MM

The unit of measurement is millimeters.

IN

The unit of measurement is inches.

PT

The unit of measurement is points.

PC

The unit of measurement is picas.

EM

The unit of measurement is ems.

EX

The unit of measurement is ex.

Parent elements

bottom , height , indent , kashidaSpace , left , right , size , top , width

url

Specifies an absolute URI path to an image file.

Content model

Content type is string.

Parent elements

drill , img

url

Specifies an absolute URI path to a chart image file.

Content model

Content type is string.

Parent elements

cht

URLParameters

Reserved.

Content model

Empty element.

Parent elements

drill

use

Specifies the values used by the prompt object. These values can be different than the ones that are rendered to the user.

Content model

Content type is string.

Parent elements

child , max , min , sval

val

Specifies the raw value used to render the text frame contents.

Content model

Content type is string.

Parent elements

txt

val

Specifies the number of units that specifies the size.

Content model

Content type is double.

Parent elements

bottom , height , indent , kashidaSpace , left , right , size , top , width

val

Contains the custom HTML from the resource.

Content model

Content type is string.

Parent elements

html

valErrorState

Specifies the error that caused the cell to be empty.

If the value is `OK`, the cell should not contain any data.

Content model

Content type is string.

The possible values of this element are restricted to the following.

OK

The cell should appear empty, there is no error.

NULL

The error is NULL.

NA

The error is not applicable.

DIV0

The error is division by zero.

OVERFLOW

The error is overflow.

SECURITY

The error is security.

CASTING

The error is casting.

OTHER_ERROR

Specifies an error not listed above.

Parent elements

txt

vAlign

Specifies the vertical alignment.

Content model

Content type is string.

The possible values of this element are restricted to the following.

TOP

The top of the element is aligned with the top of the tallest element on the line.

MIDDLE

The element is placed in the middle of the parent element.

BOTTOM

The bottom of the element is aligned with the lowest element on the line.

SUPER

Aligns the element as it was superscript.

SUB

Aligns the element as it was subscript.

TEXT-TOP

The top of the element is aligned with the top of the parent element's font.

TEXT-BOTTOM

The bottom of the element is aligned with the bottom of the parent element's font.

LENGTH

Reserved.

%

Raises or lower an element as a percentage of the normal line height. Negative values are allowed.

BASELINE

Align the baseline of the element with the baseline of the parent element. This is default.

Parent elements

styleGroup

valTyp

Specifies the text frame data value type.

Content model

Content type is string.

The possible values of this element are restricted to the following.

date

The text frame data value type is date.

time

The text frame data value type is time.

datetime

The text frame data value type is datetime.

number

The text frame data value type is number.

currency

The text frame data value type is currency.

percent

The text frame data value type is percent.

text

The text frame data value type is text.

timeInterval

The text frame data value type is time interval.

Parent elements

txt

---

value

Specifies the value in the name/value pair of a parameter.

Content model

Content type is string.

Parent elements

parm

---

value

Contains an enumeration of possible secondary operations.

Content model

Content type is string.

The possible values of this element are restricted to the following.

FORWARD

Can move to next prompt page.

BACK

Can move to previous prompt page.

FINISH

Can finish prompt collection.

LAST

Can move to last report page.

NEXT

Can move to next report page.

PREVIOUS

Can move to previous report page.

DRILL

Can drill up or down.

FIRST

Can move to the first prompt page.

EXTENSION

Reserved.

RELEASE

Can release the session.

Parent elements

secondaryOperations

versionBase

Specifies the location of the saved output.

The existence of this element in a report output indicates that the report output came from a saved version of the report.

Content model

Content type is string.

Parent elements

document , filterResultSet

verticalSize

Specifies the number of repeater table rows per column.

Content model

Content type is int.

Parent elements

reptbl

widget

Specifies an IBM Cognos Business Insight widget object.

Content model

id *(optional)* then ref *(optional)* then style *(any number)* then widgetURI then annURL *(optional)* then extension *(optional)*

Parent elements

item , reportElement

widgetURI

Specifies the url of a Web page widget.

Content model

Content type is string.

Parent elements

widget

width

Specifies the width of the border or box.

Content model

val then units then extension *(optional)*

Parent elements

bottom , boxStyle , left , right , top

wordBreak

Specifies whether word breaking is enabled.

Content model

Content type is boolean.

Parent elements

textStyle

wordBreakStyle

Specifies the Report Studio word break style.

Content model

Content type is string.

The possible values of this element are restricted to the following.

NORMAL

Specifies the Normal word break style.

BREAK_ALL

Specifies the Break All word break style.

KEEP_ALL

Specifies the Keep All word break style.

Parent elements

textStyle

wrapping

Specifies whether word wrapping is enabled.

When `true`, word wrapping is enabled.

Content model

Content type is boolean.

Parent elements

textStyle

writingMode

Specifies the writing mode for the style. This is used for some Asian language styles.

Content model

Content type is string.

The possible values of this element are restricted to the following.

LEFT_TO_RIGHT_TOP_TO_BOTTOM

The writing mode is from left to right, and top to bottom on the page.

TOP_TO_BOTTOM_RIGHT_TO_LEFT

The writing mode is from right to left, and bottom to top on the page.

Parent elements

textStyle

---

x

Specifies the x coordinate of the chart `area`.

Content model

Content type is int.

Parent elements

coord

---

y

Specifies the y coordinate of the chart `area`.

Content model

Content type is int.

Parent elements

coord

Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing
3755 Riverside Dr
Ottawa, ON K1V 1B7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at " Copyright and trademark information " at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y