



BEA WebLogic Workshop™ Help

Version 8.1 SP2
November 2003

Copyright

Copyright © 2003 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software–Restricted Rights Clause at FAR 52.227–19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227–7013, subparagraph (d) of the Commercial Computer Software—Licensing clause at NASA FAR supplement 16–52.227–86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Liquid Data for WebLogic, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E–Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

Table of Contents

User–Installed Extensions.....	1
BEA Tuxedo Control.....	2
Tuxedo Control Overview.....	3
Who Uses the Tuxedo Control?.....	4
What is Required?.....	5
Getting Started with the Tuxedo Control.....	6
Before You Begin.....	7
What You Must Know About Tuxedo.....	8
Setting Up a WebLogic Tuxedo Connector (WTC) Connection.....	10
How Do I: Create a Java Application That Uses a Tuxedo Service.....	12
How Do I: Use a Tuxedo Service in a Java Application.....	13
How Do I: Create a Web Service with Transactional Support in the Tuxedo Service.....	18
How Do I: Call Non–transactional Tuxedo Services.....	19
How Do I: Create a Tuxedo Control.....	20
How Do I: Create a Tuxedo Control.....	21
How Do I: Create a Logical Business Control Supporting Multiple Tuxedo Services.....	27
How Do I: Create a Tuxedo Control That Uses Queue Type Services.....	28
How Do I: Configure a WTC Service.....	33
How Do I: Map Java Data Types/FML and VIEW Buffers.....	38
How Do I: Map Data Types.....	39
How Do I: Map Java Data Type Buffers to FML and VIEW.....	40
How Do I: Map FML and VIEW Buffers to Java Data Types.....	43
How Do I: Map Non–Fielded Buffers.....	44

Table of Contents

Getting Started with the Tuxedo Control Samples.....	45
Simpapp Sample.....	46
Bankapp Samples.....	47
Bankapp Web Service.....	48
Bankapp Portal.....	51
Bankapp Workflow.....	53
Bankapp Pageflow.....	55
Troubleshooting Tuxedo Control.....	56
Development-time Errors.....	57
Run-time Errors.....	58
@jc:tuxedo Tag.....	59
Crystal Reports for BEA WebLogic Workshop.....	63
Topics Included in This Section.....	64
Welcome to the Crystal Reports for BEA WebLogic Workshop Java API Reference.....	65
Crystal Common Classes.....	66
Viewer Java SDK.....	270
crystal-tags-reportviewer Tags.....	391
Viewer Tag Library.....	392
Viewer Tag Library.....	410
Welcome to Crystal Reports for BEA WebLogic Workshop.....	428
Getting Started.....	433
Using Crystal Reports for BEA WebLogic Workshop.....	436
Welcome.....	443

Table of Contents

Architecture.....	448
Viewer Functionality.....	452
Setting up the Development Environment.....	457
Getting Started.....	459
SDK Reference.....	480
Viewer Tag Library.....	485

User-Installed Extensions

When you install extensions created by third parties, associated help for the extension will appear in the Table of Contents in this section. Extensions include third-party Java controls, IDE extensions, custom tag libraries and application templates.

Note: This area may also include documentation for extensions installed with WebLogic Workshop.

Note: WebLogic Workshop can display help from a BEA documentation server ("online") or from your local disk. User-installed Java control or IDE extension help is only present in the help content on your local disk. To learn how to switch between online and local help content, see [Using WebLogic Workshop Help](#).

Related Topics

None

BEA Tuxedo Control

The following sections provide prerequisite and overview information for the Tuxedo Control.

- Tuxedo Control Overview
- Who Uses the Tuxedo Control?
- What is Required?

Tuxedo Control Overview

The Tuxedo Control is a feature addition to WebLogic Workshop that allows developers to create web services or WebLogic Workshop applications (that can utilize controls) that use Tuxedo services.

The Tuxedo Control allows you to create a web service or WebLogic Workshop application that uses one or multiple Tuxedo services in its business logic.

The Tuxedo Control provides the low-level plumbing to:

- create connections to Tuxedo services
- invoke the services and retrieve the responses
- convert your application's Java data types to and from Tuxedo buffer types

The Tuxedo Control is a generic interface. To use the control to communicate with specific Tuxedo services, you must create a Control Extension (jcx file). The Control Extension is a Java interface that tells the Tuxedo Control the names of the Tuxedo services you want to invoke, the type of interaction the services require (service with reply, queued service, etc), and the Tuxedo buffer types the services expect as input. Your application invokes the methods defined by the Control Extension interface to invoke the Tuxedo services. After you create a Control Extension for a specific set of Tuxedo services, you can reuse the Control Extension in any application that needs to access those services. WebLogic Workshop does most of the work of creating a Control Extension for you.

Who Uses the Tuxedo Control?

The Tuxedo Control has two different users. Typically, one user may be more familiar with Tuxedo services and applications. This user will develop the control extension and make it available for use in business solutions. Users who are less knowledgeable about Tuxedo, can leverage the control extension that is already built.

The following descriptions provide an example of possible users.

Enterprise Architect

A developer who understands Tuxedo services and the supported buffer types. This developer will build a Tuxedo Control extension as a JCX or JAR file to deliver as a control for a WebLogic Workshop developer who does not have any Tuxedo knowledge.

Typical tasks for an Enterprise Architect are:

- "How Do I: Create a Tuxedo Control"
- "How Do I: Create a Logical Business Control Supporting Multiple Tuxedo Services"
- "How Do I: Manage Complex Interactions"
- "How Do I: Create a Tuxedo Control That Uses Queue Type Services"

WebLogic Workshop Developer

A developer of web services or Java applications that will use the Tuxedo Control extension delivered by the Enterprise Architect to use the Tuxedo services needed to accomplish certain tasks. This developer will not require technical understanding of Tuxedo services and buffer types.

Typical tasks for an Enterprise Architect are:

- "How Do I: Use a Tuxedo Service in a Java Application"
- "How Do I: Create a Web Service With Transactional Support in the Tuxedo Service"
- "How Do I: Call Non-transactional Tuxedo Service"

What is Required?

The Tuxedo Control can access Tuxedo services running on the following versions of Tuxedo:

- BEA Tuxedo 8.1, 8.0, 7.1, or 6.5

You must configure a connection to WebLogic Tuxedo Connector to access Tuxedo services from WebLogic Workshop.

Certain Tuxedo service information is required to use the service from any client, such as the service name, the type of service, the type of buffer, and how the data is placed in the buffer and extracted from the buffer.

Related Topics

- Working with Java Controls
- Building Your Own Java Control
- Building Web Services

Getting Started with the Tuxedo Control

The following topics explain how to get started with the Tuxedo Control.

Topics Included in This Section

- Before You Begin
- What You Must Know About Tuxedo
- Setting Up a WebLogic Tuxedo Connector (WTC) Connection

Related Topics

- Getting Started With Java Controls

Before You Begin

Prior to working with the Tuxedo Control, you should have access to a working BEA Tuxedo system.

Note: Before using the Tuxedo Control, it is assumed:

- ◆ Tuxedo services you want to access are available to WebLogic Tuxedo Connector (WTC).
- ◆ WTC is configured properly to connect to Tuxedo and initiate a Tuxedo session as specified in *Setting Up a WebLogic Tuxedo Connector (WTC) Connection*.
- ◆ You have a properly configured and WebLogic Workshop–enabled WebLogic Server domain to use for creating a web service or Java application that uses Tuxedo services.

Related Topics

- Installing BEA Tuxedo
- Installing BEA WebLogic Platform

What You Must Know About Tuxedo

Before creating a web service or Java application that uses Tuxedo services, you need to understand the general representation of a Tuxedo service so that proper data mapping occurs.

Understanding How a Tuxedo Service is Represented

The Tuxedo Control provides a generic mechanism for using Tuxedo services. As provided by BEA, the Tuxedo Control has no information about the Tuxedo service the control accesses at run-time. In using the Tuxedo Control, you must create a Java Control Extension (.jcx file) that configures an instance of the control for a specific Tuxedo service. This configuration information controls the type of buffers to use, what information is placed in the buffers, how to call the Tuxedo service, and other information specific to the service that the control is calling.

The information necessary to configure the Tuxedo Control instance is the same information needed by any client of the Tuxedo service the control is calling. At a minimum, this includes the name of the service, the type of the Tuxedo buffer to send to the service, how to invoke the service, what data to place in the buffer, and what data to retrieve from any response from the service. For some buffer types, such as FML32 buffers, the field definition or field tables must be provided. Similarly, for VIEW32 buffers, the class definitions must also be provided.

Tuxedo Field Headers

Before you can begin to work with FML fielded buffers, you must:

- Define fields.
- Make field definitions available to application programs (through field table files and mapping functions at runtime). You must process the field table into a Java class using the `mkfldclass` and `mkfldclass32` utilities. The control then uses the Java class at runtime.

For more information about `mkfldclass` and `mkfldclass32` utilities, see WebLogic Server Javadoc.

Tuxedo VIEW Buffers

Before you can begin to work with VIEW buffers, you must:

- Define the VIEW fields.
- Create the `TypedView` or `TypedView32` class definitions that your application uses. You must process the VIEW definition using the `viewj` or `viewj32` compilers, which generate Java source code. The control uses the Java class at runtime.

For more information about `viewj` and `viewj32` utilities, see WebLogic Server Javadoc.

Tuxedo Typed Buffers

WebLogic Tuxedo Connector provides an interface called `TypedBuffers` that corresponds to Tuxedo typed buffers. Messages are passed to servers in typed buffers.

Note: WebLogic Tuxedo Connector does not support double-byte character sets or international character sets.

IDE Extensions

The WebLogic Tuxedo Connector provides the following buffer types.

Buffer Type	Description
STRING	Buffer type used when the data is an array of characters that terminates with the null character. Tuxedo equivalent: STRING.
CARRAY	Buffer type used when the data is an undefined array of characters (byte array), any of which can be null. Tuxedo equivalent: CARRAY.
FML	Buffer type used when the data is self-defined. Each data field carries its own identifier, an occurrence number, and possibly a length indicator. Tuxedo equivalent: FML.
FML32	Buffer type similar to FML but allows for larger character fields, more fields, and larger overall buffers. Tuxedo equivalent: FML32.
XML	Buffer type used when data is an XML based message. Tuxedo equivalent: XML for Tuxedo Release 7.1 and higher.
VIEW	Buffer type used when the application uses a Java structure to define the buffer structure using a view description file. Tuxedo equivalent: VIEW.
VIEW32	Buffer type similar to View but allows for larger character fields, more fields, and larger overall buffers. Tuxedo equivalent: VIEW32.
X_OCTET	Buffer type used when the data is an undefined array of characters (byte array) any of which can be null. X_OCTET is identical in semantics to CARRAY. Tuxedo equivalent: X_OCTET.
X_COMMON	Buffer type identical in semantics to View. Tuxedo equivalent: VIEW.
X_C_TYPE	Buffer type identical in semantics to View. Tuxedo equivalent: VIEW.

Related Topics

- FML and VIEW Features
- WebLogic Tuxedo Connector TypedBuffers
- Tuxedo Documentation

Setting Up a WebLogic Tuxedo Connector (WTC) Connection

To develop web services or Java applications using WebLogic Workshop that access Tuxedo services, you must use WebLogic Tuxedo Connector to act as the gateway between the environments. WebLogic Tuxedo Connector connects WebLogic Server to Tuxedo. You must configure your environment to connect the two environments using WebLogic Tuxedo Connector.

Tuxedo Environment

In the Tuxedo environment, you must:

1. Define a Tuxedo domain in the UBBCONFIG and make sure it is available.
 - a. You need to define a GROUP in the GROUP section. For example, a GROUP definition should be:

```
GWGRP LMID=simple GRPNO=2
```

- b. Add the following in the SERVERS section:

```
DMADM SRVGRP=GWGRP SRVID=1
GWADM SRVGRP=GWGRP SRVID=2
GWTDOMAIN SRVGRP=GWGRP SRVID=3
```

2. Define a WSL connection to Tuxedo in the UBBCONFIG and make sure it is available.
 - a. You need to define a GROUP in the GROUP section. For example, a GROUP definition should be:

```
WSLGROUP LMID=simple GRPNO=3
```

- b. Add the following in the SERVERS section:

```
WSL SRVGRP=WSLGRP SRVID=1 CLOPT="-A -- -n //machine:port"
```

3. In the DMCONFIG, specify the network address (NWADDR) where WTC is running. The access point ID for the local and remote domain definitions must match the corresponding remote and local access point IDs in the WTC service configuration.
 - ◆ The Tuxedo local access point ID must match the WTC remote access point ID.
 - ◆ The Tuxedo remote access point ID must match the WTC local access point ID.A sample DMCONFIG file follows.

```
# DMCONFIG FILE
```

```
*DM_RESOURCES
```

```
VERSION="U22"
```

```
*DM_LOCAL
```

IDE Extensions

```
<LOCAL DOMAIN NAME> GWGRP=GWGRP2
ACCESSPOINTID="<LOCAL DOMAIN NAME>"
AUDITLOG="<APPDIR>\aud"
BLOCKTIME=20
DMTLOGDEV="<APPDIR>\tlog"
DMTLOGNAME="DMTLOG_TDOM1"
MAXDATALEN=56
MAXRACCESSPOINT=89
MAXTRAN=100
BLOB_SHM_SIZE=1000000
SECURITY=NONE
```

*DM_REMOTE

```
<REMOTE DOMAIN NAME> ACCESSPOINTID="<REMOTE DOMAIN NAME>"
ACL_POLICY="LOCAL"
CREDENTIAL_POLICY="LOCAL"
```

*DM_TDOMAIN

```
<LOCAL DOMAIN NAME> NWADDR="//<LOCAL DOMAIN NETWORK ADDRESS>:<PORT NUMBER>"
```

```
<REMOTE DOMAIN NAME> NWADDR="//<REMOTE DOMAIN NETWORK ADDRESS>:<PORT NUMBER>"
```

*DM_REMOTE_SERVICES

4. Run the following commands to boot the Tuxedo application:

```
tmloadcf -y <name>.ubb
dmloadcf -y domconfig
tmboot -y
```

You should now have the Tuxedo application up and running.

WebLogic Environment

To configure WebLogic Tuxedo Connector, start WebLogic Server and use the **Insert Control – Tuxedo** dialog to configure a WTC service. For instructions on how to configure the WTC service, refer to [How Do I: Configure a WTC Service](#).

Related Topics

- [Configuring WebLogic Tuxedo Connector](#)
- [WebLogic Tuxedo Connector Documentation](#)

How Do I: Create a Java Application That Uses a Tuxedo Service

The following topics describe how to create a Java application that uses Tuxedo services.

Topics Included in This Section

- [How Do I: Use a Tuxedo Service in a Java Application](#)
- [How Do I: Create a Web Service with Transactional Support in the Tuxedo Service](#)
- [How Do I: Call Non-transactional Tuxedo Services](#)

Related Topics

- [Working with Java Controls](#)
- [Building Web Services](#)
- [WebLogic Tuxedo Connector Documentation](#)

How Do I: Use a Tuxedo Service in a Java Application

The Tuxedo Control allows you to create a web service or WebLogic Workshop application that uses one or multiple Tuxedo services in its business logic. The Tuxedo Control is a generic interface. To use the control to communicate with specific Tuxedo services, you must create a Control Extension (jcx file). The Control Extension is a Java interface that tells the Tuxedo Control the names of the Tuxedo services you want to invoke, the type of interaction the services require (service with reply, queued service, etc), and the Tuxedo buffer types the services expect as input. Your application invokes the methods defined by the Control Extension interface to invoke the Tuxedo services. After you create a Control Extension for a specific set of Tuxedo services, you can reuse the Control Extension in any application that needs to access those services. WebLogic Workshop does most of the work of creating a Control Extension for you.

In this section, the example shown represents creating a web service. The code for a web service is contained in a Java Web Service (JWS) file. A JWS file is a Java file in that it contains code for a Java class as well as implementation code intended specifically for a web service class targeted to run under WebLogic Server.

To create a web service that uses a Tuxedo service:

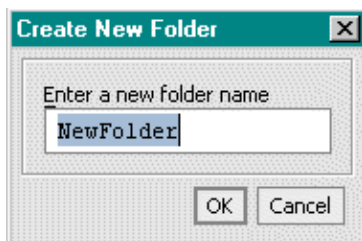
- Set Up Your Environments
- Create a Java Web Application
- Add a Tuxedo Control to Your Java Application
- Add a Method to a Web Service

Set Up Your Environments

1. Start the Tuxedo server that advertises the Tuxedo service you want to use.
2. Start WebLogic Workshop. From the **Start** menu, choose **Programs ->WebLogic Platform 8.1 ->WebLogic Workshop 8.1**.
3. Start WebLogic Server. You can confirm the status of WebLogic Server by checking the status bar at the bottom of WebLogic Workshop. If WebLogic Server is running, the status bar displays a green ball. If WebLogic Server is not running, the status bar displays a red ball. If the status bar shows a red ball and Server Stopped, then start WebLogic Server from the **Tools** menu. Choose **Tools -> WebLogic Server -> Start WebLogic Server**.

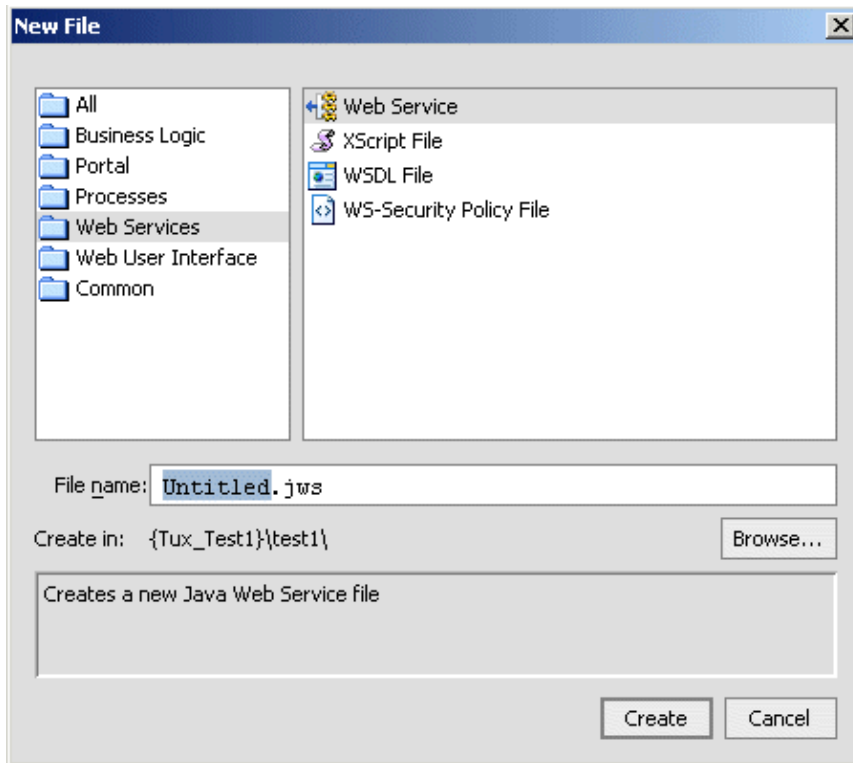
Create a Java Web Application

1. Create or open a Webapp Application and Project in WebLogic Workshop.
2. In the **Enter a new folder name** field, type the name of the folder where you want your web service to reside.



3. Click **OK**. The new folder appears inside the existing WebLogic Workshop Project folder in the **Application** tab.

4. Right-click the new folder and select **New** -> **Web Service** or other application type you want. The **New File** dialog box displays.



5. In the upper left-hand pane, confirm that **Web Services** is selected. In the upper right-hand pane, confirm that **Web Service** is selected. In the **File Name** field, type the name of the web service. The File Name you enter becomes the name of the web service class WebLogic Workshop creates.

Note: If you selected an alternative application type other than **Web Service**, confirm the selection and enter an appropriate **File Name**.

6. Click **Create**. In the case of a **Web Service**, a new JWS file is created and displayed in the Design View.

Add a Tuxedo Control to Your Java Application

Controls act as interfaces between your web service and other data sources. Therefore, you want to add a Tuxedo Control to your web service to provide your web service with access to Tuxedo services.

Java Control Extensions (JCX File) define an interface, but no implementation. Using the control is a matter of declarative programming instead of procedural programming.

The tasks in this step are:

- Use a Tuxedo Control Extension
- Create a WebLogic Tuxedo Connector (WTC) Service
- Use a Tuxedo Service from the Control Extension
- Test the Web Service Using Debugger

Use a Tuxedo Control Extension

In this task you will use a Tuxedo Control file and then add a method to the control file.

1. Select the web service in which you want to use a Tuxedo service.
2. Click the **Design View** tab.
3. From the **Insert** menu, choose **Controls -> Tuxedo**. The **Insert Control – Tuxedo** dialog displays.

Insert Control - Tuxedo

STEP 1 Variable name for this control:

STEP 2 I would like to :

☐ Use a Tuxedo control already defined by a JCX file

JCX file:

☒ Create a new Tuxedo control to use.

New JCX name:

☐ Make this a control factory that can create multiple instances at runtime

STEP 3

Service Type: Buffer Type:

Service Name:

Send Queue Name:

Receive Queue Space:

Receive Queue Name:

Field Tables:

Return View Classes:

Send View Class:

4. In Step 1 enter a variable name for the control.
5. In step 2 click one of the following radio buttons:

Click **Use a Tuxedo Control already defined by a JCX file** to use an existing Tuxedo Control file. Browse to the JCX file. If you select an existing JCX file, the remaining fields in the Tuxedo Control – **Insert Tuxedo** dialog become inactive.

Click **Create a new Tuxedo Control to use** to create a new Tuxedo Control. You can follow the procedures described in How Do I: Create a Tuxedo Control, to create a new Tuxedo Control to use in your application.

Create a WebLogic Tuxedo Connector (WTC) Service

WTC acts as a gateway between the Tuxedo environment and WebLogic environment; therefore, a WTC service definition must exist that maps to the Tuxedo service you want to use in your web service.

Import the Tuxedo service that you want to use in your web service. For information on how to import WTC services, see *How Do I: Configure a WTC Service*.

Use Public Methods from the Control Extension

The Tuxedo Control provides the following public methods to use in your web service in addition to the methods defined by your control extension. If you want to use these methods you must call them explicitly.

Public Method	Description
getConnection()	Gets the WTC connection currently used by this control. This allows you to use more of the WTC APIs defined for a connection. getConnection() returns an instance of an ApplicationToMonitorInterface object that can be used to interact with Tuxedo through JATMI calls. Note: This method will not allow you to terminate the connection or use tpacall with asynchronous calls. Using the getConnection() method assumes that the developer has a working knowledge of WTC calls.
getMappingIssues()	Returns an array of MappingIssues associated with the last request made. Mapping issues are generated by the control when there is a discrepancy between the Java data type and the Tuxedo buffer.
gettpurcode()	Returns the tpurcode associated with the last exception that occurred.
gettperrno()	Returns the tperrno associated with the last request made by the control.

Use a Tuxedo Service from the Control Extension

Next you must incorporate one or more methods from the control extension just created in the previous step in your web service.

1. Select the web service file (JWS).
2. Click the **Design View** tab.
3. From the **Data Palette**, drag a method from the control extension to the web service **Design View**. This adds a method to the web service that shows up as a SOAP service. The implementation of this service simply calls the Tuxedo Control extension; therefore, exposing the Tuxedo service as a web service.

For the web service to provide additional processing before or after the call to the Tuxedo service or if the web service needs a different signature, click the **Source View** tab and edit the Java code as needed.

4. From the menu, select **File -> Save**.

Test the Web Service Using Debugger

Next you will test your web service using debugger.

1. From the **Debug** menu, select **Start**.
2. The WebLogic Workshop test browser displays with a place to enter the parameters for the web service method that was previously added. Enter the appropriate parameters and click the method name button.
3. Verify the results that return at the bottom of the page when the test browser refreshed the window.

Add a Method to a Web Service

Web services expose their functionality through methods. These methods allow clients to request something from the web service.

To enhance the web service and add a method:

1. Select the web service to which you want to add a method and click the **Design View** tab.

Note: It is helpful, but not required, to be in the Design View so you can see the changes as you add elements to your web service.

2. From the **Insert** menu, select **Method**.
3. Select the method you want to add to the web service and click **Enter**.
4. Provide the implementation of the web service. Write the method code that performs the function of the web service. For example, implementation of the web service may require calling one or more controls.
5. From the menu select **File** -> **Save** to save your work.

Related Topics

- Building Web Services

How Do I: Create a Web Service with Transactional Support in the Tuxedo Service

Tuxedo services are often transactional; therefore, the Tuxedo Control provides a declarative means of propagating the transaction from the web service to the Tuxedo system.

To create a web service that uses a transactional Tuxedo service:

1. Select the web service in which you want to use a Tuxedo service.
2. Double-click the JCX file in the **Application Pane**.
3. Click on the **Property Editor** tab and select the transaction attribute tag.
4. Specify the transaction attribute that applies to the Tuxedo service. The possible attribute values are:

Transaction Attribute	Description
Supports	The transaction state of the caller of the control is propagated to Tuxedo via WTC and the Tuxedo service participates in a distributed transaction with the web service implementation. The Tuxedo Control does not start a transaction automatically or check for a transaction before executing.
NotSupported	<p>The Tuxedo Control effectively suspends any existing transaction context before invoking the Tuxedo service. This means that any action that occurs within the Tuxedo service is not part of the transaction of the caller of the control.</p> <p>Specifying NotSupported also sets TPNOTRAN flag when making a request to Tuxedo.</p>

Note: If an exception occurs inside the web service and is not caught, any transaction in progress is aborted.

5. From the menu, select **File** -> **Save** to save your work.

How Do I: Call Non-transactional Tuxedo Services

Currently, WebLogic Workshop always starts a transaction before invoking the control. If the method on the control extension is marked as "Supports", then that transaction context is sent to Tuxedo by the control via WebLogic Tuxedo Connector. If the Tuxedo server called has not performed a tlopen or if the Tuxedo domain is not set up with a transaction manager, then the service call fails when the domain gateway tries to import the transaction. The service call fails with a TPETRAN error.

To use a non-transactional service, use the "NotSupported" value for the transaction attribute.

1. Select the web service in which you want to use a Tuxedo service.
2. Double-click the JCX file in the **Application Pane**.
3. Click on the **Property Editor** tab and select the transaction attribute tag.
4. Specify NotSupported for the transaction attribute that applies to the Tuxedo service.
5. From the menu, select **File** -> **Save** to save your work.

Related Topics

- @jc:tuxedo Tag

How Do I: Create a Tuxedo Control

The following topics describe how to create a Tuxedo Control:

Topics Included in This Section

- [How Do I: Create a Tuxedo Control](#)
- [How Do I: Create a Logical Business Control Supporting Multiple Tuxedo Services](#)
- [How Do I: Create a Tuxedo Control That Uses Queue Type Services](#)

Related Topics

- [Working with Java Controls](#)
- [Building Web Services](#)
- [WebLogic Tuxedo Connector Documentation](#)

How Do I: Create a Tuxedo Control

In this task you will create a Tuxedo Control file and then add a method to the control file.

1. Select the web service in which you want to use a Tuxedo service.
2. Click the **Design View** tab.
3. From the **Insert** menu, choose **Controls** → **Tuxedo**. The **Insert Control – Tuxedo** dialog displays.

Insert Control - Tuxedo

STEP 1 Variable name for this control:

STEP 2 I would like to :

☐ Use a Tuxedo control already defined by a JCX file

JCX file:

☒ Create a new Tuxedo control to use.

New JCX name:

☐ Make this a control factory that can create multiple instances at runtime

STEP 3

Service Type: Buffer Type:

Service Name:

Send Queue Name:

Receive Queue Space:

Receive Queue Name:

Field Tables:

Return View Classes:

Send View Class:

4. In Step 1 enter a variable name for the control.
5. In step 2 click the following radio buttons:

Click **Create a new Tuxedo Control to use** and enter a name for the JCX file in the **New JCX name** field.

Note: The Tuxedo Control you create must be in a folder within the Project. For example, the JCX name will be *folder.name* or *folder\name*.

6. In the Tuxedo Control – **Insert Tuxedo** dialog specify the following Tuxedo service attributes in

Step 3:

Tuxedo Control Field	Description
Service Type	<p>Select the type of service from the menu. This specifies the type of interaction that the Tuxedo Control supports in the Tuxedo service.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> ◆ service which supports synchronous request/response. ◆ oneway which supports request without response. ◆ queue which supports tpenqueue and tpdequeue. <p>Note: For more information about specifying queue service types, see How Do I: Create a Tuxedo Control That Uses Queue Type Services.</p> <p>The default is service.</p>
Buffer Type	<p>Select the type of buffer from the menu. This specifies the type of buffer that the Tuxedo Control will construct to send to the Tuxedo service.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> ◆ carray used when the data is an unspecified array of characters (byte array), any of which can be null. Tuxedo equivalent: CARRAY. ◆ fml used when the data contains named fields of simple types each with a maximum length of 65 K bytes. ◆ fml32 used when the data contains named fields including nested FML32 fields each having a maximum length of 4 GB. ◆ string used when the data contains a simple string as its only contents. ◆ view used to define C or COBOL structures for a Tuxedo application and to define an equivalent Java TypedView buffer for a WebLogic application. A view description file in which the fields and types that appear in the data structure are defined, must be available to client and server processes that use a data structure described in a VIEW typed buffer. Encoding and decoding are performed automatically if the buffer is passed between machines of different types. ◆ view32 is equivalent to VIEW but uses 32 bits for length and count fields, which allows for larger and more fields. ◆ xml used when data is an XML document. Data input to methods must be in the form of a Java String or an XML Bean. Tuxedo equivalent: XML for Tuxedo Release 7.1 and higher. <p>Note: For input XML using data dependent routing, remove all comments generated by the WebLogic Workshop test browser in the XML buffer. Depending on the location of the comments in the buffer, Tuxedo may have problems parsing the buffer.</p>
Buffer Type (cont.)	<ul style="list-style-type: none"> ◆ x_common is a synonym for the view buffer type. An x_common buffer is represented by a Java class extending the TypedXCommon class. It is identical in semantics to a view buffer.

IDE Extensions

	<ul style="list-style-type: none"> ◆ x_c_type is a synonym for the view buffer type. An x_c_type buffer is represented by a Java class extending the TypedXCType class. It is identical in semantics to a view buffer. ◆ x_octet is simply a byte array and is a synonym for the CARRAY buffer type in Tuxedo. ◆ none used when no input buffer is to be used in the request to the Tuxedo service. If the method signature contains input parameters, an error is generated. The default is fml32.
Service Name	<p>Type the WTC imported service name. This name should match the service name that is mapped in Create a WebLogic Tuxedo Connector (WTC) Service.</p> <p>If you are not sure of the service name, you can click Browse and a list of imported WTC service names display. If the service is listed in the WTC Imported Services dialog, you can select it and click OK. If the service you want is not listed, then the service has not been imported into WebLogic Server.</p> <p>To import a service from the WTC Imported Services dialog, click Import and the Services Exported by Tuxedo Application dialog displays.</p> <p>For information about importing WTC services, see Import New Tuxedo Services.</p>
Field Tables	<p>Type the name of the field table for the Tuxedo service when using fml or fml32 buffers. You should specify the field table when using fml or fml32 buffers that are sent or act as reply buffers from the called service. The name of the field table should match the classes generated with the mkfldclass or mkfldclass32 utilities.</p> <p>For buffer types fml and fml32, WTC needs the names and types of the fields the buffer can contain. These are described in field table classes created with mkfldclass and mkfldclass32 utilities.</p> <p>For more information about mkfldclass and mkfldclass32 utilities, see WebLogic Server Javadoc.</p>
Return View Classes	<p>Type the names of the view classes that correspond to the view buffer types that can be received from the Tuxedo service. This field is only required when the Tuxedo service returns a view or view32 buffers. These classes are independent from the Send View Class attribute which is used to specify the view or view32 class that represents the view buffer the Tuxedo service expects as an input.</p> <p>Note: The value of this attribute may be a list of classes so that a single Tuxedo service can respond with different VIEW buffers.</p> <p>For information about viewj and viewj32 utilities, see WebLogic Server Javadoc.</p>
Send View Class	<p>If you have selected view or view32 in the Buffer type field, the Send View Class field becomes active. This attribute provides the name of the class that represents the view/view32 buffer the service is expecting as an input buffer. This view class is used to create the TypedView buffer.</p>

7. Click **Create**. A new JCX file is created on the **Application** tab and a new icon appears in **Design View** of the web service.
8. In **Design View**, double-click the image of the newly created Tuxedo Control.
9. The new Tuxedo Control file (*controlname.jcx*) displays in the **Design View**.

IDE Extensions

10. Click the **Source View** tab. Edit the signature of this method in the JCX file to match the buffer contents expected for the Tuxedo service being called.

When using the Insert Wizard to create a new Java Control Extension, a single method is defined that corresponds to the **Service Type** and **Service Name** specified in the Insert Wizard. The signature and possibly the name of this method in the JCX file needs to be edited to match the name and buffer contents expected for the Tuxedo service being called. In addition, any other services that this control calls must have methods defined for them in this interface.

For queued services, the method name created by the wizard is enqueue, dequeue, or enqueueDequeue. You can change the wizard-created name.

- ◆ If send queue information is provided and no receive queue information is provided then the method inserted is enqueue.
- ◆ If receive queue information is provided and no send queue is provided, then the insert method is dequeue.
- ◆ If both send and receive queue information are provided, then the insert method is enqueueDequeue.

For information about mapping data and fields to create request/response buffers, see How Do I: Map Java Data Types/FML and VIEW Buffers.

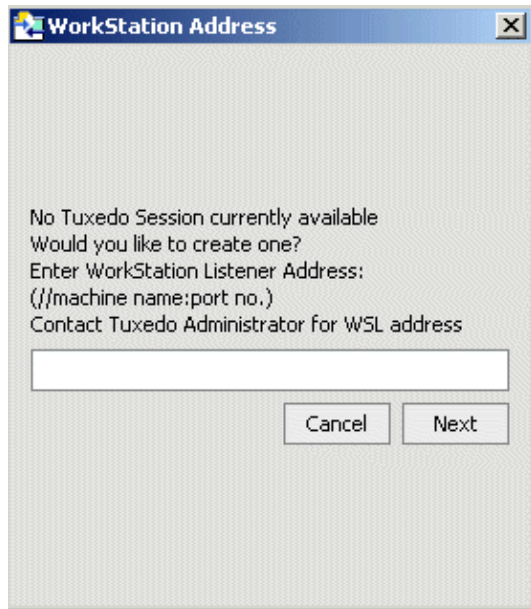
11. Select **File** -> **Save** from the menu.
12. Press **Ctrl+F4** to close the Tuxedo Control file (*controlname.jcx*.)

Import New Tuxedo Services

Within the web service or Java application, you can browse available Tuxedo services from the **Insert Control – Tuxedo** dialog. In the **Service Name** or **Receive Queue Space** fields, you can click **Browse** and see a list of available WTC imported services. If the service you want is not in the list, then the service has not been imported into WebLogic Server.

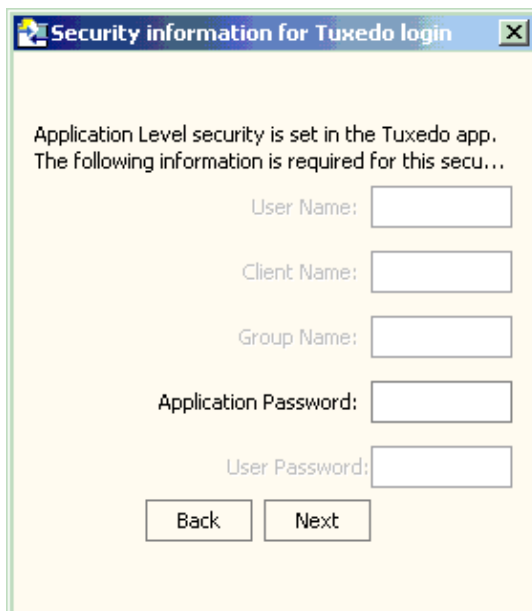
Note: Before you can import the service to WebLogic Server, you must configure the WTC service. To configure a WTC service, click **Configure** on the **WTC Imported Services** dialog box. Refer to How Do I: Configure a WTC Service.

To import a service from the **WTC Imported Services** dialog, click **Import** and the **Workstation Address** dialog displays.



1. Enter the network address of the workstation listener on a remote Tuxedo domain, such as a *machine_name:port number* combination *//mach1:5100*. This is the address of the Tuxedo application so the service names can be retrieved. Click **Next**.
2. When you click **Next** on the **Workstation Address** dialog, if the Tuxedo domain has security turned on, the **Security Information for Tuxedo**

Note: **login** dialog displays. Enter the Tuxedo application information and click **Next**.



A connection to the Tuxedo domain is made and a list of services exported by the Remote Tuxedo Domain displays.

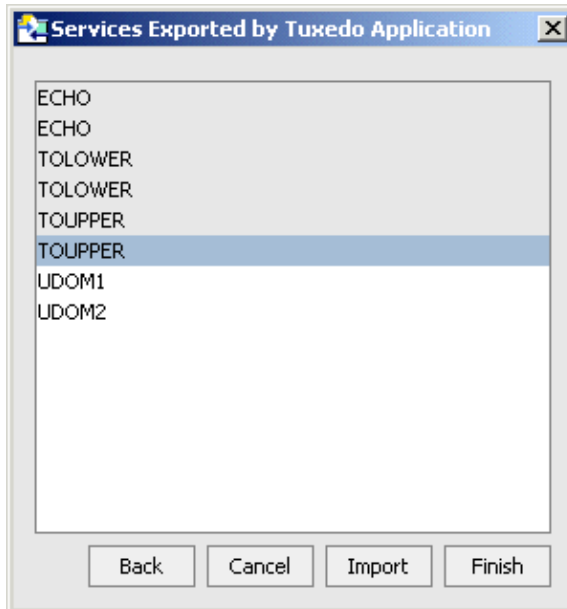
3. Select the services to import. If there is more than one local WTC domain configured in WebLogic Server, then the list of local domains configured displays and you can select the local domain that offers this service. Click **Import** to create the associated import services.

IDE Extensions

If you click **Cancel**, the WTC service is deleted and the WTC Imported Services dialog displays.

If you click **Back**, the **Workstation Address** dialog displays.

Click **Finish** when you have completed importing all required services and the configured WTC service will deploy. The **WTC Import Services** dialog displays. The services you have imported should display in the list.



For more information about importing WTC services, see "Setting Up a WebLogic Tuxedo Connector (WTC) Connection."

How Do I: Create a Logical Business Control Supporting Multiple Tuxedo Services

A web service can access a single Tuxedo service or many Tuxedo services. Each Tuxedo Control file that you add to your web service maps to one or many Tuxedo service(s). To create a Tuxedo Control that supports multiple Tuxedo services, you must edit the JCX file in the Design View or Source View.

In the **Design View**:

1. Right-click and select **Insert Method** from the menu.
2. Set the appropriate properties, such as service name, service type, etc. in the **Property Editor**.

In the **Source View**:

1. Define new methods.
2. Add the @jc:tuxedo tag and the service-type attribute.

Related Topics

- [How Do I: Create a Java Application That Uses a Tuxedo Service](#)

How Do I: Create a Tuxedo Control That Uses Queue Type Services

When the service type is queue, the Tuxedo Control uses Tuxedo queuing services via `tpenqueue` and `tpdequeue`. As in other interaction styles, the data from the application to the Tuxedo queue is mapped based upon the method signature and the buffer type. The determination of whether a method is an enqueue, a dequeue, or both is determined by the method signature.

- If the method has input parameters, an enqueue is performed.
- If the method has a return type other than void, a dequeue is performed.
- If the method has both input parameters and a non-void return type, then an enqueue is performed followed by a dequeue.

Warning: If an enqueue and dequeue is performed in the same queue with transaction attribute set to Supports, you will not see the enqueue message until the transaction commits. The dequeue will dequeue another message if one is in the queue or hang until the message is in the queue. If the dequeue hangs, you should turn off transaction support so that the dequeue can proceed.

Enqueue Behavior

The name of the queue space used to enqueue messages is determined by the `send-queue-space` attribute of the Tuxedo Control `jc:tuxedo` tag. The `send-queue-name` attribute determines the queue name. The `reply-queue-name` determines the name of the reply queue to associate with the message. The `failure-queue-name` gives the name of the failure queue to associate with the message.

Note: For information about using `tmqforward`, refer to *Services Using `tmqforward`*.

Dequeue Behavior

For dequeuing messages, the `receive-queue-space`, if set, determines the queue space; otherwise, the `send-queue-space` is used. This simply allows for you to only specify one queue space and it is used for both the enqueue and dequeue. The `receive-queue-name` attribute determines the queue name.

Create a Tuxedo Control for Queue Service Types

To specify using a queue service type you will create a Tuxedo Control file and then add a method to the control file.

1. Select the web service to which you want to use a Tuxedo service.
2. Click the **Design View** tab.
3. From the **Insert** menu, choose **Controls -> Tuxedo**. The **Insert Control – Tuxedo** dialog displays.

4. In Step 1 enter a variable name for the control.
5. In step 2 click the following radio buttons:

Click **Create a new Tuxedo Control to use** and enter a name for the JCX file in the **New JCX name** field.

Note: The Tuxedo Control you create must be in a folder within the Project. For example, the JCX name will be *folder.name* or *folder\name*.

6. In Step 3 specify the following Tuxedo service attributes:

Tuxedo Control Field	Description
Service Type	Specify queue which supports tpenqueue and tpdequeue.
Buffer	Select the type of buffer from the menu. This specifies the type of buffer that the Tuxedo

Type	<p>Control supports in the Tuxedo service.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> ◆ carray used when the data is an unspecified array of characters (byte array), any of which can be null. Tuxedo equivalent: CARRAY. ◆ fml used when the data contains named fields of simple types each with a maximum length of 65 K bytes. ◆ fml32 used when the data contains named fields including nested FML32 fields each having a maximum length of 4 GB. ◆ string used when the data contains a simple string as its only contents. ◆ view used to define C or COBOL structures for a Tuxedo application and to define an equivalent Java TypedView buffer for a WebLogic application. A view description file in which the fields and types that appear in the data structure are defined, must be available to client and server processes that use a data structure described in a VIEW typed buffer. Encoding and decoding are performed automatically if the buffer is passed between machines of different types. ◆ view32 is equivalent to VIEW but uses 32 bits for length and count fields, which allows for larger and more fields. ◆ xml used when data is an XML document. Tuxedo equivalent: XML for Tuxedo Release 7.1 and higher. <p>Note: For input XML using data dependent routing, remove all comments generated by the WebLogic Workshop test browser in the XML buffer. Depending on the location of the comments in the buffer, Tuxedo may have problems parsing the buffer.</p> <ul style="list-style-type: none"> ◆ x_common is a synonym for the view buffer type. An x_common buffer is represented by a Java class extending the TypedXCommon class. It is identical in semantics to a view buffer.
Buffer Type (cont.)	<ul style="list-style-type: none"> ◆ x_c_type is a synonym for the view buffer type. An x_c_type buffer is represented by a Java class extending the TypedXCType class. It is identical in semantics to a view buffer. ◆ x_octet is simply a byte array and is a synonym for the CARRAY buffer type in Tuxedo. ◆ none used when no input buffer is to be used in the request to the Tuxedo service. If the method signature contains input parameters, an error is generated. <p>The default is fml32.</p>
Send Queue Space	<p>If you have selected queue in the Service type field, the Send Queue Space field becomes active. Type the name of the queue space in which the sending queue name is located. This name is the WTC imported service name. This name should match the service name that is mapped in Create a WebLogic Tuxedo Connector (WTC) Service.</p> <p>If you are not sure of the name of the queue space, you can click Browse and a list of imported WTC service names display. If the service is listed in the WTC Imported Services dialog, you can select it and click OK. If the service you want is not listed, then the service has not been imported into WebLogic Server.</p> <p>To import a service from the WTC Imported Services dialog, click Import and the Services Exported by Tuxedo Application dialog displays.</p>

IDE Extensions

	For information about importing WTC services, see Import New Tuxedo Services.
Send Queue Name	If you have selected queue in the Service type field, the Send Queue Name field becomes active. Type the name of the queue to which messages should be sent.
Receive Queue Space	<p>If you have selected a queue in the Service type field, the Receive Queue Space field becomes active. Type the name of the queue space in which the receiving queue name is located.</p> <p>If you are not sure of the name of the queue space, you can click Browse and a list of imported WTC service names display. If the service is listed in the WTC Imported Services dialog, you can select it and click OK. If the service you want is not listed, then the service has not been imported into WebLogic Server.</p> <p>To import a service from the WTC Imported Services dialog, click Import and the Workstation Address dialog displays.</p> <p>For information about importing WTC services, see Import New Tuxedo Services.</p>
Receive Queue Name	If you have selected a queue in the Service Type field, the Receive Queue Name field becomes active. Type the name of the queue to receive queued messages.
Field Tables	<p>Type the name of the field table for the Tuxedo service when using fml or fml32 buffers. You should specify the field table when using fml or fml32 buffers that are sent or act as reply buffers from the called service. The name of the field table should match the classes generated with the mkfldclass or mkfldclass32 utilities.</p> <p>For buffer types fml and fml32, WTC needs the names and types of the fields the buffer can contain. These are described in field table classes created with mkfldclass and mkfldclass32 utilities.</p> <p>For more information about mkfldclass and mkfldclass32 utilities, see WebLogic Server Javadoc.</p>
Return View Classes	<p>Type the names of the view classes that correspond to the view buffer types that can be received from the Tuxedo service. This field is only required when the Tuxedo service returns a view or view32 buffers. These classes are independent from the Send View Class attribute which is used to specify the view or view32 class that represents the view buffer the Tuxedo service expects as an input.</p> <p>Note: The value of this attribute may be a list of classes so that a single Tuxedo service can respond with different VIEW buffers.</p> <p>For information about viewj and viewj32 utilities, see WebLogic Server Javadoc.</p>
Send View Class	If you have selected view or view32 in the Buffer type field, the Send View Class field becomes active. This attribute provides the name of the class that represents the view/view32 buffer the service is expecting as an input buffer. This view class is used to create the TypedView buffer.

7. Click **Create**. A new JCX file is created on the **Application** tab and a new icon appears in **Design View** of the web service.
8. In **Design View**, double-click the image of the newly created Tuxedo Control.

IDE Extensions

9. The new Tuxedo Control file (*controlname.jcx*) displays in the **Design View**.
10. Click the **Source View** tab. Edit the signature of this method in the JCX file to match the buffer contents expected for the Tuxedo service being called.

When using the Insert Wizard to create a new Java Control Extension, a single method is defined that corresponds to the **Service Type** and **Service Name** specified in the Insert Wizard. The signature and possibly the name of this method in the JCX file needs to be edited to match the name and buffer contents expected for the Tuxedo service being called. In addition, any other services that this control calls must have methods defined for them in this interface.

For queued services, the method name created by the wizard is enqueue, dequeue, or enqueueDequeue. You can change the wizard-created name.

- ◆ If send queue information is provided and no receive queue information is provided then the method inserted is enqueue.
 - ◆ If receive queue information is provided and no send queue is provided, then the insert method is dequeue.
 - ◆ If both send and receive queue information are provided, then the insert method is enqueueDequeue.
11. Select **File** → **Save** from the menu.
 12. Press **Ctrl+F4** to close the Tuxedo Control file (*controlname.jcx*.)

Note: For additional assistance with queue behavior, refer to sample code in `<BEA_HOME>/ext_components/Tuxedo/controls/Samples` directory.

Services Using tmqforward

If the queue is being serviced by tmqforward and the service that tmqforward calls generates a reply, a reply-queue-name must be specified for tmqforward from the **Property Editor** in the `jc:tuxedo` tag. If reply-queue-name is not specified in this situation, the reply is dropped.

Related Topics

- How Do I: Create a Java Application That Uses a Tuxedo Service
- @jc:tuxedo Tag

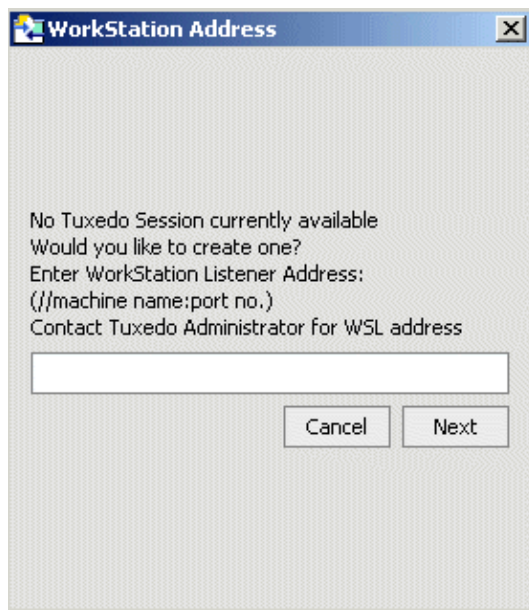
How Do I: Configure a WTC Service

WTC acts as a gateway between the Tuxedo environment and WebLogic environment; therefore, a WTC service definition must exist that maps to the Tuxedo service you want to use in your web service. You can use the **Insert Control – Tuxedo** dialog to configure WTC services as described in this topic.

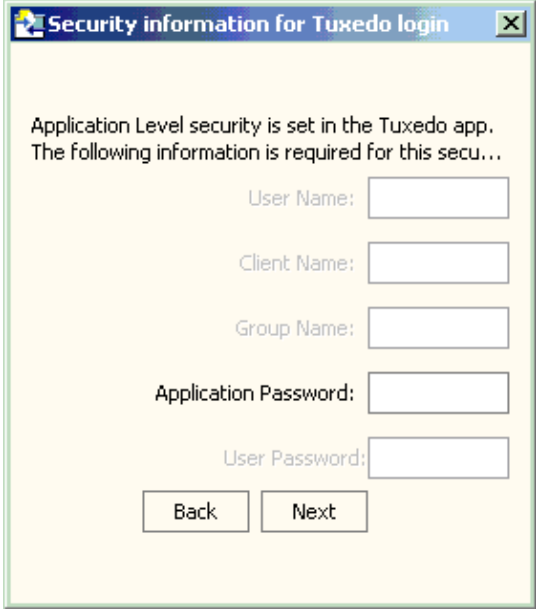
Note: An alternative way to configure WTC services is via the WebLogic Server Administration Console. See the WebLogic Tuxedo Connector Administration Guide for information on configuring WTC using the WebLogic Server Administration Console.

To configure a WTC service within Tuxedo Control you must access the **WTC Imported Services** dialog.

1. Within the web service or Java application, choose **Insert->Controls->Tuxedo**. The **Insert Control – Tuxedo** dialog displays. In the **Service Name** or **Receive Queue Space** fields, you can click **Browse** and see a list of available WTC imported services. If the service you want is not in the list, then the service has not been imported into WebLogic Server.
2. In step 2 of the **Insert Control – Tuxedo** dialog, select **Create a new Tuxedo Control to use** option. This will activate the Step 3 section of the dialog and allow you to browse the **Service Name** or **Receive Queue Space** fields. A list of available WTC imported services displays. If the service you want is not in the list, then the service has not been imported into WebLogic Server.
3. To configure a WTC service from the **WTC Imported Services** dialog, click **Configure** and the **Workstation Address** dialog displays. Domain information is required from the Tuxedo application.
4. Enter the network address of the workstation listener on a remote Tuxedo domain, such as a *machine_name:port number* combination *//mach1:5100*. This is the address of the Tuxedo application so the service names and domain information can be retrieved. Click **Next**. After the Workstation Address dialog, one of the following scenarios will occur depending on whether the domain information is determined from the Tuxedo application or from a default configuration.



- ◆ When you click **Next** on the **Workstation Address** dialog, if the Tuxedo domain has security turned on, the **Security Information for Tuxedo login** dialog displays. Enter the Tuxedo application information and click **Next**.



Security information for Tuxedo login

Application Level security is set in the Tuxedo app.
The following information is required for this secu...

User Name:

Client Name:

Group Name:

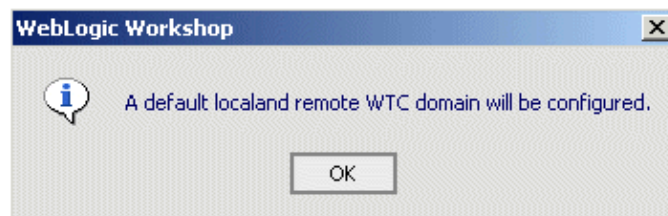
Application Password:

User Password:

Back Next

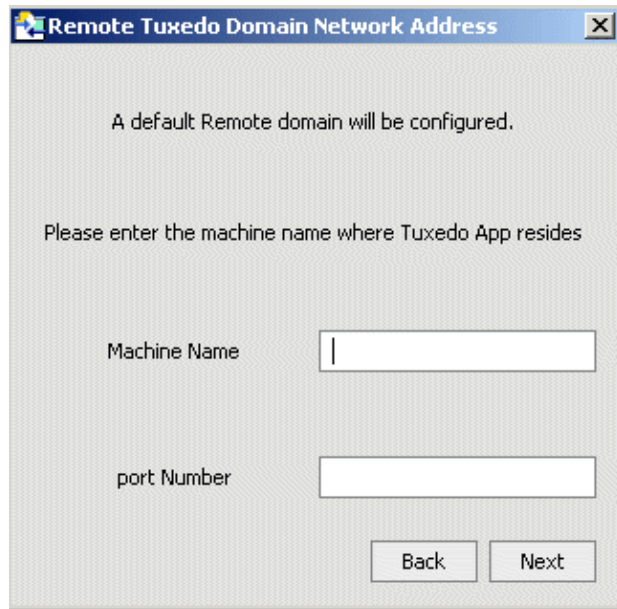
A connection to the Tuxedo domain is made and local and remote domain information is imported from the Tuxedo application. If the local and remote domains are configured in WTC successfully, a list of services exported by the Remote Tuxedo Domain displays. Proceed to step 5.

- ◆ If the machine where the Tuxedo application resides is not reachable and/or the domain information cannot be retrieved from Tuxedo, then a default local and remote domain is configured in WebLogic Server. A prompt displays stating that a default domain will be configured. Click **OK**.



The **Remote Tuxedo Domain Network Address** dialog displays. Enter the machine name and the port number where the Tuxedo application resides.

IDE Extensions



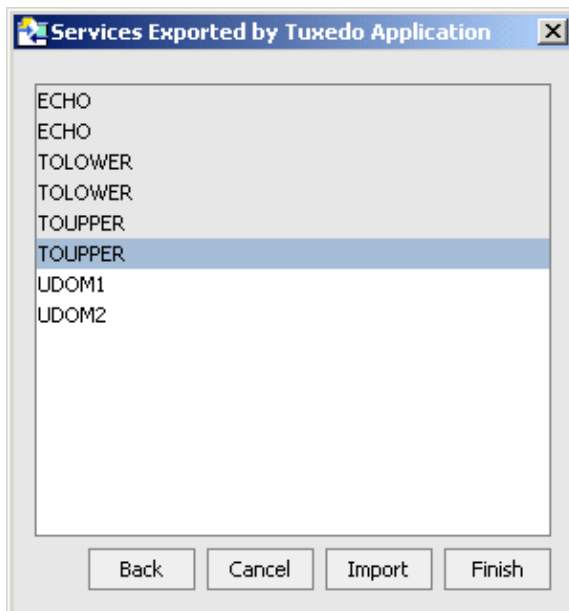
The image shows a Windows-style dialog box titled "Remote Tuxedo Domain Network Address". Inside the dialog, there is a message: "A default Remote domain will be configured." followed by "Please enter the machine name where Tuxedo App resides". Below this, there are two input fields: "Machine Name" and "port Number". At the bottom of the dialog, there are two buttons: "Back" and "Next".

If you click **Back**, the WTC server Mbean that was created is deleted from the WebLogic Server–WTC configuration.

If you click **Next**, a default Remote Domain is created. However, because a default Remote Domain was configured, no services are imported. Proceed to step 6.

5. If the domain information was retrieved from the Tuxedo application and there is more than one remote domain configured, then a list of the Tuxedo Remote domains displays. Select one or more of the remote domains from the list and click **OK**. Similarly, if there is more than one Local domain configured in the Tuxedo application, then a list of Local domains displays. Select the Local domain that offers the service you want and click **Select**. After selecting the Local and Remote domains, the list of exported services by the Tuxedo application displays.

After the Workstation Address dialog, if there is only one Local and Remote domain configured in the Tuxedo application, the list of exported services by the Tuxedo application displays. Select the service you want.



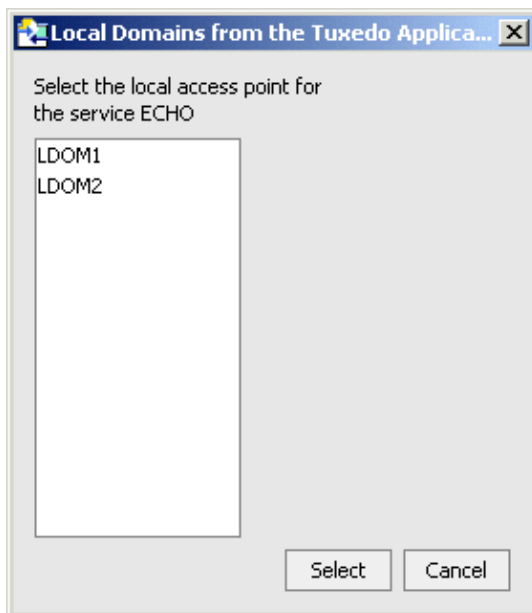
Click **Import** to create the associated import services.

If you click **Cancel**, the WTC service is deleted and the WTC Imported Services dialog displays.

If you click **Back**, the **Workstation Address** dialog displays.

Click **Finish** when you have completed importing all required services and the configured WTC service will deploy.

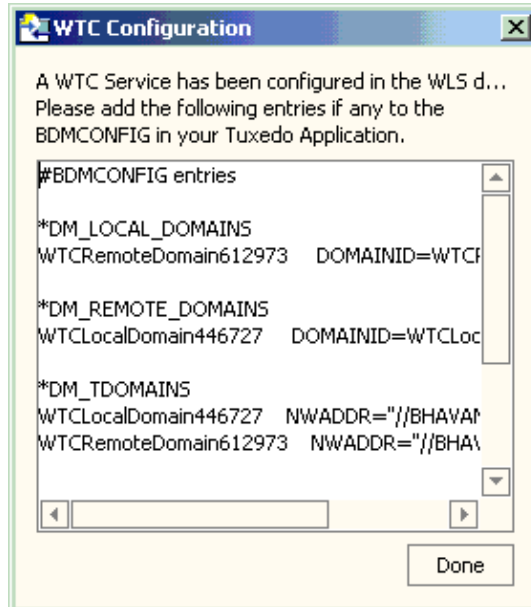
After selecting the services to export, if there is more than one Local domain configured in WTC, a dialog displays prompting you to select the Local domain that will be exporting the previously selected service. Click **Select** after choosing the Local domain.



6. The **WTC Configuration** dialog displays.

IDE Extensions

- ◆ If all the information to configure WTC was imported from the Tuxedo application, an empty **WTC Configuration** pane displays. Click **Done** to complete the WTC service configuration process.
- ◆ If a default domain is configured, the following **WTC Configuration** dialog displays with configuration information for *DM_LOCAL_DOMAINS, *DM_REMOTE_DOMAINS and *DM_TDOMAINS. Copy the entries in the pane and paste them into the BDMCONFIG file in your Tuxedo application. Be sure and save the updates to your BDMCONFIG file. Click **Done** to complete the WTC service configuration process.



The **WTC Import Services** dialog displays. The services you have imported should display in the list. It may be helpful to review step 5 when selecting services.

Note: When you have completed the configuration, the **WTC Imported Services** dialog displays. If you want to configure a WTC service when using an existing Tuxedo Control extension, click **Back** and the **Insert Control – Tuxedo** dialog displays. Click **Cancel** on the **Insert Control – Tuxedo** dialog to abort the creation of a new Tuxedo Control extension.

7. If you have security configured in your WebLogic Server domain, you should set up security for WTC using the WebLogic Server Administration Console. For information on setting up WTC security, refer to the WebLogic Tuxedo Connector Documentation.

Related Topics

- WebLogic Tuxedo Connector Documentation
- Create a WebLogic Tuxedo Connector (WTC) Service

How Do I: Map Java Data Types/FML and VIEW Buffers

In order to map Tuxedo buffers to Java data types, you must be familiar with the Tuxedo header files and buffer types of the Tuxedo service as mentioned in "What You Must Know About Tuxedo."

The following topics explain how to map Tuxedo buffers to Java data types.

Topics Included in This Section

- How Do I: Map Data Types
- How Do I: Map Java Data Type Buffers to FML and VIEW
- How Do I: Map FML and VIEW Buffers to Java Data Types
- How Do I: Map Non-Fielded Buffers

Related Topics

- What You Must Know About Tuxedo
- FML and VIEW Features
- Managing Typed Buffers

How Do I: Map Data Types

When you are requesting or providing data from a Tuxedo buffer using the Tuxedo Control, you will need to understand how Java data types map to Tuxedo buffers. The Tuxedo Control does the mapping of the Java data types into and out of the Tuxedo buffers that are sent and received. The names of the arguments or fields in complex objects match up with their corresponding field names for FML/ FML32 and VIEW/VIEW32 buffers. Other buffer types simply concatenate all the fields into the buffers. Arrays map to occurrences of fields for FML and to VIEW arrays for VIEW.

The Tuxedo Control supports the following ATMI field types and maps them to the specified Java data types.

Java Data Type	Field Type
Short or short	FLD_SHORT
Integer or int	FLD_LONG
** Decimal	FLD_DECIMAL
Character or char	FLD_CHAR
Float or float	FLD_FLOAT
Double or double	FLD_DOUBLE
String	FLD_STRING
Byte[] or byte[]	FLD_CARRAY
* object	FLD_FML32

* This represents an instance of a class.

** This field type is only supported for View/View32 buffers.

Note: Decimal is a Java equivalent to the Tuxedo packed decimal type, dec_t. It is contained in the weblogic.wtc.jatmi package.

How Do I: Map Java Data Type Buffers to FML and VIEW

For type mapping to occur properly, you must edit the JCX file to define a method that matches field names from complex Java data types to the FML/FML32 and VIEW/VIEW32 buffers. This means that the control matches parameter names and Java member names to fielded buffer field names. This field matching allows the control to build the correct request/response buffers that are expected. This field matching defines what data the control places in the buffer. The service defines or determines the buffer setup. In other words, the service type defines the communication model which determines request buffers and response buffers.

Communication Model	Service Type
Synchronous Request/Response (tpcall)	service
Request without Response (tpacall)	oneway
Queued (tpenqueue)	queue

The fields defined in the method in the control extension must match the fields defined in the field tables or VIEW class. For example, if the control extension defines a method of:

```
String anOp (String s1, Float f1, Short s2);
```

At runtime when the anOp method is called, assuming a fielded buffer type is used, the control accesses the field tables for fields named "s1", "f1", and "s2". For the fields that are found, the control places the parameter with the same name into the buffer as that field.

For complex Java data types such as beans or Java classes with only data members, this field matching process is more complex. The control performs the same matching algorithm as previously described, but it is performed recursively on each of the complex Java data types. For example, classes as follows:

```
static public class Name {
    String      First;
    String      Last;
    Character    MI;
}

static public class Person {
    Name        CompleteName;
    Float       Age;
    Integer     Weight;
}

static public class People {
    Person[]    PEOPLE;
}
```

and an operation signature as follows:

```
People addPerson(Person aPerson);
```

In the previous example, the control at runtime looks for a field in the field table by the name of aPerson. If the control finds aPerson and it is an embedded type field such as FLD_FML32, the control creates the

embedded buffer and begins filling the embedded buffer with the fields from the value passed as `aPerson`. Otherwise, because `Person` is a complex type, the control starts looking for fields matching the members of the complex type `Person`. Thus the control looks for a field named `CompleteName`. Again, if `CompleteName` matches an embedded buffer type, the control creates the embedded buffer and fills it. Next the control looks for a field in the field tables named `First`. If a match occurs, the control places the Java field `aPerson.Name.First` in the buffer with the field ID associated with the name `First`. Next it looks for a field named `Last`. The control performs a depth first traversal of the passed parameters and then looks for matching fields in the field table.

The control performs a similar process to pull the returned buffer apart. In the previous example, the control examines the return type of the method. As it is an array of a complex type, the control looks for occurrences of a field named `PEOPLE`, and if found, determines the number of occurrences and creates an array of `Person` of that size. If no field of name `PEOPLE` is found, it looks for occurrences of fields `CompleteName`, `Age`, and `Weight`. it then creates an array of `Person` large enough to hold the least number of occurrences of those fields found. Then the control starts populating the array by pulling fields out of the buffer that match the data member names of the `Person` and creating instances of `Person`.

Using JavaBean–style Data Accessors

A JavaBean is a Java class that hides its state behind methods that conform to certain naming standards. This Java class uses private data fields with public accessors. To determine the fields for a class, the control constructs a list of all public fields and then adds any additional fields found by examining the public methods of the class looking for methods of the following form:

```
public void setFIELD(type val);
public void setFIELD(type[] val);
public type getFIELD();
public type[] getFIELD();
```

In the previous code example,

- *FIELD* is the name of the field/property
- *type* is the Java type of the field

The JavaBean naming conventions require that a field name begins with a lowercase character. By default, the Tuxedo Control will follow the JavaBean naming conventions. For example, a field called *myField* would have a getter named *getMyField* and a setter named *setMyField*. To convert an accessor name to a field name, the Tuxedo Control strips the get or set prefix and converts the first character of the resulting field name to lowercase, *myField*.

For some existing Tuxedo field tables that have field names that begin with an uppercase character, this naming convention conversion is a problem. To resolve this naming issue, use the `accessor-name-conventions` attribute in the `@jc:tuxedo` tag. For more information about `@jc:tuxedo` tag, refer to `@jc:tuxedo` Tag.

JavaBean Accessors allow you to control movement of data to and from the JavaBean. Using public data members, the control directly accesses the fields and the ability to control that movement of data is limited to the coercion capabilities of the control. An accessor allows user–provided methods to perform arbitrary checks or data modification before a field's data is retrieved or set. An example of where this may be helpful is in retrieving a string field from a buffer that was space padded as in a `VIEW` buffer generated by `COBOL`.

A custom-written setter could trim the field before storing the string.

Data Type Mapping and Coercion

When the types of fields with the same name do not match exactly, coercion specifications determine the mapping behavior. You can specify the mapping-strictness attribute in the `jc:tuxedo` tag using the **Property Editor**. Possible coercion values for mapping-strictness are:

- **normal** is the most expected type of coercion. If the field is a numeric field type and the Java Number class can provide the appropriate coercion, then it is used. Exceptions are raised only if the coercion required is out of the ordinary, such as trying to convert the string "one" to a numeric field. Missing fields and some type coercion problems are logged as issues.
- **strict** only allows matching types with little or no coercion provided. All disallowed combinations cause a `ControlException`. Few issues are logged as most problems result in an exception.
- **loose** attempts any feasible coercion. For example, a string containing "true" or "The" when mapped to a Boolean field will both result in a value of true as the control will only look at the first character of the string. This setting can yield uncertain results. Some data may not make it through the control and as a result may be lost in mapping. Few errors raise an exception and most problems are logged as issues.

The default is **normal**.

In the previous example from How Do I: Map Java Data Type Buffers to FML and VIEW, the field in the field table associated with Weight might be a String field. In this case, for mapping strictness of either **loose** or **normal**, the control converts the Integer data member of the Person instance to a String before trying to place the data in the buffer. That is a normal and reasonable coercion to perform, so the control performs that coercion when the mapping strictness is anything but **strict**. For a mapping strictness of **normal**, the control places an issue in the issue list to indicate that it had to coerce some data. With a mapping strictness of **strict**, the control raises an exception.

How Do I: Map FML and VIEW Buffers to Java Data Types

The mapping of FML and VIEW buffers to Java data types is performed in a similar manner as that described in How Do I: Map Java Data Type Buffers to FML and VIEW. The mapping is controlled by the fields or data members of the objects specified as the method's return type and not by the fields in the FML or VIEW buffer. Each field in the return type is retrieved from the FML or VIEW buffer based on matching the object's field name to the FML field name or VIEW class. The return type must be an instantiable type, a class not an interface or even abstract class. The control uses that return type as the type it will instantiate. Because interfaces or abstract classes cannot be instantiated, they are not allowed as the return type.

How Do I: Map Non-Fielded Buffers

When mapping to or from buffer types other than FML or VIEW, the control fills the buffer contents with the arguments of the method call or the return type is populated from the contents of the buffer. Type coercion occurs where possible. If the return type for a method is defined as Long and a STRING buffer returned from the service, a new Long is created passing the contents of the STRING buffer to the constructor.

Getting Started with the Tuxedo Control Samples

The Tuxedo Control includes various samples and sample code to assist you in understanding and developing control extensions to access the Tuxedo services for your business needs.

Be sure to explore the contents of the samples files in the <BEA_HOME>/weblogic81/samples/partners/Tuxedo directory for specific samples. The Tuxedo directory contains three subdirectories: setup, TuxSampleApp, and TuxSampleCode.

- The setup directory has a subdirectory for each Tuxedo sample application, for example, setup/bankapp and setup/simpapp. Each application subdirectory contains a README file explaining how to set up the Tuxedo sample application to communicate with WebLogic Server and how to use the control sample for that application. The directory also contains a DMCONFIG file, a replacement UBBCONFIG file, and possibly other files for use or editing with the samples.
- The TuxSampleApp directory contains a full WebLogic Workshop application with several Web projects that demonstrate the use of the Tuxedo Control with different types of Tuxedo services and different types of Tuxedo data buffers.
- The TuxSampleCode directory contains several subdirectories, each containing sample files to use for creating different types of WebLogic Workshop applications that communicate with Tuxedo.

The following topics explain how to get started with the Tuxedo Control and cover simpapp and bankapp samples.

Note: Other samples and sample code are available for assistance but not documented.

Topics Included in This Section

- Simpapp Sample
- Bankapp Samples

Related Topics

- Getting Started With Java Controls

Simpapp Sample

A sample that demonstrates use of the Tuxedo Control by exposing the Tuxedo service, TOUPPER in the Tuxedo simpapp application. TOUPPER is a Tuxedo service that converts text to uppercase. The simpapp sample is delivered as one of the Web Service projects within the TuxSampleApp application.

Concepts Demonstrated By This Sample

Use of the Tuxedo Control in a simple web service.

To Run the Sample

Before running the simpapp.jws sample, be sure you have accomplished the tasks defined in "Before You Begin."

1. Start the Tuxedo server that advertises TOUPPER in the simpapp application.
2. Start WebLogic Workshop. From the **Start** menu, choose **Programs -> WebLogic Platform 8.1 -> WebLogic Workshop**.
3. From the **File** menu, select **Open -> Application** to open an existing Application in WebLogic Workshop. Click **Browse** and select
<BEA_HOME>/weblogic81/samples/partners/Tuxedo/TuxSampleApp/TuxSampleApp.work.
4. Expand the simpleWS node in the **Application** pane by clicking the + sign in front of simpleWS. When you expand the simpleWS node, you will see a **controls** folder that contains simpapp.jcx. The simpleWS node also contains Simpapp.jws.
5. Open the **controls** folder and double-click the simpapp.jcx file.
6. Click the **Source View** tab. Verify the source code has the following method signature.

```
public String toUpper (String msg) ;
```

7. Select Simpapp.jws from the **Application** pane and click the **Design View** tab. Verify that **remoteSvc** specifies **toUpper**.
8. From the **Debug** menu, select **Start**.
9. When the test browser displays, enter text in the **msg** field and click the **toUpper** button.
10. The results display at the bottom of the browser window when the browser refreshes.

Related Topics

- Tutorial for simpapp, a Simple C Application
- JCX File

Bankapp Samples

Bankapp sample is available as part of the Tuxedo product. The following bankapp samples may be used with Tuxedo Control.

- Bankapp Web Service
- Bankapp Portal
- Bankapp Workflow
- Bankapp Pageflow

Related Topics

- Tutorial for bankapp, a Full C Application

Bankapp Web Service

This sample shows using the Tuxedo services of the bankapp sample as shipped with Tuxedo. It provides BankCtrl.jcx, a pre-configured Java Control Extension, that uses the DEPOSIT, WITHDRAWAL, TRANSFER, INQUIRY, OPEN_ACCT, and CLOSE_ACCT services. For each of these services, an operation is defined in the Java Control Extension interface providing an easy to use mechanism for accessing these services.

Concepts Demonstrated By This Sample

Defining the method signatures in the JWS and JCX files to ensure correct mapping among the incoming web document, the application's Java data types, and the Tuxedo buffer.

To Run the Sample

Before running the bankapp sample, be sure you have accomplished the tasks defined in "Getting Started with the Tuxedo Control." This includes making sure you have modified the DMCONFIG and UBBCONFIG files to run the bankapp sample. Also, refer to the README file in <BEA_HOME>/weblogic81/samples/partners/Tuxedo/setup/bankapp directory. This README file explains what you must do to set up bankapp on the Tuxedo side.

1. Start the Tuxedo server that advertises the services in the bankapp application.
2. Start WebLogic Workshop. From the **Start** menu, choose **Programs -> WebLogic Platform 8.1 -> WebLogic Workshop**.
3. From the **File** menu, select **New -> Application** to create a new Application in WebLogic Workshop.
4. In the New Application dialog, select **All** in the left pane and select **Empty Application** in the right-pane. Type bankapp in the **Name** field. Browse to an appropriate temporary directory for this sample.
5. Click **Create**.
6. Right-click on the bankapp in the **Application** tab. Select **New -> Project**.
7. In the **New Project** dialog, select **Web Services** in the left pane. Type TestWS in the **Project Name** field.
8. Click **Create**.
9. Right-click on TestWS project in the **Application** tab and select **New -> Folder**.
10. Create a folder labeled **controls**.
11. Expand the TestWS node in the **Application** pane by clicking the + sign in front of TestWS. Right-click on TestWS and select **New -> Web Service**. A file creation wizard displays. Type bankapp.jws for the file name and click **Create**.
12. Import the BankCtrl.jcx and bankflds.java files to the **controls** folder. Right-click on the **controls** folder and select **Import**. In the **Import Files** dialog, browse to <BEA_HOME>/weblogic81/samples/partners/Tuxedo/TuxSampleCode/bankapp/BankWS/controls directory and select BankCtrl.jcx and bankflds.java and click **Import**.
13. Open the **controls** folder and double-click the BankCtrl.jcx file.
14. Click the **Source View** tab. Edit the source code with the following change.

Change the method signature (source line) to:

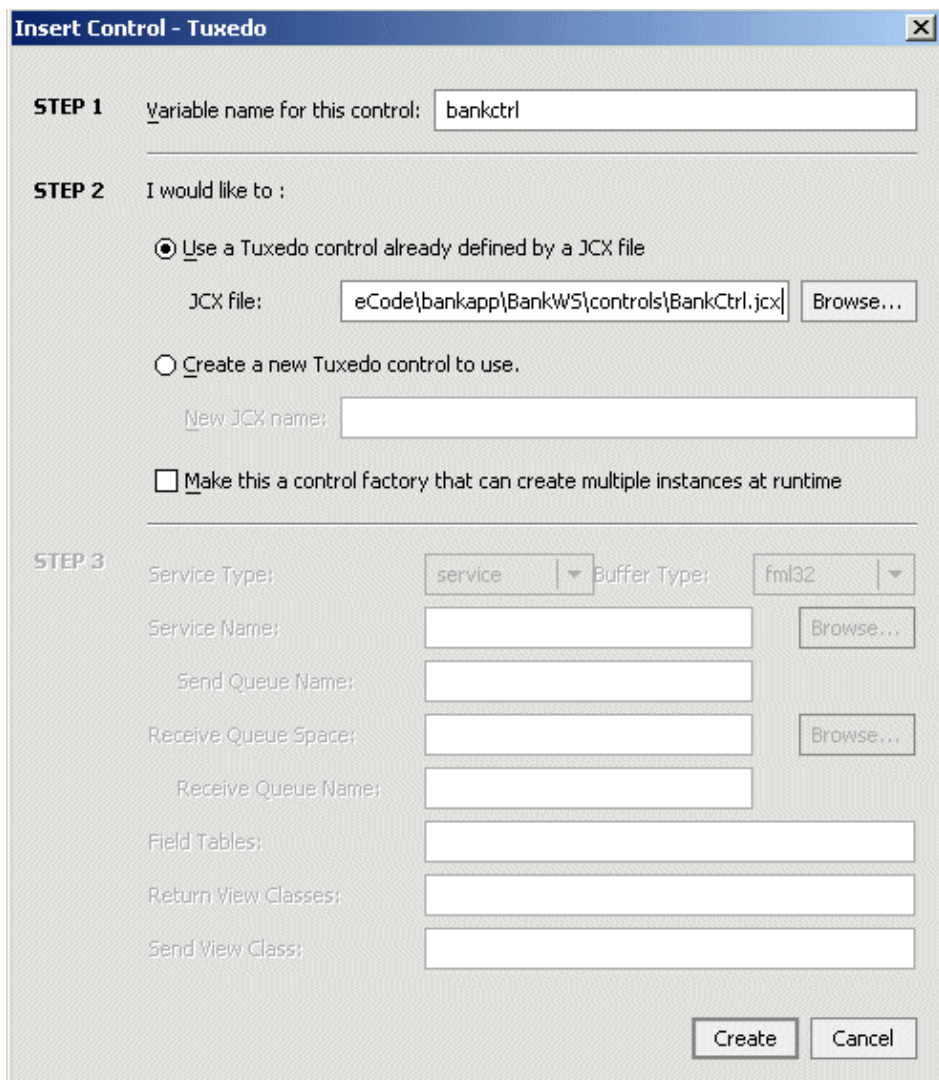
```
public void OPEN_ACCT ();
```

IDE Extensions

to:

```
public OPEN_ACCT_RETURN OpenAccount (String LAST_NAME,  
String FIRST_NAME,  
String MID_INIT,  
String SSN,  
String ADDRESS,  
String PHONE,  
String ACCT_TYPE,  
Integer BRANCH_ID,  
String SAMOUNT);
```

15. Select bankapp.jws from the **Application** pane and click the **Design View** tab.
16. From the **Data Palette**, select **Controls** -> **Add** -> **Tuxedo**. The **Insert Control – Tuxedo** dialog displays.



The image shows the 'Insert Control - Tuxedo' dialog box, which is divided into three steps. Step 1 is 'Variable name for this control:' with a text field containing 'bankctrl'. Step 2 is 'I would like to :', with two radio buttons: 'Use a Tuxedo control already defined by a JCX file' (selected) and 'Create a new Tuxedo control to use.'. The 'Use a Tuxedo control already defined by a JCX file' option has a 'JCX file:' label and a text field containing 'eCode\bankapp\BankWS\controls\BankCtrl.jcx', with a 'Browse...' button to its right. The 'Create a new Tuxedo control to use.' option has a 'New JCX name:' label and an empty text field. There is also a checkbox labeled 'Make this a control factory that can create multiple instances at runtime'. Step 3 is 'Service Type:' with a dropdown menu showing 'service', and 'Buffer Type:' with a dropdown menu showing 'fml32'. Below these are several text fields: 'Service Name:', 'Send Queue Name:', 'Receive Queue Name:', 'Field Tables:', 'Return View Classes:', and 'Send View Class:'. There are 'Browse...' buttons next to the 'Service Name:', 'Receive Queue Name:', and 'Send View Class:' fields. At the bottom right are 'Create' and 'Cancel' buttons.

In Step 1 enter a variable name for the control.

In step 2 click **Use a Tuxedo Control already defined by a JCX file** to use an existing Tuxedo Control file. Browse to controls/BankCtrl.jcx file.

IDE Extensions

Click **Create** to add the methods to the control.

17. Edit the JWS file by inserting the control methods. In the Data Palette, expand the node for the control variable you created by clicking on the + sign. With the Design View of bankapp.jws in the central pane, drag and drop each of the six control methods onto the design pane. WebLogic Workshop creates web service methods to invoke the control methods.
18. Add the following code to the bankapp.jws to change the Transfer service. The changes allow the XML sent to the web service and the XML returned from the service to reflect more clearly the nature of the data being sent. The Tuxedo input buffer has a single field with two occurrences to represent the source and target account IDs. Similarly, the return buffer has a single field with two occurrences to represent the source and target account balances. These multiple occurrence fields are most naturally represented as arrays in Java. The arrays are translated into separately named Java fields to make the XML representation clearer.

Enter a new nested class declaration for the return value of Transfer:

```
public static class TransferBalances
{
    public String FROM_BALANCE;
    public String TO_BALANCE;

    public TransferBalances (String fromBal, String toBal)
    {
        FROM_BALANCE = fromBal;
        TO_BALANCE = toBal;
    }
}
```

Change the definition of Transfer created by WebLogic Workshop to the following:

```
public TransferBalances Transfer(Integer fromAcct, Integer toAcct, String SAMOUNT)
{
    controls.BankCtrl.Accounts inputAccts;
    controls.BankCtrl.Balances retval;
    TransferBalances tbal;

    inputAccts = new controls.BankCtrl.Accounts(fromAcct, toAcct);
    retval = bankSvc.Transfer(inputAccts, SAMOUNT);
    tbal = new TransferBalances(retval.SBALANCE[0], retval.SBALANCE[1]);
    return tbal;
}
```

19. From the **Debug** menu, select **Start**.
20. When the test browser displays, enter text in the fields and click the **OpenAccount** button.
21. The results display at the bottom of the browser window when the browser refreshes.

Related Topics

- Tutorial for bankapp, a Full C Application

Bankapp Portal

This sample shows how to use the Tuxedo Control extension to call bankapp from a portal application.

This sample application uses the Tuxedo bankapp services DEPOSIT, WITHDRAWAL, and INQUIRY in a WebLogic Portal application.

To use the sample, you must have WebLogic Portal installed. The sample application needs to be targeted to a Portal-enabled domain. Make sure you configure WTC in this domain as instructed for the Bankapp Web Service sample. To use this sample, you must create a new Portal application and import the sample files into the new application, as described below. Once the application is setup, built and deployed, you can use the Workshop test browser to test it. With the file bank.portal open in the Workshop, select **Portal -> Open Current Portal** from the toolbar. You should see a web page open on the INQUIRY service portlet. Links at the top of the page take you to the other services.

Concepts Demonstrated By This Sample

Calling a Tuxedo application, bankapp, and using bankapp services from a portal application.

Sample Code

A sample portal file is available in `<BEA_HOME>/weblogic81/samples/partners/Tuxedo/TuxSampleCode/bankPortal` to assist you in understanding how to specify access to Tuxedo services from a portal application.

To use the sample, you must complete the following tasks.

1. You must have WebLogic Portal installed.
2. The sample application needs to be targeted to a Portal-enabled domain.
3. Make sure you configure WTC in this domain as instructed for the Bankapp Web Service sample.
4. Create a new portal application. In the WebLogic Workshop IDE select **File->New->Application**.
5. In the New Application dialog, select **Portal** in the left pane and select **Portal Application** in the right-pane. Type a name for the portal application in the **Name** field and be sure the application is targeted to a portal domain, such as the samples portal domain in `<BEA_HOME>/samples/domains/portal`.
6. Click **Create**.
7. Create a new portal project. In the **Application** tab, right-click on the name of the application you created in the previous steps. Select **New ->Project**.
8. In the **New Project** dialog, select **Portal** in the left pane and Portal Web Project in the right pane. Type a project name in the **Project Name** field.
9. Click **Create**.
10. Import the bankPortal sample files. In the **Application** tab, right-click the name of the Portal Web Project you created and select **Import**. In the **Import** dialog, browse to `<BEA_HOME>/weblogic81/samples/partners/Tuxedo/TuxSampleCode/bankPortal/bankPortal`. Select each of the files and folders in this directory and import them into your portal project.
11. Insert the Tuxedo Control into your application. Open the file Controller.jspf in the Portal Web Project you created. Make sure you are in the **Design View**. Select **Insert->Controls->Tuxedo** from the main tool bar. The **Insert Control - Tuxedo** dialog displays.
 - a. In step 1 of the dialog, enter myControl in the **Variable name** field.

IDE Extensions

- b. In step 2 of the dialog click **Use a Tuxedo Control already defined by a JCX file** to use an existing Tuxedo Control file. Browse to controls/BankCtrl.jcx file and click **Select**.
 - c. Click **Create** in the **Insert Control – Tuxedo** dialog.
12. You should now be able to build, deploy and test your portal application.

Once the application is setup, built and deployed, you can use the WebLogic Workshop test browser to test it. With the file bank.portal open in WebLogic Workshop, select **Portal –>Open Current Portal** from the toolbar. You should see a web page open on the INQUIRY service portlet. Links at the top of the page take you to the other services.

Related Topics

- Tutorial for bankapp, a Full C Application
- Pageflow and JSP Samples

Bankapp Workflow

This sample application uses the Tuxedo bankapp service, DEPOSIT, in a Client Request with Response workflow conversation.

Concepts Demonstrated By This Sample

Calling a Tuxedo application, bankapp, and using bankapp services from a WebLogic Integration workflow.

Sample Code

A sample WebLogic Integration workflow is available in `<BEA_HOME>/weblogic81/samples/partners/Tuxedo/TuxSampleCode/bankBusinessProcess` to assist you in understanding how to specify access to Tuxedo services from a workflow.

This sample application uses the Tuxedo bankapp service, DEPOSIT, in a Client Request with Response workflow conversation.

To use this sample, you must complete the following tasks.

1. You must have WebLogic Integration installed.
2. The sample application needs to be targeted to an Integration-enabled WebLogic domain.
3. Make sure you configure WTC in this domain as instructed for the Bankapp Web Service sample.
4. Create a new Process application. In the WebLogic Workshop IDE select **File->New->Application**.
5. In the **New Application** dialog, select **Process** in the left pane and **Process Application** in the right pane. Type a name for the process application in the **Name** field. Be sure the application is targeted to an Integration domain, such as the samples Integration domain in `<BEA_HOME>/weblogic81/samples/domains/integration`.
6. Click **Create**.
7. Import the sample files into your new application. First, select the **Schemas** node in the Application Pane, then right-click and select **Import**. Browse to `<BEA_HOME>/weblogic81/samples/partners/Tuxedo/TuxSampleCode/bankBusinessProcess/bankBusinessPr` and select both xsd files in that directory for Import.
8. Next, select the **Process Web Project** node created by Workshop (the name should be your application's name followed by Web, for example myProcessAppWeb). Right-click and select **Import**. Browse to `<BEA_HOME>/weblogic81/samples/partners/Tuxedo/TuxSampleCode/bankBusinessProcess/bankBusinessPr` and select the bankBusinessProcessTransformations and controls folders for Import.
9. Finally, select the processes folder under the **Process Web Project**, right click and select **Import**. Browse to `<BEA_HOME>/weblogic81/samples/partners/Tuxedo/TuxSampleCode/bankBusinessProcess/bankBusinessPr` and select the bankBusinessProcess.jpd file for Import.
10. Now delete the default process.jpd file created by WebLogic Workshop. Right-click on process.jpd in the processes folder and select **Delete**.
11. Insert the Tuxedo Control into your application. Open the file bankBusinessProcess.jpd and make sure you are in the **Design View**. In the **Data Palette Pane**, select **Controls->Add->Tuxedo**. The **Insert Control – Tuxedo** dialog displays.
 - a. In step 1 of the dialog, enter bankSvc in the **Variable name** field.
 - b. In step 2 of the dialog click **Use a Tuxedo Control already defined by a JCX file** to use an existing Tuxedo Control file. Browse to controls/BankCtrl.jcx file and click **Select**.

IDE Extensions

- c. Click **Create** in the **Insert Control – Tuxedo** dialog.
12. Open the file `bankBusinessProcess.jpdl` in WebLogic Workshop and then start the test browser.
Choose the tab to test the SOAP message.
13. Replace the text in the **ACCOUNT_ID** and **SAMOUNT** fields in the test XML with a valid bankapp account identifier, for example 10001, and dollar amount.

Related Topics

- Tutorial for bankapp, a Full C Application
- Tutorial: Building Your First Workflow

Bankapp Pageflow

This sample application uses the Tuxedo bankapp services DEPOSIT, WITHDRAWAL, OPEN_ACCOUNT, CLOSE_ACCOUNT, INQUIRY, and TRANSFER in a WebLogic Workshop pageflow. The bankPageFlow sample is delivered as a Web Project within the TuxSampleApp application.

Concepts Demonstrated By This Sample

Calling a Tuxedo application, bankapp, and using bankapp services from within a pageflow.

Sample Code

1. Start the Tuxedo server that advertises the bankapp service.
2. Start WebLogic Workshop.
3. From the **File** menu, select **Open->Application**. Click **Browse** and select `<BEA_HOME>/weblogic81/samples/partners/Tuxedo/TuxSampleApp/TuxSampleApp.work`.
4. Install WebLogic Workshop's page flow libraries and resources into the sample application. Select the bankPageFlow node in the Application Pane, then right-click and select **Install->Web Project Libraries**. A pop-up dialog displays. Be sure that the check boxes to the **Libraries** and **Resources** are both selected and click **Install**.
5. Build the project. To do this right-click the bankPageFlow node in the **Application Pane** and select **Build bankPageFlow**.
6. Open the file bankPageFlow/bankPageFlow/bankPageFlowController.jspf and start the WebLogic Workshop test browser.
7. You should see a web page with links to each of the six bankapp FML services. Each page lets you enter some information and invoke a Tuxedo service. Each page also has a Home link that takes you back to the original page.

Related Topics

- Tutorial for bankapp, a Full C Application
- Tutorial: Your First Pageflow Application
- Pageflow and JSP Samples

Troubleshooting Tuxedo Control

The following sections provide troubleshooting information for the Tuxedo Control.

- Development-time Errors
- Run-time Errors

Development-time Errors

Field Tables and View Buffers

Problem: In the Source View of the control, if there is a red line under the field table classes or view classes, this indicates that WebLogic Workshop cannot load the named classes.

Cause/Resolution: The error is typically caused by either not having compiled the field table or view table or not having updated/added the package name in the generated source file.

Buffer-type Mismatch

Problem: If buffer-type attribute is set to fml32 and the field-tables attribute specifies an FML table and mapping-strictness is strict, a development-time error similar to the following results:

```
simpapp.jcx.20:Conversion of String to Short disallowed, name=StringField  
SUGGESTION: Check argument and field types.
```

Resolution: Open <field-table>.java file and check whether it is FldTbl or FldTbl32.

Run-time Errors

TPETTRAN

Problem: TPETTRAN error is typically caused by problems in propagating the transaction from WebLogic to Tuxedo. The most common causes of this are:

- The application has not performed a `tpopen()`
- No TMS is running in the Tuxedo domain
- The TLOG file for the domain gateway hasn't been created.

Cause/Resolution: If the Tuxedo application does not need to participate in the same transaction as WebLogic started, then simply set the transaction attribute on the `@jc:tuxedo` tag to `NotSupported`. See [How Do I: Create a Web Service with Transactional Support in the Tuxedo Service](#), for more information on enabling transaction support in your Tuxedo application.

TPENOENT

Problem: The TPENOENT error is typically caused when the service listed in the service attribute (or in the `send-queue-space` attribute) of the `@jc:tuxedo` tag has not been imported.

Resolution: Verify with the WebLogic Console that the service has been imported and that the WTC server has been deployed.

TPESYSTEM from a Service that Returns a VIEW Buffer

Problem: TPESYSTEM indicates a generic system error and can be caused by many different types of problems. In a service that returns a VIEW buffer, TPESYSTEM often indicates that WTC has not been given the class for the type of buffer that is being returned. Therefore, the reply cannot be constructed.

Resolution: Verify that WTC knows about the VIEW buffer class. The class should either be named in the method's `view-class` attribute, or should be configured as a resource in the WTC configuration.

@jc:tuxedo Tag

Specifies which services and buffer formats are used with the control. The tag can be specified in the operation definition with the exception of the mapping-strictness and the debug-level tags.

Syntax

@jc:tuxedo

```
service-type="service | oneway | queue"
service-name="name"
buffer-type="carray | fml | fml32 | string | xml | view |
            view32 | x_c_type | x_common | x_octet | none"
field-fml32="field_name"
mapping-strictness="strict | normal | loose"
transaction="Supports | NotSupported"
send-queue-space="queue_space_name"
send-queue-name="queue_name"
receive-queue-space="queue_space_name"
receive-queue-name="queue_name"
reply-queue-name="queue_name"
failure-queue-name="failure_name"
view-class="view_class_name"
send-view-class="view_class_name"
accessor-name-conventions="bean-conventions |
                           tux-conventions"
```

Attributes

service-type

Specifies the type of Tuxedo service this control uses.

Default: **service**

Possible values are:

- **service** specifies a typical synchronous request/response (tpcall). The control sends a request to a remote Tuxedo service and waits for a reply.
- **oneway** specifies a request without a response expected (tpacall).
- **queue** specifies a queued model of interaction (tpenqueue). In this case other attributes specifying the queue space and the queue name such as send-queue-space, send-queue-name, receive-queue-space, receive-queue-name, reply-queue-name, and failure-queue-name should be defined.

service-name

Specifies the WTC imported service name. This attribute can only be specified at the method level (not at the control level).

buffer-type

Specifies the type of buffer the Tuxedo Control supports in the Tuxedo service.

Default: **fml32**

Possible values are:

- **carray** used when the data is an unspecified array of characters (byte array), any of which can be null. Tuxedo equivalent: CARRAY.
- **fml** used when the data contains named fields of simple types each with a maximum length of 65 K bytes. In this case, the field-tables attribute should also be defined.
- **fml32** used when the data contains named fields including nested FML32 fields each having a maximum length of 4 GB. In this case, the field-tables attribute should also be defined.
- **string** used when the data contains a simple string as its only contents.
- **xml** used when data is an XML document. Data input to methods that specify a buffer-type of xml must be in the form of a Java String or an XML Bean (an instance of a class that implements XmlObject). Tuxedo equivalent: XML for Tuxedo Release 7.1 and higher.

Note: For input XML using data dependent routing, remove all comments generated by the WebLogic Workshop test browser in the XML buffer. Depending on the location of the comments in the buffer, Tuxedo may have problems parsing the buffer.

- **view** used to define C or COBOL structures for a Tuxedo application and to define an equivalent Java TypedView buffer for a WebLogic application. A view description file in which the fields and types that appear in the data structure are defined, must be available to client and server processes that use a data structure described in a VIEW typed buffer. Encoding and decoding are performed automatically if the buffer is passed between machines of different types.
- **view32** is equivalent to VIEW but uses 32 bits for length and count fields, which allows for larger and more fields.
- **x_common** is a synonym for the VIEW buffer type. An **x_common** buffer is represented by a Java class extending the TypedXCommon class. It is identical in semantics to a VIEW buffer.
- **x_c_type** is a synonym for the VIEW buffer type. An **x_c_type** buffer is represented by a Java class extending the TypedXCType class. It is identical in semantics to a VIEW buffer.
- **x_octet** is simply a byte array and is a synonym for the CARRAY buffer type in Tuxedo.
- **none** is used when no input buffer is to be used in the request to the Tuxedo service. If the method signature contains input parameters, an error is generated.

field-tables

Specifies the class names that describe the buffer for the Tuxedo service if you have specified **buffer-type="fml | fml32"**. These class names are space delimited and must be fully qualified.

The name of the field table should match the classes generated with the **mkfldclass** or **mkfldclass32** utilities.

For buffer types **fml** and **fml32**, WTC needs the names and types of the fields the buffer can contain. These are described in field table classes created with **mkfldclass** and **mkfldclass32** utilities.

For more information about mkfldclass and mkfldclass32 utilities, see WebLogic Server Javadoc.

mapping-strictness

Specifies the type of coercion strictness you want for the buffer mapping.

Default: **normal**

Possible values are:

- **normal** is the most expected type of coercion. If the field is a numeric field type and the Java Number class can provide the appropriate coercion, then it is used. Exceptions are raised only if the coercion required is out of the ordinary, such as trying to convert the string "one" to a numeric field. Missing fields and some type coercion problems are logged as issues.
- **strict** only allows matching types with little or no coercion provided. Most disallowed combinations cause a ControlException. Few issues are logged as most problems result in an exception.
- **loose** attempts any feasible coercion. For example, a string containing "true" or "The" when mapped to a Boolean field will both result in a value of true as the control will only look at the first character of the string. This setting can yield uncertain results. Some data may not make it through the control and as a result may be lost in mapping. Few errors raise an exception and most problems are logged as issues.

transaction

Provides a declarative mechanism for controlling the transaction context under which the control executes. If an exception occurs inside the web service and is not caught, any transaction in progress is aborted.

Default: **Supports**

Possible values are:

- **Supports** the web service uses the transaction state of the caller of the web service. The control neither starts a transaction automatically nor checks for a transaction before executing.
- **NotSupported** the control sets the TPNOTRAN flag when making a request to Tuxedo. This causes the Tuxedo service to operate outside the context of the transaction started by the web service.

send-queue-space

Specifies the name of the queue space in which the sending queue name is located. For more information on queues, refer to the Tuxedo Documentation.

send-queue-name

Specifies the name of the queue to send messages to for **service-type="queue"**. For more information on queues, refer to the Tuxedo Documentation.

receive-queue-space

Specifies the name of the queue space in which the receiving queue name is located. For more information on queues, refer to the Tuxedo Documentation.

receive-queue-name

Specifies the name of the queue to receive queued messages for **service-type="queue"**. For more information on queues, refer to the Tuxedo Documentation.

reply-queue-name

Specifies the name of the reply queue to associate with the message for **service-type="queue"**. For more information on queues, refer to the Tuxedo Documentation.

failure-queue-name

Specifies the name of the failure queue to associate with the message for **service-type="queue"**. For more information on queues, refer to the Tuxedo Documentation.

view-classes

Specifies the names of the view classes that may be returned by Tuxedo services. Specify one or more view classes when the service you are calling may return TypedView, TypedView32, TypedXCommon, or TypedXCType buffers. The class names should match the classes generated by the viewj or viewj32 utilities and should be space delimited and fully qualified.

The control uses this attribute to pass view class types to WTC. WTC needs to know the view class type so it can create a TypedView instance to contain the data it receives from Tuxedo. Alternatively, this information can be provided to WTC in the Resource section of the WTC configuration.

For more information about viewj and viewj32 utilities, see WebLogic Server Javadoc.

send-view-class

Specifies the name of the view class that is passed to the Tuxedo service when you specify **buffer-type="view | view32 | x_common | x_c_type"**. This view class is used to create the TypedView buffer.

accessor-name-conventions

To convert an accessor name to a field name, specify the convention to follow for naming.

Default: **bean-conventions**

Possible values are:

- **bean-conventions** is the default naming convention. The control only recognizes accessor names that begin with set, get, or is followed by an uppercase letter. It will strip the set, get, or is prefix from the accessor name and convert the first character of the resulting field name to lowercase.
- **tux-conventions** allows the control to accept any name beginning with set, get, or is as a valid accessor. The control strips the set, get, or is prefix and uses the resulting name unchanged.

Crystal Reports for BEA WebLogic Workshop

Crystal Reports for BEA WebLogic Workshop gives you the ability to create, import, and edit Crystal Reports on a JSP page. These abilities are integrated into the IDE.

Topics Included in This Section

Crystal Reports for BEA WebLogic Workshop Java API Reference

This library consolidates the developer documentation that is included with Crystal Reports for BEA WebLogic Workshop.

JSP Tags Reference

Provides a brief overview of the Viewer Tag Library, including how to set up your JSP pages to use the tag library.

Crystal Reports for BEA WebLogic Workshop User's Guide

This guide provides you with procedures for ensuring Crystal Reports for BEA WebLogic Workshop is correctly installed and how to obtain and install Crystal Reports. It also shows you how to use Crystal Reports for BEA WebLogic Workshop to integrate report viewing into BEA WebLogic Workshop projects.

Crystal Reports for BEA WebLogic Workshop Developer's Guide

This guide provides the information necessary to help you to get started with the Viewers Java SDK. It includes code samples and a number of step-by-step examples that you can follow in order to help you integrate the Java viewer in your own web applications.

Related Topics

None.

Welcome to the Crystal Reports for BEA WebLogic Workshop Java API Reference

This library consolidates the developer documentation that is included with Crystal Reports for BEA WebLogic Workshop. You can access information by browsing the various sets of JavaDocs and supplementary documentation.

Click the appropriate link to jump to a specific guide:

Crystal Common Classes

- [JavaDocs API Reference](#)

Viewer Java SDK Guide

- [JavaDocs API Reference](#)
-

Crystal Decisions, Inc.
<http://www.crystaldecisions.com/>
Support services:
<http://support.crystaldecisions.com/>

Crystal Common Classes

The Crystal Common Classes provide you with tools to retrieve and set properties in a report.

See:

Description

Packages	
com.crystaldecisions.sdk.occa.report.data	This package is used to provide a definition for the report's data.
com.crystaldecisions.sdk.occa.report.exportoptions	This package allows you to specify the export format of a report document.
com.crystaldecisions.sdk.occa.report.lib	This package is a general utility that provides exception and container classes.

The Crystal Common Classes provide you with tools to retrieve and set properties in a report. The classes provide a definition of the primary report properties, such as parameter fields, fields, and their associated values. The Crystal Common Classes are a subset of the Report Application Server (RAS) SDK, which is part of an enterprise level report creation and modification solution—the Report Application Server. Unlike the RAS SDK, the Crystal Common Classes are not able to make permanent changes to report properties. Any properties that are modified will only be effective for the duration of the session. For more information on how to permanently modify reports or dynamically create reports, see the *Crystal Reports for BEA WebLogic Workshop Developer's Guide*.

The Crystal Common Classes consist of two packages, `com.crystaldecisions.sdk.occa.report.data` and `com.crystaldecisions.sdk.occa.report.lib`. The data package provides the report definition classes, while the lib package provides utility classes that are used by the classes in data.

Package `com.crystaldecisions.sdk.occa.report.data`

This package is used to provide a definition for the report's data.

See:

Description

Interface Summary	
IConnectionInfo	This interface enables you to get and set information for the data source connection.
IField	This interface defines a report field in general.
IFormulaField	This interface defines a formula field in the report.
IParameterField	This interface is used to get and set values for the parameter field.

IDE Extensions

IParameterFieldDiscreteValue	This interface is used to get and set a discrete value belonging to a parameter.
IParameterFieldRangeValue	This interface is used to get and set values for a ranged parameter.
IParameterFieldValue	This interface enables you to get and set a parameter's description.
IValue	This interface is used as a base class to represent different kinds of values (values in formulas, parameters, and so on).

Class Summary	
ConnectionInfo	This object enables you to get and set information for the data source connection.
ConnectionInfos	This object returns connection information.
Field	This object represents a report field in general.
FieldDisplayNameType	This class contains constants that indicate how the name of the field is displayed.
FieldKind	This class contains constants which indicate the kind of a particular field.
FieldValueType	This class contains constants that indicate what type of data is stored by the field.
FormulaField	This object implements a formula field in the report.
ParameterDefaultValueDisplayType	For internal use only.
ParameterField	This object enables you to get and set values for the parameter field.
ParameterFieldDiscreteValue	This object represents a discrete value belonging to a parameter.
ParameterFieldRangeValue	This object stores the value for a ranged parameter.
ParameterFieldType	This class contains constants for the parameter field type and methods for using these constants.
ParameterFieldValue	This object represents a parameter field value.
ParameterSortMethod	This class contains constants that specify how the parameters are sorted.
ParameterSortOrder	This class contains constants that indicate how a parameter list is sorted.
ParameterValueRangeKind	This class contains constants that indicate whether the parameter is discrete, ranged, or both.
RangeValueBoundType	This class contains constants that indicate the range bound; that is, how the bound on a range is treated.
Value	This object defines a value of a field.
Values	

This object defines a collection that contains an array of Value objects and allows you to add, remove, search for, and add new values to and from the collection.
--

Package com.crystaldecisions.sdk.occa.report.data Description

This package is used to provide a definition for the report's data. It contains all the data structures necessary to define the data that the report contains. It is concerned with the retrieval of data, where and how this data is stored, the structures that are used to store the data while processing it, and the structures that are used to store it once it is processed. The report's data definition is not concerned with how the report is laid out. That is, it does not store information regarding graphics, charts, borders, text objects, and so on.

Although this library provides functionality to modify and manipulate the data definition, any changes made to the report will not be permanent. In order to dynamically permanently modify the report, you must have a Report Application Server. For more information on this, consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide*.

com.crystaldecisions.sdk.occa.report.data Class ConnectionInfo

```
java.lang.Object
└─ com.crystaldecisions.sdk.occa.report.data.ConnectionInfo
```

All Implemented Interfaces:

IClone, IConnectionInfo

```
public class ConnectionInfo
extends java.lang.Object
implements IConnectionInfo, IClone
```

This object enables you to get and set information for the data source connection. Use the IConnectionInfo interface to manipulate this object.

Constructor Summary

ConnectionInfo ()
ConnectionInfo (IConnectionInfo src)

Method Summary

java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, com.crystaldecisions.xml.serialization.XMLSerializationContext ctx, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
PropertyBag	getAttributes () Returns the property bag for the data source connection.
ConnectionInfoKind	getKind () Returns the kind of connection.
java.lang.String	getPassword () Returns the password used to connect to the data source.
java.lang.String	getUserName () Returns the user name used to connect to the data source.
boolean	hasContent (java.lang.Object obj) Returns true if this object contains the same elements as the passed in object.
boolean	isMatch (IConnectionInfo info, boolean completeMatching) Checks whether two connections match.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, com.crystaldecisions.xml.serialization.XMLSerializationContext ctx) For internal use only.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, java.lang.String sTag, com.crystaldecisions.xml.serialization.XMLSerializationContext ctx) For internal use only.
void	saveContents (com.crystaldecisions.xml.serialization.XMLWriter writer, com.crystaldecisions.xml.serialization.XMLSerializationContext ctx) For internal use only.
void	setAttributes (PropertyBag attributes) Sets the property bag for the data source connection.
void	setKind (ConnectionInfoKind kind) Returns the kind of connection.
void	setPassword (java.lang.String password) Returns the password used to connect to the data source.

IDE Extensions

void	setUserName (java.lang.String userName) Sets the user name used to connect to the data source.
boolean	skipWritingIdenticalObject () For internal use only.
void	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ConnectionInfo

```
public ConnectionInfo(IConnectionInfo src)
```

ConnectionInfo

```
public ConnectionInfo()
```

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: IClone

Returns the new object that has been cloned.

Specified by:

clone in interface IClone

Parameters:

deepClone – true to use deep clone, false to use shallow.

Returns:

The new object that has been cloned.

copyTo

```
public void copyTo(java.lang.Object destObject,
                  boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

copyTo in interface IClone

Parameters:

destObject – The destination object to copy to.

deepCopy – true to use deep copy, false to use shallow.

createMember

```
public java.lang.Object createMember(java.lang.String eleName,
                                       org.xml.sax.Attributes attrs,
                                       com.crystaldecisions.xml.serialization.XMLSerializationCon
                                       java.util.Map objState,
                                       boolean[] bLoaded)
```

For internal use only.

endElement

```
public void endElement(java.lang.String eleName,
                      java.util.Map objState)
```

For internal use only.

getAttributes

```
public PropertyBag getAttributes()
```

Description copied from interface: IConnectionInfo

Returns the property bag for the data source connection.

Specified by:

getAttributes in interface IConnectionInfo

Returns:

The property bags as a PropertyBag object.

getKind

```
public ConnectionInfoKind getKind()
```

Description copied from interface: IConnectionInfo

Returns the kind of connection. For example, connection kinds include SQL, query, meta data, database file, and Crystal Report Query Engine (CRQE).

Specified by:

getKind in interface `IConnectionInfo`

Returns:

The connection kind as a `ConnectionInfoKind` object.

getPassword

```
public java.lang.String getPassword()
```

Description copied from interface: `IConnectionInfo`

Returns the password used to connect to the data source.

Specified by:

getPassword in interface `IConnectionInfo`

Returns:

The password as a `String`.

getUserName

```
public java.lang.String getUserName()
```

Description copied from interface: `IConnectionInfo`

Returns the user name used to connect to the data source.

Specified by:

getUserName in interface `IConnectionInfo`

Returns:

The user name as a `String`.

hasContent

```
public boolean hasContent(java.lang.Object obj)
```

Description copied from interface: `IClone`

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface `IClone`

Parameters:

obj – The object to check for content.

Returns:

true if this object contains the same elements as the passed in object, otherwise false.

readElement

```
public void readElement(java.lang.String eleName,  
                        java.lang.String sVal,
```

IDE Extensions

```
org.xml.sax.Attributes attrs,  
java.util.Map objState)
```

For internal use only.

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,  
                 com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

```
java.io.IOException
```

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,  
                 java.lang.String sTag,  
                 com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

```
java.io.IOException
```

saveContents

```
public void saveContents(com.crystaldecisions.xml.serialization.XMLWriter writer,  
                         com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

```
java.io.IOException
```

setAttributes

```
public void setAttributes(PropertyBag attributes)
```

Description copied from interface: IConnectionInfo

Sets the property bag for the data source connection.

Specified by:

```
setAttributes in interface IConnectionInfo
```

Parameters:

```
attributes – The property bag as a PropertyBag object.
```

setKind

```
public void setKind(ConnectionInfoKind kind)
```

Description copied from interface: IConnectionInfo

Returns the kind of connection. For example, connection kinds include SQL, query, meta data, database file, and Crystal Report Query Engine (CRQE).

Specified by:

setKind in interface IConnectionInfo

Parameters:

kind – The connection kind as a ConnectionInfoKind object.

setPassword

```
public void setPassword(java.lang.String password)
```

Description copied from interface: IConnectionInfo

Returns the password used to connect to the data source.

Specified by:

setPassword in interface IConnectionInfo

Parameters:

password – The password as a String.

setUserName

```
public void setUserName(java.lang.String userName)
```

Description copied from interface: IConnectionInfo

Sets the user name used to connect to the data source.

Specified by:

setUserName in interface IConnectionInfo

Parameters:

userName – The user name as a String.

startElement

```
public void startElement(java.lang.String eleName,  
                          java.util.Map objState,  
                          org.xml.sax.Attributes attrs)
```

For internal use only.

skipWritingIdenticalObject

```
public boolean skipWritingIdenticalObject()
```

For internal use only.

isMatch

```
public boolean isMatch(IConnectionInfo info,
                       boolean completeMatching)
```

Description copied from interface: IConnectionInfo

Checks whether two connections match. Returns true if they match, and false otherwise.

Specified by:

isMatch in interface IConnectionInfo

Parameters:

info – The IConnectionInfo object that this connection will be compared against.
 completeMatching – A boolean specifying whether to use complete matching. By default, this is set to true. If true, all members of the object are compared. When all members of the object have been compared, a value of true is returned if all members are exactly the same. If set to false, all members of the object are compared and a value of true is returned if the mismatch is caused because one of the comparing members has an empty string, but all of the remaining members match.

Returns:

true if they match, and false otherwise.

com.crystaldecisions.sdk.occa.report.data Class ConnectionInfos

```
java.lang.Object
├─ java.util.AbstractCollection
│   └─ java.util.AbstractList
│       └─ java.util.ArrayList
│           └─ com.crystaldecisions.sdk.occa.report.lib.ReportSDKVector
│               └─ com.crystaldecisions.sdk.occa.report.data.ConnectionInfos
```

All Implemented Interfaces:

java.lang.Cloneable, java.util.Collection, IClone, java.util.List, java.util.RandomAccess, java.io.Serializable

```
public class ConnectionInfos
  extends ReportSDKVector
  implements IClone
```

This object returns connection information.

See Also:

Serialized Form

Constructor Summary

ConnectionInfos ()
ConnectionInfos (ConnectionInfos src)

Method Summary	
void	add (int index, java.lang.Object element) Inserts the specified element at the specified position in this collection.
boolean	add (java.lang.Object o) Appends the specified element to the end of this collection.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
IConnectionInfo	getConnectionInfo (int index) Returns an index value for the connection from IConnectionInfo.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
void	save (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.

Methods inherited from class com.crystaldecisions.sdk.occa.report.lib.ReportSDKVector
addElement, clone, copyTo, elementAt, findIndexOf, hasContent, insertElementAt, removeAllElements

Methods inherited from class java.util.ArrayList
--

IDE Extensions

<code>addAll, addAll, clear, clone, contains, ensureCapacity, get, indexOf, isEmpty, lastIndexOf, remove, set, size, toArray, toArray, trimToSize</code>
--

Methods inherited from class <code>java.util.AbstractList</code>

<code>equals, hashCode, iterator, listIterator, listIterator, subList</code>
--

Methods inherited from class <code>java.util.AbstractCollection</code>

<code>containsAll, remove, removeAll, retainAll, toString</code>
--

Methods inherited from class <code>java.lang.Object</code>

<code>getClass, notify, notifyAll, wait, wait, wait</code>
--

Methods inherited from interface <code>com.crystaldecisions.sdk.occa.report.lib.IClone</code>
--

<code>clone, copyTo, hasContent</code>
--

Methods inherited from interface <code>java.util.List</code>

<code>containsAll, equals, hashCode, iterator, listIterator, listIterator, remove, removeAll, retainAll, subList</code>

Constructor Detail

ConnectionInfos

```
public ConnectionInfos(ConnectionInfos src)
```

ConnectionInfos

```
public ConnectionInfos()
```

Method Detail

getConnectionInfo

```
public IConnectionInfo getConnectionInfo(int index)
```

Returns an index value for the connection from `IConnectionInfo`.

Parameters:

`index` – The index value for the `IConnectionInfo` as an `int`.

Returns:

An IConnectionInfo object.

readElement

```
public void readElement(java.lang.String eleName,  
                        java.lang.String sVal,  
                        org.xml.sax.Attributes attrs,  
                        java.util.Map objState)
```

For internal use only.

createMember

```
public java.lang.Object createMember(java.lang.String eleName,  
                                     org.xml.sax.Attributes attrs,  
                                     XMLSerializationContext ctxt,  
                                     java.util.Map objState,  
                                     boolean[] bLoaded)
```

For internal use only.

endElement

```
public void endElement(java.lang.String eleName,  
                      java.util.Map objState)
```

For internal use only.

save

```
public void save(XMLWriter writer,  
                java.lang.String sTag,  
                XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

save

```
public void save(XMLWriter writer,  
                XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

saveContents

```
public void saveContents(XMLWriter writer,
                        XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

startElement

```
public void startElement(java.lang.String eleName,
                        java.util.Map objState,
                        org.xml.sax.Attributes attrs)
```

For internal use only.

add

```
public boolean add(java.lang.Object o)
```

Appends the specified element to the end of this collection.

Specified by:

add in interface java.util.List

Parameters:

o – element to be added into the collection.

Returns:

true if this collection changed as a result of the call.

Throws:

java.lang.ClassCastException – class of the specified element prevents it from being added to this collection.

java.lang.NullPointerException – if the specified element is null and this collection does not support null elements.

add

```
public void add(int index,
                java.lang.Object element)
```

Inserts the specified element at the specified position in this collection. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Specified by:

add in interface java.util.List

Parameters:

index – index at which the specified element is to be inserted.

element – element to be inserted.

Throws:

`java.lang.IndexOutOfBoundsException` – if index is out of range (`index < 0` || `index > size()`).

`java.lang.ClassCastException` – class of the specified element prevents it from being added to this collection.

`java.lang.NullPointerException` – if the specified element is null and this collection does not support null elements.

com.crystaldecisions.sdk.occa.report.data Class Field

```
java.lang.Object
└─ com.crystaldecisions.sdk.occa.report.data.Field
```

All Implemented Interfaces:

IClone, IField

Direct Known Subclasses:

FormulaField, ParameterField

*public abstract class **Field***

extends java.lang.Object

implements IField, IClone

This object represents a report field in general.

Constructor Summary

Field()

Method Summary

java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attributes, com.crystaldecisions.xml.serialization.XMLSerializationContext context, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	

IDE Extensions

	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
java.lang.String	getDescription () Returns a description of the field.
java.lang.String	getDisplayName (FieldDisplayNameType displayName, java.util.Locale locale) Returns the name of the field that has been formatted according to the value of the param displayNameType.
java.lang.String	getFormulaForm () Returns the name of the field as it will be used in a formula.
java.lang.String	getHeadingText () Returns the text used as a heading when the field is added to the report.
boolean	getIsRecurring () Returns true if the field is recurring, and false otherwise.
int	getLength () Returns the maximum length the field's data may be (in bytes).
java.lang.String	getLongName (java.util.Locale locale) Returns the fully qualified name of the field.
java.lang.String	getName () Returns the name of the field.
java.lang.String	getShortName (java.util.Locale locale) Returns the short name of the field.
FieldValueType	getType () Returns what type of field this is.
boolean	hasContent (java.lang.Object srcField) Returns true if this object contains the same elements as the passed in object.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, com.crystaldecisions.xml.serialization.XMLSerializationContext ctx) For internal use only.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, java.lang.String sTag, com.crystaldecisions.xml.serialization.XMLSerializationContext ctx) For internal use only.
void	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	setDescription (java.lang.String description) Sets the description of the field.
void	setHeadingText (java.lang.String headingText) Sets the text used as a heading when the field is added to the report.

IDE Extensions

void	setLength (int length) Sets the maximum length the field's data may be (in bytes).
void	setName (java.lang.String name) Sets the name of the field.
void	setType (FieldValueType valueType) Sets what type of field this is.
void	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface com.crystaldecisions.sdk.occa.report.data.IField

getKind

Constructor Detail

Field

```
public Field()
```

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: IClone

Returns the new object that has been cloned.

Specified by:

clone in interface IClone

Parameters:

deepClone – true to use deep clone, false to use shallow.

Returns:

The new object that has been cloned.

copyTo

```
public void copyTo(java.lang.Object destObject,
                   boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

copyTo in interface IClone

Parameters:

destObject – The destination object to copy to.

deepCopy – true to use deep copy, false to use shallow.

endElement

```
public void endElement(java.lang.String eleName,
                       java.util.Map objState)
```

For internal use only.

getDescription

```
public java.lang.String getDescription()
```

Description copied from interface: IField

Returns a description of the field.

Specified by:

getDescription in interface IField

Returns:

A String containing a description of the field.

getFormulaForm

```
public java.lang.String getFormulaForm()
```

Description copied from interface: IField

Returns the name of the field as it will be used in a formula. This is a unique identifier to avoid ambiguity in formula code. It is the same value as the one used in the CRW formula editor.

Specified by:

getFormulaForm in interface IField

Returns:

A String containing the name of the field as it will be used in a formula.

getHeadingText

```
public java.lang.String getHeadingText()
```

Description copied from interface: IField

Returns the text used as a heading when the field is added to the report.

Specified by:

getHeadingText in interface IField

Returns:

A String containing the text used as a heading when the field is added to the report.

getIsRecurring

```
public boolean getIsRecurring()
```

Description copied from interface: IField

Returns true if the field is recurring, and false otherwise. A field is recurring if its contents change during the formatting of a report. A constant field (not recurring) is one that doesn't change. The special field "Page Number," for example, is a recurring field, while the special field "Report Title" is not.

Specified by:

getIsRecurring in interface IField

Returns:

true if the field is recurring, and false otherwise.

getLength

```
public int getLength()
```

Description copied from interface: IField

Returns the maximum length the field's data may be (in bytes).

Specified by:

getLength in interface IField

Returns:

A String containing the maximum length the field's data may be (in bytes).

getLongName

```
public java.lang.String getLongName(java.util.Locale locale)
```

Description copied from interface: IField

Returns the fully qualified name of the field.

Specified by:

getLongName in interface IField

Parameters:

locale – The locale that will be used to localize the field.

Returns:

A String containing the fully qualified name of the field.

getName

```
public java.lang.String getName()
```

Description copied from interface: IField

Returns the name of the field. For DBField objects, the value returned by this method is equivalent to the getDisplayName method.

Specified by:

getName in interface IField

Returns:

A String containing the name of the field.

getShortName

```
public java.lang.String getShortName(java.util.Locale locale)
```

Description copied from interface: IField

Returns the short name of the field.

Specified by:

getShortName in interface IField

Parameters:

locale – The locale that will be used to localize the field name.

Returns:

A String containing the short name of the field.

readElement

```
public void readElement(java.lang.String eleName,  
                        java.lang.String sVal,  
                        org.xml.sax.Attributes attrs,  
                        java.util.Map objState)
```

For internal use only.

saveContents

```
public void saveContents(XMLWriter writer,  
                        XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

setDescription

```
public void setDescription(java.lang.String description)
```

Description copied from interface: IField

Sets the description of the field.

Specified by:

setDescription in interface IField

Parameters:

description – A String containing a description of the field.

setHeadingText

```
public void setHeadingText(java.lang.String headingText)
```

Description copied from interface: IField

Sets the text used as a heading when the field is added to the report.

Specified by:

setHeadingText in interface IField

Parameters:

headingText – A String containing the text used as a heading when the field is added to the report.

setLength

```
public void setLength(int length)
```

Description copied from interface: IField

Sets the maximum length the field's data may be (in bytes).

Specified by:

setLength in interface IField

Parameters:

length – A String containing the maximum length the field's data may be (in bytes).

setName

```
public void setName(java.lang.String name)
```

Description copied from interface: IField

Sets the name of the field. For DBField objects, the value returned by this method is equivalent to the getDisplayName method.

Specified by:

setName in interface IField

Parameters:

name – A String containing the name of the field.

createMember

```
public java.lang.Object createMember(java.lang.String eleName,
                                     org.xml.sax.Attributes attrs,
                                     com.crystaldecisions.xml.serialization.XMLSerializationCon
                                     java.util.Map objState,
                                     boolean[] bLoaded)
```

For internal use only.

getDisplayName

```
public java.lang.String getDisplayName(FieldDisplayNameType displayName,
                                     java.util.Locale locale)
```

Description copied from interface: IField

Returns the name of the field that has been formatted according to the value of the parameter `displayNameType`.

Specified by:

`getDisplayName` in interface `IField`

Parameters:

`displayName` – Indicates how the display text should be formatted. It may be one of the values listed under `FieldDisplayNameType`.

`locale` – Formats the string according to a locale.

Returns:

A `String` containing the name of the field that has been formatted according to the value of the parameter `displayNameType`.

getType

```
public FieldValueType getType()
```

Description copied from interface: IField

Returns what type of field this is. Do not confuse this property with `Kind`. `Kind` identifies what sort of report field this is, while `Type` identifies what data it stores. For example, a bitmap, string, 8 bit integer, chart, etc.

Specified by:

`getType` in interface `IField`

Returns:

A `FieldValueType` object that specifies what type of field this is.

hasContent

```
public boolean hasContent(java.lang.Object srcField)
```

Description copied from interface: IClone

Returns `true` if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Parameters:

srcField – The object to check for content.

Returns:

true if this object contains the same elements as the passed in object, otherwise false.

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,
                 com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,
                 java.lang.String sTag,
                 com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

setType

```
public void setType(FieldValueType valueType)
```

Description copied from interface: IField

Sets what type of field this is. Do not confuse this property with Kind. Kind identifies what sort of report field this is, while Type identifies what data it stores. For example, a bitmap, string, 8 bit integer, chart, etc.

Specified by:

setType in interface IField

Parameters:

valueType – A FieldValueType object that specifies what type of field this is.

startElement

```
public void startElement(java.lang.String eleName,
                        java.util.Map objState,
                        org.xml.sax.Attributes attrs)
```

For internal use only.

com.crystaldecisions.sdk.occa.report.data

Class FieldDisplayNameType

```
java.lang.Object
└─ com.crystaldecisions.sdk.occa.report.data.FieldDisplayNameType
```

```
public final class FieldDisplayNameType
    extends java.lang.Object
```

This class contains constants that indicate how the name of the field is displayed. Not all fields supply different forms of its name. If a particular form is not available, its short name is returned.

Field Summary	
static int	_description Some databases provide a description of a field.
static int	_fieldName The field's name without its prefix.
static int	_formulaName The name as it could be used in a formula.
static int	_headingText The name of the field as it would appear as the heading of a column.
static int	_longName The fully qualified field name.
static int	_shortName The field's name including the prefix indicating what kind of field it is.
static FieldDisplayNameType	description Some databases provide a description of a field.
static FieldDisplayNameType	fieldName The field's name without its prefix.
static FieldDisplayNameType	formulaName The name as it could be used in a formula.
static FieldDisplayNameType	headingText The name of the field as it would appear as the heading of a column.
static FieldDisplayNameType	longName The fully qualified field name.
static FieldDisplayNameType	shortName The field's name including the prefix indicating what kind of field it is.

Method Summary

<code>static FieldDisplayNameType</code>	from_int (int i) Returns the FieldDisplayNameType corresponding to the specified int.
<code>int</code>	value () Returns the value of this FieldDisplayNameType object.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail

`_fieldName`

```
public static final int _fieldName
```

The field's name without its prefix. For a DBFields object, this will be equivalent to its short name.

See Also:

Constant Field Values

`_shortName`

```
public static final int _shortName
```

The field's name including the prefix indicating what kind of field it is. Formula fields are prefixed with @. SQL Expression Fields are prefixed with %. Parameter fields are prefixed with ?. Running total fields are prefixed with #.

See Also:

Constant Field Values

`_longName`

```
public static final int _longName
```

The fully qualified field name.

See Also:

Constant Field Values

_description

```
public static final int _description
```

Some databases provide a description of a field. If there is no description for the field the method returns an empty string.

See Also:

Constant Field Values

_formulaName

```
public static final int _formulaName
```

The name as it could be used in a formula.

See Also:

Constant Field Values

_headingText

```
public static final int _headingText
```

The name of the field as it would appear as the heading of a column.

See Also:

Constant Field Values

fieldName

```
public static final FieldDisplayNameType fieldName
```

The field's name without its prefix. For a `DBFields` object, this will be equivalent to its short name.

shortName

```
public static final FieldDisplayNameType shortName
```

The field's name including the prefix indicating what kind of field it is. Formula fields are prefixed with `@`. SQL Expression Fields are prefixed with `%`. Parameter fields are prefixed with `?`. Running total fields are prefixed with `#`.

longName

```
public static final FieldDisplayNameType longName
```

The fully qualified field name.

description

```
public static final FieldDisplayNameType description
```

Some databases provide a description of a field. If there is no description for the field the method returns an empty string.

formulaName

```
public static final FieldDisplayNameType formulaName
```

The name as it could be used in a formula.

headingText

```
public static final FieldDisplayNameType headingText
```

The name of the field as it would appear as the heading of a column.

Method Detail

from_int

```
public static final FieldDisplayNameType from_int(int i)
```

Returns the `FieldDisplayNameType` corresponding to the specified `int`.

Parameters:

`i` – The `int` value of the desired `FieldDisplayNameType`.

Returns:

The `FieldDisplayNameType` corresponding to the specified `int`.

value

```
public int value()
```

Returns the value of this `FieldDisplayNameType` object.

Returns:

An `int` that specifies the value of this `FieldDisplayNameType` object.

com.crystaldecisions.sdk.occa.report.data

Class FieldKind

```
java.lang.Object
└─ com.crystaldecisions.sdk.occa.report.data.FieldKind
```

```
public final class FieldKind
extends java.lang.Object
```

This class contains constants which indicate the kind of a particular field. This is not the same as the field's value type, which indicates what data type (integer or string) is used to store the field's data.

Field Summary	
static int	_DBField Database field.
static int	_formulaField Formula field.
static int	_groupNameField Group name field.
static int	_parameterField Parameter field.
static int	_runningTotalField Running total field.
static int	_specialField Special field.
static int	_summaryField Summary field.
static int	_unknownField Unknown field type.
static FieldKind	DBField A FieldKind object that specifies a database field.
static FieldKind	formulaField A FieldKind object that specifies a formula field.
static FieldKind	groupNameField A FieldKind object that specifies a group name field.
static FieldKind	parameterField A FieldKind object that specifies a parameter field.
static FieldKind	runningTotalField A FieldKind object that specifies a running total field.
static FieldKind	specialField A FieldKind object that specifies a special field.

<code>static FieldKind</code>	summaryField A <code>FieldKind</code> object that specifies a summary field.
<code>static FieldKind</code>	unknownField A <code>FieldKind</code> object that specifies an unknown field type.

Method Summary	
<code>static FieldKind</code>	from_int (int i) Returns the <code>FieldKind</code> object with the specified int value.
<code>static FieldKind</code>	from_string (java.lang.String sVal) Returns the <code>FieldKind</code> object with the specified String value.
<code>java.lang.String</code>	toString () Returns the String value of this <code>FieldKind</code> object.
<code>int</code>	value () Returns the int value of this <code>FieldKind</code> object.

Methods inherited from class java.lang.Object
<code>equals</code> , <code>getClass</code> , <code>hashCode</code> , <code>notify</code> , <code>notifyAll</code> , <code>wait</code> , <code>wait</code> , <code>wait</code>

Field Detail

_DBField

```
public static final int _DBField
```

Database field.

See Also:

Constant Field Values

_formulaField

```
public static final int _formulaField
```

Formula field.

See Also:

Constant Field Values

_parameterField

```
public static final int _parameterField
```

Parameter field.

See Also:

Constant Field Values

_specialField

```
public static final int _specialField
```

Special field.

See Also:

Constant Field Values

_summaryField

```
public static final int _summaryField
```

Summary field.

See Also:

Constant Field Values

_groupNameField

```
public static final int _groupNameField
```

Group name field.

See Also:

Constant Field Values

DBField

```
public static final FieldKind DBField
```

A `FieldKind` object that specifies a database field.

formulaField

```
public static final FieldKind formulaField
```

A FieldKind object that specifies a formula field.

parameterField

```
public static final FieldKind parameterField
```

A FieldKind object that specifies a parameter field.

specialField

```
public static final FieldKind specialField
```

A FieldKind object that specifies a special field.

summaryField

```
public static final FieldKind summaryField
```

A FieldKind object that specifies a summary field.

groupNameField

```
public static final FieldKind groupNameField
```

A FieldKind object that specifies a group name field.

_runningTotalField

```
public static final int _runningTotalField
```

Running total field.

See Also:

Constant Field Values

_unknownField

```
public static final int _unknownField
```

Unknown field type.

See Also:

Constant Field Values

runningTotalField

```
public static final FieldKind runningTotalField
```

A FieldKind object that specifies a running total field.

unknownField

```
public static final FieldKind unknownField
```

A FieldKind object that specifies an unknown field type.

Method Detail

from_int

```
public static final FieldKind from_int(int i)
```

Returns the FieldKind object with the specified int value.

Parameters:

i – An int that specifies the value of the desired FieldKind object.

Returns:

The FieldKind object with the specified int value.

toString

```
public java.lang.String toString()
```

Returns the String value of this FieldKind object.

Returns:

The String value of this FieldKind object.

value

```
public int value()
```

Returns the int value of this FieldKind object.

Returns:

The int value of this FieldKind object.

from_string

```
public static final FieldKind from_string(java.lang.String sVal)
```

Returns the FieldKind object with the specified String value.

Parameters:

sVal – A String that specifies the value of the desired FieldKind object.

Returns:

The FieldKind object with the specified String value.

com.crystaldecisions.sdk.occa.report.data**Class FieldValueType**

```
java.lang.Object
```

```
└ com.crystaldecisions.sdk.occa.report.data.FieldValueType
```

```
public final class FieldValueType
```

```
extends java.lang.Object
```

This class contains constants that indicate what type of data is stored by the field.

Field Summary	
static int	_bitmapField A bitmap.
static int	_blobField A BLOB is simply a chunk of binary data.
static int	_booleanField A boolean type.
static int	_chartField A chart.
static int	_currencyField A currency type.
static int	_dateField A date type.
static int	_dateTimeField A date type.
static int	_iconField An icon.
static int	_int16sField 16-bit signed integer.
static int	_int16uField 16-bit unsigned integer.

IDE Extensions

static int	_int32sField 32-bit signed integer.
static int	_int32uField 32-bit unsigned integer.
static int	_int8sField 8-bit signed integer.
static int	_int8uField 8-bit unsigned integer.
static int	_numberField A floating point number.
static int	_oleField An OLE field.
static int	_persistentMemoField The data of persistent memos can be read at any time.
static int	_pictureField A picture.
static int	_sameAsInputField Same data type as input field.
static int	_stringField A string.
static int	_timeField A time field.
static int	_transientMemoField A variable length array of characters.
static int	_unknownField A unknown field.
static FieldValueType	bitmapField A FieldValueType that specifies a bitmap.
static FieldValueType	blobField A FieldValueType that specifies a BLOB or in other words, a chunk of binary data.
static FieldValueType	booleanField A FieldValueType that specifies a boolean type.
static FieldValueType	chartField A FieldValueType that specifies a chart.
static FieldValueType	currencyField A FieldValueType that specifies a currency type.
static FieldValueType	dateField A FieldValueType that specifies a date type.
static FieldValueType	dateTimeField A FieldValueType that specifies a date type.
static FieldValueType	

IDE Extensions

	iconField A <code>FieldValueType</code> that specifies an icon.
<code>static FieldValueType</code>	int16sField A <code>FieldValueType</code> that specifies a 16-bit signed integer.
<code>static FieldValueType</code>	int16uField A <code>FieldValueType</code> that specifies a 16-bit unsigned integer.
<code>static FieldValueType</code>	int32sField A <code>FieldValueType</code> that specifies a 32-bit signed integer.
<code>static FieldValueType</code>	int32uField A <code>FieldValueType</code> that specifies a 32-bit unsigned integer.
<code>static FieldValueType</code>	int8sField A <code>FieldValueType</code> that specifies an 8-bit signed integer.
<code>static FieldValueType</code>	int8uField A <code>FieldValueType</code> that specifies an 8-bit unsigned integer.
<code>static FieldValueType</code>	numberField A <code>FieldValueType</code> that specifies a floating point number.
<code>static FieldValueType</code>	oleField A <code>FieldValueType</code> that specifies an OLE field.
<code>static FieldValueType</code>	persistentMemoField A <code>FieldValueType</code> that specifies a persistent memo.
<code>static FieldValueType</code>	pictureField A <code>FieldValueType</code> that specifies a picture.
<code>static FieldValueType</code>	sameAsInputField A <code>FieldValueType</code> that specifies the same data type as the input field.
<code>static FieldValueType</code>	stringField A <code>FieldValueType</code> that specifies a string.
<code>static FieldValueType</code>	timeField A <code>FieldValueType</code> that specifies a time field.
<code>static FieldValueType</code>	transientMemoField A <code>FieldValueType</code> that specifies a variable length array of characters.
<code>static FieldValueType</code>	unknownField A <code>FieldValueType</code> that specifies an unknown field.

Method Summary

static FieldValueType	from_int (int i) Returns the FieldValueType object corresponding to the specified int.
static FieldValueType	from_string (java.lang.String sVal) Returns the FieldValueType object corresponding to the specified String.
static boolean	isPrimitiveType (FieldValueType type) Returns true if the specified FieldValueType object is a primitive data type.
java.lang.String	toString () Returns the String value of this FieldValueType object.
java.lang.String	toVariantTypeString () Returns the variant type of the String value of this FieldValueType object.
int	value () Returns the int value of this FieldValueType object.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

int8sField

```
public static final int _int8sField
```

8-bit signed integer.

See Also:

Constant Field Values

int8uField

```
public static final int _int8uField
```

8-bit unsigned integer.

See Also:

Constant Field Values

`_int16sField`

```
public static final int _int16sField
```

16-bit signed integer.

See Also:

Constant Field Values

`_int16uField`

```
public static final int _int16uField
```

16-bit unsigned integer.

See Also:

Constant Field Values

`_int32sField`

```
public static final int _int32sField
```

32-bit signed integer.

See Also:

Constant Field Values

`_int32uField`

```
public static final int _int32uField
```

32-bit unsigned integer.

See Also:

Constant Field Values

`_numberField`

```
public static final int _numberField
```

A floating point number. It has a range of approximately 1.7E-308 to 1.7E+308

See Also:

Constant Field Values

`_currencyField`

```
public static final int _currencyField
```

A currency type. This is a floating point number but some databases distinguish between the two.

See Also:

Constant Field Values

`_booleanField`

```
public static final int _booleanField
```

A boolean type.

See Also:

Constant Field Values

`_dateField`

```
public static final int _dateField
```

A date type. The DATE type is implemented as a floating-point value, measuring days from midnight, 30 December 1899. If the number is negative it is the number of days before 30 December 1899.

See Also:

Constant Field Values

`_timeField`

```
public static final int _timeField
```

A time field. It is the number of seconds since midnight.

See Also:

Constant Field Values

`_stringField`

```
public static final int _stringField
```

A string. It has a maximum length of 254 bytes. Strings longer than this are treated as memos.

See Also:

Constant Field Values

`_transientMemoField`

```
public static final int _transientMemoField
```

A variable length array of characters. Transient memos are memos whose data must be read at the same time the database record to which it belongs is being read.

See Also:

Constant Field Values

`_persistentMemoField`

```
public static final int _persistentMemoField
```

The data of persistent memos can be read at any time. That is, at the time the record is read only the memo tag is read and the data can be collected later.

See Also:

Constant Field Values

`_blobField`

```
public static final int _blobField
```

A BLOB is simply a chunk of binary data. There is a 4GB limit on BLOBS.

See Also:

Constant Field Values

`_dateTimeField`

```
public static final int _dateTimeField
```

A date type. The DATE type is implemented as a floating-point value, measuring days from midnight, 30 December 1899. If the number is negative it is the number of days before 30 December 1899. To interpret the time portion, take the absolute value of the fractional part of the number.

See Also:

Constant Field Values

`_bitmapField`

```
public static final int _bitmapField
```

A bitmap.

See Also:

Constant Field Values

`_iconField`

```
public static final int _iconField
```

An icon.

See Also:

Constant Field Values

`_pictureField`

```
public static final int _pictureField
```

A picture.

See Also:

Constant Field Values

`_oleField`

```
public static final int _oleField
```

An OLE field.

See Also:

Constant Field Values

`_chartField`

```
public static final int _chartField
```

A chart.

See Also:

Constant Field Values

`_sameAsInputField`

```
public static final int _sameAsInputField
```

Same data type as input field.

See Also:

Constant Field Values

_unknownField

```
public static final int _unknownField
```

A unknown field. That is, it is not one of the types defined above.

See Also:

Constant Field Values

int8sField

```
public static final FieldValueType int8sField
```

A FieldValueType that specifies an 8-bit signed integer.

int8uField

```
public static final FieldValueType int8uField
```

A FieldValueType that specifies an 8-bit unsigned integer.

int16sField

```
public static final FieldValueType int16sField
```

A FieldValueType that specifies a 16-bit signed integer.

int16uField

```
public static final FieldValueType int16uField
```

A FieldValueType that specifies a 16-bit unsigned integer.

int32sField

```
public static final FieldValueType int32sField
```

A FieldValueType that specifies a 32-bit signed integer.

int32uField

```
public static final FieldValueType int32uField
```

A FieldValueType that specifies a 32-bit unsigned integer.

numberField

```
public static final FieldValueType numberField
```

A FieldValueType that specifies a floating point number.

currencyField

```
public static final FieldValueType currencyField
```

A FieldValueType that specifies a currency type.

booleanField

```
public static final FieldValueType booleanField
```

A FieldValueType that specifies a boolean type.

dateField

```
public static final FieldValueType dateField
```

A FieldValueType that specifies a date type.

timeField

```
public static final FieldValueType timeField
```

A FieldValueType that specifies a time field.

stringField

```
public static final FieldValueType stringField
```

A FieldValueType that specifies a string.

transientMemoField

```
public static final FieldValueType transientMemoField
```

A FieldValueType that specifies a variable length array of characters.

persistentMemoField

```
public static final FieldValueType persistentMemoField
```

A FieldValueType that specifies a persistent memo. The data of persistent memos can be read at any time.

blobField

```
public static final FieldValueType blobField
```

A FieldValueType that specifies a BLOB or in other words, a chunk of binary data.

dateTimeField

```
public static final FieldValueType dateTimeField
```

A FieldValueType that specifies a date type.

bitmapField

```
public static final FieldValueType bitmapField
```

A FieldValueType that specifies a bitmap.

iconField

```
public static final FieldValueType iconField
```

A FieldValueType that specifies an icon.

pictureField

```
public static final FieldValueType pictureField
```

A `FieldValueType` that specifies a picture.

oleField

```
public static final FieldValueType oleField
```

A `FieldValueType` that specifies an OLE field.

chartField

```
public static final FieldValueType chartField
```

A `FieldValueType` that specifies a chart.

sameAsInputField

```
public static final FieldValueType sameAsInputField
```

A `FieldValueType` that specifies the same data type as the input field.

unknownField

```
public static final FieldValueType unknownField
```

A `FieldValueType` that specifies an unknown field.

Method Detail

from_int

```
public static final FieldValueType from_int(int i)
```

Returns the `FieldValueType` object corresponding to the specified `int`.

Parameters:

`i` – The `int` value of the desired `FieldValueType`.

Returns:

The `FieldValueType` object corresponding to the specified `int`.

from_string

```
public static final FieldValueType from_string(java.lang.String sVal)
```

Returns the `FieldValueType` object corresponding to the specified `String`.

Parameters:

sVal – The String value of the desired FieldValueType.

Returns:

The FieldValueType object corresponding to the specified String.

isPrimitiveType

```
public static final boolean isPrimitiveType(FieldValueType type)
```

Returns true if the specified FieldValueType object is a primitive data type.

Parameters:

type – The FieldValueType object to be tested.

Returns:

true if the specified FieldValueType object is a primitive data type, and false otherwise.

toString

```
public java.lang.String toString()
```

Returns the String value of this FieldValueType object.

Returns:

The String value of this FieldValueType object.

toVariantTypeString

```
public java.lang.String toVariantTypeString()
```

Returns the variant type of the String value of this FieldValueType object.

Returns:

A String that specifies the variant type of this FieldValueType object.

value

```
public int value()
```

Returns the int value of this FieldValueType object.

Returns:

The int value of this FieldValueType object.

com.crystaldecisions.sdk.occa.report.data Class FormulaField

```
java.lang.Object
├─ com.crystaldecisions.sdk.occa.report.data.Field
│   └─ com.crystaldecisions.sdk.occa.report.data.FormulaField
```

All Implemented Interfaces:

IClone, IField, IFormulaField

public class FormulaField*extends Field**implements IFormulaField, IClone*

This object implements a formula field in the report. Use the IFormulaField interface to access this object. To permanently modify formula fields in a report, a Report Application Server is required. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

Constructor Summary	
FormulaField()	
FormulaField(IFormulaField src)	

Method Summary	
java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
boolean	getIsRecurring () Returns true if the field is recurring, and false otherwise.
FieldKind	getKind () Returns what kind of field this is.
int	getOptions () For internal use only.
java.lang.String	getShortName (java.util.Locale locale) Returns the short name of the field.
FormulaSyntax	getSyntax () Returns the syntax used to write the formula.

IDE Extensions

java.lang.String	getText() Returns the formula string.
boolean	hasContent (java.lang.Object srcFormulaField) Returns true if this object contains the same elements as the passed in object.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
void	save (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	setOptions (int options) For internal use only.
void	setSyntax (FormulaSyntax syntax) Sets the syntax used to write the formula.
void	setText (java.lang.String text) Sets the formula string.
void	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.

Methods inherited from class com.crystaldecisions.sdk.occa.report.data.Field

createMember, getDescription, getDisplayName, getFormulaForm, getHeadingText, getLength, getLongName, getName, getType, save, save, setDescription, setHeadingText, setLength, setName, setType

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface com.crystaldecisions.sdk.occa.report.data.IField

getDescription, getDisplayName, getFormulaForm, getHeadingText, getLength, getLongName, getName, getType, setDescription, setHeadingText, setLength, setName, setType

Constructor Detail

FormulaField

```
public FormulaField(IFormulaField src)
```

FormulaField

```
public FormulaField()
```

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: IClone

Returns the new object that has been cloned.

Specified by:

clone in interface IClone

Overrides:

clone in class Field

copyTo

```
public void copyTo(java.lang.Object destObject,
                  boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

copyTo in interface IClone

Overrides:

copyTo in class Field

createMember

```
public java.lang.Object createMember(java.lang.String eleName,
                                     org.xml.sax.Attributes attrs,
                                     XMLSerializationContext ctxt,
                                     java.util.Map objState,
                                     boolean[] bLoaded)
```

For internal use only.

Overrides:

createMember in class Field

endElement

```
public void endElement(java.lang.String eleName,
                       java.util.Map objState)
```

Description copied from class: Field

For internal use only.

Overrides:

endElement in class Field

getShortName

```
public java.lang.String getShortName(java.util.Locale locale)
```

Description copied from interface: IField

Returns the short name of the field.

Specified by:

getShortName in interface IField

Overrides:

getShortName in class Field

getSyntax

```
public FormulaSyntax getSyntax()
```

Description copied from interface: IFormulaField

Returns the syntax used to write the formula.

Specified by:

getSyntax in interface IFormulaField

Returns:

The FormulaSyntax object.

getText

```
public java.lang.String getText()
```

Description copied from interface: IFormulaField

Returns the formula string.

Specified by:

getText in interface IFormulaField

Returns:

The formula as a String.

readElement

```
public void readElement(java.lang.String eleName,
                        java.lang.String sVal,
                        org.xml.sax.Attributes attrs,
                        java.util.Map objState)
```

Description copied from class: Field

For internal use only.

Overrides:

readElement in class Field

save

```
public void save(XMLWriter writer,
                 java.lang.String sTag,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Overrides:

save in class Field

Throws:

java.io.IOException

save

```
public void save(XMLWriter writer,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Overrides:

save in class Field

Throws:

java.io.IOException

saveContents

```
public void saveContents(XMLWriter writer,
                         XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Overrides:

saveContents in class Field

Throws:

java.io.IOException

setSyntax

```
public void setSyntax(FormulaSyntax syntax)
```

Description copied from interface: IFormulaField

Sets the syntax used to write the formula.

Specified by:

setSyntax in interface IFormulaField

Parameters:

syntax – The FormulaSyntax object.

setText

```
public void setText(java.lang.String text)
```

Description copied from interface: IFormulaField

Sets the formula string.

Specified by:

setText in interface IFormulaField

Parameters:

text – The formula as a String.

startElement

```
public void startElement(java.lang.String eleName,  
                           java.util.Map objState,  
                           org.xml.sax.Attributes attrs)
```

Description copied from class: Field

For internal use only.

Overrides:

startElement in class Field

getIsRecurring

```
public boolean getIsRecurring()
```

Description copied from interface: IField

Returns true if the field is recurring, and false otherwise. A field is recurring if its contents change during the formatting of a report. A constant field (not recurring) is one that doesn't change. The special field "Page Number," for example, is a recurring field, while the special field "Report Title" is not.

Specified by:

getIsRecurring in interface IField

Overrides:

getIsRecurring in class Field

getKind

```
public FieldKind getKind()
```

Description copied from interface: IField

Returns what kind of field this is. Do not confuse this property with Type. Kind identifies what sort of report field this is, while Type identifies what the field is composed of. For example, a bitmap, string, chart, number, date, and so on.

Specified by:

getKind in interface IField

Returns:

A String that specifies what kind of field this is.

getOptions

```
public int getOptions()
```

Description copied from interface: IFormulaField

For internal use only.

Specified by:

getOptions in interface IFormulaField

hasContent

```
public boolean hasContent(java.lang.Object srcFormulaField)
```

Description copied from interface: IClone

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Overrides:

hasContent in class Field

setOptions

```
public void setOptions(int options)
```

Description copied from interface: IFormulaField

For internal use only.

Specified by:

setOptions in interface IFormulaField

com.crystaldecisions.sdk.occa.report.data Interface IConnectionInfo

All Known Implementing Classes:

public interface **ICConnectionInfo**

This interface enables you to get and set information for the data source connection.

Method Summary

PropertyBag	getAttributes () Returns the property bag for the data source connection.
ConnectionInfoKind	getKind () Returns the kind of connection.
java.lang.String	getPassword () Returns the password used to connect to the data source.
java.lang.String	getUserName () Returns the user name used to connect to the data source.
boolean	isMatch (ICConnectionInfo info, boolean completeMatching) Checks whether two connections match.
void	setAttributes (PropertyBag attributes) Sets the property bag for the data source connection.
void	setKind (ConnectionInfoKind kind) Returns the kind of connection.
void	setPassword (java.lang.String password) Returns the password used to connect to the data source.
void	setUserName (java.lang.String userName) Sets the user name used to connect to the data source.

Method Detail**getAttributes**

```
public PropertyBag getAttributes()
```

Returns the property bag for the data source connection.

Returns:

The property bags as a PropertyBag object.

getKind

```
public ConnectionInfoKind getKind()
```

Returns the kind of connection. For example, connection kinds include SQL, query, meta data, database file, and Crystal Report Query Engine (CRQE).

Returns:

The connection kind as a `ConnectionInfoKind` object.

getPassword

```
public java.lang.String getPassword()
```

Returns the password used to connect to the data source.

Returns:

The password as a `String`.

getUserName

```
public java.lang.String getUserName()
```

Returns the user name used to connect to the data source.

Returns:

The user name as a `String`.

isMatch

```
public boolean isMatch(IConnectionInfo info,  
                        boolean completeMatching)
```

Checks whether two connections match. Returns `true` if they match, and `false` otherwise.

Parameters:

`info` – The `IConnectionInfo` object that this connection will be compared against.
`completeMatching` – A boolean specifying whether to use complete matching. By default, this is set to `true`. If `true`, all members of the object are compared. When all members of the object have been compared, a value of `true` is returned if all members are exactly the same. If set to `false`, all members of the object are compared and a value of `true` is returned if the mismatch is caused because one of the comparing members has an empty string, but all of the remaining members match.

Returns:

`true` if they match, and `false` otherwise.

setAttributes

```
public void setAttributes(PropertyBag attributes)
```

Sets the property bag for the data source connection.

Parameters:

`attributes` – The property bag as a `PropertyBag` object.

setKind

```
public void setKind(ConnectionInfoKind kind)
```

Returns the kind of connection. For example, connection kinds include SQL, query, meta data, database file, and Crystal Report Query Engine (CRQE).

Parameters:

`kind` – The connection kind as a `ConnectionInfoKind` object.

setPassword

```
public void setPassword(java.lang.String password)
```

Returns the password used to connect to the data source.

Parameters:

`password` – The password as a `String`.

setUserName

```
public void setUserName(java.lang.String userName)
```

Sets the user name used to connect to the data source.

Parameters:

`userName` – The user name as a `String`.

com.crystaldecisions.sdk.occa.report.data Interface IField

All Known Subinterfaces:

`IFormulaField`, `IParameterField`

All Known Implementing Classes:

`Field`, `FormulaField`, `ParameterField`

public interface **IField**

This interface defines a report field in general. It is used as an abstract base from which other report fields inherit, and it contains methods and properties that are common to all report fields.

Method Summary	
java.lang.String	getDescription() Returns a description of the field.
java.lang.String	getDisplayName (FieldDisplayNameType displayNameType, java.util.Locale locale) Returns the name of the field that has been formatted according to the value of the parameter displayNameType.
java.lang.String	getFormulaForm() Returns the name of the field as it will be used in a formula.
java.lang.String	getHeadingText() Returns the text used as a heading when the field is added to the report.
boolean	getIsRecurring() Returns true if the field is recurring, and false otherwise.
FieldKind	getKind() Returns what kind of field this is.
int	getLength() Returns the maximum length the field's data may be (in bytes).
java.lang.String	getLongName (java.util.Locale locale) Returns the fully qualified name of the field.
java.lang.String	getName() Returns the name of the field.
java.lang.String	getShortName (java.util.Locale locale) Returns the short name of the field.
FieldValueType	getType() Returns what type of field this is.
void	setDescription (java.lang.String description) Sets the description of the field.
void	setHeadingText (java.lang.String headingText) Sets the text used as a heading when the field is added to the report.
void	setLength (int length) Sets the maximum length the field's data may be (in bytes).
void	setName (java.lang.String name) Sets the name of the field.
void	setType (FieldValueType valueType) Sets what type of field this is.

Method Detail

getDescription

```
public java.lang.String getDescription()
```

Returns a description of the field.

Returns:

A `String` containing a description of the field.

getHeadingText

```
public java.lang.String getHeadingText()
```

Returns the text used as a heading when the field is added to the report.

Returns:

A `String` containing the text used as a heading when the field is added to the report.

getFormulaForm

```
public java.lang.String getFormulaForm()
```

Returns the name of the field as it will be used in a formula. This is a unique identifier to avoid ambiguity in formula code. It is the same value as the one used in the CRW formula editor.

Returns:

A `String` containing the name of the field as it will be used in a formula.

getIsRecurring

```
public boolean getIsRecurring()
```

Returns `true` if the field is recurring, and `false` otherwise. A field is recurring if its contents change during the formatting of a report. A constant field (not recurring) is one that doesn't change. The special field "Page Number," for example, is a recurring field, while the special field "Report Title" is not.

Returns:

`true` if the field is recurring, and `false` otherwise.

getKind

```
public FieldKind getKind()
```

Returns what kind of field this is. Do not confuse this property with `Type`. `Kind` identifies what sort of report field this is, while `Type` identifies what the field is composed of. For example, a bitmap, string, chart, number, date, and so on.

Returns:

A `String` that specifies what kind of field this is.

getLength

```
public int getLength()
```

Returns the maximum length the field's data may be (in bytes).

Returns:

A String containing the maximum length the field's data may be (in bytes).

getLongName

```
public java.lang.String getLongName(java.util.Locale locale)
```

Returns the fully qualified name of the field.

Parameters:

locale – The locale that will be used to localize the field.

Returns:

A String containing the fully qualified name of the field.

getName

```
public java.lang.String getName()
```

Returns the name of the field. For DBField objects, the value returned by this method is equivalent to the getDisplayName method.

Returns:

A String containing the name of the field.

getShortName

```
public java.lang.String getShortName(java.util.Locale locale)
```

Returns the short name of the field.

Parameters:

locale – The locale that will be used to localize the field name.

Returns:

A String containing the short name of the field.

setDescription

```
public void setDescription(java.lang.String description)
```

Sets the description of the field.

Parameters:

description – A String containing a description of the field.

setHeadingText

```
public void setHeadingText(java.lang.String headingText)
```

Sets the text used as a heading when the field is added to the report.

Parameters:

headingText – A String containing the text used as a heading when the field is added to the report.

setLength

```
public void setLength(int length)
```

Sets the maximum length the field's data may be (in bytes).

Parameters:

length – A String containing the maximum length the field's data may be (in bytes).

setName

```
public void setName(java.lang.String name)
```

Sets the name of the field. For DBField objects, the value returned by this method is equivalent to the getDisplayName method.

Parameters:

name – A String containing the name of the field.

getDisplayName

```
public java.lang.String getDisplayName(FieldDisplayNameType displayNameType,  
                                         java.util.Locale locale)
```

Returns the name of the field that has been formatted according to the value of the parameter displayNameType.

Parameters:

displayNameType – Indicates how the display text should be formatted. It may be one of the values listed under FieldDisplayNameType.
locale – Formats the string according to a locale.

Returns:

A String containing the name of the field that has been formatted according to the value of the parameter displayNameType.

getType

```
public FieldValueType getType()
```

Returns what type of field this is. Do not confuse this property with Kind. Kind identifies what sort of report field this is, while Type identifies what data it stores. For example, a bitmap, string, 8 bit integer, chart, etc.

Returns:

A `FieldValueType` object that specifies what type of field this is.

setType

```
public void setType(FieldValueType valueType)
```

Sets what type of field this is. Do not confuse this property with Kind. Kind identifies what sort of report field this is, while Type identifies what data it stores. For example, a bitmap, string, 8 bit integer, chart, etc.

Parameters:

`valueType` – A `FieldValueType` object that specifies what type of field this is.

com.crystaldecisions.sdk.occa.report.data

Interface IFormulaField

All Superinterfaces:

`IField`

All Known Implementing Classes:

`FormulaField`

```
public interface IFormulaField
extends IField
```

This interface defines a formula field in the report.

Method Summary

<code>int</code>	getOptions () For internal use only.
<code>FormulaSyntax</code>	getSyntax () Returns the syntax used to write the formula.
<code>java.lang.String</code>	getText () Returns the formula string.
<code>void</code>	setOptions (int options) For internal use only.
<code>void</code>	setSyntax (FormulaSyntax syntax) Sets the syntax used to write the formula.
<code>void</code>	setText (java.lang.String text) Sets the formula string.

Methods inherited from interface com.crystaldecisions.sdk.occa.report.data.IField

getDescription, getDisplayName, getFormulaForm, getHeadingText, getIsRecurring, getKind, getLength, getLongName, getName, getShortName, getType, setDescription, setHeadingText, setLength, setName, setType

Method Detail**getText**

```
public java.lang.String getText()
```

Returns the formula string.

Returns:

The formula as a String.

setText

```
public void setText(java.lang.String text)
```

Sets the formula string.

Parameters:

text – The formula as a String.

getSyntax

```
public FormulaSyntax getSyntax()
```

Returns the syntax used to write the formula.

Returns:

The FormulaSyntax object.

setSyntax

```
public void setSyntax(FormulaSyntax syntax)
```

Sets the syntax used to write the formula.

Parameters:

syntax – The FormulaSyntax object.

getOptions

```
public int getOptions()
```

For internal use only.

setOptions

```
public void setOptions(int options)
```

For internal use only.

com.crystaldecisions.sdk.occa.report.data Interface IParameterField

All Superinterfaces:

IField

All Known Implementing Classes:

ParameterField

public interface IParameterField

extends IField

This interface is used to get and set values for the parameter field. Parameters prompt the user of a report to enter information.

Think of a parameter as a question that the user needs to answer before the report is generated. The information users enter, or the way they respond, determines what appears in the report. For example, in a report used by sales people, there might be a parameter that asks the user to choose a region. The report would return the results for the specific region instead of returning the results for all of the regions.

By using parameter fields in formulas, selection formulas, and in the report itself, you can create a single report that you can modify whenever your needs change. Parameter fields can also be used in subreports. Parameter values are discrete or ranged. A discrete value represents a particular, single value, while a range value represents a value between a certain range. Additionally, parameters can have one value (discrete or ranged) or multiple values (discrete or ranged).

The `ParameterField` object allows you to manipulate a parameter in the report.

Method Summary	
boolean	getAllowCustomCurrentValues() Returns <code>true</code> if the user can enter custom values for a parameter when the report is refreshed and <code>false</code> otherwise.
boolean	

IDE Extensions

	getAllowMultiValue() Returns true whether the user can select more than one value for a parameter when the report is refreshed; returns false if the user can only choose one value for the parameter.
boolean	getAllowNullValue() Returns true if the value for the parameter may be null and false otherwise.
IField	getBrowseField() Returns the database field whose values are being used as default values for the parameter.
Values	getCurrentValues() Returns the values that are currently being used for the parameter.
ParameterDefaultValueDisplayType	getDefaultValueDisplayType() For internal use only.
Values	getDefaultValues() Returns the default values that may be used for the parameter.
ParameterSortMethod	getDefaultValueSortMethod() Returns whether the default values are sorted by value or by the value's description.
ParameterSortOrder	getDefaultValueSortOrder() Returns the manner in which the values or the values' descriptions are sorted.
java.lang.String	getEditMask() Returns the edit mask for the parameter values.
IParameterFieldDiscreteValue	getMaximumValue() Returns the maximum value the parameter can have, providing that a minimum value is also specified.
IParameterFieldDiscreteValue	getMinimumValue() Returns the minimum value the parameter can have, providing that a maximum value is also specified.
ParameterFieldType	getParameterType() Returns how the parameter is being used—that is, its type.
java.lang.String	getReportName() Returns the name of the report to which the parameter belongs.
ParameterValueRangeKind	getValueRangeKind() Returns whether the parameter is discrete, ranged, or both.
Values	getValues() Returns the current values for the parameter, or the default values if no current values are defined.
void	setAllowCustomCurrentValues(boolean allowCustomCurr Sets if the user can enter custom values for a parameter when

IDE Extensions

		the report is refreshed.
void	setAllowMultiValue (boolean allowMultiValue)	Sets whether the user can select more than one value for a parameter when the report is refreshed.
void	setAllowNullValue (boolean allowNullValue)	Set to true if the value for the parameter may be null and false otherwise.
void	setBrowseField (IField browseField)	Sets the database field whose values are being used as default values for the parameter.
void	setCurrentValues (Values currentValues)	Sets the values that are currently being used for the parameter.
void	setDefaultValueDisplayType (ParameterDefaultValueDisplayType defaultValueDisplayType)	For internal use only.
void	setDefaultValues (Values defaultValues)	Sets the default values that may be used for the parameter.
void	setDefaultValueSortMethod (ParameterSortMethod defaultValueSortMethod)	Sets whether the default values are sorted by value or by the value's description.
void	setDefaultValueSortOrder (ParameterSortOrder defaultValueSortOrder)	Sets the manner in which the values or the values' descriptions are sorted.
void	setEditMask (java.lang.String editMask)	Sets the edit mask for the parameter values.
void	setMaximumValue (IPParameterFieldDiscreteValue maximumValue)	Sets the maximum value the parameter can have, providing that a minimum value is also specified.
void	setMinimumValue (IPParameterFieldDiscreteValue minimumValue)	Sets the minimum value the parameter can have, providing that a maximum value is also specified.
void	setParameterType (ParameterFieldType parameterType)	Sets how the parameter is being used—that is, its type.
void	setReportName (java.lang.String reportName)	Sets the name of the report to which the parameter belongs.
void	setValueRangeKind (ParameterValueRangeKind valueRangeKind)	Sets whether the parameter is discrete, ranged, or both.

Methods inherited from interface com.crystaldecisions.sdk.occa.report.data.IField

getDescription, getDisplayName, getFormulaForm, getHeadingText, getIsRecurring, getKind, getLength, getLongName, getName, getShortName,

<code>getType, setDescription, setHeadingText, setLength, setName, setType</code>

Method Detail

getBrowseField

```
public IField getBrowseField()
```

Returns the database field whose values are being used as default values for the parameter. This is used for UI-display purposes.

Returns:

The database field whose values are being used as default values for the parameter as an IField interface.

setBrowseField

```
public void setBrowseField(IField browseField)
```

Sets the database field whose values are being used as default values for the parameter. This is used for UI-display purposes.

Parameters:

`browseField` – The database field whose values are being used as default values for the parameter as an IField interface.

getAllowMultiValue

```
public boolean getAllowMultiValue()
```

Returns `true` whether the user can select more than one value for a parameter when the report is refreshed; returns `false` if the user can only choose one value for the parameter.

Returns:

`true` if the user can select more than one value for a parameter when the report is refreshed, and `false` otherwise.

setAllowMultiValue

```
public void setAllowMultiValue(boolean allowMultiValue)
```

Sets whether the user can select more than one value for a parameter when the report is refreshed.

Parameters:

`allowMultiValue` – `true` if the user can select more than one value for a parameter when the report is refreshed, and `false` otherwise.

getReportName

```
public java.lang.String getReportName()
```

Returns the name of the report to which the parameter belongs. This string is empty if the parameter belongs in a main report; if the parameter is contained in a subreport, this contains the name of the subreport.

Returns:

The name of the report as a `String`.

getDefaultValueDisplayType

```
public ParameterDefaultValueDisplayType getDefaultValueDisplayType()
```

For internal use only.

setReportName

```
public void setReportName(java.lang.String reportName)
```

Sets the name of the report to which the parameter belongs. This string is empty if the parameter belongs in a main report; if the parameter is contained in a subreport, this contains the name of the subreport.

Parameters:

`reportName` – The name of the report as a `String`. Set this string as empty if the parameter belongs in a main report; if the parameter is contained in a subreport, set this to contain the name of the subreport.

getParameterType

```
public ParameterFieldType getParameterType()
```

Returns how the parameter is being used—that is, its type.

Returns:

The parameter field type as a `ParameterFieldType` object.

setParameterType

```
public void setParameterType(ParameterFieldType parameterType)
```

Sets how the parameter is being used—that is, its type.

Parameters:

`parameterType` – The parameter field type as a `ParameterFieldType` object.

getAllowNullValue

```
public boolean getAllowNullValue()
```

Returns `true` if the value for the parameter may be `null` and `false` otherwise. This is used only for stored SQL procedures.

Returns:

`true` if the value for the parameter may be `null`, and `false` otherwise.

setAllowNullValue

```
public void setAllowNullValue(boolean allowNullValue)
```

Set to `true` if the value for the parameter may be `null` and `false` otherwise. This is used only for stored SQL procedures.

Parameters:

`allowNullValue` – `true` if the value for the parameter may be `null`, and `false` otherwise.

getMinimumValue

```
public IParameterFieldDiscreteValue getMinimumValue()
```

Returns the minimum value the parameter can have, providing that a maximum value is also specified.

If the parameter is discrete, the discrete values must be between the minimum and maximum values. If the parameter has a range limit, both the maximum and minimum values are defined. The `BeginValue` and/or `EndValue` of each ranged value must be between the minimum and maximum values.

If the parameter is a number, this property represents the smallest allowable number the parameter may have. If it is a string, this is the minimum length the string may be. If it is a Date/Time value, this is the earliest date and time the parameter may be. This property does not apply to Boolean values.

Returns:

The `IParameterFieldDiscreteValue` object.

setMinimumValue

```
public void setMinimumValue(IParameterFieldDiscreteValue minimumValue)
```

Sets the minimum value the parameter can have, providing that a maximum value is also specified.

If the parameter is discrete, the discrete values must be between the minimum and maximum values. If the parameter has a range limit, both the maximum and minimum values are defined. The `BeginValue` and/or `EndValue` of each ranged value must be between the minimum and maximum values.

If the parameter is a number, this property represents the smallest allowable number the parameter may have. If it is a string, this is the minimum length the string may be. If it is a Date/Time value, this is the earliest date and time the parameter may be. This property does not apply to Boolean values.

Parameters:

`minimumValue` – The `IPParameterFieldDiscreteValue` object.

getMaximumValue

```
public IPParameterFieldDiscreteValue getMaximumValue()
```

Returns the maximum value the parameter can have, providing that a minimum value is also specified.

If the parameter is discrete, the discrete values must be between the minimum and maximum values. If the parameter has a range limit, both the maximum and minimum values are defined. The `BeginValue` and/or `EndValue` of each ranged value must be between the minimum and maximum values.

If the parameter is a number, this property represents the largest allowable number the parameter may have. If it is a string, this is the maximum length the string may be. If it is a Date/Time value, this is the earliest date and time the parameter may be. This property does not apply to Boolean values.

Returns:

The `IPParameterFieldDiscreteValue` object.

setMaximumValue

```
public void setMaximumValue(IPParameterFieldDiscreteValue maximumValue)
```

Sets the maximum value the parameter can have, providing that a minimum value is also specified.

If the parameter is discrete, the discrete values must be between the minimum and maximum values. If the parameter has a range limit, both the maximum and minimum values are defined. The `BeginValue` and/or `EndValue` of each ranged value must be between the minimum and maximum values.

If the parameter is a number, this property represents the largest allowable number the parameter may have. If it is a string, this is the maximum length the string may be. If it is a Date/Time value, this is the earliest date and time the parameter may be. This property does not apply to Boolean values.

Parameters:

`maximumValue` – The `IPParameterFieldDiscreteValue` object.

getEditMask

```
public java.lang.String getEditMask()
```

Returns the edit mask for the parameter values. This is used for UI-display purposes. The edit mask can be any of a set of masking characters used to restrict the values you can enter as parameter values.

IDE Extensions

The edit mask also limits the values you can enter as default prompting values.

Note: The value of the edit mask is not checked if you use the SDK to change parameter values. You must check to ensure that any values the user enters respect the edit mask. If a user enters an invalid value according to the edit mask, it will not be possible to view or schedule the report.

The edit mask follows the following guidelines:

- ◇ "A" (allows an alphanumeric character and requires the entry of a character in the parameter value)
- ◇ "a" (allows an alphanumeric character and does not require the entry of a character in the parameter value)
- ◇ "0" (allows a digit [0 to 9] and requires the entry of a character in the parameter value)
- ◇ "9" (allows a digit or a space, and does not require the entry of a character in the parameter value)
- ◇ "#" (allows a digit, space, or plus/minus sign, and does not require the entry of a character in the parameter value)
- ◇ "L" (allows a letter [A to Z], and requires the entry of a character in the parameter value)
- ◇ "?" (allows a letter, and does not require the entry of a character in the parameter value)
- ◇ "" (allows any character or space, and requires the entry of a character in the parameter value)
- ◇ "C" (allows any character or space, and does not require the entry of a character in the parameter value)
- ◇ ". , ; - /" (separator characters). Inserting separator characters into an Edit Mask is something like hard coding the formatting for the parameter field. When the field is placed on the report, the separator character will appear in the field object frame, like this: LLLL/0000. This example depicts an edit mask that requires four letters followed by four numbers.
- ◇ ">" (causes subsequent characters to be converted to uppercase)
- ◇ "\" (causes the subsequent character to be displayed as a literal). For example, the edit mask "\"A" would display a parameter value of "A." If the edit mask is "00\\A00," then a valid parameter value would consist of two digits, the letter "A," and then two additional digits.
- ◇ "Password". Allows you to set the edit mask to "Password." You can create conditional formulas that specify that certain sections of the report become visible only when certain user passwords are entered.

Note: Some of the edit mask characters require that you enter a character in their place (when entering a parameter value), while others allow you to leave a space, if needed. For example, if the edit mask is 000099, you can enter a parameter value with four digits, five digits, or six digits, since the '9' edit mask character does not require the entry of a character. However, since '0' does require such an entry, you could not enter a parameter value with less than four digits.

Returns:

The edit mask as a `String`.

setEditMask

```
public void setEditMask(java.lang.String editMask)
```

Sets the edit mask for the parameter values. This is used for UI–display purposes. The edit mask can be any of a set of masking characters used to restrict the values you can enter as parameter values. The edit mask also limits the values you can enter as default prompting values.

Note: The value of the edit mask is not checked if you use the SDK to change parameter values. You must check to ensure that any values the user enters respect the edit mask. If a user enters an invalid value according to the edit mask, it will not be possible to view or schedule the report.

The edit mask follows the following guidelines:

- ◇ "A" (allows an alphanumeric character and requires the entry of a character in the parameter value)
- ◇ "a" (allows an alphanumeric character and does not require the entry of a character in the parameter value)
- ◇ "0" (allows a digit [0 to 9] and requires the entry of a character in the parameter value)
- ◇ "9" (allows a digit or a space, and does not require the entry of a character in the parameter value)
- ◇ "#" (allows a digit, space, or plus/minus sign, and does not require the entry of a character in the parameter value)
- ◇ "L" (allows a letter [A to Z], and requires the entry of a character in the parameter value)
- ◇ "?" (allows a letter, and does not require the entry of a character in the parameter value)
- ◇ "" (allows any character or space, and requires the entry of a character in the parameter value)
- ◇ "C" (allows any character or space, and does not require the entry of a character in the parameter value)
- ◇ ". , : ; - /" (separator characters). Inserting separator characters into an edit mask is something like hard coding the formatting for the parameter field. When the field is placed on the report, the separator character will appear in the field object frame, like this: LLLL/0000. This example depicts an edit mask that requires four letters followed by four numbers.
- ◇ "<" (causes subsequent characters to be converted to lowercase)
- ◇ ">" (causes subsequent characters to be converted to uppercase)
- ◇ "\" (causes the subsequent character to be displayed as a literal). For example, the edit mask "\"A" would display a parameter value of "A." If the edit mask is "00\\A00," then a valid parameter value would consist of two digits, the letter "A," and then two additional digits.
- ◇ "Password". Allows you to set the edit mask to "Password." You can create conditional formulas that specify that certain sections of the report become visible only when certain user passwords are entered.

Note: Some of the edit mask characters require that you enter a character in their place (when entering a parameter value), while others allow you to leave a space, if needed. For example, if the Edit Mask is 000099, you can enter a parameter value with four digits, five digits, or six digits, since the '9' edit mask character does not require the entry of a character. However, since '0' does require such an entry, you could not enter a parameter value with less than four digits.

Parameters:

editMask – The edit mask as a String.

getAllowCustomCurrentValues

```
public boolean getAllowCustomCurrentValues()
```

Returns `true` if the user can enter custom values for a parameter when the report is refreshed and `false` otherwise. If this property is `true`, users can enter any value for the parameter. If it is `false`, they must choose from one of the default values.

Returns:

`true` if the user can enter custom values for a parameter when the report is refreshed, and `false` otherwise.

setAllowCustomCurrentValues

```
public void setAllowCustomCurrentValues(boolean allowCustomCurrentValues)
```

Sets if the user can enter custom values for a parameter when the report is refreshed. If this property is true, users can enter any value for the parameter. If it is false, they must choose from one of the default values.

Parameters:

allowCustomCurrentValues – true if the user can enter custom values for a parameter when the report is refreshed, and false otherwise.

getDefaultValueSortOrder

```
public ParameterSortOrder getDefaultValueSortOrder()
```

Returns the manner in which the values or the values' descriptions are sorted.

Returns:

The manner in which the values or the values' descriptions are sorted as a ParameterSortOrder object.

setDefaultValueSortOrder

```
public void setDefaultValueSortOrder(ParameterSortOrder defaultValueSortOrder)
```

Sets the manner in which the values or the values' descriptions are sorted.

Parameters:

defaultValueSortOrder – The manner in which the values or the values' descriptions are sorted as a ParameterSortOrder object.

setDefaultValueDisplayType

```
public void setDefaultValueDisplayType(ParameterDefaultValueDisplayType type)
```

For internal use only.

getDefaultValueSortMethod

```
public ParameterSortMethod getDefaultValueSortMethod()
```

Returns whether the default values are sorted by value or by the value's description. This is used for UI–display purposes.

Returns:

Whether the default values are sorted by value or by the value's description as a ParameterSortMethod object.

setDefaultValueSortMethod

```
public void setDefaultValueSortMethod(ParameterSortMethod defaultValueSortMethod)
```

Sets whether the default values are sorted by value or by the value's description. This is used for UI-display purposes.

Parameters:

defaultValueSortMethod – Whether the default values are sorted by value or by the value's description as a ParameterSortMethod object.

getValueRangeKind

```
public ParameterValueRangeKind getValueRangeKind()
```

Returns whether the parameter is discrete, ranged, or both.

Returns:

Whether the parameter is discrete, ranged, or both as a ParameterValueRangeKind object.

setValueRangeKind

```
public void setValueRangeKind(ParameterValueRangeKind valueRangeKind)
```

Sets whether the parameter is discrete, ranged, or both.

Parameters:

valueRangeKind – Whether the parameter is discrete, ranged, or both as a ParameterValueRangeKind object.

getCurrentValues

```
public Values getCurrentValues()
```

Returns the values that are currently being used for the parameter. To permanently set this value, a Report Application Server is required. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

Returns:

The current values as a Values object.

getDefaultValues

```
public Values getDefaultValues()
```

Returns the default values that may be used for the parameter.

Returns:

The default values that may be used for the parameter as a `Values` object.

getValues

```
public Values getValues()
```

Returns the current values for the parameter, or the default values if no current values are defined. This is equivalent to the `getCurrentValues()` method unless it is empty, in which case it is equivalent to the `getDefaultValues()` method.

Returns:

The values as a `Values` object.

setCurrentValues

```
public void setCurrentValues(Values currentValues)
```

Sets the values that are currently being used for the parameter. To permanently set this value, a Report Application Server is required. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

Parameters:

`currentValues` – The current values as a `Values` object.

setDefaultValues

```
public void setDefaultValues(Values defaultValues)
```

Sets the default values that may be used for the parameter.

Parameters:

`defaultValues` – The default values that may be used for the parameter as a `Values` object.

com.crystaldecisions.sdk.occa.report.data Interface IParameterFieldDiscreteValue

All Superinterfaces:

`IParameterFieldValue`, `IValue`

All Known Implementing Classes:

`ParameterFieldDiscreteValue`

```
public interface IParameterFieldDiscreteValue
extends IParameterFieldValue
```

This interface is used to get and set a discrete value belonging to a parameter. A discrete value belonging to a parameter. A discrete value is a value that does not have a range—that is, it is singular. Examples of discrete

values are: 1, 56.4, "Jeremy", and so on.

Method Summary	
<code>java.lang.Object</code>	getValue() Returns the discrete value of the parameter.
<code>void</code>	setValue(java.lang.Object value) Sets the discrete value of the parameter.

Methods inherited from interface <code>com.crystaldecisions.sdk.occa.report.data.IParameterFieldValue</code>
<code>getDescription, setDescription</code>

Methods inherited from interface <code>com.crystaldecisions.sdk.occa.report.data.IValue</code>
<code>computeText, displayText</code>

Method Detail

getValue

```
public java.lang.Object getValue()
```

Returns the discrete value of the parameter.

Returns:

The value as an object.

setValue

```
public void setValue(java.lang.Object value)
```

Sets the discrete value of the parameter.

Parameters:

`value` – The value as an object.

`com.crystaldecisions.sdk.occa.report.data` Interface **IParameterFieldRangeValue**

All Superinterfaces:

`IParameterFieldValue, IValue`

All Known Implementing Classes:
 ParameterFieldRangeValue

*public interface **IPParameterFieldRangeValue***
*extends **IPParameterFieldValue***

This interface is used to get and set values for a ranged parameter. The `ParameterFieldRangeValue` object stores the value for a ranged parameter. The object contains the upper bound, the lower bound range, and how the two are treated: that is if they exist, are included, and so on.

Method Summary	
<code>java.lang.Object</code>	getBeginValue () Returns the lower-bound value of the range.
<code>java.lang.Object</code>	getEndValue () Returns the upper-bound value of the range.
<code>RangeValueBoundType</code>	getLowerBoundType () Returns how the lower bound of the range is treated.
<code>RangeValueBoundType</code>	getUpperBoundType () Returns how the upper bound of the range is treated.
<code>void</code>	setBeginValue (<code>java.lang.Object beginValue</code>) Sets the lower-bound value of the range.
<code>void</code>	setEndValue (<code>java.lang.Object endValue</code>) Returns the upper-bound value of the range.
<code>void</code>	setLowerBoundType (<code>RangeValueBoundType lowerBoundType</code>) Sets how the lower bound of the range is treated.
<code>void</code>	setUpperBoundType (<code>RangeValueBoundType upperBoundType</code>) Sets how the upper bound of the range is treated.

Methods inherited from interface <code>com.crystaldecisions.sdk.occa.report.data.IPParameterFieldValue</code>
<code>getDescription</code> , <code>setDescription</code>

Methods inherited from interface <code>com.crystaldecisions.sdk.occa.report.data.IValue</code>
<code>computeText</code> , <code>displayText</code>

Method Detail

getUpperBoundType

```
public RangeValueBoundType getUpperBoundType()
```

Returns how the upper bound of the range is treated. Can also specify that the range has no upper bound.

Returns:

How the upper bound of the range is treated as a RangeValueBoundType object.

setUpperBoundType

```
public void setUpperBoundType(RangeValueBoundType upperBoundType)
```

Sets how the upper bound of the range is treated. This method can also be used to specify that the range has no upper bound.

Parameters:

upperBoundType – How the upper bound of the range is treated as a RangeValueBoundType object.

getLowerBoundType

```
public RangeValueBoundType getLowerBoundType()
```

Returns how the lower bound of the range is treated. Can also specify that the range has no lower bound.

Returns:

How the lower bound of the range is treated as a RangeValueBoundType Object.

setLowerBoundType

```
public void setLowerBoundType(RangeValueBoundType lowerBoundType)
```

Sets how the lower bound of the range is treated. This method can also be used to specify that the range has no lower bound.

Parameters:

lowerBoundType – How the lower bound of the range is treated as a RangeValueBoundType Object.

getBeginValue

```
public java.lang.Object getBeginValue()
```

Returns the lower-bound value of the range. Use the getEndValue() method to determine the upper-bound value.

Returns:

The lower-bound value as an Object.

setBeginValue

```
public void setBeginValue(java.lang.Object beginValue)
```

Sets the lower-bound value of the range. Use the `getEndValue()` method to determine the upper-bound value.

Parameters:

`beginValue` – The lower-bound value as an `Object`.

getEndValue

```
public java.lang.Object getEndValue()
```

Returns the upper-bound value of the range. Use the `getBeginValue()` method to determine the lower-bound value.

Returns:

The upper-bound value as an `Object`.

setEndValue

```
public void setEndValue(java.lang.Object endValue)
```

Returns the upper-bound value of the range. Use the `getBeginValue()` method to determine the lower-bound value.

Parameters:

`endValue` – The upper-bound value as an `Object`.

com.crystaldecisions.sdk.occa.report.data Interface IParameterFieldValue

All Superinterfaces:

`IValue`

All Known Subinterfaces:

`IParameterFieldDiscreteValue`, `IParameterFieldRangeValue`

All Known Implementing Classes:

`ParameterFieldDiscreteValue`, `ParameterFieldRangeValue`, `ParameterFieldValue`

```
public interface IParameterFieldValue  
extends IValue
```

This interface enables you to get and set a parameter's description.

Method Summary

java.lang.String	getDescription() Returns a description of the parameter's value.
void	setDescription (java.lang.String description) Sets a description of the parameter's value.

Methods inherited from interface com.crystaldecisions.sdk.occa.report.data.IValue

computeText, displayText

Method Detail

getDescription

```
public java.lang.String getDescription()
```

Returns a description of the parameter's value.

Returns:

A description of the parameter's value as a String.

setDescription

```
public void setDescription(java.lang.String description)
```

Sets a description of the parameter's value.

Parameters:

description – A description of the parameter's value as a String.

com.crystaldecisions.sdk.occa.report.data Interface IValue

All Known Subinterfaces:

IParameterFieldDiscreteValue, IParameterFieldRangeValue, IParameterFieldValue

All Known Implementing Classes:

ParameterFieldDiscreteValue, ParameterFieldRangeValue, ParameterFieldValue, Value

```
public interface IValue
```

This interface is used as a base class to represent different kinds of values (values in formulas, parameters, and so on). It is not to be used directly, and the actual value is defined in the implementing class.

Method Summary

java.lang.String	computeText () The computeText method returns a String representation of the field value that can be used in a report formula.
java.lang.String	displayText (java.util.Locale locale) Returns the field's value as a localized and formatted String.

Method Detail

computeText

```
public java.lang.String computeText ( )
```

The computeText method returns a String representation of the field value that can be used in a report formula.

The String is not formatted or localized. The syntax will always be Crystal syntax. If the field contains a date, computeText returns Date(x, x, x); if it contains a date and time, computeText returns DateTime(x, x, x, x, x, x). However, when the date is 1900, 0, 1 (the Java start date), ComputeText returns Time(x, x, x). If you want to set a field to a time value only (that is, with no date), you should set the date to 1900, 0, 1, so that computeText will retrieve the correct information.

Returns:

A String representation of the field value that can be used in a report formula.

displayText

```
public java.lang.String displayText (java.util.Locale locale)
```

Returns the field's value as a localized and formatted String. The Locale can be a java.util.Locale value.

The value is formatted according to the locale. The user is required to pass in a locale; the system default locale or user default locale is not used. If the field contains a date, displayText returns a locale-specific date string; if it contains a date and time, displayText returns a locale-specific date time String; if it contains a time String, displayText returns a locale-specific time String.

Parameters:

locale – The locale as specified by java.util.Locale .

Returns:

The field's value as a localized and formatted String.

com.crystaldecisions.sdk.occa.report.data

Class ParameterDefaultValueDisplayType

java.lang.Object

└ com.crystaldecisions.sdk.occa.report.data.ParameterDefaultValueDisplayType

*public final class **ParameterDefaultValueDisplayType**
extends java.lang.Object*

For internal use only.

Field Summary	
static int	_displayDescriptionAndValue
static int	_displayDescriptionOnly
static ParameterDefaultValueDisplayType	displayDescriptionAndValue
static ParameterDefaultValueDisplayType	displayDescriptionOnly

Method Summary	
static ParameterDefaultValueDisplayType	from_int (int i)
static ParameterDefaultValueDisplayType	from_string (java.lang.String sVal)
java.lang.String	toString ()
int	value ()

Methods inherited from class java.lang.Object
<code>equals, getClass, hashCode, notify, notifyAll, wait, wait, wait</code>

Field Detail

_displayDescriptionOnly

```
public static final int _displayDescriptionOnly
```

See Also:

Constant Field Values

_displayDescriptionAndValue

```
public static final int _displayDescriptionAndValue
```

See Also:

Constant Field Values

displayDescriptionOnly

```
public static final ParameterDefaultValueDisplayType displayDescriptionOnly
```

displayDescriptionAndValue

```
public static final ParameterDefaultValueDisplayType displayDescriptionAndValue
```

Method Detail

from_int

```
public static final ParameterDefaultValueDisplayType from_int(int i)
```

from_string

```
public static final ParameterDefaultValueDisplayType from_string(java.lang.String sVal)
```

toString

```
public java.lang.String toString()
```

value

```
public int value()
```

com.crystaldecisions.sdk.occa.report.data

Class ParameterField

```
java.lang.Object
├─ com.crystaldecisions.sdk.occa.report.data.Field
│   └─ com.crystaldecisions.sdk.occa.report.data.ParameterField
```

All Implemented Interfaces:

IClone, IField, IParameterField

```
public class ParameterField
    extends Field
    implements IParameterField, IClone
```

This object enables you to get and set values for the parameter field. When possible, use the IParameterField interface, to manipulate this object. Parameters prompt the user of a report to enter information.

Think of a parameter as a question that the user needs to answer before the report is generated. The information users enter, or the way they respond, determines what appears in the report. For example, in a report used by sales people, there might be a parameter that asks the user to choose a region. The report would return the results for the specific region instead of returning the results for all of the regions.

By using parameter fields in formulas, selection formulas, and in the report itself, you can create a single report that you can modify whenever your needs change. Parameter fields can also be used in subreports. Parameter values are discrete or ranged. A discrete value represents a particular, single value, while a range value represents a value between a certain range. Additionally, parameters can have one value (discrete or ranged) or multiple values (discrete or ranged).

The ParameterField object allows you to manipulate a parameter in the report.

Constructor Summary

ParameterField()

ParameterField(IParameterField src)

Method Summary

java.lang.Object

IDE Extensions

	clone (boolean deepClone) Returns the new object that has been cloned.
void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
boolean	getAllowCustomCurrentValues () Returns true if the user can enter custom values for a parameter when the report is refreshed and false otherwise.
boolean	getAllowMultiValue () Returns true whether the user can select more than one value for a parameter when the report is refreshed; returns false if the user can only choose one value for the parameter.
boolean	getAllowNullValue () Returns true if the value for the parameter may be null and false otherwise.
IField	getBrowseField () Returns the database field whose values are being used as default values for the parameter.
Values	getCurrentValues () Returns the values that are currently being used for the parameter.
ParameterDefaultValueDisplayType	getDefaultValueDisplayType () For internal use only.
Values	getDefaultValue () Returns the default values that may be used for the parameter.
ParameterSortMethod	getDefaultValueSortMethod () Returns whether the default values are sorted by value or by the value's description.
ParameterSortOrder	getDefaultValueSortOrder () Returns the manner in which the values or the values' descriptions are sorted.
java.lang.String	getEditMask () Returns the edit mask for the parameter values.

IDE Extensions

FieldKind	getKind() Returns what kind of field this is.
IParameterFieldDiscreteValue	getMaximumValue() Returns the maximum value the parameter can have, providing that a minimum value is also specified.
IParameterFieldDiscreteValue	getMinimumValue() Returns the minimum value the parameter can have, providing that a maximum value is also specified.
ParameterFieldType	getParameterType() Returns how the parameter is being used—that is, its type.
java.lang.String	getReportName() Returns the name of the report to which the parameter belongs.
java.lang.String	getShortName(java.util.Locale locale) Returns the short name of the field.
ParameterFieldUsage	getUsage() For internal use only.
ParameterValueRangeKind	getValueRangeKind() Returns whether the parameter is discrete, ranged, or both.
Values	getValues() Returns the current values for the parameter, or the default values if no current values are defined.
boolean	hasContent(java.lang.Object srcParameterField) Returns true if this object contains the same elements as the passed in object.
void	readElement(java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save(XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
void	save(XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	saveContents(XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	setAllowCustomCurrentValues(boolean allowCustom) Sets if the user can enter custom values for a

IDE Extensions

		parameter when the report is refreshed.
void	setAllowMultiValue (boolean allowMultiValue)	Sets whether the user can select more than one value for a parameter when the report is refreshed.
void	setAllowNullValue (boolean allowNullValue)	Set to true if the value for the parameter may be null and false otherwise.
void	setBrowseField (IField browseField)	Sets the database field whose values are being used as default values for the parameter.
void	setCurrentValues (Values currentValues)	Sets the values that are currently being used for the parameter.
void	setDefaultValueDisplayType (ParameterDefaultValueDisplayType defaultValueDisplayType)	For internal use only.
void	setDefaultValues (Values defaultValues)	Sets the default values that may be used for the parameter.
void	setDefaultValuesSortMethod (ParameterSortMethod defaultValueSortMethod)	Sets whether the default values are sorted by value or by the value's description.
void	setDefaultValuesSortOrder (ParameterSortOrder defaultValueSortOrder)	Sets the manner in which the values or the values' descriptions are sorted.
void	setEditMask (java.lang.String editMask)	Sets the edit mask for the parameter values.
void	setMaximumValue (IParameterFieldDiscreteValue maximumValue)	Sets the maximum value the parameter can have, providing that a minimum value is also specified.
void	setMinimumValue (IParameterFieldDiscreteValue minimumValue)	Sets the minimum value the parameter can have, providing that a maximum value is also specified.
void	setParameterType (ParameterFieldType parameterType)	Sets how the parameter is being used—that is, its type.
void	setReportName (java.lang.String reportName)	Sets the name of the report to which the parameter belongs.
void	setUsage (int usage)	For internal use only.
void	setValueRangeKind (ParameterValueRangeKind valueRangeKind)	Sets whether the parameter is discrete, ranged, or both.

IDE Extensions

	<code>void</code>	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.
--	-------------------	--

Methods inherited from class com.crystaldecisions.sdk.occa.report.data.Field

createMember, getDescription, getDisplayName, getFormulaForm,
getHeadingText, getIsRecurring, getLength, getLongName, getName,
getType, save, save, setDescription, setHeadingText, setLength, setName,
setType

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait,
wait

Methods inherited from interface com.crystaldecisions.sdk.occa.report.data.IField

getDescription, getDisplayName, getFormulaForm, getHeadingText,
getIsRecurring, getLength, getLongName, getName, getType,
setDescription, setHeadingText, setLength, setName, setType

Constructor Detail

ParameterField

```
public ParameterField(IParameterField src)
```

ParameterField

```
public ParameterField()
```

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: IClone

Returns the new object that has been cloned.

Specified by:

clone in interface IClone

Overrides:

clone in class Field

copyTo

```
public void copyTo(java.lang.Object destObject,
                  boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

copyTo in interface IClone

Overrides:copyTo in class Field

createMember

```
public java.lang.Object createMember(java.lang.String eleName,
                                     org.xml.sax.Attributes attrs,
                                     XMLSerializationContext ctxt,
                                     java.util.Map objState,
                                     boolean[] bLoaded)
```

For internal use only.

Overrides:createMember in class Field

endElement

```
public void endElement(java.lang.String eleName,
                      java.util.Map objState)
```

For internal use only.

Overrides:endElement in class Field

getAllowCustomCurrentValues

```
public boolean getAllowCustomCurrentValues()
```

Description copied from interface: IParameterField

Returns true if the user can enter custom values for a parameter when the report is refreshed and false otherwise. If this property is true, users can enter any value for the parameter. If it is false, they must choose from one of the default values.

Specified by:

getAllowCustomCurrentValues in interface IParameterField

Returns:

true if the user can enter custom values for a parameter when the report is refreshed, and false otherwise.

getAllowMultiValue

```
public boolean getAllowMultiValue()
```

Description copied from interface: IParameterField

Returns true whether the user can select more than one value for a parameter when the report is refreshed; returns false if the user can only choose one value for the parameter.

Specified by:

getAllowMultiValue in interface IParameterField

Returns:

true if the user can select more than one value for a parameter when the report is refreshed, and false otherwise.

getAllowNullValue

```
public boolean getAllowNullValue()
```

Description copied from interface: IParameterField

Returns true if the value for the parameter may be null and false otherwise. This is used only for stored SQL procedures.

Specified by:

getAllowNullValue in interface IParameterField

Returns:

true if the value for the parameter may be null, and false otherwise.

getBrowseField

```
public IField getBrowseField()
```

Description copied from interface: IParameterField

Returns the database field whose values are being used as default values for the parameter. This is used for UI-display purposes.

Specified by:

getBrowseField in interface IParameterField

Returns:

The database field whose values are being used as default values for the parameter as an IField interface.

getCurrentValues

```
public Values getCurrentValues()
```

Description copied from interface: IParameterField

Returns the values that are currently being used for the parameter. To permanently set this value, a Report Application Server is required. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

Specified by:

getCurrentValues in interface IParameterField

Returns:

The current values as a Values object.

getDefaultValues

```
public Values getDefaultValues()
```

Description copied from interface: IParameterField

Returns the default values that may be used for the parameter.

Specified by:

getDefaultValues in interface IParameterField

Returns:

The default values that may be used for the parameter as a Values object.

getDefaultValueSortMethod

```
public ParameterSortMethod getDefaultValueSortMethod()
```

Description copied from interface: IParameterField

Returns whether the default values are sorted by value or by the value's description. This is used for UI-display purposes.

Specified by:

getDefaultValueSortMethod in interface IParameterField

Returns:

Whether the default values are sorted by value or by the value's description as a ParameterSortMethod object.

getDefaultValueSortOrder

```
public ParameterSortOrder getDefaultValueSortOrder()
```

Description copied from interface: IParameterField

Returns the manner in which the values or the values' descriptions are sorted.

Specified by:

getDefaultValueSortOrder in interface IParameterField

Returns:

The manner in which the values or the values' descriptions are sorted as a ParameterSortOrder object.

getEditMask

```
public java.lang.String getEditMask()
```

Description copied from interface: IParameterField

Returns the edit mask for the parameter values. This is used for UI–display purposes. The edit mask can be any of a set of masking characters used to restrict the values you can enter as parameter values. The edit mask also limits the values you can enter as default prompting values.

Note: The value of the edit mask is not checked if you use the SDK to change parameter values. You must check to ensure that any values the user enters respect the edit mask. If a user enters an invalid value according to the edit mask, it will not be possible to view or schedule the report.

The edit mask follows the following guidelines:

- ◇ "A" (allows an alphanumeric character and requires the entry of a character in the parameter value)
- ◇ "a" (allows an alphanumeric character and does not require the entry of a character in the parameter value)
- ◇ "0" (allows a digit [0 to 9] and requires the entry of a character in the parameter value)
- ◇ "9" (allows a digit or a space, and does not require the entry of a character in the parameter value)
- ◇ "#" (allows a digit, space, or plus/minus sign, and does not require the entry of a character in the parameter value)
- ◇ "L" (allows a letter [A to Z], and requires the entry of a character in the parameter value)
- ◇ "?" (allows a letter, and does not require the entry of a character in the parameter value)
- ◇ "" (allows any character or space, and requires the entry of a character in the parameter value)
- ◇ "C" (allows any character or space, and does not require the entry of a character in the parameter value)
- ◇ ". , : ; - /" (separator characters). Inserting separator characters into an Edit Mask is something like hard coding the formatting for the parameter field. When the field is placed on the report, the separator character will appear in the field object frame, like this: LLLL/0000. This example depicts an edit mask that requires four letters followed by four numbers.
- ◇ ">" (causes subsequent characters to be converted to uppercase)
- ◇ "\" (causes the subsequent character to be displayed as a literal). For example, the edit mask "\A" would display a parameter value of "A." If the edit mask is "00\A00," then a valid parameter value would consist of two digits, the letter "A," and then two additional digits.
- ◇ "Password". Allows you to set the edit mask to "Password." You can create conditional formulas that specify that certain sections of the report become visible only when certain user passwords are entered.

Note: Some of the edit mask characters require that you enter a character in their place (when entering a parameter value), while others allow you to leave a space, if needed. For example, if the edit mask is 000099, you can enter a parameter value with four digits, five digits, or six digits, since the '9' edit mask character does not require the entry of a character. However, since '0' does require such an entry, you could not enter a parameter value with less than four digits.

Specified by:

getEditMask in interface IParameterField

Returns:

The edit mask as a String.

getKind

```
public FieldKind getKind()
```

Description copied from interface: IField

Returns what kind of field this is. Do not confuse this property with Type. Kind identifies what sort of report field this is, while Type identifies what the field is composed of. For example, a bitmap, string, chart, number, date, and so on.

Specified by:

getKind in interface IField

Returns:

A String that specifies what kind of field this is.

getMaximumValue

```
public IParameterFieldDiscreteValue getMaximumValue()
```

Description copied from interface: IParameterField

Returns the maximum value the parameter can have, providing that a minimum value is also specified.

If the parameter is discrete, the discrete values must be between the minimum and maximum values. If the parameter has a range limit, both the maximum and minimum values are defined. The BeginValue and/or EndValue of each ranged value must be between the minimum and maximum values.

If the parameter is a number, this property represents the largest allowable number the parameter may have. If it is a string, this is the maximum length the string may be. If it is a Date/Time value, this is the earliest date and time the parameter may be. This property does not apply to Boolean values.

Specified by:

getMaximumValue in interface IParameterField

Returns:

The IParameterFieldDiscreteValue object.

getMinimumValue

```
public IParameterFieldDiscreteValue getMinimumValue()
```

Description copied from interface: IParameterField

Returns the minimum value the parameter can have, providing that a maximum value is also specified.

If the parameter is discrete, the discrete values must be between the minimum and maximum values. If the parameter has a range limit, both the maximum and minimum values are defined. The BeginValue and/or EndValue of each ranged value must be between the minimum and maximum values.

If the parameter is a number, this property represents the smallest allowable number the parameter

may have. If it is a string, this is the minimum length the string may be. If it is a Date/Time value, this is the earliest date and time the parameter may be. This property does not apply to Boolean values.

Specified by:

getMinimumValue in interface IParameterField

Returns:

The IParameterFieldDiscreteValue object.

getParameterType

```
public ParameterFieldType getParameterType()
```

Description copied from interface: IParameterField

Returns how the parameter is being used—that is, its type.

Specified by:

getParameterType in interface IParameterField

Returns:

The parameter field type as a ParameterFieldType object.

getReportName

```
public java.lang.String getReportName()
```

Description copied from interface: IParameterField

Returns the name of the report to which the parameter belongs. This string is empty if the parameter belongs in a main report; if the parameter is contained in a subreport, this contains the name of the subreport.

Specified by:

getReportName in interface IParameterField

Returns:

The name of the report as a String.

getShortName

```
public java.lang.String getShortName(java.util.Locale locale)
```

Description copied from interface: IField

Returns the short name of the field.

Specified by:

getShortName in interface IField

Overrides:

getShortName in class Field

getValueRangeKind

```
public ParameterValueRangeKind getValueRangeKind()
```

Description copied from interface: IParameterField

Returns whether the parameter is discrete, ranged, or both.

Specified by:

getValueRangeKind in interface IParameterField

Returns:

Whether the parameter is discrete, ranged, or both as a ParameterValueRangeKind object.

getValues

```
public Values getValues()
```

Description copied from interface: IParameterField

Returns the current values for the parameter, or the default values if no current values are defined.

This is equivalent to the IParameterField.getCurrentValues() method unless it is empty, in which case it is equivalent to the IParameterField.getDefaultValues() method.

Specified by:

getValues in interface IParameterField

Returns:

The values as a Values object.

hasContent

```
public boolean hasContent(java.lang.Object srcParameterField)
```

Description copied from interface: IClone

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Overrides:

hasContent in class Field

readElement

```
public void readElement(java.lang.String eleName,  
                        java.lang.String sVal,  
                        org.xml.sax.Attributes attrs,  
                        java.util.Map objState)
```

For internal use only.

Overrides:

readElement in class Field

save

```
public void save(XMLWriter writer,  
                java.lang.String sTag,
```

```
XMLSerializationContext ctxt)
throws java.io.IOException
```

For internal use only.

Overrides:

save in class Field

Throws:

java.io.IOException

save

```
public void save(XMLWriter writer,
XMLSerializationContext ctxt)
throws java.io.IOException
```

For internal use only.

Overrides:

save in class Field

Throws:

java.io.IOException

saveContents

```
public void saveContents(XMLWriter writer,
XMLSerializationContext ctxt)
throws java.io.IOException
```

For internal use only.

Overrides:

saveContents in class Field

Throws:

java.io.IOException

setAllowCustomCurrentValues

```
public void setAllowCustomCurrentValues(boolean allowCustomCurrentValues)
```

Description copied from interface: IParameterField

Sets if the user can enter custom values for a parameter when the report is refreshed. If this property is true, users can enter any value for the parameter. If it is false, they must choose from one of the default values.

Specified by:

setAllowCustomCurrentValues in interface IParameterField

Parameters:

allowCustomCurrentValues – true if the user can enter custom values for a parameter when the report is refreshed, and false otherwise.

setAllowMultiValue

```
public void setAllowMultiValue(boolean allowMultiValue)
```

Description copied from interface: IParameterField

Sets whether the user can select more than one value for a parameter when the report is refreshed.

Specified by:

setAllowMultiValue in interface IParameterField

Parameters:

allowMultiValue – true if the user can select more than one value for a parameter when the report is refreshed, and false otherwise.

setAllowNullValue

```
public void setAllowNullValue(boolean allowNullValue)
```

Description copied from interface: IParameterField

Set to true if the value for the parameter may be null and false otherwise. This is used only for stored SQL procedures.

Specified by:

setAllowNullValue in interface IParameterField

Parameters:

allowNullValue – true if the value for the parameter may be null, and false otherwise.

setBrowseField

```
public void setBrowseField(IField browseField)
```

Description copied from interface: IParameterField

Sets the database field whose values are being used as default values for the parameter. This is used for UI–display purposes.

Specified by:

setBrowseField in interface IParameterField

Parameters:

browseField – The database field whose values are being used as default values for the parameter as an IField interface.

setCurrentValues

```
public void setCurrentValues(Values currentValues)
```

Description copied from interface: IParameterField

Sets the values that are currently being used for the parameter. To permanently set this value, a Report Application Server is required. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

Specified by:

setCurrentValues in interface IParameterField

Parameters:

currentValues – The current values as a Values object.

setDefaultValues

```
public void setDefaultValues(Values defaultValues)
```

Description copied from interface: IParameterField

Sets the default values that may be used for the parameter.

Specified by:

setDefaultValues in interface IParameterField

Parameters:

defaultValues – The default values that may be used for the parameter as a Values object.

setDefaultValueSortMethod

```
public void setDefaultValueSortMethod(ParameterSortMethod defaultValueSortMethod)
```

Description copied from interface: IParameterField

Sets whether the default values are sorted by value or by the value's description. This is used for UI–display purposes.

Specified by:

setDefaultValueSortMethod in interface IParameterField

Parameters:

defaultValueSortMethod – Whether the default values are sorted by value or by the value's description as a ParameterSortMethod object.

setDefaultValueSortOrder

```
public void setDefaultValueSortOrder(ParameterSortOrder defaultValueSortOrder)
```

Description copied from interface: IParameterField

Sets the manner in which the values or the values' descriptions are sorted.

Specified by:

setDefaultValueSortOrder in interface IParameterField

Parameters:

defaultValueSortOrder – The manner in which the values or the values' descriptions are sorted as a ParameterSortOrder object.

setEditMask

```
public void setEditMask(java.lang.String editMask)
```

Description copied from interface: IParameterField

IDE Extensions

Sets the edit mask for the parameter values. This is used for UI–display purposes. The edit mask can be any of a set of masking characters used to restrict the values you can enter as parameter values. The edit mask also limits the values you can enter as default prompting values.

Note: The value of the edit mask is not checked if you use the SDK to change parameter values. You must check to ensure that any values the user enters respect the edit mask. If a user enters an invalid value according to the edit mask, it will not be possible to view or schedule the report.

The edit mask follows the following guidelines:

- ◇ "A" (allows an alphanumeric character and requires the entry of a character in the parameter value)
- ◇ "a" (allows an alphanumeric character and does not require the entry of a character in the parameter value)
- ◇ "0" (allows a digit [0 to 9] and requires the entry of a character in the parameter value)
- ◇ "9" (allows a digit or a space, and does not require the entry of a character in the parameter value)
- ◇ "#" (allows a digit, space, or plus/minus sign, and does not require the entry of a character in the parameter value)
- ◇ "L" (allows a letter [A to Z], and requires the entry of a character in the parameter value)
- ◇ "?" (allows a letter, and does not require the entry of a character in the parameter value)
- ◇ "" (allows any character or space, and requires the entry of a character in the parameter value)
- ◇ "C" (allows any character or space, and does not require the entry of a character in the parameter value)
- ◇ ". , : ; - /" (separator characters). Inserting separator characters into an edit mask is something like hard coding the formatting for the parameter field. When the field is placed on the report, the separator character will appear in the field object frame, like this: LLLL/0000. This example depicts an edit mask that requires four letters followed by four numbers.
- ◇ "<" (causes subsequent characters to be converted to lowercase)
- ◇ ">" (causes subsequent characters to be converted to uppercase)
- ◇ "\" (causes the subsequent character to be displayed as a literal). For example, the edit mask "\"A" would display a parameter value of "A." If the edit mask is "00\\A00," then a valid parameter value would consist of two digits, the letter "A," and then two additional digits.
- ◇ "Password". Allows you to set the edit mask to "Password." You can create conditional formulas that specify that certain sections of the report become visible only when certain user passwords are entered.

Note: Some of the edit mask characters require that you enter a character in their place (when entering a parameter value), while others allow you to leave a space, if needed. For example, if the Edit Mask is 000099, you can enter a parameter value with four digits, five digits, or six digits, since the '9' edit mask character does not require the entry of a character. However, since '0' does require such an entry, you could not enter a parameter value with less than four digits.

Specified by:

setEditMask in interface IParameterField

Parameters:

editMask – The edit mask as a String.

setMaximumValue

```
public void setMaximumValue(IParameterFieldDiscreteValue maximumValue)
```

Description copied from interface: IParameterField

Sets the maximum value the parameter can have, providing that a minimum value is also specified.

If the parameter is discrete, the discrete values must be between the minimum and maximum values. If the parameter has a range limit, both the maximum and minimum values are defined. The BeginValue and/or EndValue of each ranged value must be between the minimum and maximum values.

If the parameter is a number, this property represents the largest allowable number the parameter may have. If it is a string, this is the maximum length the string may be. If it is a Date/Time value, this is the earliest date and time the parameter may be. This property does not apply to Boolean values.

Specified by:

setMaximumValue in interface IParameterField

Parameters:

maximumValue – The IParameterFieldDiscreteValue object.

setMinimumValue

```
public void setMinimumValue(IParameterFieldDiscreteValue minimumValue)
```

Description copied from interface: IParameterField

Sets the minimum value the parameter can have, providing that a maximum value is also specified.

If the parameter is discrete, the discrete values must be between the minimum and maximum values. If the parameter has a range limit, both the maximum and minimum values are defined. The BeginValue and/or EndValue of each ranged value must be between the minimum and maximum values.

If the parameter is a number, this property represents the smallest allowable number the parameter may have. If it is a string, this is the minimum length the string may be. If it is a Date/Time value, this is the earliest date and time the parameter may be. This property does not apply to Boolean values.

Specified by:

setMinimumValue in interface IParameterField

Parameters:

minimumValue – The IParameterFieldDiscreteValue object.

setParameterType

```
public void setParameterType(ParameterFieldType parameterType)
```

Description copied from interface: IParameterField

Sets how the parameter is being used—that is, its type.

Specified by:

setParameterType in interface IParameterField

Parameters:

parameterType – The parameter field type as a ParameterFieldType object.

setReportName

```
public void setReportName(java.lang.String reportName)
```

Description copied from interface: IParameterField

Sets the name of the report to which the parameter belongs. This string is empty if the parameter belongs in a main report; if the parameter is contained in a subreport, this contains the name of the subreport.

Specified by:

setReportName in interface IParameterField

Parameters:

reportName – The name of the report as a String. Set this string as empty if the parameter belongs in a main report; if the parameter is contained in a subreport, set this to contain the name of the subreport.

setUsage

```
public void setUsage(int usage)
```

For internal use only.

setValueRangeKind

```
public void setValueRangeKind(ParameterValueRangeKind valueRangeKind)
```

Description copied from interface: IParameterField

Sets whether the parameter is discrete, ranged, or both.

Specified by:

setValueRangeKind in interface IParameterField

Parameters:

valueRangeKind – Whether the parameter is discrete, ranged, or both as a ParameterValueRangeKind object.

startElement

```
public void startElement(java.lang.String eleName,
                        java.util.Map objState,
                        org.xml.sax.Attributes attrs)
```

For internal use only.

Overrides:

startElement in class Field

getDefaultValueDisplayType

```
public ParameterDefaultValueDisplayType getDefaultValueDisplayType()
```

For internal use only.

Specified by:

getDefaultValueDisplayType in interface IParameterField

getUsage

```
public ParameterFieldUsage getUsage()
```

For internal use only.

setDefaultValueDisplayType

```
public void setDefaultValueDisplayType(ParameterDefaultValueDisplayType type)
```

For internal use only.

Specified by:

setDefaultValueDisplayType in interface IParameterField

com.crystaldecisions.sdk.occa.report.data Class ParameterFieldDiscreteValue

```
java.lang.Object
├── com.crystaldecisions.sdk.occa.report.data.Value
│   ├── com.crystaldecisions.sdk.occa.report.data.ParameterFieldValue
│   └── com.crystaldecisions.sdk.occa.report.data.ParameterFieldDiscreteValue
```

All Implemented Interfaces:

IClone, IParameterFieldDiscreteValue, IParameterFieldValue, IValue

```
public class ParameterFieldDiscreteValue
extends ParameterFieldValue
implements IParameterFieldDiscreteValue, IClone
```

This object represents a discrete value belonging to a parameter. A discrete value is a value that does not have a range—that is, it is singular. Examples of discrete values are: 1, 56.4, "Jeremy", and so on. When possible, use the interfaces IParameterFieldValue and IValue to manipulate this object.

Constructor Summary

ParameterFieldDiscreteValue ()
--

ParameterFieldDiscreteValue (IParameterFieldDiscreteValue src)

Method Summary

java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
java.lang.String	computeText () The computeText method returns a String representation of the field value that can be used in a report formula.
void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
java.lang.String	displayText (java.util.Locale locale) Returns the field's value as a localized and formatted String.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
java.lang.Object	getValue () Returns the discrete value of the parameter.
boolean	hasContent (java.lang.Object srcParameterFieldDiscreteValue) Returns true if this object contains the same elements as the passed in object.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
void	save (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	setValue (java.lang.Object value) Sets the discrete value of the parameter.
void	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs)

	For internal use only.
--	------------------------

Methods inherited from class <code>com.crystaldecisions.sdk.occa.report.data.ParameterFieldValue</code>
--

<code>getDescription, setDescription, startElement</code>

Methods inherited from class <code>java.lang.Object</code>

<code>equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</code>
--

Methods inherited from interface <code>com.crystaldecisions.sdk.occa.report.data.IParameterFieldValue</code>

<code>getDescription, setDescription</code>

Constructor Detail

ParameterFieldDiscreteValue

```
public ParameterFieldDiscreteValue(IParameterFieldDiscreteValue src)
```

ParameterFieldDiscreteValue

```
public ParameterFieldDiscreteValue()
```

Method Detail

computeText

```
public java.lang.String computeText()
```

Description copied from interface: `IValue`

The `computeText` method returns a `String` representation of the field value that can be used in a report formula.

The `String` is not formatted or localized. The syntax will always be Crystal syntax. If the field contains a date, `computeText` returns `Date(x, x, x)`; if it contains a date and time, `computeText` returns `DateTime(x, x, x, x, x, x)`. However, when the date is 1900, 0, 1 (the Java start date), `computeText` returns `Time(x, x, x)`. If you want to set a field to a time value only (that is, with no date), you should set the date to 1900, 0, 1, so that `computeText` will retrieve the correct information.

Specified by:

`computeText` in interface `IValue`

Specified by:

displayText

```
public java.lang.String displayText(java.util.Locale locale)
```

Description copied from interface: IValue

Returns the field's value as a localized and formatted String. The Locale can be a `java.util.Locale` value.

The value is formatted according to the locale. The user is required to pass in a locale; the system default locale or user default locale is not used. If the field contains a date, `displayText` returns a locale-specific date string; if it contains a date and time, `displayText` returns a locale-specific date time String; if it contains a time String, `displayText` returns a locale-specific time String.

Specified by:

`displayText` in interface `IValue`

Specified by:

`displayText` in class `Value`

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: IClone

Returns the new object that has been cloned.

Specified by:

`clone` in interface `IClone`

Specified by:

`clone` in class `ParameterFieldValue`

copyTo

```
public void copyTo(java.lang.Object destObject,  
                  boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

`copyTo` in interface `IClone`

Overrides:

`copyTo` in class `ParameterFieldValue`

createMember

```
public java.lang.Object createMember(java.lang.String eleName,
                                     org.xml.sax.Attributes attrs,
                                     XMLSerializationContext ctxt,
                                     java.util.Map objState,
                                     boolean[] bLoaded)
```

Description copied from class: Value

For internal use only.

Overrides:

createMember in class ParameterFieldValue

endElement

```
public void endElement(java.lang.String eleName,
                       java.util.Map objState)
```

Description copied from class: Value

For internal use only.

Overrides:

endElement in class ParameterFieldValue

getValue

```
public java.lang.Object getValue()
```

Description copied from interface: IParameterFieldDiscreteValue

Returns the discrete value of the parameter.

Specified by:

getValue in interface IParameterFieldDiscreteValue

Returns:

The value as an object.

readElement

```
public void readElement(java.lang.String eleName,
                        java.lang.String sVal,
                        org.xml.sax.Attributes attrs,
                        java.util.Map objState)
```

Description copied from class: Value

For internal use only.

Overrides:

readElement in class ParameterFieldValue

save

```
public void save(XMLWriter writer,
                 java.lang.String sTag,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

Description copied from class: Value

For internal use only.

Overrides:

save in class ParameterFieldValue

Throws:

java.io.IOException

save

```
public void save(XMLWriter writer,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

Description copied from class: Value

For internal use only.

Overrides:

save in class ParameterFieldValue

Throws:

java.io.IOException

setValue

```
public void setValue(java.lang.Object value)
```

Description copied from interface: IParameterFieldDiscreteValue

Sets the discrete value of the parameter.

Specified by:

setValue in interface IParameterFieldDiscreteValue

Parameters:

value – The value as an object.

hasContent

```
public boolean hasContent(java.lang.Object srcParameterFieldDiscreteValue)
```

Description copied from interface: IClone

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Overrides:

hasContent in class ParameterFieldValue

saveContents

```
public void saveContents(XMLWriter writer,
                        XMLSerializationContext ctxt)
    throws java.io.IOException
```

Description copied from class: Value

For internal use only.

Overrides:

saveContents in class ParameterFieldValue

Throws:

java.io.IOException

startElement

```
public void startElement(java.lang.String eleName,
                        java.util.Map objState,
                        org.xml.sax.Attributes attrs)
```

For internal use only.

com.crystaldecisions.sdk.occa.report.data Class ParameterFieldRangeValue

```
java.lang.Object
├─ com.crystaldecisions.sdk.occa.report.data.Value
│   └─ com.crystaldecisions.sdk.occa.report.data.ParameterFieldValue
│       └─ com.crystaldecisions.sdk.occa.report.data.ParameterFieldRangeValue
```

All Implemented Interfaces:

IClone, IParameterFieldRangeValue, IParameterFieldValue, IValue

```
public class ParameterFieldRangeValue
    extends ParameterFieldValue
    implements IParameterFieldRangeValue, IClone
```

This object stores the value for a ranged parameter. This object contains the upper bound, the lower bound range, and how the two are treated: that is if they exist, are included, and so on.

Constructor Summary

ParameterFieldRangeValue()

ParameterFieldRangeValue(IParameterFieldRangeValue src)

Method Summary	
java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
java.lang.String	computeText () The computeText method returns a String representation of the field value that can be used in a report formula.
void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
java.lang.String	displayText (java.util.Locale locale) Returns the field's value as a localized and formatted String.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
java.lang.Object	getBeginValue () Returns the lower-bound value of the range.
java.lang.Object	getEndValue () Returns the upper-bound value of the range.
RangeValueBoundType	getLowerBoundType () Returns how the lower bound of the range is treated.
RangeValueBoundType	getUpperBoundType () Returns how the upper bound of the range is treated.
boolean	hasContent (java.lang.Object srcParameterFieldRangeValue) Returns true if this object contains the same elements as the passed in object.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
void	save (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.

IDE Extensions

void	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	setBeginValue (java.lang.Object beginValue) Sets the lower-bound value of the range.
void	setEndValue (java.lang.Object endValue) Returns the upper-bound value of the range.
void	setLowerBoundType (RangeValueBoundType lowerBoundType) Sets how the lower bound of the range is treated.
void	setUpperBoundType (RangeValueBoundType upperBoundType) Sets how the upper bound of the range is treated.
void	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.

Methods inherited from class com.crystaldecisions.sdk.occa.report.data.ParameterFieldValue

getDescription, setDescription, startElement

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface com.crystaldecisions.sdk.occa.report.data.IParameterFieldValue

getDescription, setDescription

Constructor Detail

ParameterFieldRangeValue

```
public ParameterFieldRangeValue(IParameterFieldRangeValue src)
```

ParameterFieldRangeValue

```
public ParameterFieldRangeValue()
```

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: IClone

Returns the new object that has been cloned.

Specified by:

clone in interface IClone

Specified by:

clone in class ParameterFieldValue

computeText

```
public java.lang.String computeText()
```

Description copied from interface: IValue

The computeText method returns a String representation of the field value that can be used in a report formula.

The String is not formatted or localized. The syntax will always be Crystal syntax. If the field contains a date, computeText returns Date(x, x, x); if it contains a date and time, computeText returns DateTime(x, x, x, x, x, x). However, when the date is 1900, 0, 1 (the Java start date), ComputeText returns Time(x, x, x). If you want to set a field to a time value only (that is, with no date), you should set the date to 1900, 0, 1, so that computeText will retrieve the correct information.

Specified by:

computeText in interface IValue

Specified by:

computeText in class Value

copyTo

```
public void copyTo(java.lang.Object destObject,  
                  boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

copyTo in interface IClone

Overrides:

copyTo in class ParameterFieldValue

createMember

```
public java.lang.Object createMember(java.lang.String eleName,  
                                     org.xml.sax.Attributes attrs,  
                                     XMLSerializationContext ctxt,  
                                     java.util.Map objState,
```

```
boolean[] bLoaded)
```

Description copied from class: Value

For internal use only.

Overrides:

createMember in class ParameterFieldValue

displayText

```
public java.lang.String displayText(java.util.Locale locale)
```

Description copied from interface: IValue

Returns the field's value as a localized and formatted String. The Locale can be a java.util.Locale value.

The value is formatted according to the locale. The user is required to pass in a locale; the system default locale or user default locale is not used. If the field contains a date, displayText returns a locale-specific date string; if it contains a date and time, displayText returns a locale-specific date time String; if it contains a time String, displayText returns a locale-specific time String.

Specified by:

displayText in interface IValue

Specified by:

displayText in class Value

endElement

```
public void endElement(java.lang.String eleName,  
                        java.util.Map objState)
```

Description copied from class: Value

For internal use only.

Overrides:

endElement in class ParameterFieldValue

getBeginValue

```
public java.lang.Object getBeginValue()
```

Description copied from interface: IParameterFieldRangeValue

Returns the lower-bound value of the range. Use the IParameterFieldRangeValue.getEndValue() method to determine the upper-bound value.

Specified by:

getBeginValue in interface IParameterFieldRangeValue

Returns:

The lower-bound value as an Object.

getEndValue

```
public java.lang.Object getEndValue()
```

Description copied from interface: IParameterFieldRangeValue

Returns the upper-bound value of the range. Use the `IParameterFieldRangeValue.getBeginValue()` method to determine the lower-bound value.

Specified by:

`getEndValue` in interface `IParameterFieldRangeValue`

Returns:

The upper-bound value as an `Object`.

getLowerBoundType

```
public RangeValueBoundType getLowerBoundType()
```

Description copied from interface: IParameterFieldRangeValue

Returns how the lower bound of the range is treated. Can also specify that the range has no lower bound.

Specified by:

`getLowerBoundType` in interface `IParameterFieldRangeValue`

Returns:

How the lower bound of the range is treated as a `RangeValueBoundType` `Object`.

getUpperBoundType

```
public RangeValueBoundType getUpperBoundType()
```

Description copied from interface: IParameterFieldRangeValue

Returns how the upper bound of the range is treated. Can also specify that the range has no upper bound.

Specified by:

`getUpperBoundType` in interface `IParameterFieldRangeValue`

Returns:

How the upper bound of the range is treated as a `RangeValueBoundType` object.

hasContent

```
public boolean hasContent(java.lang.Object srcParameterFieldRangeValue)
```

Description copied from interface: IClone

Returns `true` if this object contains the same elements as the passed in object.

Specified by:

`hasContent` in interface `IClone`

Overrides:

hasContent in class ParameterFieldValue

readElement

```
public void readElement(java.lang.String eleName,
                        java.lang.String sVal,
                        org.xml.sax.Attributes attrs,
                        java.util.Map objState)
```

Description copied from class: Value

For internal use only.

Overrides:

readElement in class ParameterFieldValue

save

```
public void save(XMLWriter writer,
                 java.lang.String sTag,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

Description copied from class: Value

For internal use only.

Overrides:

save in class ParameterFieldValue

Throws:

java.io.IOException

save

```
public void save(XMLWriter writer,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

Description copied from class: Value

For internal use only.

Overrides:

save in class ParameterFieldValue

Throws:

java.io.IOException

saveContents

```
public void saveContents(XMLWriter writer,
                         XMLSerializationContext ctxt)
    throws java.io.IOException
```

Description copied from class: Value

For internal use only.

Overrides:

saveContents in class ParameterFieldValue

Throws:

java.io.IOException

setBeginValue

```
public void setBeginValue(java.lang.Object beginValue)
```

Description copied from interface: IParameterFieldRangeValue

Sets the lower-bound value of the range. Use the

IParameterFieldRangeValue.getEndValue() method to determine the upper-bound value.

Specified by:

setBeginValue in interface IParameterFieldRangeValue

Parameters:

beginValue – The lower-bound value as an Object.

setEndValue

```
public void setEndValue(java.lang.Object endValue)
```

Description copied from interface: IParameterFieldRangeValue

Returns the upper-bound value of the range. Use the

IParameterFieldRangeValue.getBeginValue() method to determine the lower-bound value.

Specified by:

setEndValue in interface IParameterFieldRangeValue

Parameters:

endValue – The upper-bound value as an Object.

setLowerBoundType

```
public void setLowerBoundType(RangeValueBoundType lowerBoundType)
```

Description copied from interface: IParameterFieldRangeValue

Sets how the lower bound of the range is treated. This method can also be used to specify that the range has no lower bound.

Specified by:

setLowerBoundType in interface IParameterFieldRangeValue

Parameters:

lowerBoundType – How the lower bound of the range is treated as a RangeValueBoundType Object.

setUpperBoundType

```
public void setUpperBoundType(RangeValueBoundType upperBoundType)
```

Description copied from interface: **IParameterFieldRangeValue**

Sets how the upper bound of the range is treated. This method can also be used to specify that the range has no upper bound.

Specified by:

`setUpperBoundType` in interface **IParameterFieldRangeValue**

Parameters:

`upperBoundType` – How the upper bound of the range is treated as a **RangeValueBoundType** object.

startElement

```
public void startElement(java.lang.String eleName,  
                          java.util.Map objState,  
                          org.xml.sax.Attributes attrs)
```

For internal use only.

com.crystaldecisions.sdk.occa.report.data

Class ParameterFieldType

```
java.lang.Object
```

```
└─ com.crystaldecisions.sdk.occa.report.data.ParameterFieldType
```

```
public final class ParameterFieldType
```

```
extends java.lang.Object
```

This class contains constants for the parameter field type and methods for using these constants.

Field Summary	
static int	_queryParameter Specifies parameter field type as a query parameter.
static int	_reportParameter Specifies parameter field type as a report parameter.
static int	_storedProcedureParameter Specifies parameter field type as a stored procedure parameter.
static ParameterFieldType	queryParameter Specifies parameter field type as a query parameter.
static ParameterFieldType	reportParameter Specifies parameter field type as a report parameter.

static ParameterFieldType	storedProcedureParameter Specifies parameter field type as a stored procedure parameter.
---------------------------	--

Method Summary

static ParameterFieldType	from_int (int i) Returns the ParameterFieldType object corresponding to the specified int.
static ParameterFieldType	from_string (java.lang.String sVal) Returns the ParameterFieldType object corresponding to the specified String.
java.lang.String	toString () Returns the String value of this ParameterFieldType object.
int	value () Returns the int value of this ParameterFieldType object.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

_reportParameter

```
public static final int _reportParameter
```

Specifies parameter field type as a report parameter.

See Also:

Constant Field Values

_storedProcedureParameter

```
public static final int _storedProcedureParameter
```

Specifies parameter field type as a stored procedure parameter.

See Also:

Constant Field Values

_queryParameter

```
public static final int _queryParameter
```

Specifies parameter field type as a query parameter.

See Also:

Constant Field Values

reportParameter

```
public static final ParameterFieldType reportParameter
```

Specifies parameter field type as a report parameter.

storedProcedureParameter

```
public static final ParameterFieldType storedProcedureParameter
```

Specifies parameter field type as a stored procedure parameter.

queryParameter

```
public static final ParameterFieldType queryParameter
```

Specifies parameter field type as a query parameter.

Method Detail

from_int

```
public static final ParameterFieldType from_int(int i)
```

Returns the ParameterFieldType object corresponding to the specified int.

Parameters:

i – The int value of the desired FieldValueType.

Returns:

The ParameterFieldType object corresponding to the specified int.

toString

```
public java.lang.String toString()
```

Returns the String value of this ParameterFieldType object.

Returns:

The String value of this ParameterFieldType object.

value

```
public int value()
```

Returns the int value of this ParameterFieldType object.

Returns:

The int value of this ParameterFieldType object.

from_string

```
public static final ParameterFieldType from_string(java.lang.String sVal)
```

Returns the ParameterFieldType object corresponding to the specified String.

Parameters:

sVal – The String value of the desired ParameterFieldType.

Returns:

The ParameterFieldType object corresponding to the specified String.

com.crystaldecisions.sdk.occa.report.data**Class ParameterFieldValue**

```
java.lang.Object
```

```
└ com.crystaldecisions.sdk.occa.report.data.Value
```

```
└ com.crystaldecisions.sdk.occa.report.data.ParameterFieldValue
```

All Implemented Interfaces:

IClone, IParameterFieldValue, IValue

Direct Known Subclasses:

ParameterFieldDiscreteValue, ParameterFieldRangeValue

```
public abstract class ParameterFieldValue
```

```
extends Value
```

```
implements IParameterFieldValue, IClone
```

This object represents a parameter field value. When possible, use IParameterFieldValue to manipulate this object.

Constructor Summary

ParameterFieldValue()

Method Summary	
abstract java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
java.lang.String	getDescription () Returns a description of the parameter's value.
boolean	hasContent (java.lang.Object srcParameterFieldValue) Returns true if this object contains the same elements as the passed in object.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
void	save (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	setDescription (java.lang.String description) Sets a description of the parameter's value.
void	startElement (java.lang.String eleName, java.util.Map objState) For internal use only.

Methods inherited from class com.crystaldecisions.sdk.occa.report.data.Value
computeText, displayText

Methods inherited from class java.lang.Object

```
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait,
wait
```

Methods inherited from interface `com.crystaldecisions.sdk.occa.report.data.IValue`

```
computeText, displayText
```

Constructor Detail

ParameterFieldValue

```
public ParameterFieldValue()
```

Method Detail

clone

```
public abstract java.lang.Object clone(boolean deepClone)
```

Description copied from interface: `IClone`

Returns the new object that has been cloned.

Specified by:

clone in interface `IClone`

Specified by:

clone in class `Value`

copyTo

```
public void copyTo(java.lang.Object destObject,
                  boolean deepCopy)
```

Description copied from interface: `IClone`

Copies the object.

Specified by:

copyTo in interface `IClone`

Specified by:

copyTo in class `Value`

createMember

```
public java.lang.Object createMember(java.lang.String eleName,
                                     org.xml.sax.Attributes attrs,
                                     XMLSerializationContext ctxt,
```

IDE Extensions

```
java.util.Map objState,  
boolean[] bLoaded)
```

Description copied from class: **Value**

For internal use only.

Overrides:

createMember in class Value

endElement

```
public void endElement(java.lang.String eleName,  
java.util.Map objState)
```

Description copied from class: **Value**

For internal use only.

Overrides:

endElement in class Value

getDescription

```
public java.lang.String getDescription()
```

Description copied from interface: **IParameterFieldValue**

Returns a description of the parameter's value.

Specified by:

getDescription in interface IParameterFieldValue

Returns:

A description of the parameter's value as a String.

hasContent

```
public boolean hasContent(java.lang.Object srcParameterFieldValue)
```

Description copied from interface: **IClone**

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Specified by:

hasContent in class Value

readElement

```
public void readElement(java.lang.String eleName,  
java.lang.String sVal,  
org.xml.sax.Attributes attrs,  
java.util.Map objState)
```

Description copied from class: Value

For internal use only.

Overrides:

readElement in class Value

save

```
public void save(XMLWriter writer,
                java.lang.String sTag,
                XMLSerializationContext ctxt)
    throws java.io.IOException
```

Description copied from class: Value

For internal use only.

Overrides:

save in class Value

Throws:

java.io.IOException

save

```
public void save(XMLWriter writer,
                XMLSerializationContext ctxt)
    throws java.io.IOException
```

Description copied from class: Value

For internal use only.

Overrides:

save in class Value

Throws:

java.io.IOException

saveContents

```
public void saveContents(XMLWriter writer,
                        XMLSerializationContext ctxt)
    throws java.io.IOException
```

Description copied from class: Value

For internal use only.

Overrides:

saveContents in class Value

Throws:

java.io.IOException

setDescription

```
public void setDescription(java.lang.String description)
```

Description copied from interface: IParameterFieldValue

Sets a description of the parameter's value.

Specified by:

setDescription in interface IParameterFieldValue

Parameters:

description – A description of the parameter's value as a String.

startElement

```
public void startElement(java.lang.String eleName,
                        java.util.Map objState)
```

For internal use only.

com.crystaldecisions.sdk.occa.report.data Class ParameterSortMethod

```
java.lang.Object
└─ com.crystaldecisions.sdk.occa.report.data.ParameterSortMethod
```

```
public final class ParameterSortMethod
extends java.lang.Object
```

This class contains constants that specify how the parameters are sorted. Use IParameterField to get and set the sort method.

Field Summary	
static int	_basedOnDescription Indicates sorting based on descriptions.
static int	_basedOnValue Indicates sorting based on values.
static ParameterSortMethod	basedOnDescription Indicates that parameters are sorted based on their descriptions.
static ParameterSortMethod	basedOnValue Indicates that parameters are sorted based on their values.

Method Summary

IDE Extensions

<code>static ParameterSortMethod</code>	<code>from_int(int i)</code> Converts a parameter sort method from an int to a value.
<code>static ParameterSortMethod</code>	<code>from_string(java.lang.String sVal)</code> Converts the parameter sort method to a String.
<code>java.lang.String</code>	<code>toString()</code> Converts the parameter sort method to a String.
<code>int</code>	<code>value()</code> Returns the value for the parameter sort method.

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Field Detail

`_basedOnValue`

```
public static final int _basedOnValue
```

Indicates sorting based on values.

See Also:

Constant Field Values

`_basedOnDescription`

```
public static final int _basedOnDescription
```

Indicates sorting based on descriptions.

See Also:

Constant Field Values

`basedOnValue`

```
public static final ParameterSortMethod basedOnValue
```

Indicates that parameters are sorted based on their values.

basedOnDescription

```
public static final ParameterSortMethod basedOnDescription
```

Indicates that parameters are sorted based on their descriptions.

Method Detail

from_int

```
public static final ParameterSortMethod from_int(int i)
```

Converts a parameter sort method from an `int` to a value.

Parameters:

`i` – The `int` value that you want to convert from.

Returns:

The `ParameterSortMethod` corresponding to the specified value.

toString

```
public java.lang.String toString()
```

Converts the parameter sort method to a `String`.

Returns:

The parameter sort method as a `String`.

value

```
public int value()
```

Returns the value for the parameter sort method.

Returns:

The value of the parameter sort method as an `int`.

from_string

```
public static final ParameterSortMethod from_string(java.lang.String sVal)
```

Converts the parameter sort method to a `String`.

Parameters:

`sVal` – The `String` value of the desired `ParameterSortMethod`.

Returns:

The parameter sort method as a `String`.

com.crystaldecisions.sdk.occa.report.data Class ParameterSortOrder

```
java.lang.Object
```

```
└ com.crystaldecisions.sdk.occa.report.data.ParameterSortOrder
```

```
public final class ParameterSortOrder  
extends java.lang.Object
```

This class contains constants that indicate how a parameter list is sorted.

Field Summary	
static int	_alphabeticalAscending The list is sorted in ascending alphabetical order.
static int	_alphabeticalDescending The list is sorted in descending alphabetical order.
static int	_noSort The list is not sorted.
static int	_numericalAscending The list is sorted in ascending numerical order.
static int	_numericalDescending The list is sorted in descending numerical order.
static ParameterSortOrder	alphabeticalAscending A ParameterSortOrder object that specifies that the list is sorted in ascending alphabetical order.
static ParameterSortOrder	alphabeticalDescending A ParameterSortOrder object that specifies that the list is sorted in descending alphabetical order.
static ParameterSortOrder	noSort A ParameterSortOrder object that specifies that the list is not sorted.
static ParameterSortOrder	numericalAscending A ParameterSortOrder object that specifies that the list is sorted in ascending numerical order.
static ParameterSortOrder	numericalDescending A ParameterSortOrder object that specifies that the list is sorted in descending numerical order.

Method Summary

IDE Extensions

<code>static ParameterSortOrder</code>	<code>from_int(int i)</code> Returns the <code>ParameterSortOrder</code> corresponding to the specified value.
<code>static ParameterSortOrder</code>	<code>from_string(java.lang.String sVal)</code> Returns the <code>ParameterSortOrder</code> corresponding to the specified type.
<code>java.lang.String</code>	<code>toString()</code> Returns the <code>String</code> value of this <code>ParameterSortOrder</code> object.
<code>int</code>	<code>value()</code> Returns the <code>int</code> value of this <code>ParameterSortOrder</code> object.

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

`_noSort`

```
public static final int _noSort
```

The list is not sorted.

See Also:

Constant Field Values

`_alphabeticalAscending`

```
public static final int _alphabeticalAscending
```

The list is sorted in ascending alphabetical order.

See Also:

Constant Field Values

`_alphabeticalDescending`

```
public static final int _alphabeticalDescending
```

The list is sorted in descending alphabetical order.

See Also:

Constant Field Values

_numericalAscending

```
public static final int _numericalAscending
```

The list is sorted in ascending numerical order.

See Also:

Constant Field Values

_numericalDescending

```
public static final int _numericalDescending
```

The list is sorted in descending numerical order.

See Also:

Constant Field Values

noSort

```
public static final ParameterSortOrder noSort
```

A ParameterSortOrder object that specifies that the list is not sorted.

alphabeticalAscending

```
public static final ParameterSortOrder alphabeticalAscending
```

A ParameterSortOrder object that specifies that the list is sorted in ascending alphabetical order.

alphabeticalDescending

```
public static final ParameterSortOrder alphabeticalDescending
```

A ParameterSortOrder object that specifies that the list is sorted in descending alphabetical order.

numericalAscending

```
public static final ParameterSortOrder numericalAscending
```

A ParameterSortOrder object that specifies that the list is sorted in ascending numerical order.

numericalDescending

```
public static final ParameterSortOrder numericalDescending
```

A ParameterSortOrder object that specifies that the list is sorted in descending numerical order.

Method Detail

from_int

```
public static final ParameterSortOrder from_int(int i)
```

Returns the ParameterSortOrder corresponding to the specified value.

Parameters:

i – The int value of the desired ParameterSortOrder object.

Returns:

The ParameterSortOrder corresponding to the specified value.

toString

```
public java.lang.String toString()
```

Returns the String value of this ParameterSortOrder object.

Returns:

The String value of this ParameterSortOrder object.

value

```
public int value()
```

Returns the int value of this ParameterSortOrder object.

Returns:

The int value of this ParameterSortOrder object.

from_string

```
public static final ParameterSortOrder from_string(java.lang.String sVal)
```

Returns the ParameterSortOrder corresponding to the specified type.

Parameters:

sVal – The String value of the desired ParameterSortOrder object.

Returns:

The ParameterSortOrder corresponding to the specified type.

com.crystaldecisions.sdk.occa.report.data Class ParameterValueRangeKind

java.lang.Object

└ com.crystaldecisions.sdk.occa.report.data.ParameterValueRangeKind

*public final class **ParameterValueRangeKind***
extends java.lang.Object

This class contains constants that indicate whether the parameter is discrete, ranged, or both.

Field Summary	
static int	_discrete The parameter is discrete.
static int	_discreteAndRange The parameter supports both ranged and discrete values.
static int	_range The parameter is ranged.
static ParameterValueRangeKind	discrete A ParameterValueRangeKind object that specifies the parameter is discrete.
static ParameterValueRangeKind	discreteAndRange A ParameterValueRangeKind object that specifies the parameter supports both ranged and discrete values.
static ParameterValueRangeKind	range A ParameterValueRangeKind object that specifies the parameter is ranged.

Method Summary	
static ParameterValueRangeKind	from_int (int i) Returns the ParameterValueRangeKind corresponding to the specified value.
static ParameterValueRangeKind	from_string (java.lang.String sVal) Returns the ParameterValueRangeKind corresponding to the specified type.
java.lang.String	toString () Returns the String value of this ParameterValueRangeKind object.

<code>int</code>	value() Returns the <code>int</code> value of this <code>ParameterValueRangeKind</code> object.
------------------	---

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

`_range`

`public static final int _range`

The parameter is ranged.

See Also:

Constant Field Values

`_discrete`

`public static final int _discrete`

The parameter is discrete.

See Also:

Constant Field Values

`_discreteAndRange`

`public static final int _discreteAndRange`

The parameter supports both ranged and discrete values.

See Also:

Constant Field Values

`range`

`public static final ParameterValueRangeKind range`

A `ParameterValueRangeKind` object that specifies the parameter is ranged.

discrete

```
public static final ParameterValueRangeKind discrete
```

A ParameterValueRangeKind object that specifies the parameter is discrete.

discreteAndRange

```
public static final ParameterValueRangeKind discreteAndRange
```

A ParameterValueRangeKind object that specifies the parameter supports both ranged and discrete values.

Method Detail

from_int

```
public static final ParameterValueRangeKind from_int(int i)
```

Returns the ParameterValueRangeKind corresponding to the specified value.

Parameters:

i – The int value of the desired ParameterValueRangeKind object.

Returns:

The ParameterValueRangeKind corresponding to the specified value.

toString

```
public java.lang.String toString()
```

Returns the String value of this ParameterValueRangeKind object.

Returns:

The String value of this ParameterValueRangeKind object.

value

```
public int value()
```

Returns the int value of this ParameterValueRangeKind object.

Returns:

The int value of this ParameterValueRangeKind object.

from_string

```
public static final ParameterValueRangeKind from_string(java.lang.String sVal)
```

Returns the ParameterValueRangeKind corresponding to the specified type.

Parameters:

sVal – The String value of the desired ParameterValueRangeKind object.

Returns:

The ParameterValueRangeKind corresponding to the specified type.

com.crystaldecisions.sdk.occa.report.data**Class RangeValueBoundType**

```
java.lang.Object
```

```
└ com.crystaldecisions.sdk.occa.report.data.RangeValueBoundType
```

```
public final class RangeValueBoundType
```

```
extends java.lang.Object
```

This class contains constants that indicate the range bound; that is, how the bound on a range is treated.

Field Summary	
static int	_exclusive The range does not include the value.
static int	_inclusive The range includes the value.
static int	_noBound The range is not bound to a value.
static RangeValueBoundType	exclusive A RangeValueBoundType that specifies the range does not include the value.
static RangeValueBoundType	inclusive A RangeValueBoundType that specifies the range includes the value.
static RangeValueBoundType	noBound A RangeValueBoundType that specifies the range is not bound to a value.

Method Summary	
static RangeValueBoundType	from_int (int i) Returns the RangeValueBoundType corresponding to the

IDE Extensions

	specified value.
static RangeValueBoundType	from_string (java.lang.String sVal) Returns the RangeValueBoundType corresponding to the specified type.
java.lang.String	toString () Returns the String value of this RangeValueBoundType object.
int	value () Returns the int value of this RangeValueBoundType object.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

_noBound

```
public static final int _noBound
```

The range is not bound to a value.

See Also:

Constant Field Values

_exclusive

```
public static final int _exclusive
```

The range does not include the value.

See Also:

Constant Field Values

_inclusive

```
public static final int _inclusive
```

The range includes the value.

See Also:

Constant Field Values

noBound

```
public static final RangeValueBoundType noBound
```

A RangeValueBoundType that specifies the range is not bound to a value.

exclusive

```
public static final RangeValueBoundType exclusive
```

A RangeValueBoundType that specifies the range does not include the value.

inclusive

```
public static final RangeValueBoundType inclusive
```

A RangeValueBoundType that specifies the range includes the value.

Method Detail

from_int

```
public static final RangeValueBoundType from_int(int i)
                                         throws java.lang.IndexOutOfBoundsException
```

Returns the RangeValueBoundType corresponding to the specified value.

Parameters:

i – The int value of the desired RangeValueBoundType object.

Returns:

The RangeValueBoundType corresponding to the specified value.

Throws:

`java.lang.IndexOutOfBoundsException` – This is thrown if the specified int does not correspond to a bound type.

toString

```
public java.lang.String toString()
```

Returns the String value of this RangeValueBoundType object.

Returns:

The String value of this RangeValueBoundType object.

value

```
public int value()
```

Returns the int value of this RangeValueBoundType object.

Returns:

The int value of this RangeValueBoundType object.

from_string

```
public static final RangeValueBoundType from_string(java.lang.String sVal)
                                         throws java.lang.IndexOutOfBoundsException
```

Returns the RangeValueBoundType corresponding to the specified type. Possible bound types are Unbounded, Exclusive, and Inclusive.

Parameters:

sVal – The String value of the desired RangeValueBoundType object.

Returns:

The RangeValueBoundType corresponding to the specified type.

Throws:

java.lang.IndexOutOfBoundsException – This is thrown if the specified String does not correspond to a valid bound type.

com.crystaldecisions.sdk.occa.report.data**Class Value**

```
java.lang.Object
└─ com.crystaldecisions.sdk.occa.report.data.Value
```

All Implemented Interfaces:

IClone, IValue

Direct Known Subclasses:

ParameterFieldValue

public abstract class Value

extends java.lang.Object

implements IValue, IClone

This object defines a value of a field. This is used as a base class to represent different kinds of values (values in formulas, parameters, and so on). It is not to be used directly, and the actual value is defined in a sub-class.

Constructor Summary

Value ()	
------------------	--

Method Summary

abstract java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
abstract java.lang.String	computeText () The computeText method returns a String representation of the field value that can be used in a report formula.
abstract void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
abstract java.lang.String	displayText (java.util.Locale locale) Returns the field's value as a localized and formatted String.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
abstract boolean	hasContent (java.lang.Object srcValue) Returns true if this object contains the same elements as the passed in object.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
void	save (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait,
wait

Constructor Detail

Value

```
public Value()
```

Method Detail

computeText

```
public abstract java.lang.String computeText()
```

Description copied from interface: IValue

The `computeText` method returns a `String` representation of the field value that can be used in a report formula.

The `String` is not formatted or localized. The syntax will always be Crystal syntax. If the field contains a date, `computeText` returns `Date(x, x, x)`; if it contains a date and time, `computeText` returns `DateTime(x, x, x, x, x, x)`. However, when the date is 1900, 0, 1 (the Java start date), `computeText` returns `Time(x, x, x)`. If you want to set a field to a time value only (that is, with no date), you should set the date to 1900, 0, 1, so that `computeText` will retrieve the correct information.

Specified by:

`computeText` in interface `IValue`

Returns:

A `String` representation of the field value that can be used in a report formula.

displayText

```
public abstract java.lang.String displayText(java.util.Locale locale)
```

Description copied from interface: IValue

Returns the field's value as a localized and formatted `String`. The `Locale` can be a `java.util.Locale` value.

The value is formatted according to the locale. The user is required to pass in a locale; the system default locale or user default locale is not used. If the field contains a date, `displayText` returns a locale-specific date string; if it contains a date and time, `displayText` returns a locale-specific date time `String`; if it contains a time `String`, `displayText` returns a locale-specific time `String`.

Specified by:

`displayText` in interface `IValue`

Parameters:

`locale` – The locale as specified by `java.util.Locale`.

Returns:

The field's value as a localized and formatted `String`.

clone

```
public abstract java.lang.Object clone(boolean deepClone)
```

Description copied from interface: IClone

Returns the new object that has been cloned.

Specified by:

clone in interface IClone

Parameters:

deepClone – true to use deep clone, false to use shallow.

Returns:

The new object that has been cloned.

copyTo

```
public abstract void copyTo(java.lang.Object destObject,  
                             boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

copyTo in interface IClone

Parameters:

destObject – The destination object to copy to.

deepCopy – true to use deep copy, false to use shallow.

createMember

```
public java.lang.Object createMember(java.lang.String eleName,  
                                       org.xml.sax.Attributes attrs,  
                                       XMLSerializationContext ctxt,  
                                       java.util.Map objState,  
                                       boolean[] bLoaded)
```

For internal use only.

endElement

```
public void endElement(java.lang.String eleName,  
                       java.util.Map objState)
```

For internal use only.

readElement

```
public void readElement(java.lang.String eleName,  
                        java.lang.String sVal,
```


IDE Extensions

```
org.xml.sax.Attributes attrs,  
java.util.Map objState)
```

For internal use only.

save

```
public void save(XMLWriter writer,  
                 java.lang.String sTag,  
                 XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

```
java.io.IOException
```

save

```
public void save(XMLWriter writer,  
                 XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

```
java.io.IOException
```

hasContent

```
public abstract boolean hasContent(java.lang.Object srcValue)
```

Description copied from interface: IClone

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Parameters:

srcValue – The object to check for content.

Returns:

true if this object contains the same elements as the passed in object, otherwise false.

saveContents

```
public void saveContents(XMLWriter writer,  
                         XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

com.crystaldecisions.sdk.occa.report.data**Class Values**

```

java.lang.Object
├─ java.util.AbstractCollection
│   └─ java.util.AbstractList
│       └─ java.util.ArrayList
│           └─ com.crystaldecisions.sdk.occa.report.lib.ReportSDKVector
│               └─ com.crystaldecisions.sdk.occa.report.data.Values

```

All Implemented Interfaces:

java.lang.Cloneable, java.util.Collection, IClone, java.util.List, java.util.RandomAccess,
java.io.Serializable

*public class **Values****extends ReportSDKVector**implements IClone*

This object defines a collection that contains an array of Value objects and allows you to add, remove, search for, and add new values to and from the collection.

See Also:

Serialized Form

Constructor Summary**Values()****Values**(Values src)**Method Summary**

void	add (double dValue) Appends the specified double value to the end of this collection.
void	add (double beginValue, RangeValueBoundType lowerBoundType, double endValue, RangeValueBoundType upperBoundType) Creates an instance of an IParameterFieldRangeValue object using the specified discrete values as bounds and appends it to the end of the collection.
void	add (int index, java.lang.Object element) Inserts the specified element at the specified position in this collection.

IDE Extensions

boolean	add (java.lang.Object o) Appends the specified element to the end of this collection.
void	add (java.lang.Object beginValue, RangeValueBoundType lowerBoundType, java.lang.Object endValue, RangeValueBoundType upperBoundType) Creates an instance of an IParameterFieldRangeValue object using the specified bounds and appends it to the end of the collection.
void	add (java.lang.String sValue) Appends the specified String to the end of this collection.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
IValue	getValue (int index) Returns the IValue object at the specified index.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
void	save (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.

Methods inherited from class com.crystaldecisions.sdk.occa.report.lib.ReportSDKVector

addElement, clone, copyTo, elementAt, findIndexOf, hasContent, insertElementAt, removeAllElements

Methods inherited from class java.util.ArrayList

addAll, addAll, clear, clone, contains, ensureCapacity, get, indexOf, isEmpty, lastIndexOf, remove, set, size, toArray, toArray, trimToSize

Methods inherited from class java.util.AbstractList

`equals`, `hashCode`, `iterator`, `listIterator`, `listIterator`, `subList`

Methods inherited from class java.util.AbstractCollection

`containsAll`, `remove`, `removeAll`, `retainAll`, `toString`

Methods inherited from class java.lang.Object

`getClass`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Methods inherited from interface com.crystaldecisions.sdk.occa.report.lib.IClone

`clone`, `copyTo`, `hasContent`

Methods inherited from interface java.util.List

`containsAll`, `equals`, `hashCode`, `iterator`, `listIterator`, `listIterator`, `remove`, `removeAll`, `retainAll`, `subList`

Constructor Detail**Values**

```
public Values(Values src)
```

Values

```
public Values()
```

Method Detail**add**

```
public void add(java.lang.String sValue)
```

Appends the specified `String` to the end of this collection.

Parameters:

`sValue` – The `String` to be added to the collection.

add

```
public void add(double dValue)
```

Appends the specified double value to the end of this collection.

Parameters:

dValue – The double to be added to the collection.

getValue

```
public IValue getValue(int index)
```

Returns the IValue object at the specified index.

Parameters:

index – The index of the desired IValue object.

Returns:

The IValue object at the specified index.

readElement

```
public void readElement(java.lang.String eleName,  
                        java.lang.String sVal,  
                        org.xml.sax.Attributes attrs,  
                        java.util.Map objState)
```

For internal use only.

add

```
public void add(double beginValue,  
                RangeValueBoundType lowerBoundType,  
                double endValue,  
                RangeValueBoundType upperBoundType)
```

Creates an instance of an IParameterFieldRangeValue object using the specified discrete values as bounds and appends it to the end of the collection.

Parameters:

beginValue – A double specifying the lower-bound value of the IParameterFieldRangeValue instance that will be added to the collection.

lowerBoundType – Specifies the type of bound to use for the lower-bound value. This can be unbounded, inclusive, or exclusive.

endValue – A double specifying the upper-bound value of the IParameterFieldRangeValue instance that will be added to the collection.

upperBoundType – Specifies the type of bound to use for the upper-bound value. This can be unbounded, inclusive, or exclusive.

See Also:

RangeValueBoundType

add

```
public void add(java.lang.Object beginValue,
                RangeValueBoundType lowerBoundType,
                java.lang.Object endValue,
                RangeValueBoundType upperBoundType)
```

Creates an instance of an `IParameterFieldRangeValue` object using the specified bounds and appends it to the end of the collection.

Parameters:

`beginValue` – The lower-bound value of the `IParameterFieldRangeValue` instance that will be added to the collection.

`lowerBoundType` – Specifies the type of bound to use for the lower-bound value. This can be unbounded, inclusive, or exclusive.

`endValue` – The upper-bound value of the `IParameterFieldRangeValue` instance that will be added to the collection.

`upperBoundType` – Specifies the type of bound to use for the upper-bound value. This can be unbounded, inclusive, or exclusive.

See Also:

`RangeValueBoundType`

createMember

```
public java.lang.Object createMember(java.lang.String eleName,
                                     org.xml.sax.Attributes attrs,
                                     XMLSerializationContext ctxt,
                                     java.util.Map objState,
                                     boolean[] bLoaded)
```

For internal use only.

endElement

```
public void endElement(java.lang.String eleName,
                      java.util.Map objState)
```

For internal use only.

save

```
public void save(XMLWriter writer,
                 java.lang.String sTag,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

`java.io.IOException`

save

```
public void save(XMLWriter writer,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

saveContents

```
public void saveContents(XMLWriter writer,
                         XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

add

```
public boolean add(java.lang.Object o)
```

Appends the specified element to the end of this collection.

Note: Only instances of the IValue interface can be added to a Values collection.

Specified by:

add in interface java.util.List

Parameters:

o – element to be added to the collection.

Returns:

true if this collection changed as a result of the call.

Throws:

java.lang.ClassCastException – if the class of the specified element prevents it from being added to this collection.

java.lang.NullPointerException – if the specified element is null and this collection does not support null elements.

add

```
public void add(int index,
                java.lang.Object element)
```

Inserts the specified element at the specified position in this collection. Shifts the element currently at that position (if any) and any subsequent elements to the right, adding one to their indices.

Specified by:

add in interface java.util.List

Parameters:

`index` – The index at which the specified element will be inserted.
`element` – The element to be inserted.

Throws:

`java.lang.IndexOutOfBoundsException` – if index is out of range (`index < 0` || `index > size()`).
`java.lang.ClassCastException` – class of the specified element prevents it from being added to this collection.
`java.lang.NullPointerException` – if the specified element is null and this collection does not support null elements.

startElement

```
public void startElement(java.lang.String eleName,
                        java.util.Map objState,
                        org.xml.sax.Attributes attrs)
```

For internal use only.

Package com.crystaldecisions.sdk.occa.report.exportoptions

This package allows you to specify the export format of a report document.

See:

Description

Interface Summary	
IExportOptions	This interface provides properties and methods for retrieving information and setting options for exporting your report (such as export format and destination).
IPageBasedExportFormatOptions	This abstract interface defines export options used when exporting to a page based export format.
IPDFExportFormatOptions	This interface provides options that can be set when exporting a report to PDF format.
IRTFWordExportFormatOptions	This interface provides options that can be set when exporting a report to Word or RTF format.

Class Summary	
ExportOptions	This object provides properties and methods for retrieving information and setting options for exporting your report (such as export format and destination).
ExportPageAreaPairKind	This class contains constants used to indicate how the page header and footer pair should be exported.

PageBasedExportFormatOptions	This abstract object defines export options when exporting to a page based export format.
PDFExportFormatOptions	This class provides options that can be set when exporting a report to PDF format.
ReportExportFormat	This class is used to specify the export format of a document.
RTFWordExportFormatOptions	This class provides options that can be set when exporting a report to Word or RTF format.

Package com.crystaldecisions.sdk.occa.report.exportoptions

Description

This package allows you to specify the export format of a report document.

Note: With the exception of `ReportExportFormat`, these classes are used primarily by the Report Application Server Java SDK in order to export reports from the viewers.

com.crystaldecisions.sdk.occa.report.exportoptions

Class ExportOptions

```
java.lang.Object
└─ com.crystaldecisions.sdk.occa.report.exportoptions.ExportOptions
```

All Implemented Interfaces:

IClone, IExportOptions

```
public class ExportOptions
extends java.lang.Object
implements IExportOptions, IClone
```

This object provides properties and methods for retrieving information and setting options for exporting your report (such as export format and destination). When possible, use the `IExportOptions` interface.

Constructor Summary

ExportOptions()

Method Summary

java.lang.Object

IDE Extensions

	clone (boolean deepClone) Returns the new object that has been cloned.
void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
ReportExportFormat	getExportFormatType () Returns the format type for the exported report (for example, text, Excel, etc.).
java.lang.Object	getFormatOptions () Returns the formatting options specified for the export format type.
boolean	hasContent (java.lang.Object obj) Returns true if this object contains the same elements as the passed in object.
void	setExportFormatType (ReportExportFormat exportFormatType) Sets the format type for the exported report (for example, text, Excel, etc.).
void	setFormatOptions (java.lang.Object formatOptions) Sets the formatting options specified for the export format type.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ExportOptions

```
public ExportOptions()
```

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: IClone

Returns the new object that has been cloned.

Specified by:

clone in interface IClone

Parameters:

deepClone – true to use deep clone, false to use shallow.

Returns:

The new object that has been cloned.

copyTo

```
public void copyTo(java.lang.Object destObject,  
                  boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

copyTo in interface IClone

Parameters:

destObject – The destination object to copy to.

deepCopy – true to use deep copy, false to use shallow.

getExportFormatType

```
public ReportExportFormat getExportFormatType()
```

Description copied from interface: IExportOptions

Returns the format type for the exported report (for example, text, Excel, etc.).

Specified by:

getExportFormatType in interface IExportOptions

Returns:

A ReportExportFormat object that specifies the format type of the exported report.

getFormatOptions

```
public java.lang.Object getFormatOptions()
```

Description copied from interface: IExportOptions

Returns the formatting options specified for the export format type.

Specified by:

getFormatOptions in interface IExportOptions

Returns:

An Object that specifies the formatting options specified for the export format type.

hasContent

```
public boolean hasContent(java.lang.Object obj)
```

Description copied from interface: IClone

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Parameters:

obj – The object to check for content.

Returns:

true if this object contains the same elements as the passed in object, otherwise false.

setExportFormatType

```
public void setExportFormatType(ReportExportFormat exportFormatType)
```

Description copied from interface: IExportOptions

Sets the format type for the exported report (for example, text, Excel, etc.).

Specified by:

setExportFormatType in interface IExportOptions

Parameters:

exportFormatType – A ReportExportFormat object that specifies the format type of the exported report.

setFormatOptions

```
public void setFormatOptions(java.lang.Object formatOptions)
```

Description copied from interface: IExportOptions

Sets the formatting options specified for the export format type.

Specified by:

setFormatOptions in interface IExportOptions

Parameters:

formatOptions – An Object that specifies the formatting options specified for the export format type.

com.crystaldecisions.sdk.occa.report.exportoptions Class ExportPageAreaPairKind

```
java.lang.Object
└─ com.crystaldecisions.sdk.occa.report.exportoptions.ExportPageAreaPairKind
```

```
public final class ExportPageAreaPairKind
extends java.lang.Object
```

This class contains constants used to indicate how the page header and footer pair should be exported. This is used by the ExcelExportFormatOptions object.

Note: Only RTF and PDF exporting are supported with Crystal Reports for BEA WebLogic Workshop. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

Field Summary

static int	
------------	--

IDE Extensions

	_forEachPage Specifies that the page header and footer pair will appear on each page.
static int	_noPageAreaPair Specifies that no page header and footer pair will be exported.
static int	_oncePerReport Specifies that the page header and footer pair will only be exported once per report.
static ExportPageAreaPairKind	forEachPage A ExportPageAreaPairKind object that specifies that the page header and footer pair will appear on each page.
static ExportPageAreaPairKind	noPageAreaPair A ExportPageAreaPairKind object that specifies that no page header and footer pair will be exported.
static ExportPageAreaPairKind	oncePerReport A ExportPageAreaPairKind object that specifies that the page header and footer pair will only be exported once per report.

Method Summary

static ExportPageAreaPairKind	from_int (int i) Returns the ExportPageAreaPairKind object corresponding to the specified int.
static ExportPageAreaPairKind	from_string (java.lang.String sVal) Returns the ExportPageAreaPairKind object corresponding to the specified String.
java.lang.String	toString () Returns the String value of this ExportPageAreaPairKind object.
int	value () Returns the int value of this ExportPageAreaPairKind object.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

`_noPageAreaPair`

```
public static final int _noPageAreaPair
```

Specifies that no page header and footer pair will be exported.

See Also:

Constant Field Values

`_oncePerReport`

```
public static final int _oncePerReport
```

Specifies that the page header and footer pair will only be exported once per report. That is, it will only appear on the first page.

See Also:

Constant Field Values

`_forEachPage`

```
public static final int _forEachPage
```

Specifies that the page header and footer pair will appear on each page.

See Also:

Constant Field Values

`noPageAreaPair`

```
public static final ExportPageAreaPairKind noPageAreaPair
```

A `ExportPageAreaPairKind` object that specifies that no page header and footer pair will be exported.

`oncePerReport`

```
public static final ExportPageAreaPairKind oncePerReport
```

A `ExportPageAreaPairKind` object that specifies that the page header and footer pair will only be exported once per report. That is, it will only appear on the first page.

forEachPage

```
public static final ExportPageAreaPairKind forEachPage
```

A ExportPageAreaPairKind object that specifies that the page header and footer pair will appear on each page.

Method Detail

from_int

```
public static final ExportPageAreaPairKind from_int(int i)
```

Returns the ExportPageAreaPairKind object corresponding to the specified int.

Parameters:

i – The int value of the desired ExportPageAreaPairKind.

Returns:

The ExportPageAreaPairKind object corresponding to the specified int.

from_string

```
public static final ExportPageAreaPairKind from_string(java.lang.String sVal)
```

Returns the ExportPageAreaPairKind object corresponding to the specified String.

Parameters:

sVal – The String value of the desired ExportPageAreaPairKind.

Returns:

The ExportPageAreaPairKind object corresponding to the specified String.

toString

```
public java.lang.String toString()
```

Returns the String value of this ExportPageAreaPairKind object.

Returns:

The String value of this ExportPageAreaPairKind object.

value

```
public int value()
```

Returns the int value of this ExportPageAreaPairKind object.

Returns:

The int value of this ExportPageAreaPairKind object.

com.crystaldecisions.sdk.occa.report.exportoptions

Interface IExportOptions

All Known Implementing Classes:
ExportOptions

public interface IExportOptions

This interface provides properties and methods for retrieving information and setting options for exporting your report (such as export format and destination).

Method Summary	
ReportExportFormat	getExportFormatType() Returns the format type for the exported report (for example, text, Excel, etc.).
java.lang.Object	getFormatOptions() Returns the formatting options specified for the export format type.
void	setExportFormatType (ReportExportFormat exportFormat) Sets the format type for the exported report (for example, text, Excel, etc.).
void	setFormatOptions (java.lang.Object formatOptions) Sets the formatting options specified for the export format type.

Method Detail

getExportFormatType

```
public ReportExportFormat getExportFormatType()
```

Returns the format type for the exported report (for example, text, Excel, etc.).

Returns:

A ReportExportFormat object that specifies the format type of the exported report.

getFormatOptions

```
public java.lang.Object getFormatOptions()
```


Returns the formatting options specified for the export format type.

Returns:

An Object that specifies the formatting options specified for the export format type.

setExportFormatType

```
public void setExportFormatType(ReportExportFormat exportFormatType)
```

Sets the format type for the exported report (for example, text, Excel, etc.).

Parameters:

exportFormatType – A ReportExportFormat object that specifies the format type of the exported report.

setFormatOptions

```
public void setFormatOptions(java.lang.Object formatOptions)
```

Sets the formatting options specified for the export format type.

Parameters:

formatOptions – An Object that specifies the formatting options specified for the export format type.

com.crystaldecisions.sdk.occa.report.exportoptions Interface IPageBasedExportFormatOptions

All Known Subinterfaces:

IPDFExportFormatOptions, IRTFWordExportFormatOptions

All Known Implementing Classes:

PageBasedExportFormatOptions, PDFExportFormatOptions, RTFWordExportFormatOptions

public interface **IPageBasedExportFormatOptions**

This abstract interface defines export options used when exporting to a page based export format. This includes specifying a range of pages to be exported. Page based export formats include PDF and RTF.

Method Summary

int	getEndPageNumber () Returns the end page, of a page export range, when exporting to a page based format.
int	getStartPageNumber () Returns the start page, of a page export range, when exporting to a page based format.
void	setEndPageNumber (int pageNumber) Sets the start page, of a page export range, when exporting to a page based format.

void	setStartPageNumber (int pageNumber) Sets the end page, of a page export range, when exporting to a page based format.
------	---

Method Detail

getEndPageNumber

```
public int getEndPageNumber()
```

Returns the end page, of a page export range, when exporting to a page based format. By default, all pages in the report are exported.

Returns:

An int that specifies the end page.

getStartPageNumber

```
public int getStartPageNumber()
```

Returns the start page, of a page export range, when exporting to a page based format. By default, all pages in the report are exported.

Returns:

An int that specifies the start page.

setEndPageNumber

```
public void setEndPageNumber(int pageNumber)
```

Sets the start page, of a page export range, when exporting to a page based format. By default, all pages in the report are exported.

Parameters:

pageNumber – An int that specifies the start page.

setStartPageNumber

```
public void setStartPageNumber(int pageNumber)
```

Sets the end page, of a page export range, when exporting to a page based format. By default, all pages in the report are exported.

Parameters:

pageNumber – An int that specifies the end page.

com.crystaldecisions.sdk.occa.report.exportoptions Interface IPDFExportFormatOptions

All Superinterfaces:

IPageBasedExportFormatOptions

All Known Implementing Classes:

PDFExportFormatOptions

public interface **IPDFExportFormatOptions**
extends *IPageBasedExportFormatOptions*

This interface provides options that can be set when exporting a report to PDF format.

Methods inherited from interface

com.crystaldecisions.sdk.occa.report.exportoptions.IPageBasedExportFormatOptions

getEndPageNumber, getStartPageNumber, setEndPageNumber,
setStartPageNumber

com.crystaldecisions.sdk.occa.report.exportoptions Interface IRTFWordExportFormatOptions

All Superinterfaces:

IPageBasedExportFormatOptions

All Known Implementing Classes:

RTFWordExportFormatOptions

public interface **IRTFWordExportFormatOptions**
extends *IPageBasedExportFormatOptions*

This interface provides options that can be set when exporting a report to Word or RTF format.

Note: Only RTF and PDF exporting are supported with Crystal Reports for BEA WebLogic Workshop. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

Methods inherited from interface

com.crystaldecisions.sdk.occa.report.exportoptions.IPageBasedExportFormatOptions

getEndPageNumber, getStartPageNumber, setEndPageNumber,
setStartPageNumber

com.crystaldecisions.sdk.occa.report.exportoptions

Class PageBasedExportFormatOptions

java.lang.Object

└─ com.crystaldecisions.sdk.occa.report.exportoptions.PageBasedExportFormatOptions

All Implemented Interfaces:

IClone, IPageBasedExportFormatOptions

Direct Known Subclasses:

PDFExportFormatOptions, RTFWordExportFormatOptions

*public abstract class **PageBasedExportFormatOptions***

extends java.lang.Object

implements IPageBasedExportFormatOptions, IClone

This abstract object defines export options when exporting to a page based export format. This includes specifying a range of pages to be exported. Page based export formats include PDF and RTF.

Constructor Summary

PageBasedExportFormatOptions ()

Method Summary

java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attributes, com.crystaldecisions.xml.serialization.XMLSerializationContext context, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
int	getEndPageNumber () Returns the end page, of a page export range, when exporting to a page based format.
int	getStartPageNumber () Returns the start page, of a page export range, when exporting to a page based format.
boolean	

IDE Extensions

	hasContent (java.lang.Object obj) Returns true if this object contains the same elements as the passed in object.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, com.crystaldecisions.xml.serialization.XMLSerializationContext c) For internal use only.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, java.lang.String sTag, com.crystaldecisions.xml.serialization.XMLSerializationContext c) For internal use only.
void	saveContents (com.crystaldecisions.xml.serialization.XMLWriter wr, com.crystaldecisions.xml.serialization.XMLSerializationContext c) For internal use only.
void	setEndPageNumber (int pageNumber) Sets the start page, of a page export range, when exporting to a page based format.
void	setStartPageNumber (int pageNumber) Sets the end page, of a page export range, when exporting to a page based format.
void	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PageBasedExportFormatOptions

```
public PageBasedExportFormatOptions()
```

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: **IClone**

Returns the new object that has been cloned.

Specified by:

clone in interface IClone

Parameters:

deepClone – true to use deep clone, false to use shallow.

Returns:

The new object that has been cloned.

copyTo

```
public void copyTo(java.lang.Object destObject,
                  boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

copyTo in interface IClone

Parameters:

destObject – The destination object to copy to.

deepCopy – true to use deep copy, false to use shallow.

createMember

```
public java.lang.Object createMember(java.lang.String eleName,
                                     org.xml.sax.Attributes attrs,
                                     com.crystaldecisions.xml.serialization.XMLSerializationCon
                                     java.util.Map objState,
                                     boolean[] bLoaded)
```

For internal use only.

endElement

```
public void endElement(java.lang.String eleName,
                      java.util.Map objState)
```

For internal use only.

getEndPageNumber

```
public int getEndPageNumber()
```

Description copied from interface: IPageBasedExportFormatOptions

Returns the end page, of a page export range, when exporting to a page based format. By default, all pages in the report are exported.

Specified by:

getEndPageNumber in interface IPageBasedExportFormatOptions

Returns:

An int that specifies the end page.

getStartPageNumber

```
public int getStartPageNumber()
```

Description copied from interface: IPageBasedExportFormatOptions

Returns the start page, of a page export range, when exporting to a page based format. By default, all pages in the report are exported.

Specified by:

getStartPageNumber in interface IPageBasedExportFormatOptions

Returns:

An int that specifies the start page.

hasContent

```
public boolean hasContent(java.lang.Object obj)
```

Description copied from interface: IClone

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Parameters:

obj – The object to check for content.

Returns:

true if this object contains the same elements as the passed in object, otherwise false.

readElement

```
public void readElement(java.lang.String eleName,  
                        java.lang.String sVal,  
                        org.xml.sax.Attributes attrs,  
                        java.util.Map objState)
```

For internal use only.

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,  
                com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,
                 java.lang.String sTag,
                 com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

saveContents

```
public void saveContents(com.crystaldecisions.xml.serialization.XMLWriter writer,
                         com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

setEndPageNumber

```
public void setEndPageNumber(int pageNumber)
```

Description copied from interface: IPageBasedExportFormatOptions

Sets the start page, of a page export range, when exporting to a page based format. By default, all pages in the report are exported.

Specified by:

setEndPageNumber in interface IPageBasedExportFormatOptions

Parameters:

pageNumber – An int that specifies the start page.

setStartPageNumber

```
public void setStartPageNumber(int pageNumber)
```

Description copied from interface: IPageBasedExportFormatOptions

Sets the end page, of a page export range, when exporting to a page based format. By default, all pages in the report are exported.

Specified by:

setStartPageNumber in interface IPageBasedExportFormatOptions

Parameters:

pageNumber – An int that specifies the end page.

startElement

```
public void startElement(java.lang.String eleName,
                        java.util.Map objState,
                        org.xml.sax.Attributes attrs)
```

For internal use only.

com.crystaldecisions.sdk.occa.report.exportoptions

Class PDFExportFormatOptions

```
java.lang.Object
├ com.crystaldecisions.sdk.occa.report.exportoptions.PageBasedExportFormatOptions
└ com.crystaldecisions.sdk.occa.report.exportoptions.PDFExportFormatOptions
```

All Implemented Interfaces:

IClone, IPageBasedExportFormatOptions, IPDFExportFormatOptions

*public class **PDFExportFormatOptions***
*extends **PageBasedExportFormatOptions***
*implements **IPDFExportFormatOptions***

This class provides options that can be set when exporting a report to PDF format. When possible use the IPDFExportFormatOptions interface.

Constructor Summary

PDFExportFormatOptions()

Method Summary

java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, com.crystaldecisions.xml.serialization.XMLSerializationContext c) For internal use only.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, java.lang.String sTag, com.crystaldecisions.xml.serialization.XMLSerializationContext c) For internal use only.

Methods inherited from class

com.crystaldecisions.sdk.occa.report.exportoptions.PageBasedExportFormatOptions

```
copyTo, createMember, endElement, getEndPageNumber, getStartPageNumber,
hasContent, readElement, saveContents, setEndPageNumber,
setStartPageNumber, startElement
```

Methods inherited from class `java.lang.Object`

```
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait,
wait
```

Methods inherited from interface

`com.crystaldecisions.sdk.occa.report.exportoptions.IPageBasedExportFormatOptions`

```
getEndPageNumber, getStartPageNumber, setEndPageNumber,
setStartPageNumber
```

Constructor Detail

PDFExportFormatOptions

```
public PDFExportFormatOptions()
```

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: `IClone`

Returns the new object that has been cloned.

Specified by:

clone in interface `IClone`

Overrides:

clone in class `PageBasedExportFormatOptions`

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,
                 com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)
                 throws java.io.IOException
```

Description copied from class: `PageBasedExportFormatOptions`

For internal use only.

Overrides:

save in class `PageBasedExportFormatOptions`

Throws:

java.io.IOException

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,
                 java.lang.String sTag,
                 com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)
    throws java.io.IOException
```

Description copied from class: PageBasedExportFormatOptions

For internal use only.

Overrides:

save in class PageBasedExportFormatOptions

Throws:

java.io.IOException

com.crystaldecisions.sdk.occa.report.exportoptions

Class ReportExportFormat

java.lang.Object

└ com.crystaldecisions.sdk.occa.report.exportoptions.ReportExportFormat

*public final class **ReportExportFormat***
extends java.lang.Object

This class is used to specify the export format of a document.

Note: Only RTF and PDF exporting are supported with Crystal Reports for BEA WebLogic Workshop. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

Field Summary	
static int	_characterSeparatedValues Format as character separated values.
static int	_crystalReports Format as a Crystal Report file.
static int	_MSExcel Format as a Microsoft Excel spreadsheet.
static int	_MSWord Format as a Microsoft Word document.
static int	_PDF Format as an Adobe PDF document.
static int	_recordToMSExcel Format as raw Excel data values.

IDE Extensions

<code>static int</code>	_RTF Format as a Rich Text Format document.
<code>static int</code>	_text Format as plain text.
<code>static ReportExportFormat</code>	characterSeparatedValues A ReportExportFormat object that specifies to format as character separated values.
<code>static ReportExportFormat</code>	crystalReports A ReportExportFormat object that specifies to format as a Crystal Report file.
<code>static ReportExportFormat</code>	MSExcel A ReportExportFormat object that specifies to format as a Microsoft Excel spreadsheet.
<code>static ReportExportFormat</code>	MSWord A ReportExportFormat object that specifies to format as a Microsoft Word document.
<code>static ReportExportFormat</code>	PDF A ReportExportFormat object that specifies to format as an Adobe PDF document.
<code>static ReportExportFormat</code>	recordToMSExcel A ReportExportFormat object that specifies to format as raw Excel data values.
<code>static ReportExportFormat</code>	RTF A ReportExportFormat object that specifies to format as a Rich Text Format document.
<code>static ReportExportFormat</code>	text A ReportExportFormat object that specifies to format as plain text.

Method Summary

<code>static ReportExportFormat</code>	from_int (int i) Returns a ReportExportFormat object corresponding to the specified int.
<code>static ReportExportFormat</code>	from_string (java.lang.String sVal) Returns a ReportExportFormat object corresponding to the specified String.
<code>java.lang.String</code>	toString () Returns the String value of this ReportExportFormat object.
<code>int</code>	value () Returns the value of this ReportExportFormat object.

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Field Detail**_crystalReports**

```
public static final int _crystalReports
```

Format as a Crystal Report file.

See Also:

Constant Field Values

_MSWord

```
public static final int _MSWord
```

Format as a Microsoft Word document.

See Also:

Constant Field Values

_MSExcel

```
public static final int _MSExcel
```

Format as a Microsoft Excel spreadsheet.

See Also:

Constant Field Values

_RTF

```
public static final int _RTF
```

Format as a Rich Text Format document.

See Also:

Constant Field Values

_PDF

```
public static final int _PDF
```

Format as an Adobe PDF document.

See Also:

Constant Field Values

_recordToMSExcel

```
public static final int _recordToMSExcel
```

Format as raw Excel data values.

See Also:

Constant Field Values

_text

```
public static final int _text
```

Format as plain text.

See Also:

Constant Field Values

_characterSeparatedValues

```
public static final int _characterSeparatedValues
```

Format as character separated values.

See Also:

Constant Field Values

crystalReports

```
public static final ReportExportFormat crystalReports
```

A ReportExportFormat object that specifies to format as a Crystal Report file.

MSWord

```
public static final ReportExportFormat MSWord
```

A ReportExportFormat object that specifies to format as a Microsoft Word document.

MSExcel

```
public static final ReportExportFormat MSExcel
```

A ReportExportFormat object that specifies to format as a Microsoft Excel spreadsheet.

RTF

```
public static final ReportExportFormat RTF
```

A ReportExportFormat object that specifies to format as a Rich Text Format document.

PDF

```
public static final ReportExportFormat PDF
```

A ReportExportFormat object that specifies to format as an Adobe PDF document.

recordToMSExcel

```
public static final ReportExportFormat recordToMSExcel
```

A ReportExportFormat object that specifies to format as raw Excel data values.

text

```
public static final ReportExportFormat text
```

A ReportExportFormat object that specifies to format as plain text.

characterSeparatedValues

```
public static final ReportExportFormat characterSeparatedValues
```

A ReportExportFormat object that specifies to format as character separated values.

Method Detail

from_int

```
public static final ReportExportFormat from_int(int i)
```

Returns a ReportExportFormat object corresponding to the specified int.

Parameters:

i – The int value of the ReportExportFormat object to be returned.

Returns:

A ReportExportFormat object corresponding to the specified int.

from_string

```
public static final ReportExportFormat from_string(java.lang.String sVal)
```

Returns a ReportExportFormat object corresponding to the specified String.

Parameters:

sVal – The value of the ReportExportFormat object to be returned.

Returns:

A ReportExportFormat object corresponding to the specified String.

toString

```
public java.lang.String toString()
```

Returns the String value of this ReportExportFormat object. This String specifies the format that this document will be exported as.

Returns:

A String that specifies the format this document will be exported as.

value

```
public int value()
```

Returns the value of this ReportExportFormat object. This value specifies the format that this documented will be exported as.

Returns:

An int that specifies the value of this ReportExportFormat object.

com.crystaldecisions.sdk.occa.report.exportoptions

Class RTFWordExportFormatOptions

```
java.lang.Object
```

```
└ com.crystaldecisions.sdk.occa.report.exportoptions.PageBasedExportFormatOptions
```

```
└ com.crystaldecisions.sdk.occa.report.exportoptions.RTFWordExportFormatOptions
```


All Implemented Interfaces:

IClone, IPageBasedExportFormatOptions, IRTFWordExportFormatOptions

public class **RTFWordExportFormatOptions**
extends *PageBasedExportFormatOptions*
implements *IRTFWordExportFormatOptions*

This class provides options that can be set when exporting a report to Word or RTF format. When possible, use the IRTFWordExportFormatOptions interface.

Note: Only RTF and PDF exporting are supported with Crystal Reports for BEA WebLogic Workshop. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

Constructor Summary

RTFWordExportFormatOptions ()

Method Summary

java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, com.crystaldecisions.xml.serialization.XMLSerializationContext c) For internal use only.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, java.lang.String sTag, com.crystaldecisions.xml.serialization.XMLSerializationContext c) For internal use only.

Methods inherited from class

com.crystaldecisions.sdk.occa.report.exportoptions.PageBasedExportFormatOptions

copyTo, createMember, endElement, getEndPageNumber, getStartPageNumber, hasContent, readElement, saveContents, setEndPageNumber, setStartPageNumber, startElement

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface

com.crystaldecisions.sdk.occa.report.exportoptions.IPageBasedExportFormatOptions

```
getEndPageNumber, getStartPageNumber, setEndPageNumber,
setStartPageNumber
```

Constructor Detail

RTFWordExportFormatOptions

```
public RTFWordExportFormatOptions()
```

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: `IClone`

Returns the new object that has been cloned.

Specified by:

clone in interface `IClone`

Overrides:

clone in class `PageBasedExportFormatOptions`

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,
                 com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)
                 throws java.io.IOException
```

Description copied from class: `PageBasedExportFormatOptions`

For internal use only.

Overrides:

save in class `PageBasedExportFormatOptions`

Throws:

`java.io.IOException`

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,
                 java.lang.String sTag,
                 com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)
                 throws java.io.IOException
```

Description copied from class: `PageBasedExportFormatOptions`

For internal use only.

Overrides:

save in class PageBasedExportFormatOptions

Throws:

java.io.IOException

Package com.crystaldecisions.sdk.occa.report.lib

This package is a general utility that provides exception and container classes.

See:

Description

Interface Summary	
IClone	This interface provides methods for cloning and copying objects.
IStrings	This interface defines a collection of <code>String</code> objects.

Class Summary	
PropertyBag	This class defines a property bag that stores a list of properties.
PropertyBagHelper	This class defines constants that can be used as keys to retrieve or specify properties contained by certain property bags.
PropertyBags	This class defines a collection of property bags.
ReportSDKVector	This class defines a vector data structure with support for specific Report Application Server SDK functionality.
Strings	This class defines a collection of <code>String</code> objects.

Package com.crystaldecisions.sdk.occa.report.lib Description

This package is a general utility that provides exception and container classes. The classes contained in this package are used by many of the objects in this SDK.

com.crystaldecisions.sdk.occa.report.lib

Interface IClone

All Known Implementing Classes:

ConnectionInfo, ConnectionInfos, ExportOptions, Field, FormulaField, PageBasedExportFormatOptions, ParameterField, ParameterFieldDiscreteValue, ParameterFieldRangeValue, ParameterFieldValue, PropertyBag, PropertyBags, ReportSDKVector, Value, Values

*public interface **IClone***

This interface provides methods for cloning and copying objects.

Method Summary	
java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
void	copyTo (java.lang.Object destObject, boolean deepCopy) Copies the object.
boolean	hasContent (java.lang.Object obj) Returns true if this object contains the same elements as the passed in object.

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Returns the new object that has been cloned.

Parameters:

deepClone – true to use deep clone, false to use shallow.

Returns:

The new object that has been cloned.

copyTo

```
public void copyTo(java.lang.Object destObject,  
boolean deepCopy)
```

Copies the object.

Parameters:

destObject – The destination object to copy to.

deepCopy – true to use deep copy, false to use shallow.

hasContent

```
public boolean hasContent(java.lang.Object obj)
```

Returns true if this object contains the same elements as the passed in object.

Parameters:

obj – The object to check for content.

Returns:

true if this object contains the same elements as the passed in object, otherwise false.

com.crystaldecisions.sdk.occa.report.lib

Interface IStrings

All Superinterfaces:

java.util.Collection, java.util.List

All Known Implementing Classes:

Strings

*public interface **IStrings***

extends java.util.List

This interface defines a collection of String objects.

Method Summary

boolean	equalsIgnoreCase (IStrings srcStrings) Returns true if the specified IStrings object is equal to this object, regardless of case.
java.lang.String	getString (int index) Returns the String at the specified index.

Methods inherited from interface java.util.List

add, add, addAll, addAll, clear, contains, containsAll, equals, get, hashCode, indexOf, isEmpty, iterator, lastIndexOf, listIterator, listIterator, remove, remove, removeAll, retainAll, set, size, subList, toArray, toArray

Method Detail

equalsIgnoreCase

```
public boolean equalsIgnoreCase(IStrings srcStrings)
```

Returns true if the specified IStrings object is equal to this object, regardless of case. In other words, this method performs a case-insensitive equality test.

Parameters:

srcStrings – The IStrings object to be compared to.

Returns:

true if the specified IStrings object is equal to this object, and false otherwise.

getString

```
public java.lang.String getString(int index)
```

Returns the String at the specified index.

Parameters:

index – The index of the desired String.

Returns:

The String at the specified index.

com.crystaldecisions.sdk.occa.report.lib Class PropertyBag

```
java.lang.Object
├─ java.util.AbstractMap
│   └─ java.util.HashMap
│       └─ com.crystaldecisions.sdk.occa.report.lib.PropertyBag
```

All Implemented Interfaces:

java.lang.Cloneable, IClone, java.util.Map, java.io.Serializable

*public class **PropertyBag**
extends java.util.HashMap
implements IClone*

This class defines a property bag that stores a list of properties. The properties contained in the bag may be of different types: they may be any of the primitive types, or an object. Use the get and put methods to retrieve and store properties.

See Also:

Serialized Form

Constructor Summary

PropertyBag()

	Constructs an empty property bag with the default capacity and load factor.
PropertyBag (java.util.Map defaults)	Constructs a new property bag with the same mappings as the given map.
PropertyBag (PropertyBag src)	Constructs a new property bag with the same properties as the specified PropertyBag object.

Method Summary	
java.lang.Object	clone (boolean deepClone) Returns the new object that has been cloned.
void	copyTo (java.lang.Object destObj, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
boolean	getBooleanValue (java.lang.Object key) Converts a value in the property bag and returns it as a boolean value. 0 is treated as false.
double	getDoubleValue (java.lang.Object key) Converts a value in the property bag and returns it as a double value.
int	getIntValue (java.lang.Object key) Converts a value in the property bag and returns it as an int value.
IStrings	getPropertyIDs () Returns a list of IDs for all the properties stored in the property bag.
java.lang.String	getStringValue (java.lang.Object key) Converts a value in the property bag and returns it as a string value.
boolean	hasContent (java.lang.Object obj) Returns true if this object contains the same elements as the passed in object.
void	putBooleanValue (java.lang.Object key, boolean value) Associates the supplied property ID with the specified boolean.
void	putDoubleValue (java.lang.Object key, double value) Associates the supplied property ID with the specified double.
void	putIntValue (java.lang.Object key, int value) Associates the supplied property ID with the specified int.
void	putStringValue (java.lang.Object key, java.lang.String value)

IDE Extensions

	Associates the supplied property ID with the specified String.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
void	save (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.

Methods inherited from class java.util.HashMap

clear, clone, containsKey, containsValue, entrySet, get, isEmpty, keySet, put, putAll, remove, size, values

Methods inherited from class java.util.AbstractMap

equals, hashCode, toString

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.util.Map

equals, hashCode

Constructor Detail

PropertyBag

```
public PropertyBag(PropertyBag src)
```

Constructs a new property bag with the same properties as the specified PropertyBag object. This is equivalent to creating a new copy.

Parameters:

src – The PropertyBag object to be copied.

PropertyBag

```
public PropertyBag()
```

Constructs an empty property bag with the default capacity and load factor.

PropertyBag

```
public PropertyBag(java.util.Map defaults)
```

Constructs a new property bag with the same mappings as the given map.

Parameters:

`defaults` – The map whose mappings are to be placed in this property bag.

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: IClone

Returns the new object that has been cloned.

Specified by:

`clone` in interface `IClone`

Parameters:

`deepClone` – `true` to use deep clone, `false` to use shallow.

Returns:

The new object that has been cloned.

copyTo

```
public void copyTo(java.lang.Object destObj,
                  boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

`copyTo` in interface `IClone`

Parameters:

`destObj` – The destination object to copy to.

`deepCopy` – `true` to use deep copy, `false` to use shallow.

createMember

```
public java.lang.Object createMember(java.lang.String eleName,
                                     org.xml.sax.Attributes attrs,
                                     XMLSerializationContext ctxt,
                                     java.util.Map objState,
                                     boolean[] bLoaded)
```

For internal use only.

endElement

```
public void endElement(java.lang.String eleName,
                      java.util.Map objState)
```

For internal use only.

getBooleanValue

```
public boolean getBooleanValue(java.lang.Object key)
```

Converts a value in the property bag and returns it as a boolean value. 0 is treated as false. All other numbers (positive and negative) are treated as true. The string "No" is converted to false, while the string "Yes" is converted to true. The strings "True" and "False" can also be used. A non-zero, numeric string will return true. If the value cannot be converted, the property fails.

Parameters:

key – The ID of the property to convert and return.

Returns:

A boolean corresponding to the value of the specified key.

getDoubleValue

```
public double getDoubleValue(java.lang.Object key)
```

Converts a value in the property bag and returns it as a double value. If the value cannot be converted, the property fails.

Parameters:

key – The ID of the property to convert and return.

Returns:

A double corresponding to the value of the specified key.

getIntValue

```
public int getIntValue(java.lang.Object key)
```

Converts a value in the property bag and returns it as an int value.

Parameters:

key – The ID of the property to convert and return.

Returns:

An int corresponding to the value of the specified key.

getPropertyIDs

```
public IStrings getPropertyIDs()
```

Returns a list of IDs for all the properties stored in the property bag. You can use this list together with the `get` method to enumerate through all the items in the property bag.

Returns:

An IStrings object containing a list of IDs for all the properties stored in the property bag.

getStringValue

```
public java.lang.String getStringValue(java.lang.Object key)
```

Converts a value in the property bag and returns it as a string value. If the value cannot be converted, the property fails.

Parameters:

key – The ID of the property to convert and return.

Returns:

A String corresponding to the value of the specified key.

hasContent

```
public boolean hasContent(java.lang.Object obj)
```

Description copied from interface: IClone

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Parameters:

obj – The object to check for content.

Returns:

true if this object contains the same elements as the passed in object, otherwise false.

putBooleanValue

```
public void putBooleanValue(java.lang.Object key,  
                             boolean value)
```

Associates the supplied property ID with the specified boolean.

Parameters:

key – The ID of the property to assign the value to.

value – The boolean value of the specified property ID.

putDoubleValue

```
public void putDoubleValue(java.lang.Object key,  
                           double value)
```

Associates the supplied property ID with the specified double.

Parameters:

key – The ID of the property to assign the value to.

value – The double value of the specified property ID.

putIntValue

```
public void putIntValue(java.lang.Object key,  
                        int value)
```

Associates the supplied property ID with the specified int.

Parameters:

key – The ID of the property to assign the value to.

value – The int value of the specified property ID.

putStringValue

```
public void putStringValue(java.lang.Object key,  
                           java.lang.String value)
```

Associates the supplied property ID with the specified String.

Parameters:

key – The ID of the property to assign the value to.

value – The String value of the specified property ID.

readElement

```
public void readElement(java.lang.String eleName,  
                        java.lang.String sVal,  
                        org.xml.sax.Attributes attrs,  
                        java.util.Map objState)
```

For internal use only.

save

```
public void save(XMLWriter writer,  
                 XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

save

```
public void save(XMLWriter writer,
                 java.lang.String sTag,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

saveContents

```
public void saveContents(XMLWriter writer,
                         XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

startElement

```
public void startElement(java.lang.String eleName,
                         java.util.Map objState,
                         org.xml.sax.Attributes attrs)
```

For internal use only.

com.crystaldecisions.sdk.occa.report.lib Class PropertyBagHelper

```
java.lang.Object
└─ com.crystaldecisions.sdk.occa.report.lib.PropertyBagHelper
```

```
public final class PropertyBagHelper
    extends java.lang.Object
```

This class defines constants that can be used as keys to retrieve or specify properties contained by certain property bags.

Field Summary

<code>static java.lang.String</code>	CONNINFO_CONNECTION_STRING A <code>String</code> that supplies extra parameters to the SQL server if it needs them.
<code>static java.lang.String</code>	CONNINFO_CRQE_DATABASENAME The name of the database that is used to form the connection string.
<code>static java.lang.String</code>	CONNINFO_CRQE_DATABASETYPE A user-friendly description of the database being accessed.
<code>static java.lang.String</code>	CONNINFO_CRQE_LOGONPROPERTIES The logon properties that form the connection string.
<code>static java.lang.String</code>	CONNINFO_CRQE_SERVERDESCRIPTION A user-friendly description of the server.
<code>static java.lang.String</code>	CONNINFO_CRQE_SQLDB <code>true</code> if the data source supports SQL.
<code>static java.lang.String</code>	CONNINFO_DATABASE_DLL The DLL that is being used to communicate with the database.
<code>static java.lang.String</code>	CONNINFO_DATABASE_NAME The name of the database being accessed.
<code>static java.lang.String</code>	CONNINFO_SERVER_NAME The name of the server.
<code>static java.lang.String</code>	CONNINFO_SERVER_TYPE The type of the server.
<code>static java.lang.String</code>	CONNINFO_URI The URI (Uniform Resource Identifier) or path of the database file.
<code>static java.lang.String</code>	PROMPTPROPERTY_ALL Return all database prompts.
<code>static java.lang.String</code>	PROMPTPROPERTY_INCLUDEMETADATASUBCONNECTIONS Include the metadata on demand.
<code>static java.lang.String</code>	PROMPTPROPERTY_INCLUDEOLAPCONNECTIONS Include the OLAP data on demand.
<code>static java.lang.String</code>	PROMPTPROPERTY_INCLUDEONDEMANDSUBREPORT Include the subreport on demand.
<code>static java.lang.String</code>	PROMPTPROPERTY_REPORTNAME Prompt for report name.

Constructor Summary

PropertyBagHelper ()

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Field Detail**CONNINFO_DATABASE_DLL**

```
public static final java.lang.String CONNINFO_DATABASE_DLL
```

The DLL that is being used to communicate with the database. Applies to all connections. Used by `ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

CONNINFO_SERVER_TYPE

```
public static final java.lang.String CONNINFO_SERVER_TYPE
```

The type of the server. Applies to SQL connections. Used by `ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

CONNINFO_SERVER_NAME

```
public static final java.lang.String CONNINFO_SERVER_NAME
```

The name of the server. Applies to SQL connections. Used by `ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

CONNINFO_DATABASE_NAME

```
public static final java.lang.String CONNINFO_DATABASE_NAME
```

The name of the database being accessed. Applies to SQL connections. Used by

`ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

CONNINFO_CONNECTION_STRING

```
public static final java.lang.String CONNINFO_CONNECTION_STRING
```

A `String` that supplies extra parameters to the SQL server if it needs them. Applies to SQL connections. Used by `ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

CONNINFO_URI

```
public static final java.lang.String CONNINFO_URI
```

The URI (Uniform Resource Identifier) or path of the database file. Applies to `MetaData`, `Query`, and `File` connections. Used by `ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

CONNINFO_CRQE_DATABASETYPE

```
public static final java.lang.String CONNINFO_CRQE_DATABASETYPE
```

A user-friendly description of the database being accessed. Applies to `Crystal Report Query Engine (CRQE)` connections. Used by `ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

CONNINFO_CRQE_SERVERDESCRIPTION

```
public static final java.lang.String CONNINFO_CRQE_SERVERDESCRIPTION
```

A user-friendly description of the server. Applies to `CRQE` connections. Used by `ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

CONNINFO_CRQE_SQLDB

```
public static final java.lang.String CONNINFO_CRQE_SQLDB
```

true if the data source supports SQL. Used by `ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

CONNINFO_CRQE_LOGONPROPERTIES

```
public static final java.lang.String CONNINFO_CRQE_LOGONPROPERTIES
```

The logon properties that form the connection string. Applies to CRQE connections. Used by `ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

CONNINFO_CRQE_DATABASENAME

```
public static final java.lang.String CONNINFO_CRQE_DATABASENAME
```

The name of the database that is used to form the connection string. Applies to CRQE connections. Used by `ConnectionInfo.getAttributes()`.

See Also:

Constant Field Values

PROMPTPROPERTY_INCLUDEONDEMANDSUBREPORT

```
public static final java.lang.String PROMPTPROPERTY_INCLUDEONDEMANDSUBREPORT
```

Include the subreport on demand. This property is only used by the Report Application Server. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

See Also:

Constant Field Values

PROMPTPROPERTY_REPORTNAME

```
public static final java.lang.String PROMPTPROPERTY_REPORTNAME
```

Prompt for report name. This property is only used by the Report Application Server. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

See Also:

Constant Field Values

PROMPTPROPERTY_ALL

```
public static final java.lang.String PROMPTPROPERTY_ALL
```

Return all database prompts. This property is only used by the Report Application Server. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

See Also:

Constant Field Values

PROMPTPROPERTY_INCLUDEMETADATASUBCONNECTIONS

```
public static final java.lang.String PROMPTPROPERTY_INCLUDEMETADATASUBCONNECTIONS
```

Include the metadata on demand. This property is only used by the Report Application Server. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

See Also:

Constant Field Values

PROMPTPROPERTY_INCLUDEOLAPCONNECTIONS

```
public static final java.lang.String PROMPTPROPERTY_INCLUDEOLAPCONNECTIONS
```

Include the OLAP data on demand. This property is only used by the Report Application Server. Please consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide* for more information.

See Also:

Constant Field Values

Constructor Detail

PropertyBagHelper

```
public PropertyBagHelper()
```

com.crystaldecisions.sdk.occa.report.lib

Class PropertyBags

```
java.lang.Object
└─ java.util.AbstractCollection
```

```

└─ java.util.AbstractList
    └─ java.util.ArrayList
        └─ com.crystaldecisions.sdk.occa.report.lib.ReportSDKVector
            └─ com.crystaldecisions.sdk.occa.report.lib.PropertyBags

```

All Implemented Interfaces:

java.lang.Cloneable, java.util.Collection, IClone, java.util.List, java.util.RandomAccess,
java.io.Serializable

```

public class PropertyBags
extends ReportSDKVector
implements IClone

```

This class defines a collection of property bags.

See Also:

Serialized Form

Constructor Summary

PropertyBags ()

Constructs a new property bag collection.

PropertyBags(PropertyBags src)

Constructs a new PropertyBags object with the same contents as the specified PropertyBags object.

Method Summary

void	add (int index, java.lang.Object element) Inserts the specified element at the specified position in this collection.
boolean	add (java.lang.Object o) Appends the specified element to the end of this collection.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
PropertyBag	getPropertyBag (int index) Returns the property bag at the specified index.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs,

IDE Extensions

	<code>java.util.Map objState)</code> For internal use only.
<code>void</code>	save (XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
<code>void</code>	save (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
<code>void</code>	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
<code>void</code>	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.

Methods inherited from class `com.crystaldecisions.sdk.occa.report.lib.ReportSDKVector`

`addElement`, `clone`, `copyTo`, `elementAt`, `findIndexOf`, `hasContent`, `insertElementAt`, `removeAllElements`

Methods inherited from class `java.util.ArrayList`

`addAll`, `addAll`, `clear`, `clone`, `contains`, `ensureCapacity`, `get`, `indexOf`, `isEmpty`, `lastIndexOf`, `remove`, `set`, `size`, `toArray`, `toArray`, `trimToSize`

Methods inherited from class `java.util.AbstractList`

`equals`, `hashCode`, `iterator`, `listIterator`, `listIterator`, `subList`

Methods inherited from class `java.util.AbstractCollection`

`containsAll`, `remove`, `removeAll`, `retainAll`, `toString`

Methods inherited from class `java.lang.Object`

`getClass`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Methods inherited from interface `com.crystaldecisions.sdk.occa.report.lib.IClone`

`clone`, `copyTo`, `hasContent`

Methods inherited from interface `java.util.List`

`containsAll`, `equals`, `hashCode`, `iterator`, `listIterator`, `listIterator`, `remove`, `removeAll`, `retainAll`, `subList`

Constructor Detail

PropertyBags

```
public PropertyBags(PropertyBags src)
```

Constructs a new PropertyBags object with the same contents as the specified PropertyBags object. This is equivalent to creating a new copy.

Parameters:

src – The PropertyBags object to be copied.

PropertyBags

```
public PropertyBags()
```

Constructs a new property bag collection.

Method Detail

createMember

```
public java.lang.Object createMember(java.lang.String eleName,  
                                     org.xml.sax.Attributes attrs,  
                                     XMLSerializationContext ctxt,  
                                     java.util.Map objState,  
                                     boolean[] bLoaded)
```

For internal use only.

endElement

```
public void endElement(java.lang.String eleName,  
                       java.util.Map objState)
```

For internal use only.

readElement

```
public void readElement(java.lang.String eleName,  
                        java.lang.String sVal,  
                        org.xml.sax.Attributes attrs,  
                        java.util.Map objState)
```

For internal use only.

save

```
public void save(XMLWriter writer,
                 java.lang.String sTag,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

save

```
public void save(XMLWriter writer,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

saveContents

```
public void saveContents(XMLWriter writer,
                         XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

startElement

```
public void startElement(java.lang.String eleName,
                         java.util.Map objState,
                         org.xml.sax.Attributes attrs)
```

For internal use only.

getPropertyBag

```
public PropertyBag getPropertyBag(int index)
```

Returns the property bag at the specified index.

Parameters:

index – The index of the desired property bag.

Returns:

The PropertyBag object at the specified index.

add

```
public boolean add(java.lang.Object o)
```

Appends the specified element to the end of this collection.

Note: Only instances of the IValue interface can be added to a Values collection.

Specified by:

add in interface java.util.List

Parameters:

o – element to be added to the collection.

Returns:

true if this collection changed as a result of the call.

Throws:

java.lang.ClassCastException – if the class of the specified element prevents it from being added to this collection.

java.lang.NullPointerException – if the specified element is null and this collection does not support null elements.

add

```
public void add(int index,  
                java.lang.Object element)
```

Inserts the specified element at the specified position in this collection. Shifts the element currently at that position (if any) and any subsequent elements to the right, adding one to their indices.

Specified by:

add in interface java.util.List

Parameters:

index – The index at which the specified element will be inserted.

element – The element to be inserted.

Throws:

java.lang.IndexOutOfBoundsException – if index is out of range (index < 0 || index > size()).

java.lang.ClassCastException – class of the specified element prevents it from being added to this collection.

java.lang.NullPointerException – if the specified element is null and this collection does not support null elements.

com.crystaldecisions.sdk.occa.report.lib

Class ReportSDKVector

```
java.lang.Object  
├─ java.util.AbstractCollection  
│   └─ java.util.AbstractList  
│       └─ java.util.ArrayList
```

└─ `com.crystaldecisions.sdk.occa.report.lib.ReportSDKVector`

All Implemented Interfaces:

`java.lang.Cloneable`, `java.util.Collection`, `IClone`, `java.util.List`, `java.util.RandomAccess`,
`java.io.Serializable`

Direct Known Subclasses:

`ConnectionInfos`, `PropertyBags`, `Strings`, `Values`

public class **ReportSDKVector**

extends `java.util.ArrayList`

implements `IClone`

This class defines a vector data structure with support for specific Report Application Server SDK functionality.

See Also:

`Serialized Form`

Constructor Summary

ReportSDKVector()

Constructs an empty vector.

ReportSDKVector(`ReportSDKVector src`)

Constructs a new `ReportSDKVector` object with the same contents as the specified `ReportSDKVector` object.

Method Summary

<code>void</code>	addElement (<code>java.lang.Object obj</code>) Appends the specified object to the end of this collection.
<code>java.lang.Object</code>	clone (<code>boolean deepClone</code>) Returns the new object that has been cloned.
<code>void</code>	copyTo (<code>java.lang.Object dest</code> , <code>boolean deepCopy</code>) Copies the object.
<code>java.lang.Object</code>	elementAt (<code>int index</code>) Returns the element at the specified index.
<code>int</code>	findIndexOf (<code>java.lang.Object object</code>) Returns the index of the first occurrence of the specified <code>Object</code> in this collection.
<code>boolean</code>	hasContent (<code>java.lang.Object obj</code>) Returns <code>true</code> if this object contains the same elements as the passed in object.
<code>void</code>	

IDE Extensions

	insertElementAt (java.lang.Object obj, int index) Inserts the supplied Object at the specified index.
void	removeAllElements () Removes all elements from this collection.

Methods inherited from class java.util.ArrayList

add, add, addAll, addAll, clear, clone, contains, ensureCapacity, get, indexOf, isEmpty, lastIndexOf, remove, set, size, toArray, toArray, trimToSize

Methods inherited from class java.util.AbstractList

equals, hashCode, iterator, listIterator, listIterator, subList

Methods inherited from class java.util.AbstractCollection

containsAll, remove, removeAll, retainAll, toString

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.util.List

containsAll, equals, hashCode, iterator, listIterator, listIterator, remove, removeAll, retainAll, subList

Constructor Detail

ReportSDKVector

```
public ReportSDKVector()
```

Constructs an empty vector.

ReportSDKVector

```
public ReportSDKVector(ReportSDKVector src)
```

Constructs a new ReportSDKVector object with the same contents as the specified ReportSDKVector object. This is equivalent to creating a new copy.

Parameters:

src – The ReportSDKVector object to be copied.

Method Detail

clone

```
public java.lang.Object clone(boolean deepClone)
```

Description copied from interface: IClone

Returns the new object that has been cloned.

Specified by:

clone in interface IClone

Parameters:

deepClone – true to use deep clone, false to use shallow.

Returns:

The new object that has been cloned.

copyTo

```
public void copyTo(java.lang.Object dest,  
                  boolean deepCopy)
```

Description copied from interface: IClone

Copies the object.

Specified by:

copyTo in interface IClone

Parameters:

dest – The destination object to copy to.

deepCopy – true to use deep copy, false to use shallow.

addElement

```
public void addElement(java.lang.Object obj)
```

Appends the specified object to the end of this collection.

Parameters:

obj – The Object to be added.

Throws:

java.lang.ClassCastException – This is thrown if the class of the specified element prevents it from being added to this collection. In other words, if there is a class mismatch.

java.lang.NullPointerException – This is thrown if the specified element is null and this collection does not support null elements.

elementAt

```
public java.lang.Object elementAt(int index)
```

Returns the element at the specified index.

Parameters:

index – The index of the element to be retrieved.

Returns:

The Object at the specified index.

findIndexOf

```
public int findIndexOf(java.lang.Object object)
```

Returns the index of the first occurrence of the specified Object in this collection. Both the type and the content of the Objects must match exactly.

Parameters:

object – The Object to search for.

Returns:

An int that specifies the index of the first occurrence of the specified Object or -1 if it is not found.

hasContent

```
public boolean hasContent(java.lang.Object obj)
```

Description copied from interface: IClone

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Parameters:

obj – The object to check for content.

Returns:

true if this object contains the same elements as the passed in object, otherwise false.

insertElementAt

```
public void insertElementAt(java.lang.Object obj,
                           int index)
```

Inserts the supplied Object at the specified index. All elements at index or greater will be shifted upward by 1.

Parameters:

obj – The Object to be inserted.

index – The index at which to insert the Object.

Throws:

java.lang.ClassCastException – This is thrown if the class of the specified element prevents it from being added to this collection. In other words, if there is a class mismatch.

java.lang.NullPointerException – This is thrown if the specified element is null and this collection does not support null elements.

removeAllElements

```
public void removeAllElements()
```

Removes all elements from this collection.

com.crystaldecisions.sdk.occa.report.lib Class Strings

```
java.lang.Object
├ java.util.AbstractCollection
│   └ java.util.AbstractList
│       └ java.util.ArrayList
│           └ com.crystaldecisions.sdk.occa.report.lib.ReportSDKVector
│               └ com.crystaldecisions.sdk.occa.report.lib.Strings
```

All Implemented Interfaces:

java.lang.Cloneable, java.util.Collection, IClone, IStrings, java.util.List, java.util.RandomAccess, java.io.Serializable

```
public class Strings
extends ReportSDKVector
implements IStrings
```

This class defines a collection of String objects.

See Also:

Serialized Form

Constructor Summary

Strings ()	Constructs an empty Strings object.
Strings (Strings src)	Constructs a new String collection with the same contents as the specified Strings object.

Method Summary

void	add (int index, java.lang.Object element) Inserts the specified element at the specified position in this collection.
boolean	add (java.lang.Object o) Appends the specified element to the end of this collection.

IDE Extensions

void	copyTo (java.lang.Object dest, boolean deepCopy) Copies the object.
java.lang.Object	createMember (java.lang.String eleName, org.xml.sax.Attributes attrs, XMLSerializationContext ctxt, java.util.Map objState, boolean[] bLoaded) For internal use only.
void	endElement (java.lang.String eleName, java.util.Map objState) For internal use only.
boolean	equalsIgnoreCase (IStrings srcStrings) Returns true if the specified IStrings object is equal to this object, regardless of case.
java.lang.String	getString (int index) Returns the String at the specified index.
boolean	hasContent (java.lang.Object obj) Returns true if this object contains the same elements as the passed in object.
void	readElement (java.lang.String eleName, java.lang.String sVal, org.xml.sax.Attributes attrs, java.util.Map objState) For internal use only.
void	save (com.crystaldecisions.xml.serialization.XMLWriter writer, com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt) For internal use only.
void	save (XMLWriter writer, java.lang.String sTag, XMLSerializationContext ctxt) For internal use only.
void	saveContents (XMLWriter writer, XMLSerializationContext ctxt) For internal use only.
void	startElement (java.lang.String eleName, java.util.Map objState, org.xml.sax.Attributes attrs) For internal use only.
java.lang.String[]	toStringArray () Returns the contents of this Strings collection as an array of String objects.

Methods inherited from class com.crystaldecisions.sdk.occa.report.lib.ReportSDKVector

addElement, clone, elementAt, findIndexOf, insertElementAt, removeAllElements

Methods inherited from class java.util.ArrayList

addAll, addAll, clear, clone, contains, ensureCapacity, get, indexOf, isEmpty, lastIndexOf, remove, set, size, toArray, toArray, trimToSize

Methods inherited from class java.util.AbstractList

equals, hashCode, iterator, listIterator, listIterator, subList

Methods inherited from class java.util.AbstractCollection

containsAll, remove, removeAll, retainAll, toString

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.util.List

addAll, addAll, clear, contains, containsAll, equals, get, hashCode, indexOf, isEmpty, iterator, lastIndexOf, listIterator, listIterator, remove, remove, removeAll, retainAll, set, size, subList, toArray, toArray

Constructor Detail**Strings**

```
public Strings(Strings src)
```

Constructs a new String collection with the same contents as the specified Strings object. This is equivalent to creating a new copy.

Parameters:

src – The Strings collection to be copied.

Strings

```
public Strings()
```

Constructs an empty Strings object.

Method Detail**copyTo**

```
public void copyTo(java.lang.Object dest,  
                  boolean deepCopy)
```

Copies the object.

Specified by:

copyTo in interface IClone

Overrides:

copyTo in class ReportSDKVector

Parameters:

`dest` – The destination object to copy to.

`deepCopy` – `true` to use deep copy, `false` to use shallow.

endElement

```
public void endElement(java.lang.String eleName,
                       java.util.Map objState)
```

For internal use only.

equalsIgnoreCase

```
public boolean equalsIgnoreCase(IStrings srcStrings)
```

Description copied from interface: IStrings

Returns `true` if the specified `IStrings` object is equal to this object, regardless of case. In other words, this method performs a case-insensitive equality test.

Specified by:

`equalsIgnoreCase` in interface `IStrings`

Parameters:

`srcStrings` – The `IStrings` object to be compared to.

Returns:

`true` if the specified `IStrings` object is equal to this object, and `false` otherwise.

getString

```
public java.lang.String getString(int index)
```

Description copied from interface: IStrings

Returns the `String` at the specified index.

Specified by:

`getString` in interface `IStrings`

Parameters:

`index` – The index of the desired `String`.

Returns:

The `String` at the specified index.

readElement

```
public void readElement(java.lang.String eleName,
                        java.lang.String sVal,
                        org.xml.sax.Attributes attrs,
                        java.util.Map objState)
```

For internal use only.

saveContents

```
public void saveContents(XMLWriter writer,  
                        XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

createMember

```
public java.lang.Object createMember(java.lang.String eleName,  
                                       org.xml.sax.Attributes attrs,  
                                       XMLSerializationContext ctxt,  
                                       java.util.Map objState,  
                                       boolean[] bLoaded)
```

For internal use only.

hasContent

```
public boolean hasContent(java.lang.Object obj)
```

Returns true if this object contains the same elements as the passed in object.

Specified by:

hasContent in interface IClone

Overrides:

hasContent in class ReportSDKVector

Parameters:

obj – The object to check for content.

Returns:

true if this object contains the same elements as the passed in object, otherwise false.

save

```
public void save(com.crystaldecisions.xml.serialization.XMLWriter writer,  
                com.crystaldecisions.xml.serialization.XMLSerializationContext ctxt)  
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

save

```
public void save(XMLWriter writer,
                 java.lang.String sTag,
                 XMLSerializationContext ctxt)
    throws java.io.IOException
```

For internal use only.

Throws:

java.io.IOException

startElement

```
public void startElement(java.lang.String eleName,
                         java.util.Map objState,
                         org.xml.sax.Attributes attrs)
```

For internal use only.

toStringArray

```
public java.lang.String[] toStringArray()
```

Returns the contents of this Strings collection as an array of String objects.

Returns:

An array of String objects containing the contents of this collection.

add

```
public boolean add(java.lang.Object o)
```

Appends the specified element to the end of this collection.

Specified by:

add in interface java.util.List

Parameters:

o – element to be added into the collection.

Returns:

true if this collection changed as a result of the call.

Throws:

java.lang.ClassCastException – class of the specified element prevents it from being added to this collection.

java.lang.NullPointerException – if the specified element is null and this collection does not support null elements.

add

```
public void add(int index,  
                java.lang.Object element)
```

Inserts the specified element at the specified position in this collection. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Specified by:

add in interface `java.util.List`

Parameters:

index – index at which the specified element is to be inserted.

element – element to be inserted.

Throws:

`java.lang.IndexOutOfBoundsException` – if index is out of range (`index < 0` || `index > size()`).

`java.lang.ClassCastException` – class of the specified element prevents it from being added to this collection.

`java.lang.NullPointerException` – if the specified element is null and this collection does not support null elements.

Viewer Java SDK

This documentation covers the Viewer Java SDK.

See:

Description

Packages	
com.crystaldecisions.report.htmlrender	This package contains classes for the rendering events in the Java Viewer.
com.crystaldecisions.report.web	This package defines contains the class <code>ServerControl</code> , which is an abstract class.
com.crystaldecisions.report.web.viewer	This package contains the key functionality for the display and behavior of the viewers, including report source information, event listeners, and event arguments.

This documentation covers the Viewer Java SDK. The Viewer Java SDK contains lightweight components intended to provide web developers with versatile tools for displaying reports inside JSP pages using HTML or DHTML. These components offer properties that you can use to control actions such as displaying buttons or displaying a group tree, page, or toolbar. They also support event handling, exporting, printing, displaying multiple viewers in the same page, and both automatic and developer-specified prompting for database or parameter information.

These lightweight components include the web reporting Java viewers. All web reporting Java viewers inherit functionality from the `ReportServerControl` Object. The viewers do not require frames and can co-exist with other HTML that you wish to display on the same page as the report. Each viewer provides different capabilities.

Viewer	Corresponding Object	Description
Used internally by the Java viewers.	<code>ReportServerControl</code>	Provides the ability to log on to the database or Crystal Enterprise, pass parameters or the selection formula, and define the report source.
Java Report Export Control	<code>ReportExportControl</code>	Provides exporting functionality including the ability to preview an exported report or save an exported report to disk.
Java Report Page Viewer	<code>CrystalReportViewer</code>	Provides basic web reporting viewer capabilities.

To create or modify a particular report you need the Report Application Server (RAS) and the RAS SDK. For details, see the *Crystal Reports for BEA WebLogic Workshop Developer's Guide*.

Note: It is recommended that you perform your own garbage collection in your web applications when using the Viewer Java SDK. This involves calling the appropriate dispose methods and ensuring that you use the `CrystalImageCleaner` class to remove temporary image files used by the viewer.

Package com.crystaldecisions.report.htmlrender

This package contains classes for the rendering events in the Java Viewer.

See:

Description

Interface Summary	
AfterRenderContentEventListener	This class provides a listener for a <code>AfterRenderContentEvent</code> .
AfterRenderEventListener	This class provides a listener for a <code>AfterRenderEvent</code> .
AfterRenderObjectEventListener	This class provides a listener for a <code>AfterRenderObjectEvent</code> .
BeforeRenderContentEventListener	This class provides a listener for <code>BeforeRenderContentEvent</code> .
BeforeRenderEventListener	This class provides a listener for <code>BeforeRenderEvent</code> .
BeforeRenderObjectEventListener	This class provides a listener for a <code>BeforeRenderObjectEvent</code> .
IAfterRenderViewItemEventListener	This class provides a listener for a <code>AfterRenderViewItemEvent</code> .
IBeforeRenderViewItemEventListener	This class provides a listener for a <code>BeforeRenderViewItemEvent</code> .
OnRenderScriptEventListener	This class provides a listener for a <code>OnRenderScriptEvent</code> .
OnRenderStyleEventListener	This class provides a listener for a <code>OnRenderStyleEvent</code> .

Class Summary	
AfterRenderContentEvent	This class provides arguments for the <code>AfterRenderContentEventListener</code> .
AfterRenderEvent	This class provides arguments for the <code>AfterRenderEventListener</code> .
AfterRenderObjectEvent	This class provides arguments for the <code>AfterRenderObjectEventListener</code> .
AfterRenderViewItemEvent	This class provides arguments for the <code>IAfterRenderViewItemEventListener</code> .
BeforeRenderContentEvent	This class provides arguments for the <code>BeforeRenderContentEventListener</code> .
BeforeRenderEvent	This class provides arguments for the <code>BeforeRenderEventListener</code> .
BeforeRenderObjectEvent	This class provides arguments for the <code>BeforeRenderObjectEventListener</code> .
BeforeRenderViewItemEvent	

	This class provides arguments for the <code>IBeforeRenderViewItemEventListener</code> .
OnRenderScriptEvent	This class provides arguments for the <code>OnRenderScriptEventListener</code> .
OnRenderStyleEvent	This class provides arguments for the <code>OnRenderStyleEventListener</code> .

Package `com.crystaldecisions.report.htmlrender` Description

This package contains classes for the rendering events in the Java Viewer. To improve performance, you can write a cache manager that allows the client document to share jobs and cache HTML pages. The `BeforeRenderEvent` class, `BeforeRenderContentEvent` class, and `BeforeRenderObjectEvent` class allow the viewer to accept user data that comes from the cache manager; the `AfterRenderEvent` class, `AfterRenderContentEvent` class, and `AfterRenderObjectEvent` class return HTML data that the user can put into the cache manager. The `OnRenderScriptEvent` class allows the viewer to accept user data when the HTML scripts generation starts, while the `OnRenderStyleEvent` class allows the viewer to accept user data when the HTML style class generation starts.

Note: It is recommended that you perform your own garbage collection in your web applications when using the Viewer Java SDK. This involves calling the appropriate dispose methods and ensuring that you use the `CrystallImageCleaner` class to remove temporary image files used by the viewer.

`com.crystaldecisions.report.htmlrender` Class `AfterRenderContentEvent`

```
java.lang.Object
├─ RenderEventObjectBase
│   └─ com.crystaldecisions.report.htmlrender.AfterRenderContentEvent
```

```
public class AfterRenderContentEvent
    extends RenderEventObjectBase
```

This class provides arguments for the `AfterRenderContentEventListener`. The event occurs when the report content rendering process is finished.

See Also:

Serialized Form

Constructor Summary

AfterRenderContentEvent (<code>java.lang.Object source</code>)

Method Summary

java.lang.String	getAddAfterContent () Returns the text that will be appended to the end of the report content HTML.
void	setAddAfterContent (java.lang.String value) Sets the text that will be appended to the end of the report content HTML.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AfterRenderContentEvent

```
public AfterRenderContentEvent(java.lang.Object source)
```

Method Detail

getAddAfterContent

```
public java.lang.String getAddAfterContent()
```

Returns the text that will be appended to the end of the report content HTML.

Returns:

The text that will be appended to the end of the report content HTML as a String.

See Also:

setAddAfterContent

setAddAfterContent

```
public void setAddAfterContent(java.lang.String value)
```

Sets the text that will be appended to the end of the report content HTML.

Parameters:

value – A String that specifies the text that will be appended to the end of the report content HTML.

See Also:

getAddAfterContent

com.crystaldecisions.report.htmlrender Interface AfterRenderContentEventListener

All Superinterfaces:

java.util.EventListener

*public interface **AfterRenderContentEventListener**
extends java.util.EventListener*

This class provides a listener for a AfterRenderContentEvent.

Method Summary

void	afterRenderContent (AfterRenderContentEvent e) The listener for a AfterRenderContentEvent.
------	--

Method Detail

afterRenderContent

`public void afterRenderContent(AfterRenderContentEvent e)`

The listener for a AfterRenderContentEvent. The event occurs when the report content rendering process is finished.

Parameters:

e – An AfterRenderContentEvent object.

See Also:

AfterRenderContentEvent

com.crystaldecisions.report.htmlrender Class AfterRenderEvent

```
java.lang.Object
├─ RenderEventObjectBase
└─ com.crystaldecisions.report.htmlrender.AfterRenderEvent
```

*public class **AfterRenderEvent**
extends RenderEventObjectBase*

This class provides arguments for the AfterRenderEventListener. The event occurs when the report rendering process is finished.

See Also:

Serialized Form

Constructor Summary

AfterRenderEvent (java.lang.Object source)

Method Summary

java.lang.String	getAddAtEnd () Returns the text that will be appended to the end of the HTML.
void	setAddAtEnd (java.lang.String value) Sets the text that will be appended to the end of the HTML.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AfterRenderEvent

```
public AfterRenderEvent(java.lang.Object source)
```

Method Detail

getAddAtEnd

```
public java.lang.String getAddAtEnd()
```

Returns the text that will be appended to the end of the HTML.

Returns:

The text that will be appended to the end of the HTML as a String.

See Also:

setAddAtEnd

setAddAtEnd

```
public void setAddAtEnd(java.lang.String value)
```

Sets the text that will be appended to the end of the HTML.

Parameters:

value – A String that specifies the text that will be appended to the end of the HTML.

See Also:

getAddAtEnd

com.crystaldecisions.report.htmlrender Interface AfterRenderEventListener

All Superinterfaces:

java.util.EventListener

```
public interface AfterRenderEventListener  
extends java.util.EventListener
```

This class provides a listener for a AfterRenderEvent.

Method Summary

void	afterRender (AfterRenderEvent e) The listener for AfterRenderEvents.
------	--

Method Detail

afterRender

```
public void afterRender(AfterRenderEvent e)
```

The listener for AfterRenderEvents. The event occurs when the report rendering process is finished.

Parameters:

e – An AfterRenderEvent object.

See Also:

AfterRenderEvent

com.crystaldecisions.report.htmlrender Class AfterRenderObjectEvent

```
java.lang.Object
├─ RenderEventObjectBase
│   └─ com.crystaldecisions.report.htmlrender.AfterRenderObjectEvent
```

*public class **AfterRenderObjectEvent**
extends *RenderEventObjectBase**

This class provides arguments for the AfterRenderObjectEventListener. The event occurs when the report object rendering process is finished.

See Also:

Serialized Form

Constructor Summary

AfterRenderObjectEvent (java.lang.Object source)
--

Method Summary

java.lang.Object	getObject () Returns the cached report object.
java.lang.Object	getObjectRender () Returns the HTML renderer object, which generates the resulting HTML.
void	setObject (java.lang.Object value) Sets the cached report object.
void	setObjectRender (java.lang.Object value) Sets the HTML renderer object, which generates the resulting HTML.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AfterRenderObjectEvent

```
public AfterRenderObjectEvent(java.lang.Object source)
```

Method Detail

getObject

```
public java.lang.Object getObject()
```

Returns the cached report object. This can be a `com.crystaldecisions.sdk.occa.report.definition.FieldObject`, `com.crystaldecisions.sdk.occa.report.definition.TextObject`, `com.crystaldecisions.sdk.occa.report.definition.ChartObject`, and so on.

Returns:

The cached report object as an `Object`.

See Also:

`setObject`

getObjectRender

```
public java.lang.Object getObjectRender()
```

Returns the HTML renderer object, which generates the resulting HTML. This is the event's source object.

Returns:

The HTML renderer as an `Object`.

See Also:

`setObjectRender`

setObject

```
public void setObject(java.lang.Object value)
```

Sets the cached report object. This can be a `com.crystaldecisions.sdk.occa.report.definition.FieldObject`, `com.crystaldecisions.sdk.occa.report.definition.TextObject`, `com.crystaldecisions.sdk.occa.report.definition.ChartObject`, and so on.

Parameters:

`value` – An `Object` that represents the cached report object.

See Also:

`getObject`

setObjectRender

```
public void setObjectRender(java.lang.Object value)
```

Sets the HTML renderer object, which generates the resulting HTML. This is the event's source object.

Parameters:

value – An Object that represents the HTML renderer.

See Also:

getObjectRender

com.crystaldecisions.report.htmlrender Interface AfterRenderObjectEventListener

All Superinterfaces:

java.util.EventListener

```
public interface AfterRenderObjectEventListener
extends java.util.EventListener
```

This class provides a listener for a AfterRenderObjectEvent.

Method Summary

void	afterRenderObject (AfterRenderObjectEvent e) The listener for AfterRenderObjectEvents.
------	--

Method Detail

afterRenderObject

```
public void afterRenderObject(AfterRenderObjectEvent e)
```

The listener for AfterRenderObjectEvents. The event occurs when the report object rendering process is finished.

Parameters:

e – An AfterRenderObjectEvent object.

See Also:

AfterRenderObjectEvent

com.crystaldecisions.report.htmlrender Class AfterRenderViewItemEvent

```
java.lang.Object
├─ RenderEventObjectBase
│   └─ com.crystaldecisions.report.htmlrender.AfterRenderViewItemEvent
```

*public class **AfterRenderViewItemEvent**
extends *RenderEventObjectBase**

This class provides arguments for the `IAfterRenderViewItemEventListener`. The event occurs after rendering a `ViewItem` (Toolbar, GroupTree, or Page).

See Also:

Serialized Form

Constructor Summary

AfterRenderViewItemEvent (java.lang.Object source)
--

AfterRenderViewItemEvent (java.lang.Object source, CrystalHtmlTextWriter writer)
--

Method Summary

java.lang.String	getType () Returns the type of view item being rendered.
void	setType (java.lang.String newM_Type) Sets the type of view item being rendered.

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

AfterRenderViewItemEvent

```
public AfterRenderViewItemEvent(java.lang.Object source)
```

AfterRenderViewItemEvent

```
public AfterRenderViewItemEvent(java.lang.Object source,
                                CrystalHtmlTextWriter writer)
```

Method Detail

getType

```
public java.lang.String getType()
```

Returns the type of view item being rendered.

Returns:

The type of view item being rendered as a String.

See Also:

setType

setType

```
public void setType(java.lang.String newM_Type)
```

Sets the type of view item being rendered.

Note: The type of view item must be either Toolbar, GroupTree, or Page.

Parameters:

newM_Type – A String that specifies the type of view item being rendered.

See Also:

getType

com.crystaldecisions.report.htmlrender Class BeforeRenderContentEvent

```
java.lang.Object
├─ RenderEventObjectBase
│   └─ com.crystaldecisions.report.htmlrender.BeforeRenderContentEvent
```

```
public class BeforeRenderContentEvent
    extends RenderEventObjectBase
```

This class provides arguments for the BeforeRenderContentEventListener. The event occurs when the report content rendering process starts.

See Also:

Serialized Form

Constructor Summary

BeforeRenderContentEvent (java.lang.Object source)

Method Summary

java.lang.String	getAddBeforeContent () Returns the text that will be written immediately following the <BODY> tag.
boolean	getNeedRenderBodyTag () Returns whether the <BODY> tag is to be rendered.
void	setAddBeforeContent (java.lang.String value) Sets the text that will be written immediately following the <BODY> tag.
void	setNeedRenderBodyTag (boolean value) Sets whether the <BODY> tag will be rendered.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BeforeRenderContentEvent

```
public BeforeRenderContentEvent(java.lang.Object source)
```

Method Detail

getAddBeforeContent

```
public java.lang.String getAddBeforeContent()
```

Returns the text that will be written immediately following the <BODY> tag.

Returns:

The text that will be written immediately following the <BODY> tag as a String.

See Also:

`setAddBeforeContent`

getNeedRenderBodyTag

```
public boolean getNeedRenderBodyTag()
```

Returns whether the <BODY> tag is to be rendered.

Returns:

true if the <BODY> tag is to be rendered and false otherwise.

See Also:

`setNeedRenderBodyTag`

setAddBeforeContent

```
public void setAddBeforeContent(java.lang.String value)
```

Sets the text that will be written immediately following the <BODY> tag.

Parameters:

value – A String that specifies the text that will be written immediately following the <BODY> tag.

See Also:

`getAddBeforeContent`

setNeedRenderBodyTag

```
public void setNeedRenderBodyTag(boolean value)
```

Sets whether the <BODY> tag will be rendered.

Parameters:

value – true to render the <BODY> tag.

See Also:

`getNeedRenderBodyTag`

com.crystaldecisions.report.htmlrender Interface BeforeRenderContentEventListener

All Superinterfaces:

`java.util.EventListener`

```
public interface BeforeRenderContentEventListener  
extends java.util.EventListener
```

This class provides a listener for BeforeRenderContentEvent.

Method Summary

void	beforeRenderContent (BeforeRenderContentEvent e) The listener for BeforeRenderContentEvents.
------	--

Method Detail

beforeRenderContent

```
public void beforeRenderContent(BeforeRenderContentEvent e)
```

The listener for BeforeRenderContentEvents. The event occurs when the report content rendering process starts.

Parameters:

e – A BeforeRenderContentEvent object.

See Also:

BeforeRenderContentEvent

com.crystaldecisions.report.htmlrender

Class BeforeRenderEvent

```
java.lang.Object
├─ RenderEventObjectBase
├─ com.crystaldecisions.report.htmlrender.BeforeRenderEvent
```

```
public class BeforeRenderEvent
    extends RenderEventObjectBase
```

This class provides arguments for the BeforeRenderEventListener. The event occurs when the report rendering process starts.

See Also:

Serialized Form

Constructor Summary

```
BeforeRenderEvent(java.lang.Object source)
```

Method Summary

IDE Extensions

<code>java.lang.String</code>	<code>getAddAtBeginning()</code> Returns the text that will be written after the <HTML> tag but before the <BODY> tag, if there is one.
<code>boolean</code>	<code>getNeedRenderHTMLTag()</code> Returns whether to render the <HTML> tag.
<code>void</code>	<code>setAddAtBeginning(java.lang.String value)</code> Sets the text that will be written after the <HTML> tag but before the <BODY> tag, if there is one.
<code>void</code>	<code>setNeedRenderHTMLTag(boolean value)</code> Sets whether the <HTML> tag will be rendered.

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Detail

BeforeRenderEvent

```
public BeforeRenderEvent(java.lang.Object source)
```

Method Detail

getAddAtBeginning

```
public java.lang.String getAddAtBeginning()
```

Returns the text that will be written after the <HTML> tag but before the <BODY> tag, if there is one.

Returns:

The text that will be written as a `String`.

See Also:

`setAddAtBeginning`

getNeedRenderHTMLTag

```
public boolean getNeedRenderHTMLTag()
```

Returns whether to render the <HTML> tag.

Returns:

`true` if the <HTML> tag is to be rendered and `false` otherwise.

See Also:

setAddAtBeginning

```
public void setAddAtBeginning(java.lang.String value)
```

Sets the text that will be written after the <HTML> tag but before the <BODY> tag, if there is one.

Parameters:

value – A String that specifies the text that will be written.

See Also:

getAddAtBeginning

setNeedRenderHTMLTag

```
public void setNeedRenderHTMLTag(boolean value)
```

Sets whether the <HTML> tag will be rendered.

Parameters:

value – true to render the <HTML> tag.

See Also:

getNeedRenderHTMLTag

com.crystaldecisions.report.htmlrender Interface BeforeRenderEventListener

All Superinterfaces:

java.util.EventListener

```
public interface BeforeRenderEventListener  
extends java.util.EventListener
```

This class provides a listener for BeforeRenderEvent.

Method Summary

void	beforeRender (BeforeRenderEvent e) The listener for BeforeRenderEvents.
------	---

Method Detail

beforeRender

```
public void beforeRender(BeforeRenderEvent e)
```

The listener for BeforeRenderEvents. The event occurs when the report rendering process starts.

Parameters:

e – A BeforeRenderEvent object.

See Also:

BeforeRenderEvent

com.crystaldecisions.report.htmlrender Class BeforeRenderObjectEvent

```
java.lang.Object
├─ RenderEventObjectBase
└─ com.crystaldecisions.report.htmlrender.BeforeRenderObjectEvent
```

```
public class BeforeRenderObjectEvent
    extends RenderEventObjectBase
```

This class provides arguments for the BeforeRenderObjectEventListener. The event occurs when the report object rendering process starts.

See Also:

Serialized Form

Constructor Summary

BeforeRenderObjectEvent (java.lang.Object source)

Method Summary

java.lang.String	getAttribute () For internal use only.
java.lang.Object	getObject () Returns the report object to be cached.
java.lang.String	getOnBlurHandler () Returns the JavaScript code that is executed when the contents of an object are made to appear out of focus.
java.lang.String	getOnClickHandler () Returns the JavaScript code that is executed when an object is clicked.

IDE Extensions

<code>java.lang.String</code>	<code>getContextMenuHandler()</code> Returns the JavaScript code that is executed when the context menu is displayed.
<code>java.lang.String</code>	<code>getOnDBClickHandler()</code> Returns the JavaScript code that is executed when an object is double-clicked.
<code>java.lang.String</code>	<code>getOnDragEndHandler()</code> Returns the JavaScript code that is executed at the end of a drag operation.
<code>java.lang.String</code>	<code>getOnDragEnterHandler()</code> Returns the JavaScript code that is executed when an object is dragged into the controls bounds.
<code>java.lang.String</code>	<code>getOnDragHandler()</code> Returns the JavaScript code that is executed during a drag operation.
<code>java.lang.String</code>	<code>getOnDragLeaveHandler()</code> Returns the JavaScript code that is executed when an object is dragged out of the controls bounds.
<code>java.lang.String</code>	<code>getOnDragOverHandler()</code> Returns the JavaScript code that is executed when a control is being dragged over a target object.
<code>java.lang.String</code>	<code>getOnDragStartHandler()</code> Returns the JavaScript code that is executed at the start of a drag operation.
<code>java.lang.String</code>	<code>getOnDropHandler()</code> Returns the JavaScript code that is executed when the drag and drop operation is complete.
<code>java.lang.String</code>	<code>getOnFocusHandler()</code> Returns the JavaScript code that is executed when the object receives focus.
<code>java.lang.String</code>	<code>getOnFocusInHandler()</code> Returns the JavaScript code that is executed just prior to the object receiving focus.
<code>java.lang.String</code>	<code>getOnFocusOutHandler()</code> Returns the JavaScript code that is executed immediately after the object loses focus.
<code>java.lang.String</code>	<code>getOnKeyDownHandler()</code> Returns the JavaScript code that is executed when an alphanumeric key is pressed.
<code>java.lang.String</code>	<code>getOnKeyPressHandler()</code> Returns the JavaScript code that is executed when a key is pressed.
<code>java.lang.String</code>	<code>getOnKeyUpHandler()</code> Returns the JavaScript code that is executed when a key is

IDE Extensions

	released.
<code>java.lang.String</code>	getOnMouseDownHandler () Returns the JavaScript code that is executed when an object is clicked with either mouse button.
<code>java.lang.String</code>	getOnMouseEnterHandler () Returns the JavaScript code that is executed when the mouse pointer is moved into the boundaries of an object.
<code>java.lang.String</code>	getOnMouseLeaveHandler () Returns the JavaScript code that is executed when the mouse pointer is moved outside of the boundaries of an object.
<code>java.lang.String</code>	getOnMouseMoveHandler () Returns the JavaScript code that is executed when the mouse pointer is moved over an object.
<code>java.lang.String</code>	getOnMouseOutHandler () Returns the JavaScript code that is executed when the mouse pointer is moved outside of the boundaries of an object.
<code>java.lang.String</code>	getOnMouseOverHandler () Returns the JavaScript code that is executed when the mouse pointer is moved into an object.
<code>java.lang.String</code>	getOnMouseUpHandler () Returns the JavaScript code that is executed when either mouse button is released over an object.
<code>java.lang.String</code>	getOnSelectHandler () Returns the JavaScript code that is executed when the current selection changes.
<code>java.lang.String</code>	getOnSelectionChangeHandler () Returns the JavaScript code that is executed when the selection state of the document changes.
<code>java.lang.String</code>	getOnSelectStartHandler () Returns the JavaScript code that is executed when the object is being selected.
<code>java.lang.String</code>	getOnStopHandler () Returns the JavaScript code that is executed when the user clicks the stop button or leaves the web page.
<code>void</code>	setAttribute (java.lang.String value) For internal use only.
<code>void</code>	setObject (java.lang.Object value) Sets the report object to be cached.
<code>void</code>	setOnBlurHandler (java.lang.String value) Sets the JavaScript code that is executed when the contents of an object are made to appear out of focus.
<code>void</code>	setOnClickHandler (java.lang.String value) Sets the JavaScript code that is executed when an object is clicked.

IDE Extensions

void	setOnContextMenuHandler (java.lang.String value) Sets the JavaScript code that is executed when the context menu is displayed.
void	setOnDBClickHandler (java.lang.String value) Sets the JavaScript code that is executed when an object is double-clicked.
void	setOnDragEndHandler (java.lang.String value) Sets the JavaScript code that is executed at the end of a drag operation.
void	setOnDragEnterHandler (java.lang.String value) Sets the JavaScript code that is executed when an object is dragged into the controls bounds.
void	setOnDragHandler (java.lang.String value) Sets the JavaScript code that is executed during a drag operation.
void	setOnDragLeaveHandler (java.lang.String value) Sets the JavaScript code that is executed when an object is dragged out of the controls bounds.
void	setOnDragOverHandler (java.lang.String value) Sets the JavaScript code that is executed when a control is being dragged over a target object.
void	setOnDragStartHandler (java.lang.String value) Sets the JavaScript code that is executed at the start of a drag operation.
void	setOnDropHandler (java.lang.String value) Sets the JavaScript code that is executed when the drag and drop operation is complete.
void	setOnFocusHandler (java.lang.String value) Sets the JavaScript code that is executed when the object receives focus.
void	setOnFocusInHandler (java.lang.String value) Sets the JavaScript code that is executed just prior to the object receiving focus.
void	setOnFocusOutHandler (java.lang.String value) Sets the JavaScript code that is executed immediately after the object loses focus.
void	setOnKeyDownHandler (java.lang.String value) Sets the JavaScript code that is executed when an alphanumeric key is pressed.
void	setOnKeyPressHandler (java.lang.String value) Sets the JavaScript code that is executed when a key is pressed.
void	setOnKeyUpHandler (java.lang.String value) Sets the JavaScript code that is executed when a key is

IDE Extensions

	released.
void	setOnMouseDownHandler (java.lang.String value) Sets the JavaScript code that is executed when an object is clicked with either mouse button.
void	setOnMouseEnterHandler (java.lang.String value) Sets the JavaScript code that is executed when the mouse pointer is moved into the boundaries of an object.
void	setOnMouseLeaveHandler (java.lang.String value) Sets the JavaScript code that is executed when the mouse pointer is moved outside of the boundaries of an object.
void	setOnMouseMoveHandler (java.lang.String value) Sets the JavaScript code that is executed when the mouse pointer is moved over an object.
void	setOnMouseOutHandler (java.lang.String value) Sets the JavaScript code that is executed when the mouse pointer is moved outside of the boundaries of an object.
void	setOnMouseOverHandler (java.lang.String value) Sets the JavaScript code that is executed when the mouse pointer is moved into an object.
void	setOnMouseUpHandler (java.lang.String value) Sets the JavaScript code that is executed when either mouse button is released over an object.
void	setOnSelectHandler (java.lang.String value) Sets the JavaScript code that is executed when the current selection changes.
void	setOnSelectionChangeHandler (java.lang.String value) Sets the JavaScript code that is executed when the selection state of the document changes.
void	setOnSelectStartHandler (java.lang.String value) Sets the JavaScript code that is executed when the object is being selected.
void	setOnStopHandler (java.lang.String value) Sets the JavaScript code that is executed when the user clicks the stop button or leaves the web page.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BeforeRenderObjectEvent

```
public BeforeRenderObjectEvent(java.lang.Object source)
```

Method Detail

getAttribute

```
public java.lang.String getAttribute()
```

For internal use only.

getObject

```
public java.lang.Object getObject()
```

Returns the report object to be cached. This can be a `com.crystaldecisions.sdk.occa.report.definition.FieldObject`, `com.crystaldecisions.sdk.occa.report.definition.TextObject`, `com.crystaldecisions.sdk.occa.report.definition.ChartObject`, and so on.

Returns:

The report object to be cached as an `Object`.

See Also:

`setObject`

getOnBlurHandler

```
public java.lang.String getOnBlurHandler()
```

Returns the JavaScript code that is executed when the contents of an object are made to appear out of focus.

Returns:

The JavaScript code as a `String`.

See Also:

`setOnBlurHandler`

getOnClickHandler

```
public java.lang.String getOnClickHandler()
```

Returns the JavaScript code that is executed when an object is clicked.

Returns:

The JavaScript code as a `String`.

See Also:

`setOnClickHandler`

getOnContextMenuHandler

```
public java.lang.String getOnContextMenuHandler()
```

Returns the JavaScript code that is executed when the context menu is displayed.

Returns:

The JavaScript code as a String.

See Also:

setOnContextMenuHandler

getOnDBClickHandler

```
public java.lang.String getOnDBClickHandler()
```

Returns the JavaScript code that is executed when an object is double-clicked.

Returns:

The JavaScript code as a String.

See Also:

setOnDBClickHandler

getOnDragEndHandler

```
public java.lang.String getOnDragEndHandler()
```

Returns the JavaScript code that is executed at the end of a drag operation.

Returns:

The JavaScript code as a String.

See Also:

setOnDragEndHandler

getOnDragEnterHandler

```
public java.lang.String getOnDragEnterHandler()
```

Returns the JavaScript code that is executed when an object is dragged into the controls bounds.

Returns:

The JavaScript code as a String.

See Also:

setOnDragEnterHandler

getOnDragHandler

```
public java.lang.String getOnDragHandler()
```

Returns the JavaScript code that is executed during a drag operation.

Note: The code is cached on the client in the HTML page.

Returns:

The JavaScript code as a String.

See Also:

`setOnDragHandler`

getOnDragLeaveHandler

```
public java.lang.String getOnDragLeaveHandler()
```

Returns the JavaScript code that is executed when an object is dragged out of the controls bounds.

Returns:

The JavaScript code as a String.

See Also:

`setOnDragLeaveHandler`

getOnDragOverHandler

```
public java.lang.String getOnDragOverHandler()
```

Returns the JavaScript code that is executed when a control is being dragged over a target object.

Returns:

The JavaScript code as a String.

See Also:

`setOnDragOverHandler`

getOnDragStartHandler

```
public java.lang.String getOnDragStartHandler()
```

Returns the JavaScript code that is executed at the start of a drag operation.

Returns:

The JavaScript code as a String.

See Also:

`setOnDragStartHandler`

getOnDropHandler

```
public java.lang.String getOnDropHandler()
```

Returns the JavaScript code that is executed when the drag and drop operation is complete.

Returns:

The JavaScript code as a String.

See Also:

`setOnDropHandler`

getOnFocusHandler

```
public java.lang.String getOnFocusHandler()
```

Returns the JavaScript code that is executed when the object receives focus.

Returns:

The JavaScript code as a String.

See Also:

`setOnFocusHandler`

getOnFocusInHandler

```
public java.lang.String getOnFocusInHandler()
```

Returns the JavaScript code that is executed just prior to the object receiving focus.

Returns:

The JavaScript code as a String.

See Also:

`setOnFocusInHandler`

getOnFocusOutHandler

```
public java.lang.String getOnFocusOutHandler()
```

Returns the JavaScript code that is executed immediately after the object loses focus.

Returns:

The JavaScript code as a String.

See Also:

`setOnFocusOutHandler`

getOnKeyDownHandler

```
public java.lang.String getOnKeyDownHandler()
```

Returns the JavaScript code that is executed when an alphanumeric key is pressed.

Returns:

The JavaScript code as a String.

See Also:

`setOnKeyDownHandler`

getOnKeyPressHandler

```
public java.lang.String getOnKeyPressHandler()
```

Returns the JavaScript code that is executed when a key is pressed.

Returns:

The JavaScript code as a String.

See Also:

setOnKeyPressHandler

getOnKeyUpHandler

```
public java.lang.String getOnKeyUpHandler()
```

Returns the JavaScript code that is executed when a key is released.

Returns:

The JavaScript code as a String.

See Also:

setOnKeyUpHandler

getOnMouseDownHandler

```
public java.lang.String getOnMouseDownHandler()
```

Returns the JavaScript code that is executed when an object is clicked with either mouse button.

Returns:

The JavaScript code as a String.

See Also:

setOnMouseDownHandler

getOnMouseEnterHandler

```
public java.lang.String getOnMouseEnterHandler()
```

Returns the JavaScript code that is executed when the mouse pointer is moved into the boundaries of an object.

Returns:

The JavaScript code as a String.

See Also:

setOnMouseEnterHandler

getOnMouseLeaveHandler

```
public java.lang.String getOnMouseLeaveHandler()
```

Returns the JavaScript code that is executed when the mouse pointer is moved outside of the boundaries of an object.

Returns:

The JavaScript code as a String.

See Also:

`setOnMouseLeaveHandler`

getOnMouseMoveHandler

```
public java.lang.String getOnMouseMoveHandler()
```

Returns the JavaScript code that is executed when the mouse pointer is moved over an object.

Returns:

The JavaScript code as a String.

See Also:

`setOnMouseMoveHandler`

getOnMouseOutHandler

```
public java.lang.String getOnMouseOutHandler()
```

Returns the JavaScript code that is executed when the mouse pointer is moved outside of the boundaries of an object.

Returns:

The JavaScript code as a String.

See Also:

`setOnMouseOutHandler`

getOnMouseOverHandler

```
public java.lang.String getOnMouseOverHandler()
```

Returns the JavaScript code that is executed when the mouse pointer is moved into an object.

Returns:

The JavaScript code as a String.

See Also:

`setOnMouseOverHandler`

getOnMouseUpHandler

```
public java.lang.String getOnMouseUpHandler()
```

Returns the JavaScript code that is executed when either mouse button is released over an object.

Returns:

The JavaScript code as a String.

See Also:

`setOnMouseUpHandler`

getOnSelectHandler

```
public java.lang.String getOnSelectHandler()
```

Returns the JavaScript code that is executed when the current selection changes.

Returns:

The JavaScript code as a String.

See Also:

`setOnSelectHandler`

getOnSelectionChangeHandler

```
public java.lang.String getOnSelectionChangeHandler()
```

Returns the JavaScript code that is executed when the selection state of the document changes.

Returns:

The JavaScript code as a String.

See Also:

`setOnSelectionChangeHandler`

getOnSelectStartHandler

```
public java.lang.String getOnSelectStartHandler()
```

Returns the JavaScript code that is executed when the object is being selected.

Returns:

The JavaScript code as a String.

See Also:

`setOnSelectStartHandler`

getOnStopHandler

```
public java.lang.String getOnStopHandler()
```

Returns the JavaScript code that is executed when the user clicks the stop button or leaves the web page.

Returns:

The JavaScript code as a String.

See Also:

`setOnStopHandler`

setAttribute

```
public void setAttribute(java.lang.String value)
```

For internal use only.

setObject

```
public void setObject(java.lang.Object value)
```

Sets the report object to be cached. This can be a `com.crystaldecisions.sdk.occa.report.definition.FieldObject`, `com.crystaldecisions.sdk.occa.report.definition.TextObject`, `com.crystaldecisions.sdk.occa.report.definition.ChartObject`, and so on.

Parameters:

value – An Object that represents the report object to be cached.

See Also:

`getObject`

setOnBlurHandler

```
public void setOnBlurHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the contents of an object are made to appear out of focus.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

`getOnBlurHandler`

setOnClickHandler

```
public void setOnClickHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when an object is clicked.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

`getOnClickHandler`

setOnContextMenuHandler

```
public void setOnContextMenuHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the context menu is displayed.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnContextMenuHandler

setOnDBClickHandler

```
public void setOnDBClickHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when an object is double-clicked.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnDBClickHandler

setOnDragEndHandler

```
public void setOnDragEndHandler(java.lang.String value)
```

Sets the JavaScript code that is executed at the end of a drag operation.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnDragEndHandler

setOnDragEnterHandler

```
public void setOnDragEnterHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when an object is dragged into the controls bounds.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnDragEnterHandler

setOnDragHandler

```
public void setOnDragHandler(java.lang.String value)
```

Sets the JavaScript code that is executed during a drag operation.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnDragHandler

setOnDragLeaveHandler

```
public void setOnDragLeaveHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when an object is dragged out of the controls bounds.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnDragLeaveHandler

setOnDragOverHandler

```
public void setOnDragOverHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when a control is being dragged over a target object.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnDragOverHandler

setOnDragStartHandler

```
public void setOnDragStartHandler(java.lang.String value)
```

Sets the JavaScript code that is executed at the start of a drag operation.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

`getOnDragStartHandler`

setOnDropHandler

```
public void setOnDropHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the drag and drop operation is complete.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

`getOnDropHandler`

setOnFocusHandler

```
public void setOnFocusHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the object receives focus.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

`getOnFocusHandler`

setOnFocusInHandler

```
public void setOnFocusInHandler(java.lang.String value)
```

Sets the JavaScript code that is executed just prior to the object receiving focus.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

`getOnFocusInHandler`

setOnFocusOutHandler

```
public void setOnFocusOutHandler(java.lang.String value)
```

Sets the JavaScript code that is executed immediately after the object loses focus.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnFocusOutHandler

setOnKeyDownHandler

```
public void setOnKeyDownHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when an alphanumeric key is pressed.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnKeyDownHandler

setOnKeyPressHandler

```
public void setOnKeyPressHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when a key is pressed.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnKeyPressHandler

setOnKeyUpHandler

```
public void setOnKeyUpHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when a key is released.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnKeyUpHandler

setOnMouseDownHandler

```
public void setOnMouseDownHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when an object is clicked with either mouse button.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnMouseDownHandler

setOnMouseEnterHandler

```
public void setOnMouseEnterHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the mouse pointer is moved into the boundaries of an object.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnMouseEnterHandler

setOnMouseLeaveHandler

```
public void setOnMouseLeaveHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the mouse pointer is moved outside of the boundaries of an object.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnMouseLeaveHandler

setOnMouseMoveHandler

```
public void setOnMouseMoveHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the mouse pointer is moved over an object.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnMouseMoveHandler

setOnMouseOutHandler

```
public void setOnMouseOutHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the mouse pointer is moved outside of the boundaries of an object.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnMouseOutHandler

setOnMouseOverHandler

```
public void setOnMouseOverHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the mouse pointer is moved into an object.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnMouseOverHandler

setOnMouseUpHandler

```
public void setOnMouseUpHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when either mouse button is released over an object.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnMouseUpHandler

setOnSelectHandler

```
public void setOnSelectHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the current selection changes.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnSelectHandler

setOnSelectionChangeHandler

```
public void setOnSelectionChangeHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the selection state of the document changes.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnSelectionChangeHandler

setOnSelectStartHandler

```
public void setOnSelectStartHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the object is being selected.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnSelectStartHandler

setOnStopHandler

```
public void setOnStopHandler(java.lang.String value)
```

Sets the JavaScript code that is executed when the user clicks the stop button or leaves the web page.

Note: The code is cached on the client in the HTML page.

Parameters:

value – A String that specifies the JavaScript code.

See Also:

getOnStopHandler

com.crystaldecisions.report.htmlrender Interface BeforeRenderObjectEventListener

All Superinterfaces:

java.util.EventListener

```
public interface BeforeRenderObjectEventListener
```

extends java.util.EventListener

This class provides a listener for a BeforeRenderObjectEvent.

Method Summary

void	beforeRenderObject (BeforeRenderObjectEvent e) The listener for a BeforeRenderObjectEvent.
------	--

Method Detail

beforeRenderObject

```
public void beforeRenderObject(BeforeRenderObjectEvent e)
```

The listener for a BeforeRenderObjectEvent. The event occurs fires when the report object rendering process starts.

Parameters:

e – A BeforeRenderObjectEvent object.

See Also:

BeforeRenderObjectEvent

com.crystaldecisions.report.htmlrender Class BeforeRenderViewItemEvent

```
java.lang.Object
├─ RenderEventObjectBase
│   └─ com.crystaldecisions.report.htmlrender.BeforeRenderViewItemEvent
```

```
public class BeforeRenderViewItemEvent
    extends RenderEventObjectBase
```

This class provides arguments for the IBeforeRenderViewItemEventListener. The event occurs before rendering a ViewItem (Toolbar, GroupTree, or Page).

See Also:

Serialized Form

Constructor Summary

BeforeRenderViewItemEvent (java.lang.Object source)	
--	--

BeforeRenderViewItemEvent (java.lang.Object source, CrystalHtmlTextWriter writer)

Method Summary

java.lang.String	getType () Returns the type of view item to be rendered.
void	setType (java.lang.String newM_Type) Sets the type of view item to be rendered.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BeforeRenderViewItemEvent

```
public BeforeRenderViewItemEvent(java.lang.Object source)
```

BeforeRenderViewItemEvent

```
public BeforeRenderViewItemEvent(java.lang.Object source,  
                                CrystalHtmlTextWriter writer)
```

Method Detail

getType

```
public java.lang.String getType()
```

Returns the type of view item to be rendered.

Returns:

The type of view item to be rendered as a String.

See Also:

setType

setType

```
public void setType(java.lang.String newM_Type)
```

Sets the type of view item to be rendered.

Note: The type of view item must be either Toolbar, GroupTree, or Page.

Parameters:

newM_Type – A String that specifies the type of view item to be rendered.

See Also:

getType

com.crystaldecisions.report.htmlrender Interface IAfterRenderViewItemEventListener

All Superinterfaces:

java.util.EventListener

```
public interface IAfterRenderViewItemEventListener  
extends java.util.EventListener
```

This class provides a listener for a AfterRenderViewItemEvent.

Method Summary

void	afterRenderViewItem (AfterRenderViewItemEvent e) The listener for a AfterRenderViewItemEvent.
------	---

Method Detail

afterRenderViewItem

```
public void afterRenderViewItem(AfterRenderViewItemEvent e)
```

The listener for a AfterRenderViewItemEvent. The event occurs after rendering a view item (Toolbar, GroupTree, or Page).

Parameters:

e – An AfterRenderViewItemEvent object.

See Also:

AfterRenderViewItemEvent

com.crystaldecisions.report.htmlrender Interface **IBeforeRenderViewItemEventListener**

All Superinterfaces:

java.util.EventListener

*public interface **IBeforeRenderViewItemEventListener**
extends java.util.EventListener*

This class provides a listener for a BeforeRenderViewItemEvent.

Method Summary

void	beforeRenderViewItem (BeforeRenderViewItemEvent e) The listener for a BeforeRenderViewItemEvent.
------	--

Method Detail

beforeRenderViewItem

```
public void beforeRenderViewItem(BeforeRenderViewItemEvent e)
```

The listener for a BeforeRenderViewItemEvent. The event occurs before rendering a ViewItem (Toolbar, GroupTree, or Page).

Parameters:

e – A BeforeRenderViewItemEvent object.

See Also:

BeforeRenderViewItemEvent

com.crystaldecisions.report.htmlrender Class **OnRenderScriptEvent**

```
java.lang.Object
├─ RenderEventObjectBase
└─ com.crystaldecisions.report.htmlrender.OnRenderScriptEvent
```

*public class **OnRenderScriptEvent**
extends RenderEventObjectBase*

This class provides arguments for the OnRenderScriptEventListener. The event occurs when the HTML script generation starts.

See Also:

Serialized Form

Constructor Summary

OnRenderScriptEvent (java.lang.Object source)
--

Method Summary

boolean	getHandled () Returns whether the event is handled.
java.lang.String	getScript () Returns the <script> block.
void	setHandled (boolean value) Sets whether the event is handled.
void	setScript (java.lang.String value) Sets the <script> block.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

OnRenderScriptEvent

```
public OnRenderScriptEvent(java.lang.Object source)
```

Method Detail

getHandled

```
public boolean getHandled()
```

Returns whether the event is handled.

Returns:

true if the event is handled and false otherwise.

See Also:

setHandled

getScript

```
public java.lang.String getScript()
```

Returns the <script> block.

Returns:

The <script> block as a String.

See Also:

setScript

setHandled

```
public void setHandled(boolean value)
```

Sets whether the event is handled. Set to true if you do not want the default action to occur.

Parameters:

value – true if the event is handled.

See Also:

getHandled

setScript

```
public void setScript(java.lang.String value)
```

Sets the <script> block. This includes the beginning <script> tag and the end <script> tag.

Parameters:

value – A String that specifies the <script> block.

See Also:

getScript

com.crystaldecisions.report.htmlrender Interface OnRenderScriptEventListener

All Superinterfaces:

java.util.EventListener

```
public interface OnRenderScriptEventListener  
extends java.util.EventListener
```

This class provides a listener for a OnRenderScriptEvent.

Method Summary

void	onRenderScript (OnRenderScriptEvent e) The listener for a OnRenderScriptEvent.
------	--

Method Detail

onRenderScript

```
public void onRenderScript(OnRenderScriptEvent e)
```

The listener for a OnRenderScriptEvent. The event occurs when the HTML script generation starts.

Parameters:

e – An OnRenderScriptEvent object.

See Also:

OnRenderScriptEvent

com.crystaldecisions.report.htmlrender

Class OnRenderStyleEvent

```
java.lang.Object
├─ RenderEventObjectBase
├─ com.crystaldecisions.report.htmlrender.OnRenderStyleEvent
```

```
public class OnRenderStyleEvent
extends RenderEventObjectBase
```

This class provides arguments for the OnRenderStyleEventListener. The event occurs when the HTML style class generation starts.

See Also:

Serialized Form

Constructor Summary

```
OnRenderStyleEvent( java.lang.Object source)
```

Method Summary

IDE Extensions

boolean	getHandled() Returns whether the event is handled.
java.lang.String	getStyle() Returns the <style> block.
void	setHandled(boolean value) Sets whether the event is handled.
void	setStyle(java.lang.String value) Sets the <style> block.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

OnRenderStyleEvent

```
public OnRenderStyleEvent(java.lang.Object source)
```

Method Detail

getHandled

```
public boolean getHandled()
```

Returns whether the event is handled.

Returns:

true if the event is handled and false otherwise.

See Also:

setHandled

getStyle

```
public java.lang.String getStyle()
```

Returns the <style> block.

Returns:

The <style> block as a String.

See Also:

setStyle

setHandled

```
public void setHandled(boolean value)
```

Sets whether the event is handled. Set to true if you do not want the default action to occur.

Parameters:

value – true if the event is handled.

See Also:

getHandled

setStyle

```
public void setStyle(java.lang.String value)
```

Sets the <style> block. This includes the beginning <style> tag and the end <style> tag.

Parameters:

value – A String that specifies the <style> block.

See Also:

getStyle

com.crystaldecisions.report.htmlrender Interface OnRenderStyleEventListener

All Superinterfaces:

java.util.EventListener

*public interface **OnRenderStyleEventListener***
extends java.util.EventListener

This class provides a listener for a OnRenderStyleEvent.

Method Summary	
void	onRenderStyle (OnRenderStyleEvent e) The listener for a OnRenderStyleEvent.

Method Detail

onRenderStyle

```
public void onRenderStyle(OnRenderStyleEvent e)
```

The listener for a OnRenderStyleEvent. The event occurs when the HTML style class generation starts.

Parameters:

e – An OnRenderStyleEvent object.

See Also:

OnRenderStyleEvent

Package com.crystaldecisions.report.web

This package defines contains the class ServerControl, which is an abstract class.

See:

Description

Class Summary

ServerControl	This is an abstract class.
----------------------	----------------------------

Package com.crystaldecisions.report.web Description

This package defines contains the class ServerControl, which is an abstract class. Use the derived classes such as CrystalReportViewer and ReportServerControl

Note: It is recommended that you perform your own garbage collection in your web applications when using the Viewer Java SDK. This involves calling the appropriate dispose methods and ensuring that you use the CrystallImageCleaner class to remove temporary image files used by the viewer.

com.crystaldecisions.report.web

Class ServerControl

```
java.lang.Object
```

```
└─ com.crystaldecisions.report.web.ServerControl
```

Direct Known Subclasses:

ReportServerControl

```
public abstract class ServerControl
```

```
extends java.lang.Object
```

This is an abstract class. Use the derived classes such as CrystalReportViewer and ReportServerControl.

Constructor Summary

ServerControl()

Method Summary

static java.lang.Object	deserializeBase64ToObject (java.lang.String base64) For internal use only.
int	getHeight () Returns the viewer height.
java.lang.String	getHtmlContent (javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response, javax.servlet.ServletContext context) This method handles the user's request to generate the HTML for the report and returns the HTML as a String.
int	getLeft () Returns the value for the position of the left side of the viewer.
java.lang.String	getName () Returns the name of the viewer control.
int	getTop () Returns the value for the position of the top of the viewer.
java.lang.String	getURI () Returns the Universal Resource Identifier for the web page containing the viewer.
java.lang.String	getViewState () Returns the ViewState.
int	getWidth () Returns the viewer width.
boolean	isIgnoreViewStateOnLoad () Returns whether the view state will be ignored when loading the viewer.
boolean	isOwnForm () Returns whether or not the viewer control owns the form.
boolean	isOwnPage () Returns whether or not the viewer control owns the page.
void	processHttpRequest (javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response, javax.servlet.ServletContext context, java.io.Writer out) Handles the user's request to generate the HTML for the report and

IDE Extensions

	writes the HTML directly to the response object.
void	setHeight (int newM_Height) Sets the viewer height.
void	setIgnoreViewStateOnLoad (boolean newIgnoreViewStateOnLoad) Sets whether to ignore the view state when loading the viewer.
void	setLeft (int newM_Left) Sets the position for the left side of the viewer.
void	setName (java.lang.String newName) Sets the name for the viewer control.
void	setOwnForm (boolean newOwnForm) Sets whether the viewer control owns the form.
void	setOwnPage (boolean newOwnPage) Sets whether the viewer control owns the page.
void	setTop (int newM_Top) Sets the value for the position of the top of the viewer.
void	setURI (java.lang.String newURI) Sets the Universal Resource Identifier for the report.
void	setViewState (java.lang.String viewState) The ViewState is an encoded string that represents the current state of the report.
void	setWidth (int newM_Width) Sets the width of the viewer.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ServerControl

```
public ServerControl()
```

Method Detail

deserializeBase64ToObject

```
public static java.lang.Object deserializeBase64ToObject(java.lang.String base64)
    throws java.io.IOException,
           java.lang.ClassNotFoundException
```

For internal use only.

Throws:

java.io.IOException
java.lang.ClassNotFoundException

getHeight

```
public int getHeight()
```

Returns the viewer height.

Returns:

The height of the viewer as an `int`. The unit is browser-dependent.

See Also:

setHeight

getHtmlContent

```
public final java.lang.String getHtmlContent(javax.servlet.http.HttpServletRequest request,
                                             javax.servlet.http.HttpServletResponse response,
                                             javax.servlet.ServletContext context)
    throws ReportSDKExceptionBase
```

This method handles the user's request to generate the HTML for the report and returns the HTML as a `String`.

Note: Either the `getHtmlContent` method or the `processHttpRequest` method can be used to handle the user's request to generate the HTML for the report, depending on how you write your JSP. If the viewer's content will be displayed more than once, then the `getHtmlContent` method will be more efficient, because the request is processed once and the resulting HTML string can be used multiple times. In this case, you must set the content type of the HTML, as recommended in the table.

Viewing format	Recommended HTML content type
Standard web reporting	"text/html; charset=UTF-8"
Mobile	"text/html" or "text/vnd.wap.wml"
PDA	"text/html; charset=UTF-8"

Parameters:

request – A `HttpServletRequest` object containing the values passed to the server during an `Http` request.

response – A `HttpServletResponse` object that receives the HTML output and sends it to the client.

context – A `ServletContext` object that specifies the servlet context.

Returns:

The HTML content as a `String`.

Throws:

ReportSDKExceptionBase

getLeft

```
public int getLeft()
```

Returns the value for the position of the left side of the viewer.

Returns:

The position of the left side of the viewer as an `int`. The unit is browser-dependent.

See Also:

`setLeft`

getName

```
public java.lang.String getName()
```

Returns the name of the viewer control. By default the viewer control is named `CrystalViewer`.

Returns:

The name of the viewer control as a `String`.

See Also:

`setName`

getTop

```
public int getTop()
```

Returns the value for the position of the top of the viewer.

Returns:

The position of the top of the viewer as an `int` for. The unit is browser-dependent.

See Also:

`setTop`

getURI

```
public java.lang.String getURI()
```

Returns the Universal Resource Identifier for the web page containing the viewer.

Returns:

The Universal Resource Identifier for the report as a `String`.

See Also:

`setURI`

getViewState

```
public java.lang.String getViewState()
```

Returns the `ViewState`. This is an encoded string that represents the current state of the report. The `ViewState` is used in order to perform client-side caching of information about the current state of

the report. By default, the viewer will manage the `ViewState` unless `isOwnForm` is set to `false`.

Returns:

An encoded `String` that represents the current state of the report.

See Also:

`setOwnForm`, `isOwnForm`, `setViewState`

getWidth

```
public int getWidth()
```

Returns the viewer width.

Returns:

The viewer width as an `int`. The unit is browser-dependent.

See Also:

`setWidth`

isIgnoreViewStateOnLoad

```
public boolean isIgnoreViewStateOnLoad()
```

Returns whether the view state will be ignored when loading the viewer. You would want this to be set to `true` if the viewer is loaded in order to display a different page of the report.

Returns:

`true` if the view state will be ignored and `false` otherwise.

See Also:

`setIgnoreViewStateOnLoad`

isOwnForm

```
public boolean isOwnForm()
```

Returns whether or not the viewer control owns the form. If the server control owns the form it is able to get and set values for the form. In particular, by owning the form, the viewer control creates the `ViewState` object that is used in order to perform client-side caching of information about the current state of the report.

If this is set to `false`, it is the host's responsibility to use `getViewState()` and set persistence (form, session). The host may use `setViewState` before calling `processHttpRequest()`; if not the control will obtain it from the request object.

Returns:

`true` if the server control owns the form and `false` otherwise. By default, this value is `true`.

See Also:

`setViewState`, `getViewState`, `setOwnForm`

isOwnPage

```
public boolean isOwnPage()
```

Returns whether or not the viewer control owns the page. In other words, the page is not within a portal and the JSP page only contains the viewer control. If the server control owns the page, it provides the opening and closing HTML tags and it is able to get and set values for the entire page.

Returns:

true if the server control owns the page and false otherwise.

See Also:

setOwnPage

processHttpRequest

```
public final void processHttpRequest(javax.servlet.http.HttpServletRequest request,
                                     javax.servlet.http.HttpServletResponse response,
                                     javax.servlet.ServletContext context,
                                     java.io.Writer out)
    throws ReportSDKExceptionBase
```

Handles the user's request to generate the HTML for the report and writes the HTML directly to the response object.

Note: Either the `getHtmlContent` method or the `processHttpRequest` method can be used to handle the user's request to generate the HTML for the report, depending on how you write your JSP. If the viewer's content will be displayed more than once, then the `getHtmlContent` method will be more efficient, because the request is processed once and the resulting HTML string can be used multiple times.

Note: If you provide your own writer then you must set the `charset` and the `content-type`. For improved performance, you may wish to wrap your writer in a `java.io.BufferedWriter`.

Parameters:

`response` – A `HttpServletResponse` object that receives the HTML output and sends it to the client.
`context` – A `ServletContext` object that specifies the servlet context.
`request` – The `HttpServletRequest` object containing the values passed to the server during an Http request.
`out` – A `Writer` object. This value is set to null unless you are providing your own writer.

Throws:

`ReportSDKExceptionBase`

setHeight

```
public void setHeight(int newM_Height)
```

Sets the viewer height.

Note: Together, the `setHeight` and `setWidth` methods control the viewable size of the viewer. The default value is 0 (zero). If these methods are set to zero, the viewable size of the viewer is calculated based on the report the user is viewing.

If the `setBestFitPage` method is set to true, the values of the `setHeight` and `setWidth` methods are ignored.

Parameters:

`newM_Height` – An `int` that specifies the height of the viewer. The unit is browser-dependent.

See Also:

`getHeight`

setIgnoreViewStateOnLoad

```
public void setIgnoreViewStateOnLoad(boolean newIgnoreViewStateOnLoad)
```

Sets whether to ignore the view state when loading the viewer. You would want this to be set to true, if the viewer is loaded in order to display a different page of the report.

Parameters:

`newIgnoreViewStateOnLoad` – true to make the viewer ignore the `ViewState` when loading.

See Also:

`isIgnoreViewStateOnLoad`

setLeft

```
public void setLeft(int newM_Left)
```

Sets the position for the left side of the viewer.

Parameters:

`newM_Left` – An `int` that specifies the position for the left side of the viewer. The unit is browser-dependent.

See Also:

`getLeft`

setName

```
public void setName(java.lang.String newName)
```

Sets the name for the viewer control. By default the viewer control is named `CrystalViewer`.

Parameters:

`newName` – The `String` that specifies the name of the viewer control.

See Also:

`getName`

setOwnForm

```
public void setOwnForm(boolean newOwnForm)
```

Sets whether the viewer control owns the form. By default this is `true`. If this is set to `false`, it is the host's responsibility to `getViewState()` and set persistence (form, session). The host may `setViewState` before calling `processHttpRequest()`; if not the control will obtain it from the request object.

Parameters:

`newOwnForm` – `true` to make the viewer object own the form. By default this is `true`.

See Also:

`setViewState`, `getViewState`, `isOwnForm`

setOwnPage

```
public void setOwnPage(boolean newOwnPage)
```

Sets whether the viewer control owns the page. Set this to `false` if the page is within a portal. If the server control owns the page, it provides the opening and closing HTML tags and it is able to get and set values for the entire page.

Note: If you set `setOwnPage` to `false` then you must set the `charset` and the `content-type`.

Parameters:

`newOwnPage` – `true` to make the viewer object own the page. The default value is `true`.

See Also:

`isOwnPage`

setTop

```
public void setTop(int newM_Top)
```

Sets the value for the position of the top of the viewer.

Parameters:

`newM_Top` – An `int` value for the position of the top of the viewer. The unit is browser-dependent.

See Also:

`getTop`

setURI

```
public void setURI(java.lang.String newURI)
```

Sets the Universal Resource Identifier for the report.

Parameters:

`newURI` – A `String` that specifies the Universal Resource Identifier for the report.

See Also:

`getURI`

setViewState

```
public void setViewState(java.lang.String viewState)
```

The ViewState is an encoded string that represents the current state of the report. The ViewState is used in order to perform client-side caching of information about the current state of the report. By default, the viewer will manage the ViewState unless `isOwnForm` is set to `false`.

Parameters:

`viewState` – An encoded String that represents the current state of the report.

See Also:

`getViewState`, `setOwnForm`, `isOwnForm`

setWidth

```
public void setWidth(int newM_Width)
```

Sets the width of the viewer.

Note: Together, the `setHeight` and `setWidth` methods control the viewable size of the viewer. The default value is 0 (zero). If these methods are set to zero, the viewable size of the viewer is calculated based on the report the user is viewing.

If the `setBestFitPage` method is set to `true`, the values of the `setHeight` and `setWidth` methods are ignored.

Parameters:

`newM_Width` – An `int` value for the width of the viewer. The unit is browser-dependent.

See Also:

`getWidth`

Package com.crystaldecisions.report.web.viewer

This package contains the key functionality for the display and behavior of the viewers, including report source information, event listeners, and event arguments.

See:

Description

Interface Summary	
DrillDownSubreportEventListener	Listener for DrillDownSubreportEvents.
DrillEventListener	Listener for DrillEvents.
NavigateEventListener	Listener for NavigateEvents.
ReportSourceChangeEventListeners	Listener for changes in report source.
ToolbarCommandEventListeners	Listener for ToolbarCommandEvents.

Class Summary	
CrHtmlUnitEnum	This class allows you to specify the units used to determine the group tree width.
CrPrintMode	This class contains constants that specify the type of printing the viewer will use when the user clicks on the print button.
CrystalImageCleaner	This class is used to periodically scan the temporary directory to delete image files that were used by the viewer but were not subsequently deleted.
CrystalReportViewer	This class enables you to manipulate how the viewer is displayed.
CrystalReportViewerBase	This class provides listeners for the report viewer, as well as providing methods for getting and setting hyperlink targets.
DrillDownSubreportEventArgs	This class provides arguments for the Subreport DrillDown Event listener.
DrillEventArgs	This class provides event arguments for the <code>DrillEventListener</code> .
NavigateEventArgs	This class provides event arguments for the <code>NavigateEventListener</code> .
ReportExportControl	This class provides exporting functionality including the ability to preview an exported report or save an exported report to disk.
ReportServerControl	This class allows you to manage how reports interact with the server.
ToolbarCommandEventArgs	This class provides event arguments for the <code>ToolbarCommandEventListener</code> .
ViewerEventArgs	This class provides event arguments for the viewer.
ViewInfo	This class provides more information for the client when handling an event.

Package `com.crystaldecisions.report.web.viewer` Description

This package contains the key functionality for the display and behavior of the viewers, including report source information, event listeners, and event arguments. Some of the major classes include:

- `ReportServerControl` is the base class for the viewers and contains methods inherited by the viewer that enable you to set report logon, report parameters and report source.
- `CrystalReportViewer` enables you to manipulate how the viewer is displayed.
- `ReportExportControl` enables you to preview an exported report or save an exported report to disk.
- Other classes and interfaces provide specific event listeners and arguments.
- `ViewInfo` extends the event handling information provided to the client.

Note: It is recommended that you perform your own garbage collection in your web applications when using the Viewer Java SDK. This involves calling the appropriate dispose methods and ensuring that you use the

CrystallImageCleaner class to remove temporary image files used by the viewer.

com.crystaldecisions.report.web.viewer

Class CrHtmlUnitEnum

```
java.lang.Object
└─ com.crystaldecisions.report.web.viewer.CrHtmlUnitEnum
```

*public class **CrHtmlUnitEnum***
extends java.lang.Object

This class allows you to specify the units used to determine the group tree width. It allows you to choose between using pixels or percentage to determine the width.

Field Summary

static CrHtmlUnitEnum	crHtmlUnitPercentage A CrHtmlUnitEnum object that specifies to use percentage to calculate the group tree width.
static CrHtmlUnitEnum	crHtmlUnitPixel A CrHtmlUnitEnum object that specifies to use pixels to calculate the group tree width.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

crHtmlUnitPixel

```
public static final CrHtmlUnitEnum crHtmlUnitPixel
```

A CrHtmlUnitEnum object that specifies to use pixels to calculate the group tree width.

crHtmlUnitPercentage

```
public static final CrHtmlUnitEnum crHtmlUnitPercentage
```

A `CrHtmlUnitEnum` object that specifies to use percentage to calculate the group tree width.

com.crystaldecisions.report.web.viewer

Class CrPrintMode

```
java.lang.Object
└─ com.crystaldecisions.report.web.viewer.CrPrintMode
```

*public final class **CrPrintMode***
extends java.lang.Object

This class contains constants that specify the type of printing the viewer will use when the user clicks on the print button.

The print mode can be set by passing a `CrPrintMode` object to the `CrystalReportViewer` class' `setPrintMode` method for a specific JSP or in the deployment descriptor for a web application wide setting. It can be set as either a context `initParameter` or an attribute. The tag name is `crystal_print_mode` and its possible values are:

- PDF – Export to Acrobat Format. This is the default value.
- ActiveX – Use an ActiveX control to send the file to a printer.

In the situation that the print mode is set through both the deployment descriptor and the viewer, the viewer specified print mode value will take precedence.

Field Summary

<code>static CrPrintMode</code>	ACTIVEX A <code>CrPrintMode</code> object that specifies ActiveX print mode.
<code>static CrPrintMode</code>	PDF A <code>CrPrintMode</code> object that specifies PDF print mode.

Method Summary

<code>java.lang.String</code>	toString() Returns the print mode specified by this <code>CrPrintMode</code> object.
-------------------------------	---

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Field Detail

PDF

```
public static final CrPrintMode PDF
```

A CrPrintMode object that specifies PDF print mode.

ACTIVEX

```
public static final CrPrintMode ACTIVEX
```

A CrPrintMode object that specifies ActiveX print mode.

Method Detail

toString

```
public java.lang.String toString()
```

Returns the print mode specified by this CrPrintMode object.

Returns:

A String specifying the print mode represented by this object.

com.crystaldecisions.report.web.viewer

Class CrystalImageCleaner

```
java.lang.Object
```

```
└─ com.crystaldecisions.report.web.viewer.CrystalImageCleaner
```

```
public class CrystalImageCleaner  
extends java.lang.Object
```

This class is used to periodically scan the temporary directory to delete image files that were used by the viewer but were not subsequently deleted. The `CrystalImageCleaner` is provided as a means for the server-side application developer (that is, JSP developer) to easily perform this clean up. The developer can specify the interval between scans of the temporary directory and the minimum age of image files that are deleted from the directory. Each web server requires only one instance of the `CrystalImageCleaner` object. Place any code that implements this class in the main JSP page.

Example:

This example creates a `CrystalImageCleaner` object that scans for image files once every minute, but only deletes files that are at least two minutes old.

```
<%!  
public void jspInit() {  
    CrystalImageCleaner.start(getServletContext(), 60000, 12000);  
}
```

```
public void jspDestroy() {
    CrystalImageCleaner.stop(getServletContext());
}
%>
```

Method Summary

static void	start (ServletContext context, long sleep, long age) Use the start method to start scanning the temporary image directory for image files.
static void	stop (ServletContext context) Use the stop method to stop scanning the temporary directory for image files.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

start

```
public static void start(ServletContext context,
                        long sleep,
                        long age)
```

Use the start method to start scanning the temporary image directory for image files.

Note: The temporary directory is scanned for images in the interval provided in the `sleep` parameter until the `Stop` method is called.

Parameters:

`context` – A `ServletContext` object that specifies the context of the Servlet.
`sleep` – A long value that specifies the interval of time in milliseconds that the `CrystalImageCleaner` should sleep before scanning the temporary directory for images.
`age` – A long value that specifies the minimum age of an image file in milliseconds before it is deleted by the `CrystalImageCleaner` when the directory is scanned.

stop

```
public static void stop(ServletContext context)
```

Use the stop method to stop scanning the temporary directory for image files.

Parameters:

context – A ServletContext object that specifies the context of the Servlet.

com.crystaldecisions.report.web.viewer

Class CrystalReportViewer

```
java.lang.Object
├── com.crystaldecisions.report.web.ServerControl
│   └── com.crystaldecisions.report.web.viewer.ReportServerControl
│       └── com.crystaldecisions.report.web.viewer.CrystalReportViewerBase
│           └── com.crystaldecisions.report.web.viewer.CrystalReportViewer
```

*public class **CrystalReportViewer***
*extends **CrystalReportViewerBase***

This class enables you to manipulate how the viewer is displayed. As well as methods to add listeners for specific events within the displayed report, there are methods for setting various options concerning how the report is displayed.

Field Summary

static java.lang.String	PAGE_TYPE Specifies view type to be rendered as Page
static java.lang.String	TOOLBAR_TYPE Specifies view type to be rendered as Toolbar
static java.lang.String	TREE_TYPE Specifies view type to be rendered as GroupTree

Fields inherited from class com.crystaldecisions.report.web.viewer.CrystalReportViewerBase

BACKWARD, FORWARD, NOACTION

Constructor Summary

CrystalReportViewer()
 Constructs the CrystalReportViewer object.

Method Summary

void	addDrillDownSubreportEventListener (DrillDownSubreportEventListener listener) Adds a listener for DrillDownSubreportEvents.
void	addNavigateEventListener (NavigateEventListener listener) Adds a listener for NavigateEvents.
void	

IDE Extensions

	addToolBarCommandEventListener (ToolBarCommandEventListener listener) Adds a listener for ToolBarCommandEvents.
void	drillDown (IGroupPath groupPath, java.lang.String groupName) Drills down on the report.
int	getGroupTreeWidth () Returns the current width of the group tree.
CrHtmlUnitEnum	getGroupTreeWidthUnit () Returns whether pixels or percentage is being used to determine the group tree width.
double	getPageToTreeRatio () Deprecated. Use <i>getGroupTreeWidth()</i> and <i>getGroupTreeWidthUnit()</i> instead.
CrPrintMode	getPrintMode () Returns the current print mode that will be used when the user clicks the print button.
ToolBar	getToolBar () Returns the toolbar object.
ViewInfo	getViewInfo () Returns the ViewInfo object.
int	getZoomFactor () Returns the zoom factor that is currently set.
boolean	hasExportButton () Returns whether the report viewer has an export button.
boolean	hasGotoPageButton () Returns whether the report viewer has a "Go to Page" button.
boolean	hasLogo () Returns whether the report viewer is set to display the Crystal Decisions logo.
boolean	hasPageNavigationButtons () Returns whether the report viewer has page navigation buttons.
boolean	hasPrintButton () Returns whether the report viewer has a print button.
boolean	hasRefreshButton () Returns whether the report viewer has a refresh button.
boolean	hasSearchButton () Returns whether the report viewer has a search button.
boolean	hasToggleGroupTreeButton () Returns whether the report viewer has a button for toggling the display of the group tree.
boolean	hasViewList () Returns whether the report viewer has a view list.
boolean	hasZoomFactorList () Returns whether the report viewer has a zoom factor list.
boolean	isBestFitPage () Returns whether the viewer will ignore the height and width values it is given and try to find the report by taking up as much space as it needs to properly display the report.
boolean	

IDE Extensions

	isDisplayGroupTree () Returns whether the group tree is displayed.
boolean	isDisplayPage () Returns whether the report page is displayed.
boolean	isDisplayToolbar () Returns whether the toolbar is displayed.
boolean	isEnabledDrillDown () Returns whether the user can drill down on the report.
boolean	isRenderAsHTML32 () Returns whether the report is rendered as HTML 3.2.
boolean	isSeparatePages () Returns whether the report is displayed as separate pages or one long page.
boolean	isShowAllPageIds () Returns whether PageIds are added to the HTML content.
void	removeDrillDownSubreportEventListener () Removes a DrillDownSubreportEventListener.
void	removeNavigateEventListener () Removes a NavigateEventListener.
void	removeToolbarCommandEventListener () Removes a ToolbarCommandEventListener.
void	searchText (java.lang.String strTextToSearch, int searchDirection) Highlights the first occurrence of the specified text in the report and scrolls to it.
void	setBestFitPage (boolean bestFitPage) Sets whether the viewer will ignore the height and width values it is given and try to find the report by taking up as much space as it needs to properly display the report.
void	setDisplayGroupTree (boolean newM_bDisplayGroupTree) Sets whether to display the group tree.
void	setDisplayPage (boolean newM_bDisplayPage) Sets whether to display the report page.
void	setDisplayToolbar (boolean display) Sets whether to display the toolbar.
void	setEnabledDrillDown (boolean enableDrillDown) Sets whether to enable drill down.
void	setGroupTreeWidth (int newGroupTreeWidth) Sets the width of the group tree.
void	setGroupTreeWidthUnit (CrHtmlUnitEnum newGroupTreeWidthUnit) Sets whether to use pixels or percentage to determine the group tree width.
void	setHasExportButton (boolean newM_bHasExportButton) Sets whether to display the export button.
void	setHasGotoPageButton (boolean newM_bHasGotoPageButton) Sets whether to display the GotoPageButton.
void	

IDE Extensions

	setHasLogo (boolean newM_bHasLogo) Sets whether to display the Crystal Decisions logo.
void	setHasPageNavigationButtons (boolean newM_bHasPageNavigationButtons) Sets whether to display the page navigation buttons.
void	setHasPrintButton (boolean newM_bHasPrintButton) Sets whether to display the print button.
void	setHasRefreshButton (boolean newM_bHasRefreshButton) Sets whether to display the refresh button.
void	setHasSearchButton (boolean newM_bHasSearchButton) Sets whether to display the search button.
void	setHasToggleGroupTreeButton (boolean newM_bHasToggleButton) Sets whether to display the toggle group tree button.
void	setHasViewList (boolean newM_bHasViewList) Sets whether to display the view list.
void	setHasZoomFactorList (boolean newM_bHasZoomButton) Sets whether to display a zoom factor list.
void	setPageToTreeRatio (double newM_nPageToTreeRatio) Deprecated. Use <i>setGroupTreeWidth(int)</i> and <i>setGroupTreeWidthUnit(com.crystaldecisions.report.web.viewer.CrHtml)</i> instead.
void	setPrintMode (CrPrintMode printMode) Sets whether to print using PDF or ActiveX print mode when the user clicks the print button.
void	setRenderAsHTML32 (boolean newM_bRenderAsHTML32) Sets whether to render the content as HTML 3.2.
void	setSeparatePages (boolean separate) Sets whether the report is displayed as separate pages or one long page.
void	setShowAllPageIds (boolean showAllPageIds) Sets whether PageIds are added to the HTML content.
void	setZoomFactor (int newM_ZoomFactor) Sets a new zoom factor for displaying the report.
void	showFirstPage () Displays the first page of the report.
void	showLastPage () Displays the last page of the report.
void	showNextPage () Displays the next page of the report.
void	showNthPage (int pageNumber) Displays the specified page of the report.
void	showPreviousPage () Displays the previous page of the report.

Methods inherited from class `com.crystaldecisions.report.web.viewer.CrystalReportViewerBase`

```

addAfterRenderContentEventListener, addAfterRenderEventListener,
addAfterRenderObjectEventListener, addAfterRenderViewItemEventListener,
addBeforeRenderContentEventListener, addBeforeRenderEventListener,
addBeforeRenderObjectEventListener,
addBeforeRenderViewItemEventListener, addDrillEventListener,
addOnRenderScriptEventListener, addOnRenderStyleEventListener,
getHyperlinkTarget, removeAfterRenderContentEventListener,
removeAfterRenderEventListener, removeAfterRenderObjectEventListener,
removeAfterRenderViewItemEventListener,
removeBeforeRenderContentEventListener, removeBeforeRenderEventListener,
removeBeforeRenderObjectEventListener,
removeBeforeRenderViewItemEventListener, removeDrillEventListener,
removeOnRenderScriptEventListener, removeOnRenderStyleEventListener,
setHyperlinkTarget

```

Methods inherited from class `com.crystaldecisions.report.web.viewer.ReportServerControl`

```

addReportPartBookmarkNavigationEventListener,
addReportSourceChangeListener, dispose, getDatabaseLogonInfos,
getEnterpriseLogon, getParameterFields, getReportSource,
getReportSourceClassFactoryName, getSelectionFormula,
getStyleSheetFileName, isEnabledLogonPrompt, isEnabledParameterPrompt,
isReuseParameterValuesOnRefresh, navigateTo, refresh,
removeReportPartBookmarkNavigationEventListener,
removeReportPartBookmarkNavigationEventListenerr,
removeReportSourceChangeListener, setDatabaseLogonInfos,
setEnabledLogonPrompt, setEnabledParameterPrompt, setEnterpriseLogon,
setParameterFields, setReportSource, setReportSourceClassFactoryName,
setReuseParameterValuesOnRefresh, setSelectionFormula,
setStyleSheetFileName, setURI, setViewTimeSelectionFormula

```

Methods inherited from class `com.crystaldecisions.report.web.ServerControl`

```

deserializeBase64ToObject, getHeight, getHtmlContent, getLeft, getName,
getTop, getURI, getViewState, getWidth, isIgnoreViewStateOnLoad,
isOwnForm, isOwnPage, processHttpRequest, setHeight,
setIgnoreViewStateOnLoad, setLeft, setName, setOwnForm, setOwnPage,
setTop, setViewState, setWidth

```

Methods inherited from class `java.lang.Object`

```

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait,
wait

```

Field Detail

TOOLBAR_TYPE

```
public static final java.lang.String TOOLBAR_TYPE
```

Specifies view type to be rendered as Toolbar

See Also:

Constant Field Values

TREE_TYPE

```
public static final java.lang.String TREE_TYPE
```

Specifies view type to be rendered as GroupTree

See Also:

Constant Field Values

PAGE_TYPE

```
public static final java.lang.String PAGE_TYPE
```

Specifies view type to be rendered as Page

See Also:

Constant Field Values

Constructor Detail

CrystalReportViewer

```
public CrystalReportViewer()
```

Constructs the CrystalReportViewer object.

Method Detail

addDrillDownSubreportEventListener

```
public void addDrillDownSubreportEventListener(DrillDownSubreportEventListener listener)  
throws java.util.TooManyListenersException
```

Adds a listener for DrillDownSubreportEvents.

Parameters:

listener – The DrillDownSubreportEventListener.

Throws:

`java.util.TooManyListenersException` – This is thrown if you attempt to add more than one listener on a particular listener source concurrently.

See Also:

`removeDrillDownSubreportEventListener`

addNavigateEventListener

```
public void addNavigateEventListener(NavigateEventListener listener)
    throws java.util.TooManyListenersException
```

Adds a listener for `NavigateEvents`.

Parameters:

`listener` – The `NavigateEventsListener`.

Throws:

`java.util.TooManyListenersException` – This is thrown if you attempt to add more than one listener on a particular listener source concurrently.

See Also:

`removeNavigateEventListener`

addToolBarCommandEventListener

```
public void addToolBarCommandEventListener(ToolBarCommandEventListener listener)
    throws java.util.TooManyListenersException
```

Adds a listener for `ToolBarCommandEvents`.

Parameters:

`listener` – The `ToolBarCommandEventListener`.

Throws:

`java.util.TooManyListenersException` – This is thrown if you attempt to add more than one listener on a particular listener source concurrently.

See Also:

`removeToolBarCommandEventListener`

drillDown

```
public void drillDown(IGroupPath groupPath,
    java.lang.String groupName)
```

Drills down on the report.

Parameters:

`groupPath` – An `IGroupPath` object that specifies the path of the group to drill down on.

`groupName` – A `String` that specifies the name of the group to drill down on.

getGroupTreeWidth

```
public int getGroupTreeWidth()
```

Returns the current width of the group tree. This value can be either be specified in pixels or as a percentage, depending on the value returned by `getGroupTreeWidthUnit()`.

Returns:

An int specifying the current width of the group tree.

getGroupTreeWidthUnit

```
public CrHtmlUnitEnum getGroupTreeWidthUnit()
```

Returns whether pixels or percentage is being used to determine the group tree width.

Returns:

A `CrHtmlUnitEnum` object specifying the unit being used to determine group tree width.

See Also:

`CrHtmlUnitEnum`

getPageToTreeRatio

```
public double getPageToTreeRatio()
```

Deprecated. Use `getGroupTreeWidth()` and `getGroupTreeWidthUnit()` instead.

Returns the page-to-tree ratio. This indicates the percentage of the report viewer that will be used to display the page as compared to the tree.

Note: The default value for the ratio is 3, which indicates that the page takes up 3 times more space than the tree, or 75 percent of the space.

Returns:

The page-to-tree ratio as a double.

See Also:

`setPageToTreeRatio`

getToolbar

```
public Toolbar getToolbar()
```

Returns the toolbar object.

Note: This method is provided so that you can see the default toolbar in case you want to modify it.

Returns:

The toolbar object.

getViewInfo

```
public ViewInfo getViewInfo()
```

Returns the ViewInfo object. The ViewInfo object is used to obtain information for the client that is not available from the specific listener that is being used.

For example, if you were using the event handler for the NavigateEventListener, you could use the following to access the group name from the ViewInfo object:

```
public void navigate (NavigateEvents e) {
    String groupName = ((CrystalReportViewer) (e.getSource())).getViewInfo().getGroup
}
```

Returns:

The ViewInfo object.

getZoomFactor

```
public int getZoomFactor()
```

Returns the zoom factor that is currently set.

Returns:

The zoom factor as an int.

See Also:

hasZoomFactorList, setHasZoomFactorList, setZoomFactor

hasExportButton

```
public boolean hasExportButton()
```

Returns whether the report viewer has an export button.

Returns:

true if the viewer has an export button and false otherwise.

See Also:

setHasExportButton

hasGotoPageButton

```
public boolean hasGotoPageButton()
```

Returns whether the report viewer has a "Go to Page" button.

Returns:

true if the viewer has a "Go to Page" button and false otherwise.

See Also:

setHasGotoPageButton

hasLogo

```
public boolean hasLogo()
```

Returns whether the report viewer is set to display the Crystal Decisions logo.

Returns:

true if the viewer is set to display the Crystal Decisions logo and false otherwise.

See Also:

setHasLogo

hasPageNavigationButtons

```
public boolean hasPageNavigationButtons()
```

Returns whether the report viewer has page navigation buttons.

Returns:

true if the viewer has page navigation buttons and false otherwise.

See Also:

setHasPageNavigationButtons

hasPrintButton

```
public boolean hasPrintButton()
```

Returns whether the report viewer has a print button.

Returns:

true if the viewer has a print button and false otherwise.

See Also:

setHasPrintButton

hasRefreshButton

```
public boolean hasRefreshButton()
```

Returns whether the report viewer has a refresh button.

Returns:

true if the viewer has a refresh button and false otherwise.

See Also:

setHasRefreshButton

hasSearchButton

```
public boolean hasSearchButton()
```

Returns whether the report viewer has a search button.

Returns:

true if the viewer has a search button and false otherwise.

See Also:

setHasSearchButton

hasToggleGroupTreeButton

```
public boolean hasToggleGroupTreeButton()
```

Returns whether the report viewer has a button for toggling the display of the group tree.

Returns:

true if the viewer has group tree button and false otherwise.

See Also:

setHasToggleGroupTreeButton(boolean newM_bHasToggleButton)

hasViewList

```
public boolean hasViewList()
```

Returns whether the report viewer has a view list.

Returns:

true if the viewer has a view list and false otherwise.

See Also:

setHasViewList

hasZoomFactorList

```
public boolean hasZoomFactorList()
```

Returns whether the report viewer has a zoom factor list.

Returns:

true if the viewer has a zoom factor list and false otherwise.

See Also:

getZoomFactor, setHasZoomFactorList, setZoomFactor

isBestFitPage

```
public boolean isBestFitPage()
```

Returns whether the viewer will ignore the height and width values it is given and try to find the best fit for the report by taking up as much space as it needs to properly display the report.

Returns:

true if the viewer will try to find the best fit for the report, otherwise false if the viewer will use the height and width values it is given.

See Also:

setBestFitPage

isDisplayGroupTree

```
public boolean isDisplayGroupTree()
```

Returns whether the group tree is displayed.

Returns:

true if the group tree is displayed and false otherwise.

See Also:

setDisplayGroupTree

isDisplayPage

```
public boolean isDisplayPage()
```

Returns whether the report page is displayed.

Note: If isDisplayPage is false then isDisplayToolbar is also false.

Returns:

true if the report page is displayed and false otherwise.

See Also:

setDisplayPage

isDisplayToolbar

```
public boolean isDisplayToolbar()
```

Returns whether the toolbar is displayed.

Note: If isDisplayPage is false then isDisplayToolbar will also be false.

Returns:

true if the toolbar is displayed and false otherwise.

See Also:

setDisplayToolbar

isEnabledDrillDown

```
public boolean isEnabledDrillDown()
```

Returns whether the user can drill down on the report.

Returns:

true if the user can drill down on the report and false otherwise.

See Also:

setEnabledDrillDown

isRenderAsHTML32

```
public boolean isRenderAsHTML32()
```

Returns whether the report is rendered as HTML 3.2.

Returns:

true if the report is rendered as HTML 3.2 and false otherwise.

See Also:

setRenderAsHTML32

isSeparatePages

```
public boolean isSeparatePages()
```

Returns whether the report is displayed as separate pages or one long page. It is not recommended that you set to this false when displaying longer reports as the report will not display until the report has been fully loaded. When set to false searchText will not be available.

Returns:

true if the report is displayed as separate pages and false otherwise.

See Also:

setSeparatePages

isShowAllPageIds

```
public boolean isShowAllPageIds()
```

Returns whether PageIds are added to the HTML content. If page IDs are added to the HTML additional content, they will be added to the beginning of the HTML div tags to identify the page.

Returns:

true if the additional PageIds are added to the HTML content and false otherwise.

See Also:

setShowAllPageIds

removeDrillDownSubreportEventListener

```
public void removeDrillDownSubreportEventListener()
```

Removes a DrillDownSubreportEventListener.

See Also:

addDrillDownSubreportEventListener

removeNavigateEventListener

```
public void removeNavigateEventListener()
```

Removes a NavigateEventListener.

See Also:

`addNavigateEventListener`

removeToolBarCommandEventListener

```
public void removeToolBarCommandEventListener()
```

Removes a `ToolBarCommandEventListener`.

See Also:

`addToolBarCommandEventListener`

searchText

```
public void searchText(java.lang.String strTextToSearch,  
                        int searchDirection)
```

Highlights the first occurrence of the specified text in the report and scrolls to it.

Note: The text search begins on the following page. To search the first page of a report or to search a report that is displayed continuously (that is, where `setSeparatePages` is set to `false`), you must use the browser's search function.

Parameters:

`strTextToSearch` – A `String` that specifies the text string to search for in the report.

`searchDirection` – An `int` that specifies the search direction. Currently, the only available value is 0 for forward.

setBestFitPage

```
public void setBestFitPage(boolean bestFitPage)
```

Sets whether the viewer will ignore the height and width values it is given and try to find the best fit for the report by taking up as much space as it needs to properly display the report.

Parameters:

`bestFitPage` – `true` to find the best fit for the report, otherwise `false` to use the height and width values it is given.

See Also:

`isBestFitPage`

setDisplayGroupTree

```
public void setDisplayGroupTree(boolean newM_bDisplayGroupTree)
```

Sets whether to display the group tree.

Parameters:

`newM_bDisplayGroupTree` – `true` to display the group tree.

See Also:

isDisplayGroupTree

setDisplayPage

```
public void setDisplayPage(boolean newM_bDisplayPage)
```

Sets whether to display the report page.

Parameters:

newM_bDisplayPage – true to display the report page.

See Also:

isDisplayPage

setDisplayToolbar

```
public void setDisplayToolbar(boolean display)
```

Sets whether to display the toolbar.

Parameters:

display – true to display the toolbar.

See Also:

isDisplayToolbar

setEnabledDrillDown

```
public void setEnabledDrillDown(boolean enableDrillDown)
```

Sets whether to enable drill down.

Parameters:

enableDrillDown – true to enable drill down.

See Also:

isEnabledDrillDown

setGroupTreeWidth

```
public void setGroupTreeWidth(int newGroupTreeWidth)
```

Sets the width of the group tree. This value can be either be specified in pixels or as a percentage, by passing the appropriate CrHtmlUnitEnum object to
 setGroupTreeWidthUnit(com.crystaldecisions.report.web.viewer.CrHtmlUnitEnum)

Parameters:

newGroupTreeWidth – An int specifying the new width of the group tree.

setGroupTreeWidthUnit

```
public void setGroupTreeWidthUnit(CrHtmlUnitEnum newGroupTreeWidthUnit)
```

Sets whether to use pixels or percentage to determine the group tree width.

Parameters:

`newGroupTreeWidthUnit` – A `CrHtmlUnitEnum` object specifying the unit to use to determine group tree width.

See Also:

`CrHtmlUnitEnum`

setPrintMode

```
public void setPrintMode(CrPrintMode printMode)
```

Sets whether to print using PDF or ActiveX print mode when the user clicks the print button. In PDF print mode, a PDF will be displayed, allowing the user to then print it. In ActiveX print mode, an ActiveX control will be downloaded to the client machine and will be printed directly to the user. If ActiveX print mode is selected on a system that does not support ActiveX controls, the print mode will default to PDF printing.

Parameters:

`printMode` – A `CrPrintMode` object specifying the print mode to be used.

See Also:

`getPrintMode()`

getPrintMode

```
public CrPrintMode getPrintMode()
```

Returns the current print mode that will be used when the user clicks the print button.

Returns:

A `CrPrintMode` object specifying the print mode currently being used.

See Also:

`setPrintMode(com.crystaldecisions.report.web.viewer.CrPrintMode)`

setHasExportButton

```
public void setHasExportButton(boolean newM_bHasExportButton)
```

Sets whether to display the export button.

Note: `setOwnPage` must be set to true or the button is not rendered, regardless of the value set in the `setHasExportButton` method.

Parameters:

`newM_bHasExportButton` – true to display the export button.

See Also:

`hasExportButton`

setHasGotoPageButton

```
public void setHasGotoPageButton(boolean newM_bHasGotoPageButton)
```

Sets whether to display the GotoPageButton.

Note: If true, the HTML returned includes a text box, into which the user can type a page number to navigate to a specific page in the report. If false, the HTML returned displays the current page, but does not allow the user to navigate by page number. The default value is true.

Parameters:

newM_bHasGotoPageButton – true to display the GotoPageButton button.

See Also:

hasGotoPageButton

setHasLogo

```
public void setHasLogo(boolean newM_bHasLogo)
```

Sets whether to display the Crystal Decisions logo.

Parameters:

newM_bHasLogo – true to display the Crystal Decisions logo.

See Also:

hasLogo

setHasPageNavigationButtons

```
public void setHasPageNavigationButtons(boolean newM_bHasPageNavigationButtons)
```

Sets whether to display the page navigation buttons.

Parameters:

newM_bHasPageNavigationButtons – true to display page navigation buttons.

See Also:

hasPageNavigationButtons

setHasPrintButton

```
public void setHasPrintButton(boolean newM_bHasPrintButton)
```

Sets whether to display the print button.

Note: Printing is accomplished by automatically exporting the report to PDF and then printing the report from the PDF viewer.

Parameters:

newM_bHasPrintButton – true to display the print button.

See Also:

hasPrintButton

setHasRefreshButton

```
public void setHasRefreshButton(boolean newM_bHasRefreshButton)
```

Sets whether to display the refresh button.

Parameters:

newM_bHasRefreshButton – true to display the refresh button.

See Also:

hasRefreshButton

setHasSearchButton

```
public void setHasSearchButton(boolean newM_bHasSearchButton)
```

Sets whether to display the search button.

Parameters:

newM_bHasSearchButton – true to display the search button.

See Also:

hasSearchButton

setHasToggleGroupTreeButton

```
public void setHasToggleGroupTreeButton(boolean newM_bHasToggleButton)
```

Sets whether to display the toggle group tree button.

Parameters:

newM_bHasToggleButton – true to display the toggle button.

See Also:

hasToggleGroupTreeButton

setHasViewList

```
public void setHasViewList(boolean newM_bHasViewList)
```

Sets whether to display the view list.

Note: The view list will include the Main Report and any views you have drilled-down into. This can include groups, charts and subreports.

Parameters:

newM_bHasViewList – true to display the view list button.

See Also:

hasViewList

setHasZoomFactorList

```
public void setHasZoomFactorList(boolean newM_bHasZoomButton)
```

Sets whether to display a zoom factor list.

Parameters:

newM_bHasZoomButton – true to display a zoom factor list.

See Also:

getZoomFactor, hasZoomFactorList, setZoomFactor

setPageToTreeRatio

```
public void setPageToTreeRatio(double newM_nPageToTreeRatio)
```

Deprecated. Use `setGroupTreeWidth(int)` and `setGroupTreeWidthUnit(com.crystaldecisions.report.web.viewer.CrHtmlUnitEnum)` instead.

Sets the page-to-tree ratio. This indicates the percentage of the report viewer that will be used to display the page as compared to the tree.

Note: The default value for the ratio is 3, which indicates that the page takes up 3 times more space than the tree, or 75 percent of the space.

Parameters:

newM_nPageToTreeRatio – A double that specifies the page-to-tree ratio.

See Also:

getPageToTreeRatio

setRenderAsHTML32

```
public void setRenderAsHTML32(boolean newM_bRenderAsHTML32)
```

Sets whether to render the content as HTML 3.2.

Parameters:

newM_bRenderAsHTML32 – true to display the content as HTML 3.2.

See Also:

isRenderAsHTML32

setSeparatePages

```
public void setSeparatePages(boolean separate)
```

Sets whether the report is displayed as separate pages or one long page. It is not recommended that you set to this `false` when displaying longer reports as the report will not display until the report has been fully loaded. When set to `false` `searchText` will not be available.

Parameters:

separate – true if the report is displayed as separate pages.

See Also:

isSeparatePages

setShowAllPageIds

```
public void setShowAllPageIds(boolean showAllPageIds)
```

Sets whether PageIds are added to the HTML content. If page IDs are added to the HTML additional content, they will be added to the beginning of the HTML div tags to identify the page.

Parameters:

showAllPageIds – true if the additional PageIds are added to the HTML content.

See Also:

isShowAllPageIds

setZoomFactor

```
public void setZoomFactor(int newM_ZoomFactor)
```

Sets a new zoom factor for displaying the report. Acceptable values for the zoom factor range from 10 to 400 percent.

Parameters:

newM_ZoomFactor – An int that specifies the value for the zoom factor.

See Also:

getZoomFactor, hasZoomFactorList, setHasZoomFactorList

showFirstPage

```
public void showFirstPage()
```

Displays the first page of the report.

See Also:

showLastPage, showNextPage, showNthPage, showPreviousPage

showLastPage

```
public void showLastPage()
```

Displays the last page of the report.

See Also:

showFirstPage, showNextPage, showNthPage, showPreviousPage

showNextPage

```
public void showNextPage()
```

Displays the next page of the report.

Note: If the last page of the report is currently displayed, the viewer remains on this page.

See Also:

`showFirstPage`, `showLastPage`, `showNthPage`, `showPreviousPage`

showNthPage

```
public void showNthPage(int pageNumber)
```

Displays the specified page of the report.

Note: If the report has no Nth page, the last page of the report is displayed.

Parameters:

`pageNumber` – An `int` that specifies the page of the report to be displayed.

See Also:

`showFirstPage`, `showLastPage`, `showNextPage`, `showPreviousPage`

showPreviousPage

```
public void showPreviousPage()
```

Displays the previous page of the report.

Note: If the first page of the report is currently displayed, the viewer remains on this page.

See Also:

`showFirstPage`, `showLastPage`, `showNextPage`, `showNthPage`

com.crystaldecisions.report.web.viewer Class CrystalReportViewerBase

```
java.lang.Object
├── com.crystaldecisions.report.web.ServerControl
│   ├── com.crystaldecisions.report.web.viewer.ReportServerControl
│   │   └── com.crystaldecisions.report.web.viewer.CrystalReportViewerBase
```

Direct Known Subclasses:

`CrystalReportViewer`

```
public abstract class CrystalReportViewerBase
    extends ReportServerControl
```

This class provides listeners for the report viewer, as well as providing methods for getting and setting hyperlink targets.

Field Summary

IDE Extensions

static int	BACKWARD Reserved for future use.
static int	FORWARD Search direction as forward.
static int	NOACTION Reserved for future use.

Constructor Summary

CrystalReportViewerBase()

Method Summary

void	addAfterRenderContentEventListener (AfterRenderContentEventListener listener) Adds a listener for AfterRenderContentEvents.
void	addAfterRenderEventListener (AfterRenderEventListener listener) Adds a listener for AfterRenderEvents.
void	addAfterRenderObjectEventListener (AfterRenderObjectEventListener listener) Adds a listener for AfterRenderObjectEvents.
void	addAfterRenderViewItemEventListener (IAfterRenderViewItemEventListener listener) Adds a listener for AfterRenderViewItemEvents.
void	addBeforeRenderContentEventListener (BeforeRenderContentEventListener listener) Adds a listener for BeforeRenderContentEvents.
void	addBeforeRenderEventListener (BeforeRenderEventListener listener) Adds a listener for BeforeRenderEvents.
void	addBeforeRenderObjectEventListener (BeforeRenderObjectEventListener listener) Adds a listener for BeforeRenderObjectEvents.
void	addBeforeRenderViewItemEventListener (IBeforeRenderViewItemEventListener listener) Adds a listener for BeforeRenderViewItemEvents.
void	addDrillEventListener (DrillEventListener listener) Adds a listener for DrillEvents.
void	addOnRenderScriptEventListener (OnRenderScriptEventListener listener) Adds a listener for OnRenderScriptEvents.
void	addOnRenderStyleEventListener (OnRenderStyleEventListener listener) Adds a listener for OnRenderStyleEvents.
java.lang.String	getHyperlinkTarget() Returns the hyperlink target for displaying the HTML.
void	removeAfterRenderContentEventListener() Removes the listener for AfterRenderContentEvents.
void	

IDE Extensions

	removeAfterRenderEventListener() Removes the listener for AfterRenderEvents.
void	removeAfterRenderObjectEventListener() Removes the listener for AfterRenderObjectEvents.
void	removeAfterRenderViewItemEventListener() Removes the listener for AfterRenderViewItemEvents.
void	removeBeforeRenderContentEventListener() Removes the listener for BeforeRenderContentEvents.
void	removeBeforeRenderEventListener() Removes the listener for BeforeRenderEvents.
void	removeBeforeRenderObjectEventListener() Removes the listener for BeforeRenderObjectEvents.
void	removeBeforeRenderViewItemEventListener() Removes the listener for BeforeRenderViewItemEvents.
void	removeDrillEventListener() Removes the listener for DrillEvents.
void	removeOnRenderScriptEventListener() Removes the listener for OnRenderScriptEvents.
void	removeOnRenderStyleEventListener() Removes the listener for OnRenderStyleEvents.
void	setHyperlinkTarget (java.lang.String target) Sets the hyperlink target for displaying the HTML.

Methods inherited from class com.crystaldecisions.report.web.viewer.ReportServerControl

addReportPartBookmarkNavigationEventListener,
addReportSourceChangeListener, dispose, getDatabaseLogonInfos,
getEnterpriseLogon, getParameterFields, getReportSource,
getReportSourceClassFactoryName, getSelectionFormula,
getStyleSheetFileName, isEnableLogonPrompt, isEnableParameterPrompt,
isReuseParameterValuesOnRefresh, navigateTo, refresh,
removeReportPartBookmarkNavigationEventListener,
removeReportPartBookmarkNavigationEventListennerr,
removeReportSourceChangeListener, setDatabaseLogonInfos,
setEnableLogonPrompt, setEnableParameterPrompt, setEnterpriseLogon,
setParameterFields, setReportSource, setReportSourceClassFactoryName,
setReuseParameterValuesOnRefresh, setSelectionFormula,
setStyleSheetFileName, setURI, setViewTimeSelectionFormula

Methods inherited from class com.crystaldecisions.report.web.ServerControl

deserializeBase64ToObject, getHeight, getHtmlContent, getLeft, getName,
getTop, getURI, getViewState, getWidth, isIgnoreViewStateOnLoad,
isOwnForm, isOwnPage, processHttpRequest, setHeight,
setIgnoreViewStateOnLoad, setLeft, setName, setOwnForm, setOwnPage,
setTop, setViewState, setWidth

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail**FORWARD**

```
public static final int FORWARD
```

Search direction as forward.

See Also:

Constant Field Values

BACKWARD

```
public static final int BACKWARD
```

Reserved for future use.

See Also:

Constant Field Values

NOACTION

```
public static final int NOACTION
```

Reserved for future use.

See Also:

Constant Field Values

Constructor Detail**CrystalReportViewerBase**

```
public CrystalReportViewerBase()
```

Method Detail

addAfterRenderContentEventListener

```
public void addAfterRenderContentEventListener(AfterRenderContentEventListener listener)
        throws java.util.TooManyListenersException
```

Adds a listener for AfterRenderContentEvents. The event is fired after rendering the body of the report is complete.

Parameters:

listener – The AfterRenderContentEvent listener.

Throws:

TooManyListenersException – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

removeAfterRenderContentEventListener()

addAfterRenderEventListener

```
public void addAfterRenderEventListener(AfterRenderEventListener listener)
        throws java.util.TooManyListenersException
```

Adds a listener for AfterRenderEvents. The event is fired after the rendering process is complete.

Parameters:

listener – The AfterRenderEvent listener.

Throws:

TooManyListenersException – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

removeAfterRenderEventListener()

addAfterRenderObjectEventListener

```
public void addAfterRenderObjectEventListener(AfterRenderObjectEventListener listener)
        throws java.util.TooManyListenersException
```

Adds a listener for AfterRenderObjectEvents. The event is fired after rendering the report object.

Parameters:

listener – The AfterRenderObjectEvent listener.

Throws:

TooManyListenersException – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

removeAfterRenderObjectEventListener()

addAfterRenderViewItemEventListener

```
public void addAfterRenderViewItemEventListener(IAfterRenderViewItemEventListener listener)
    throws java.util.TooManyListenersException
```

Adds a listener for AfterRenderViewItemEvents. The event is fired after rendering a viewer item (Toolbar, GroupTree, or Page).

Parameters:

listener – The AfterRenderViewItemEvent listener.

Throws:

TooManyListenersException – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

removeAfterRenderViewItemEventListener()

addBeforeRenderContentEventListener

```
public void addBeforeRenderContentEventListener(BeforeRenderContentEventListener listener)
    throws java.util.TooManyListenersException
```

Adds a listener for BeforeRenderContentEvents. The event is fired before rendering the body of the report is complete.

Parameters:

listener – The BeforeRenderContentEvent listener.

Throws:

TooManyListenersException – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

removeBeforeRenderContentEventListener()

addBeforeRenderEventListener

```
public void addBeforeRenderEventListener(BeforeRenderEventListener listener)
    throws java.util.TooManyListenersException
```

Adds a listener for BeforeRenderEvents. The event is fired before the rendering process begins.

Parameters:

listener – The BeforeRenderEvent listener.

Throws:

TooManyListenersException – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

removeBeforeRenderEventListener()

addBeforeRenderObjectEventListener

```
public void addBeforeRenderObjectEventListener(BeforeRenderObjectEventListener listener)
    throws java.util.TooManyListenersException
```

Adds a listener for `BeforeRenderObjectEvents`. The event is fired before rendering the report object.

Parameters:

listener – The `BeforeRenderObjectEvent` listener.

Throws:

`TooManyListenersException` – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

`removeBeforeRenderObjectEventListener()`

addBeforeRenderViewItemEventListener

```
public void addBeforeRenderViewItemEventListener(IBeforeRenderViewItemEventListener listener)
        throws java.util.TooManyListenersException
```

Adds a listener for `BeforeRenderViewItemEvents`. The event is fired before rendering a viewer item (Toolbar, GroupTree, or Page).

Parameters:

listener – The `BeforeRenderViewItemEvent` listener.

Throws:

`TooManyListenersException` – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

`removeBeforeRenderViewItemEventListener()`

addDrillEventListener

```
public void addDrillEventListener(DrillEventListener listener)
        throws java.util.TooManyListenersException
```

Adds a listener for `DrillEvents`. The event is fired on drill down.

Parameters:

listener – The `DrillEvent` listener.

Throws:

`TooManyListenersException` – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

`removeDrillEventListener`

addOnRenderScriptEventListener

```
public void addOnRenderScriptEventListener(OnRenderScriptEventListener listener)
        throws java.util.TooManyListenersException
```

Adds a listener for `OnRenderScriptEvents`. The event is fired before rendering the "SCRIPT" block.

Parameters:

listener – The `OnRenderScriptEvent` listener.

Throws:

`TooManyListenersException` – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

`removeOnRenderScriptEventListener()`

addOnRenderStyleEventListener

```
public void addOnRenderStyleEventListener(OnRenderStyleEventListener listener)
        throws java.util.TooManyListenersException
```

Adds a listener for `OnRenderStyleEvents`. The event is fired before rendering the "STYLE" block.

Parameters:

`listener` – `OnRenderStyleEvents` listener.

Throws:

`TooManyListenersException` – This is thrown if you attempt to add more than one listener on a particular event listener source concurrently.

See Also:

`removeOnRenderStyleEventListener()`

getHyperlinkTarget

```
public java.lang.String getHyperlinkTarget()
```

Returns the hyperlink target for displaying the HTML.

The target `_self` displays the HTML document in the same frame, the target `_parent` displays the HTML document in the same frame or window that contains the current frameset, the target `_top` displays the HTML document in the entire browser window, and the target `_blank` to displays the HTML document in a new browser window.

Returns:

The hyperlink target as a `String`. By default the target is `_self`.

See Also:

`setHyperlinkTarget`

removeAfterRenderContentEventListener

```
public void removeAfterRenderContentEventListener()
```

Removes the listener for `AfterRenderContentEvents`.

See Also:

`addAfterRenderContentEventListener(com.crystaldecisions.report.htmlre`

removeAfterRenderEventListener

```
public void removeAfterRenderEventListener()
```

Removes the listener for AfterRenderEvents.

See Also:

```
addAfterRenderEventListener(com.crystaldecisions.report.htmlrender.A
```

removeAfterRenderObjectEventListener

```
public void removeAfterRenderObjectEventListener()
```

Removes the listener for AfterRenderObjectEvents.

See Also:

```
addAfterRenderObjectEventListener(com.crystaldecisions.report.htmlren
```

removeAfterRenderViewItemEventListener

```
public void removeAfterRenderViewItemEventListener()
```

Removes the listener for AfterRenderViewItemEvents.

See Also:

```
addAfterRenderViewItemEventListener(com.crystaldecisions.report.html
```

removeBeforeRenderContentEventListener

```
public void removeBeforeRenderContentEventListener()
```

Removes the listener for BeforeRenderContentEvents.

See Also:

```
addBeforeRenderContentEventListener(com.crystaldecisions.report.html
```

removeBeforeRenderEventListener

```
public void removeBeforeRenderEventListener()
```

Removes the listener for BeforeRenderEvents.

See Also:

```
addBeforeRenderEventListener(com.crystaldecisions.report.htmlrender.I
```

removeBeforeRenderObjectEventListener

```
public void removeBeforeRenderObjectEventListener()
```

Removes the listener for BeforeRenderObjectEvents.

See Also:

`addBeforeRenderObjectEventListener(com.crystaldecisions.report.htmlr`

removeBeforeRenderViewItemEventListener

```
public void removeBeforeRenderViewItemEventListener()
```

Removes the listener for BeforeRenderViewItemEvents.

See Also:

`addBeforeRenderViewItemEventListener(com.crystaldecisions.report.html`

removeDrillEventListener

```
public void removeDrillEventListener()
```

Removes the listener for DrillEvents.

See Also:

`addDrillEventListener`

removeOnRenderScriptEventListener

```
public void removeOnRenderScriptEventListener()
```

Removes the listener for OnRenderScriptEvents.

See Also:

`addOnRenderScriptEventListener(com.crystaldecisions.report.htmlrender`

removeOnRenderStyleEventListener

```
public void removeOnRenderStyleEventListener()
```

Removes the listener for OnRenderStyleEvents.

See Also:

`addOnRenderStyleEventListener(com.crystaldecisions.report.htmlrender`

setHyperlinkTarget

```
public void setHyperlinkTarget(java.lang.String target)
```

Sets the hyperlink target for displaying the HTML.

Use the target `_self` to display the HTML document in the same frame, `_parent` to display HTML document in the same frame or window that contains the current frameset, `_top` to display HTML document in the entire browser window, and `_blank` to display HTML document in a new browser window.

Parameters:

`target` – A String that specifies the target for the hyperlink. By default the target is `_self`.

See Also:

`getHyperlinkTarget`

com.crystaldecisions.report.web.viewer

Class DrillDownSubreportEventArgs

```
java.lang.Object
├─ java.util.EventObject
│   └─ com.crystaldecisions.report.web.viewer.ViewerEventArgs
│       └─ com.crystaldecisions.report.web.viewer.DrillDownSubreportEventArgs
```

All Implemented Interfaces:

`java.io.Serializable`

```
public class DrillDownSubreportEventArgs
extends ViewerEventArgs
```

This class provides arguments for the Subreport DrillDown Event listener.

See Also:

Serialized Form

Constructor Summary

```
DrillDownSubreportEventArgs( java.lang.Object source,
SubreportRequestContext oldSubreportContext,
SubreportRequestContext newSubreportContext)
```

Method Summary

<code>java.lang.String</code>	getCurrentSubreportName () Returns the name of the current subreport.
<code>int</code>	getCurrentSubreportPageNumber () Returns the page number for the current subreport.
<code>int</code>	getCurrentXOffset () Returns the current X offset.
<code>int</code>	getCurrentYOffset () Returns the current Y offset.
<code>java.lang.String</code>	getNewSubreportName () Returns the new subreport name.

IDE Extensions

int	getNewSubreportPageNumber () Returns the new subreport page number.
int	getNewXOffset () Returns the new X offset.
int	getNewYOffset () Returns the new Y offset.

Methods inherited from class `com.crystaldecisions.report.web.viewer.ViewerEventArgs`

`isHandled`, `setHandled`

Methods inherited from class `java.util.EventObject`

`getSource`, `toString`

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Detail

DrillDownSubreportEventArgs

```
public DrillDownSubreportEventArgs(java.lang.Object source,  
                                   SubreportRequestContext oldSubreportContext,  
                                   SubreportRequestContext newSubreportContext)
```

Method Detail

getCurrentSubreportName

```
public java.lang.String getCurrentSubreportName()
```

Returns the name of the current subreport.

Returns:

The name of the current subreport as a `String`.

getCurrentSubreportPageNumber

```
public int getCurrentSubreportPageNumber()
```

Returns the page number for the current subreport.

Returns:

The page number for the current subreport as an `int`.

getCurrentXOffset

```
public int getCurrentXOffset()
```

Returns the current X offset. The unit is browser-dependent.

Returns:

The value of the current X offset as an `int`.

getCurrentYOffset

```
public int getCurrentYOffset()
```

Returns the current Y offset. The unit is browser-dependent.

Returns:

The value of the current Y offset as an `int`.

getNewSubreportName

```
public java.lang.String getNewSubreportName()
```

Returns the new subreport name.

Returns:

The name of the new subreport as a `String`.

getNewSubreportPageNumber

```
public int getNewSubreportPageNumber()
```

Returns the new subreport page number.

Returns:

The new subreport page number as an `int`.

getNewXOffset

```
public int getNewXOffset()
```

Returns the new X offset. The unit is browser-dependent.

Returns:

The value of the new X offset as an `int`.

getNewYOffset

```
public int getNewYOffset()
```

Returns the new Y offset. The unit is browser-dependent.

Returns:

The value of the new X offset as an `int`.

com.crystaldecisions.report.web.viewer **Interface DrillDownSubreportEventListener**

All Superinterfaces:

`java.util.EventListener`

```
public interface DrillDownSubreportEventListener  
extends java.util.EventListener
```

Listener for `DrillDownSubreportEvents`.

Method Summary	
void	drillDownSubreport (<code>DrillDownSubreportEventArgs e</code>) Listener for <code>DrillDownSubreportEvents</code> .

Method Detail

drillDownSubreport

```
public void drillDownSubreport(DrillDownSubreportEventArgs e)
```

Listener for `DrillDownSubreportEvents`. These events occur when the user clicks on a report object in a subreport or when a different view is selected from the view list.

Parameters:

`e` – A `DrillDownSubreportEventArgs` object.

See Also:

`DrillDownSubreportEventArgs`

com.crystaldecisions.report.web.viewer **Class DrillEventArgs**

```
java.lang.Object  
└─ java.util.EventObject
```

```

└─ com.crystaldecisions.report.web.viewer.ViewerEventArgs
    └─ com.crystaldecisions.report.web.viewer.DrillEventArgs

```

All Implemented Interfaces:

java.io.Serializable

public class **DrillEventArgs**
extends **ViewerEventArgs**

This class provides event arguments for the `DrillEventListener`.

See Also:

Serialized Form

Constructor Summary

DrillEventArgs(java.lang.Object source)

DrillEventArgs(java.lang.Object source,
com.crystaldecisions.sdk.occa.report.reportsource.TotallerNodeID newNode)

Method Summary

java.lang.String	getGroupName () Returns the group name for the object drilled on.
java.lang.String	getGroupNamePath () Returns the group name path for the object drilled on.
GroupPath	getGroupPath () Returns the group path for the object drilled on.

Methods inherited from class com.crystaldecisions.report.web.viewer.ViewerEventArgs

isHandled, setHandled

Methods inherited from class java.util.EventObject

getSource, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

DrillEventArgs

```
public DrillEventArgs(java.lang.Object source)
```

DrillEventArgs

```
public DrillEventArgs(java.lang.Object source,  
    com.crystaldecisions.sdk.occa.report.reportsource.TotallerNodeID newNode)
```

Method Detail

getGroupName

```
public java.lang.String getGroupName()
```

Returns the group name for the object drilled on.

Returns:

The group name as a String.

getGroupNamePath

```
public java.lang.String getGroupNamePath()
```

Returns the group name path for the object drilled on.

Returns:

The group name path as a String.

getGroupPath

```
public GroupPath getGroupPath()
```

Returns the group path for the object drilled on.

Returns:

The GroupPath as an object.

com.crystaldecisions.report.web.viewer Interface DrillEventListener

All Superinterfaces:

java.util.EventListener

*public interface **DrillEventListener***
*extends **java.util.EventListener***

Listener for DrillEvents.

Method Summary

void	drill (DrillEventArgs e) Listener for DrillEvent.
------	---

Method Detail

drill

`public void drill(DrillEventArgs e)`

Listener for DrillEvent. A DrillEvent occurs when a report object is clicked. This event is also fired when a different view is selected from the view list.

Parameters:

e – A DrillEventArgs object.

See Also:

DrillEventArgs

com.crystaldecisions.report.web.viewer

Class NavigateEventArgs

```
java.lang.Object
├─ java.util.EventObject
│   └─ com.crystaldecisions.report.web.viewer.ViewerEventArgs
│       └─ com.crystaldecisions.report.web.viewer.NavigateEventArgs
```

All Implemented Interfaces:

java.io.Serializable

*public class **NavigateEventArgs***
*extends **ViewerEventArgs***

This class provides event arguments for the NavigateEventListener.

See Also:

Serialized Form

Constructor Summary

NavigateEventArgs(java.lang.Object source, int newPageN)

Method Summary

int	getPageNumber () Gets the current page number.
-----	---

Methods inherited from class com.crystaldecisions.report.web.viewer.ViewerEventArgs

isHandled, setHandled

Methods inherited from class java.util.EventObject

getSource, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

NavigateEventArgs

```
public NavigateEventArgs(java.lang.Object source,
                        int newPageN)
```

Method Detail

getPageNumber

```
public int getPageNumber()
```

Gets the current page number.

Returns:

The current page number as an int.

com.crystaldecisions.report.web.viewer Interface NavigateEventListener

All Superinterfaces:

java.util.EventListener

*public interface **NavigateEventListener***
extends java.util.EventListener

Listener for NavigateEvents.

Method Summary

void	navigate (NavigateEventArgs e) Listener for NavigateEvents.
------	---

Method Detail

navigate

```
public void navigate(NavigateEventArgs e)
```

Listener for NavigateEvents.

Parameters:

e – A NavigateEventArgs object.

See Also:

NavigateEventArgs

com.crystaldecisions.report.web.viewer Class ReportExportControl

```
java.lang.Object
├── com.crystaldecisions.report.web.ServerControl
│   ├── com.crystaldecisions.report.web.viewer.ReportServerControl
│   │   └── com.crystaldecisions.report.web.viewer.ReportExportControl
```

*public class **ReportExportControl***
extends ReportServerControl

This class provides exporting functionality including the ability to preview an exported report or save an exported report to disk.

The following example includes methods from the `ReportExportControl` class, `com.crystaldecisions.sdk.occa.report.exportoptions.PDFExportFormatOptions` class, and the `com.crystaldecisions.sdk.occa.report.exportoptions.ExportOptions` class.

Example:

Java Report Export Control Example

Constructor Summary

<code>ReportExportControl()</code>

Method Summary

void	setExportAsAttachment (boolean bAsAttachment) Sets whether to export the files as an attachment.
void	setExportOptions (com.crystaldecisions.sdk.occa.report.exportoptions.ExportOptions exportOptions) Sets the export options and the export format for the report.

Methods inherited from class com.crystaldecisions.report.web.viewer.ReportServerControl

```
addReportPartBookmarkNavigationEventListener,
addReportSourceChangeListener, dispose, getDatabaseLogonInfos,
getEnterpriseLogon, getParameterFields, getReportSource,
getReportSourceClassFactoryName, getSelectionFormula,
getStyleSheetFileName, isEnableLogonPrompt, isEnableParameterPrompt,
isReuseParameterValuesOnRefresh, navigateTo, refresh,
removeReportPartBookmarkNavigationEventListener,
removeReportPartBookmarkNavigationEventListenerr,
removeReportSourceChangeListener, setDatabaseLogonInfos,
setEnableLogonPrompt, setEnableParameterPrompt, setEnterpriseLogon,
setParameterFields, setReportSource, setReportSourceClassFactoryName,
setReuseParameterValuesOnRefresh, setSelectionFormula,
setStyleSheetFileName, setURI, setViewTimeSelectionFormula
```

Methods inherited from class com.crystaldecisions.report.web.ServerControl

```
deserializeBase64ToObject, getHeight, getHtmlContent, getLeft, getName,
getTop, getURI, getViewState, getWidth, isIgnoreViewStateOnLoad,
isOwnForm, isOwnPage, processHttpRequest, setHeight,
setIgnoreViewStateOnLoad, setLeft, setName, setOwnForm, setOwnPage,
setTop, setViewState, setWidth
```

Methods inherited from class java.lang.Object

```
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait,
wait
```

Constructor Detail

ReportExportControl

```
public ReportExportControl()
```

Method Detail

setExportOptions

```
public void setExportOptions(com.crystaldecisions.sdk.occa.report.exportoptions.ExportOptions e
```

Sets the export options and the export format for the report. The export options can also include the page range.

Note: You must provide a valid value for this method.

Parameters:

expOpt – A

com.crystaldecisions.sdk.occa.report.exportoptions.ExportOptions
object that specifies the export options and the export format for the report.

setExportAsAttachment

```
public void setExportAsAttachment(boolean bAsAttachment)
```

Sets whether to export the files as an attachment.

Note: The default value for this method is `false`. If you set this to `true` you will be provided with a dialog box that allows you to export the report to disk. If the value is `false` then the report will export and display in the browser.

Parameters:

bAsAttachment – `true` to export the files as attachments.

com.crystaldecisions.report.web.viewer

Class ReportServerControl

```
java.lang.Object
```

```
└ com.crystaldecisions.report.web.ServerControl
```

```
└ com.crystaldecisions.report.web.viewer.ReportServerControl
```


Direct Known Subclasses:

CrystalReportViewerBase, ReportExportControl

*public class **ReportServerControl***
extends ServerControl

This class allows you to manage how reports interact with the server. There are methods for manipulating the report's logons, parameters, and selection formulas.

The following example includes methods from the ReportServerControl class, the com.crystaldecisions.sdk.occa.report.data.Fields class, the com.crystaldecisions.sdk.occa.report.data.ParameterField class, the com.crystaldecisions.sdk.occa.report.data.ParameterFieldDiscreteValue class, the com.crystaldecisions.sdk.occa.report.data.Values class, and the CrystalReportViewer class.

Example:

Java Report Parameters Example

Constructor Summary

ReportServerControl ()

Method Summary

void	addReportPartBookmarkNavigationEventListener (IReportPartBookmarkNavigationEventArgs) Adds a listener for ReportPartBookmarkNavigationEventArgs.
void	addReportSourceChangeEvent Listener (ReportSourceChangeEvent) Adds a listener for report source changes.
void	dispose () Disposes of the ReportServerControl.
ConnectionInfos	getDatabaseLogonInfos () Returns the information required to log on to the database.
java.lang.Object	getEnterpriseLogon () Returns the information required to log on to Crystal Enterprise.
Fields	getParameterFields () Returns the parameter fields for the report.
IReportSource	getReportSource () Returns the report source object.
java.lang.String	getReportSourceClassFactoryName () Returns the name of the report source class factory for the report.

IDE Extensions

java.lang.String	getSelectionFormula() Returns the selection formula value for the report.
java.lang.String	getStyleSheetFileName() Returns the file name of the cascading style sheet applied to the report.
boolean	isEnabledLogonPrompt() Returns whether or not logon prompting for the report is enabled.
boolean	isEnabledParameterPrompt() Returns whether or not parameter prompting for the report is enabled.
boolean	isReuseParameterValuesOnRefresh() Gets whether the current parameter values will be used when the viewer is refreshed.
void	navigateTo (java.lang.String sDataContext, java.lang.String sObjectName) Navigates to a particular report object in the current report.
void	refresh() Refreshes the ReportServerControl object.
void	removeReportPartBookmarkNavigationEventListener() Removes a DrillDownSubreportEventListener.
void	removeReportPartBookmarkNavigationEventListenererr() Deprecated.
void	removeReportSourceChangeListener() Removes a report source change event listener.
void	setDatabaseLogonInfos (ConnectionInfos newDatabaseLogonInfos) Sets the information required to log on to the database.
void	setEnabledLogonPrompt (boolean newEnableLogonPrompt) Sets whether or not the logon prompt is enabled for the report.
void	setEnabledParameterPrompt (boolean newEnableParameterPrompt) Sets whether or not the parameter prompt is enabled for the report.
void	setEnterpriseLogon (java.lang.Object newEnterpriseLogon) Sets the information required to log on to Crystal Enterprise.
void	setParameterFields (Fields newParameterFields) Sets the parameter fields for the report.
void	setReportSource (java.lang.Object reportSource) Sets the report instance that the viewer will render in HTML.
void	setReportSourceClassFactoryName (java.lang.String newReportClassFactoryName) Sets the report source class factory name.
void	setReuseParameterValuesOnRefresh (boolean newReuseParameterValuesOnRefresh) Sets whether or not to re-prompt parameters on refresh.
void	setSelectionFormula (java.lang.String newSelectionFormula) Sets the selection formula that will be used when the report is displayed.
void	setStyleSheetFileName (java.lang.String newStyleSheetFileName) Sets the style sheet that will be used to display the report content.

IDE Extensions

void	setURI (java.lang.String newURI) Sets the Universal Resource Identifier for the report.
void	setViewTimeSelectionFormula (java.lang.String newSelectionFormula) Reserved for future use.

Methods inherited from class com.crystaldecisions.report.web.ServerControl

deserializeBase64ToObject, getHeight, getHtmlContent, getLeft, getName, getTop, getURI, getViewState, getWidth, isIgnoreViewStateOnLoad, isOwnForm, isOwnPage, processHttpRequest, setHeight, setIgnoreViewStateOnLoad, setLeft, setName, setOwnForm, setOwnPage, setTop, setViewState, setWidth

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ReportServerControl

```
public ReportServerControl()
```

Method Detail

addReportPartBookmarkNavigationEventListener

```
public void addReportPartBookmarkNavigationEventListener(IReportPartBookmarkNavigationEventListener listener)  
throws java.util TooManyListenersException
```

Adds a listener for ReportPartBookmarkNavigationEventArgs. Please note that report parts are not available without the advanced viewers available with the Report Application Server. For more information, see the *Crystal Reports for BEA WebLogic Workshop Developer's Guide*.

Parameters:

listener – The DrillDownSubreportEventListener.

Throws:

TooManyListenersException – This is thrown if you attempt to add more than one listener on a particular listener source concurrently.

addReportSourceChangeListener

```
public void addReportSourceChangeListener(ReportSourceChangeListener listener)
```

Adds a listener for report source changes.

Parameters:

listener – The listener for report source changes.

dispose

```
public void dispose()
```

Disposes of the ReportServerControl.

getDatabaseLogonInfos

```
public ConnectionInfos getDatabaseLogonInfos()  
                        throws ReportSDKExceptionBase
```

Returns the information required to log on to the database.

Note: The viewer sets the value of this property after you set the report source for the report.

Note: This method is only available after the processHttpRequest method or getHtmlContent method has been called.

Returns:

The information required to log on to the database as a
com.crystaldecisions.sdk.occa.report.data.ConnectionInfo object.

Throws:

ReportSDKExceptionBase – This exception is thrown if the process is unsuccessful..

See Also:

setDatabaseLogonInfos, setEnableLogonPrompt

getEnterpriseLogon

```
public java.lang.Object getEnterpriseLogon()
```

Returns the information required to log on to Crystal Enterprise.

Note: This can be a Crystal Enterprise session or a Crystal Enterprise Token.

Returns:

The information required to log on to Crystal Enterprise as an Object.

See Also:

setEnterpriseLogon

getParameterFields

```
public Fields getParameterFields()
    throws ReportSDKExceptionBase
```

Returns the parameter fields for the report.

Note: This method is only available after the `processHttpRequest` method or `getHtmlContent` method has been called.

Returns:

The parameter fields for the report as a `com.crystaldecisions.sdk.occa.report.data.Fields` object.

Throws:

`ReportSDKExceptionBase` – This exception is thrown if the process is unsuccessful..

See Also:

`isEnabledParameterPrompt`, `setEnabledParameterPrompt`, `setParameterFields`

getReportSource

```
public IReportSource getReportSource()
    throws ReportSDKExceptionBase
```

Returns the report source object.

Returns:

The report source object.

Throws:

`ReportSDKExceptionBase` – This exception is thrown if the process is unsuccessful..

See Also:

`setReportSource`

getReportSourceClassFactoryName

```
public java.lang.String getReportSourceClassFactoryName()
```

Returns the name of the report source class factory for the report.

Returns:

The name of the report source class factory for the report as a `String`.

getSelectionFormula

```
public java.lang.String getSelectionFormula()
    throws ReportSDKExceptionBase
```

Returns the selection formula value for the report.

Note: This method is only available after the `processHttpRequest` method or `getHtmlContent` method has been called.

Returns:

The selection formula value for the report as a String.

Throws:

ReportSDKExceptionBase – This exception is thrown if the process is unsuccessful..

See Also:

setSelectionFormula

getStyleSheetFileName

```
public java.lang.String getStyleSheetFileName()
```

Returns the file name of the cascading style sheet applied to the report.

Returns:

The file name of the cascading style sheet applied to the report as a String.

See Also:

setStyleSheetFileName

isEnableLogonPrompt

```
public boolean isEnableLogonPrompt()
```

Returns whether or not logon prompting for the report is enabled.

Returns:

true if logon prompting is enabled, otherwise false.

See Also:

getDatabaseLogonInfos, setDatabaseLogonInfos,
setEnableLogonPrompt

isEnableParameterPrompt

```
public boolean isEnableParameterPrompt()
```

Returns whether or not parameter prompting for the report is enabled.

Returns:

true if parameter prompting is enabled, otherwise false.

See Also:

getParameterFields, setEnableParameterPrompt, setParameterFields

isReuseParameterValuesOnRefresh

```
public boolean isReuseParameterValuesOnRefresh()
```

Gets whether the current parameter values will be used when the viewer is refreshed.

Returns:

true if the current parameter values will be used when the viewer is refreshed and false otherwise.

See Also:

```
getParameterFields, isEnabledParameterPrompt,
setEnabledParameterPrompt, setParameterFields,
setReuseParameterValuesOnRefresh
```

navigateTo

```
public void navigateTo(java.lang.String sDataContext,
                       java.lang.String sObjectName)
```

Navigates to a particular report object in the current report.

Note: The data context represents the whole group tree or a sub tree in the group tree. DataContext can be specified using either strings or integers to narrow down the search for the report object.

Example 1:

"/Country[USA]/Region[CA]" or "2-1". Each of these examples navigates to the California region, which is found by going to the third node in the first group level and then the second node in the second group level.

Example 2:

"/Country[USA]/Region[*]" or "2-*". Each of these examples navigates to all regions in the USA, which is found by going to the third node in the first group level and then the second group level.

Note: The wild card (*) can only be set for the last level.

Parameters:

sDataContext – A String that specifies the data subtree of the report.
sObjectName – A String that specifies the name of the report object.

refresh

```
public void refresh()
```

Refreshes the ReportServerControl object.

removeReportPartBookmarkNavigationEventListener

```
public void removeReportPartBookmarkNavigationEventListener()
```

Removes a DrillDownSubreportEventListener.

removeReportPartBookmarkNavigationEventListener

```
public void removeReportPartBookmarkNavigationEventListener()
```

Deprecated.

Removes a DrillDownSubreportEventListener.

removeReportSourceChangeListener

```
public void removeReportSourceChangeListener()
```

Removes a report source change event listener.

setDatabaseLogonInfos

```
public void setDatabaseLogonInfos(ConnectionInfos newDatabaseLogonInfos)
```

Sets the information required to log on to the database.

Parameters:

newDatabaseLogonInfos – A
com.crystaldecisions.sdk.occa.report.data.ConnectionInfo object
that specifies the information required to log on to the database.

See Also:

getDatabaseLogonInfos

setEnableLogonPrompt

```
public void setEnableLogonPrompt(boolean newEnableLogonPrompt)
```

Sets whether or not the logon prompt is enabled for the report.

Parameters:

newEnableLogonPrompt – true to enable logon prompt for the report.

See Also:

isEnabledLogonPrompt, getDatabaseLogonInfos, setDatabaseLogonInfos

setEnableParameterPrompt

```
public void setEnableParameterPrompt(boolean newEnableParameterPrompt)
```

Sets whether or not the parameter prompt is enabled for the report.

Parameters:

newEnableParameterPrompt – true to enable parameter prompt for this report.

setEnterpriseLogon

```
public void setEnterpriseLogon(java.lang.Object newEnterpriseLogon)
```

Sets the information required to log on to Crystal Enterprise.

Note: This can be a Crystal Enterprise session or a Crystal Enterprise Token.

Parameters:

newEnterpriseLogon – An Object that specifies the information required to log on to Crystal Enterprise.

See Also:

getEnterpriseLogon

setParameterFields

```
public void setParameterFields(Fields newParameterFields)
```

Sets the parameter fields for the report.

Note: If the value of the ParameterFields property is not set at design time, you can prompt the user for parameter fields before the report is run by setting the setEnableParameterPrompt method to true.

The viewer sets the value of this property after you set the report source for the report.

Parameters:

newParameterFields – A
com.crystaldecisions.sdk.occa.report.data.Fields object that specifies the new parameter fields for the report.

See Also:

getParameterFields, isEnableParameterPrompt,
setEnableParameterPrompt

setReportSource

```
public void setReportSource(java.lang.Object reportSource)  
    throws ReportSDKExceptionBase
```

Sets the report instance that the viewer will render in HTML.

Example 1:

```
CrystalReportViewer viewer = new CrystalReportViewer();  
IReportSourceFactory reportSourceFactory = (IReportSourceFactory) enterpriseSession.get.  
Object reportSource = reportSourceFactory.openReportSource(report, Locale.ENGLISH);  
viewer.setOwnForm(true);  
viewer.setOwnPage(true);  
viewer.setName("Viewer1");  
viewer.setReportSource(reportSource);
```

Example 2:

```
CrystalReportViewer viewer = new CrystalReportViewer();
IReportAppFactory reportAppFactory = (IReportAppFactory) enterpriseSession.getService("IReportAppFactory");
ReportClientDocument clientDoc = reportAppFactory.openDocument(report, 0, Locale.ENGLISH);
viewer.setOwnForm(true);
viewer.setOwnPage(true);
viewer.setName("Viewer2");
viewer.setReportSource(clientDoc.getReportSource());
```

Parameters:

reportSource – An Object representing the report instance that the viewer will render in HTML.

Throws:

ReportSDKExceptionBase – This exception is thrown if the process is unsuccessful.

See Also:

getReportSource

setReportSourceClassFactoryName

```
public void setReportSourceClassFactoryName(java.lang.String newReportClassFactoryName)
```

Sets the report source class factory name.

Parameters:

newReportClassFactoryName – A String that specifies the report class factory name.

setReuseParameterValuesOnRefresh

```
public void setReuseParameterValuesOnRefresh(boolean newReuseParameterValuesOnRefresh)
```

Sets whether or not to re-prompt parameters on refresh.

Parameters:

newReuseParameterValuesOnRefresh – true to re-prompt parameters on refresh.

See Also:

isReuseParameterValuesOnRefresh

setSelectionFormula

```
public void setSelectionFormula(java.lang.String newSelectionFormula)
```

Sets the selection formula that will be used when the report is displayed.

Note: The default value is an empty string.

Parameters:

newSelectionFormula – A String that specifies the selection formula to use when the report is displayed.

See Also:

getSelectionFormula

setStyleSheetFileName

```
public void setStyleSheetFileName(java.lang.String newStyleSheetFileName)
```

Sets the style sheet that will be used to display the report content. The file should be located in the directory `crystalreportviewers\css` on the machine where the RAS server is running. The cascading style sheet that you use needs to contain the same classes as the default style sheet.

Parameters:

`newStyleSheetFileName` – A String that specifies the file name for the style sheet (for example `stylesheet.css`).

setURI

```
public void setURI(java.lang.String newURI)
```

Description copied from class: `ServerControl`

Sets the Universal Resource Identifier for the report.

Overrides:

`setURI` in class `ServerControl`

Parameters:

`newURI` – A String that specifies the Universal Resource Identifier for the report.

See Also:

`getURI`

setViewTimeSelectionFormula

```
public void setViewTimeSelectionFormula(java.lang.String newSelectionFormula)
```

Reserved for future use.

com.crystaldecisions.report.web.viewer Interface ReportSourceChangeListener

All Superinterfaces:

`java.util.EventListener`

```
public interface ReportSourceChangeListener  
extends java.util.EventListener
```

Listener for changes in report source.

Method Summary

void	clientTargetChange (java.util.EventObject e) Listener for changes to the target client.
void	reportSourceChange (java.util.EventObject e) Listener for report source changes.
void	separatePagesChange (java.util.EventObject e) Listener for changes to how the pages of the report are displayed.

Method Detail

clientTargetChange

```
public void clientTargetChange(java.util.EventObject e)
```

Listener for changes to the target client. This event occurs when the client target (where the content will be displayed) has changed.

Parameters:

e – The event object.

reportSourceChange

```
public void reportSourceChange(java.util.EventObject e)
```

Listener for report source changes. This event occurs when the report source changes.

Parameters:

e – The event object.

separatePagesChange

```
public void separatePagesChange(java.util.EventObject e)
```

Listener for changes to how the pages of the report are displayed. This event occurs when the setting to display the report as separate pages or as one long page has been changed.

Parameters:

e – The event object.

com.crystaldecisions.report.web.viewer Class ToolbarCommandEventArgs

```
java.lang.Object
└─ java.util.EventObject
```

```

└─ com.crystaldecisions.report.web.viewer.ViewerEventArgs
   └─ com.crystaldecisions.report.web.viewer.ToolbarCommandEventArgs

```

All Implemented Interfaces:

java.io.Serializable

*public class **ToolbarCommandEventArgs***
*extends **ViewerEventArgs***

This class provides event arguments for the `ToolbarCommandEventListener`.

See Also:

Serialized Form

Constructor Summary

ToolbarCommandEventArgs(java.lang.Object source,
java.lang.String action)

ToolbarCommandEventArgs(java.lang.Object source,
java.lang.String action, java.lang.String value)

Method Summary

java.lang.String	getAction () Returns the action.
------------------	---

java.lang.String	getValue () Returns the name of the value.
------------------	---

Methods inherited from class com.crystaldecisions.report.web.viewer.ViewerEventArgs

isHandled, setHandled

Methods inherited from class java.util.EventObject

getSource, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ToolbarCommandEventArgs

```
public ToolbarCommandEventArgs(java.lang.Object source,
                               java.lang.String action)
```

ToolbarCommandEventArgs

```
public ToolbarCommandEventArgs(java.lang.Object source,
                               java.lang.String action,
                               java.lang.String value)
```

Method Detail

getAction

```
public java.lang.String getAction()
```

Returns the action.

Returns:

The name of the action as a String.

getValue

```
public java.lang.String getValue()
```

Returns the name of the value.

Returns:

The name of the value as a String.

com.crystaldecisions.report.web.viewer

Interface ToolbarCommandEventListener

All Superinterfaces:

java.util.EventListener

```
public interface ToolbarCommandEventListener
    extends java.util.EventListener
```

Listener for ToolbarCommandEvents.

Method Summary

void	command (ToolbarCommandEventArgs e)
------	--

Method Detail

command

```
public void command(ToolbarCommandEventArgs e)
```

Listener for ToolbarCommandEvents.

Parameters:

e – A ToolbarCommandEventArgs object.

See Also:

ToolbarCommandEventArgs

com.crystaldecisions.report.web.viewer Class ViewerEventArgs

```
java.lang.Object
├─ java.util.EventObject
│   └─ com.crystaldecisions.report.web.viewer.ViewerEventArgs
```

All Implemented Interfaces:

java.io.Serializable

Direct Known Subclasses:

DrillDownSubreportEventArgs, DrillEventArgs, NavigateEventArgs, ToolbarCommandEventArgs

```
public class ViewerEventArgs
extends java.util.EventObject
```

This class provides event arguments for the viewer.

See Also:

Serialized Form

Constructor Summary

```
ViewerEventArgs(java.lang.Object source)
```

Method Summary

boolean	isHandled() Returns if the viewer event is handled.
void	setHandled (boolean newHandled) Sets whether the event is handled.

Methods inherited from class java.util.EventObject

getSource, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ViewerEventArgs

```
public ViewerEventArgs(java.lang.Object source)
```

Method Detail

isHandled

```
public boolean isHandled()
```

Returns if the viewer event is handled.

Returns:

true if the viewer event is handled and false otherwise.

setHandled

```
public void setHandled(boolean newHandled)
```

Sets whether the event is handled.

Parameters:

newHandled – true if the viewer event is handled.

com.crystaldecisions.report.web.viewer Class ViewInfo

```
java.lang.Object
```


*public class **ViewInfo***
extends `java.lang.Object`

This class provides more information for the client when handling an event. The `ViewInfo` object extends the event handling information provided for the client, making it possible to obtain information that is not available from the specific listener that is being used.

For example, if you were using the event handler for the `NavigateEventListener`, you could use the following to access the group name from the `ViewInfo` object:

```
public void navigate (NavigateEvents e){
    String groupName = ((CrystalReportViewer) (e.getSource())).getViewInfo().getGroupName();
}
```

Method Summary	
<code>java.lang.String</code>	<code>getGroupName ()</code> Returns the group name for the view.
<code>java.lang.String</code>	<code>getGroupNamePath ()</code> Returns the path for the group name.
<code>java.lang.String</code>	<code>getGroupPath ()</code> Returns the <code>GroupPath</code> for the view.
<code>int</code>	<code>getPageNumber ()</code> Returns the page number for the view.
<code>java.lang.String</code>	<code>getSubreportContainingGroupPath ()</code> Returns the data context for the subreport.
<code>java.lang.String</code>	<code>getSubreportName ()</code> Returns the subreport name for the view.
<code>int</code>	<code>getSubreportPageNumber ()</code> Returns the page number for the subreport.
<code>int</code>	<code>getSubreportXPos ()</code> The X position of the subreport.
<code>int</code>	<code>getSubreportYPos ()</code> The Y position of the subreport.
<code>int</code>	<code>getZoomFactor ()</code> Returns the zoom factor for the view.

Methods inherited from class <code>java.lang.Object</code>
<code>equals</code> , <code>getClass</code> , <code>hashCode</code> , <code>notify</code> , <code>notifyAll</code> , <code>toString</code> , <code>wait</code> , <code>wait</code> , <code>wait</code>

Method Detail

getGroupName

```
public java.lang.String getGroupName()
```

Returns the group name for the view. This name indicates how the report content is being grouped.

Returns:

The name of the group as a `String`.

getGroupNamePath

```
public java.lang.String getGroupNamePath()
```

Returns the path for the group name.

Returns:

The path for the group name as a `String`.

getGroupPath

```
public java.lang.String getGroupPath()
```

Returns the `GroupPath` for the view.

Returns:

The `GroupPath` as an `String`.

getPageNumber

```
public int getPageNumber()
```

Returns the page number for the view.

Returns:

The page number for the view as an `int`.

getSubreportContainingGroupPath

```
public java.lang.String getSubreportContainingGroupPath()
```

Returns the data context for the subreport.

Returns:

The data context for the subreport as a `String`.

getSubreportName

```
public java.lang.String getSubreportName()
```

Returns the subreport name for the view.

Returns:

The name of the subreport as a String.

getSubreportPageNumber

```
public int getSubreportPageNumber()
```

Returns the page number for the subreport.

Returns:

The page number of the subreport as an int.

getSubreportXPos

```
public int getSubreportXPos()
```

The X position of the subreport.

Returns:

The X position of the subreport as an int.

getSubreportYPos

```
public int getSubreportYPos()
```

The Y position of the subreport.

Returns:

The Y position of the subreport as an int.

getZoomFactor

```
public int getZoomFactor()
```

Returns the zoom factor for the view.

Returns:

The zoom factor for the view as an int.

JSP Tags Reference

crystal-tags-reportviewer Tags

crystal-tags-reportviewer Tags

<i>Tag</i>	<i>Summary</i>
crviewer:viewer	The viewer tag is the main tag that instantiates the CrystalReportViewer. It handles the creation of the viewer and allows you to set viewer properties through supported attributes.
crviewer:report	The report tag is used to specify the report that is displayed by the viewer. It is required by the viewer tag, unless you are using a manually cached report source.

Viewer Tag Library

The Viewer Tag Library allows you to access the report viewing capabilities of the Crystal report viewers without needing to embed large amounts of Java code in your JSP pages.

This section first provides a brief overview of the Viewer Tag Library, including how to set up your JSP pages to use the tag library. The section then provides a list of all the tags available, their specific uses, and relevant examples.

Click the appropriate link to jump to that section:

- [Overview](#)
- [Tag reference](#)
- [Samples](#)

Overview

The Viewer Tag Library allows you to access the Crystal report viewers through modular and reusable tags. Using tags allows you to reduce the amount of embedded Java code in a JSP page by moving the functional implementation into the tag and its implementing class. Doing so effectively abstracts the details of the code from the presentation of the actual page, allowing the web page author to concentrate on developing JSP pages. This allows all the functionality of the Crystal report viewers to be accessed without knowing how to write Java code.

The Viewer Tag Library is made up of a primary tag, the viewer tag, and a collection of nested tags. The viewer tag supports attributes that relate to the way the Crystal report viewers appear and behave.

Using the Viewer Tag Library is equivalent to using the Viewer Java SDK to programmatically customize the viewer. It provides the same basic functionality that the Viewer Java SDK provides, supporting the same features and allowing the same level of customization over how the viewer is displayed. The advantage of using the Viewer Tag Library, however, is that it consistently generates efficient code, as the tag implementation has been carefully tested and optimized.

Using the Viewer Tag Library

To use the tags provided by the Viewer Tag Library in a JSP page, a reference to the tag library's descriptor file is required. The following code needs to be added to the top of each JSP page using the tag library:

```
<%@ taglib uri="/crystal-tags-reportviewer.tld" prefix="crviewer" %>
```

Tag reference

The Viewer Tag Library is comprised of several tags. The tag reference is designed to provide a concise description of the tags available, their supported attributes, and how the tags and attributes should be used. Each tag description follows the same format. First, a brief summary of the tag and its function are given. This is followed by a list of nested tags and a table containing a summary of all the attributes. The attributes are

then described in detail.

Each attribute description provides a quick summary that states whether the tag is optional or required, its default value, and the type of value it expects. This summary is followed by a more comprehensive description of the attribute, and, if required, what specific values the attribute accepts.

The following list provides a hierarchical view of the tags available:

Viewer tag

- Report tag

Viewer tag

The viewer tag is the main tag that instantiates the CrystalReportViewer. It handles the creation of the viewer and allows you to set viewer properties through supported attributes.

Each instance of the viewer tag requires a report tag to specify the report to be displayed. If multiple instances of the viewer will be used on a single HTML page, each instance must be given a unique name, specified through the viewerName attribute.

Nested tags

Report tag

Attributes

Appearance

Attribute	Description	Required
enablePageToGrow	Sets whether the viewer takes up as much space as it needs to properly display the report.	No
height	Sets the viewer height in pixels.	No
width	Sets the viewer width in pixels.	No
left	Sets the left position of the viewer.	No
top	Sets the top position of the viewer.	No
displayGroupTree	Sets whether to display the Group Tree on viewer start up.	No
displayPage		No

IDE Extensions

	Sets whether to display the report page.	
displayToolBar	Sets whether to display the toolbar.	No
groupTreeWidth	Sets the width of the Group Tree in pixels.	No
groupTreeWidthUnit	Sets whether to use pixels or percentage to determine the group tree width.	No
zoomPercentage	Sets the zoom factor used when displaying the report.	No
styleSheet	Sets the stylesheet that used to display the report content.	No

Behavior

Attribute	Description	Required
printMode	Sets whether to print using PDF or ActiveX print mode when the user clicks the print button.	No
allowDatabaseLogonPrompting	Sets whether to allow users to be prompted for database logon information.	No
allowDrillDown	Sets whether to allow the user to drill down on the report.	No
allowParameterPrompting	Sets whether to allow the user to be prompted for parameter values.	No
hyperlinkTarget	Sets the window or frame where hyperlinked documents are displayed.	No
isOwnPage	Sets whether the viewer control owns the page.	No

Toolbar

Attribute	Description	Required
displayToolBarViewList	Sets whether to display the view list on the toolbar.	No
displayToolBarExportButton	Sets whether to display the export button on the toolbar.	No
displayToolBarFindPageButton	Sets whether to display the Go To Page button on the toolbar.	No
displayToolBarPageNavigationButtons	Sets whether to display the page navigation buttons on the toolbar.	No
displayToolBarPrintButton	Sets whether to display the print button on the toolbar.	No
displayToolBarRefreshButton	Sets whether to display the refresh button on the toolbar.	No
displayToolBarFindButton	Sets whether to display the search button on the toolbar.	No
displayToolBarToggleTreeButton	Sets whether to display the Group Tree toggle button on the toolbar.	No
displayToolBarZoomList	Sets whether to display a zoom factor drop down list on the toolbar.	No

IDE Extensions

displayToolbarCrystalLogo	Sets whether to display the Crystal Decisions logo on the toolbar.	No
---------------------------	--	----

Report Source

Attribute	Description	Required
reportSourceType	Specifies the type of report source the viewer will use.	Yes
reportSourceVar	Sets the name of the variable where the report source will be cached.	No

Identification

Attribute	Description	Required
viewerName	Sets the name of the viewer.	Yes

enablePageToGrow

Optional

Defaults to true

boolean

Sets whether the viewer ignores the height and width values it is given and try to find the best fit for the report by taking up as much space as it needs to properly display the report.

height

Optional

Defaults to 600

int

Sets the viewer height in pixels. Together, the height and width attributes control the viewer's viewable size.

width

Optional

Defaults to 800

int

Sets the viewer width in pixels. Together, the height and width attributes control the viewer's viewable size.

left

Optional

Defaults to 0

int

Sets the left position of the viewer. The units used are browser dependent.

top

Optional

Defaults to 0

int

Sets the top position of the viewer. The units used are browser dependent.

displayGroupTree

Optional

Defaults to true

boolean

Sets whether to display the Group Tree on viewer start up.

displayPage

Optional

Defaults to true

boolean

Sets whether to display the report page.

displayToolbar

Optional

Defaults to true

boolean

Sets whether to display the toolbar.

groupTreeWidth

Optional

Defaults to 200

int

Sets the width of the Group Tree in pixels.

groupTreeWidthUnit

Optional

pixel

String

Sets whether to use pixels or percentage to determine the group tree width.

Use `pixel` to specify to use pixels and `percent` to specify to use percentage to determine the group tree width.

zoomPercentage

Optional

Defaults to 100

int

Sets the zoom factor used when displaying the report. Acceptable values for the zoom factor range from 10 to 400 percent.

styleSheet

Optional

Defaults to `default.css`

String

Sets the style sheet that used to display the report content. The cascading style sheet that you use needs to contain the same classes as the default style sheet. The default style sheet is located in the `/crystalreportviewers10/css` directory. When specifying a custom style sheet, the paths are always

relative to this directory.

printMode

Optional

Defaults to PDF

String

Sets whether to print using PDF or ActiveX print mode when the user clicks the print button. In PDF print mode, a PDF is displayed, allowing the user to then print it. In ActiveX print mode, an ActiveX control is downloaded to the client machine and is sent directly to the printer. If ActiveX print mode is selected on a system that does not support ActiveX controls, the print mode defaults to PDF printing.

Use PDF to specify PDF print mode and ActiveX to specify ActiveX print mode.

allowDatabaseLogonPrompting

Optional

Defaults to true

boolean

Sets whether to allow users to be prompted for database logon information.

allowDrillDown

Optional

Defaults to true

boolean

Sets whether to allow the user to drill down on the report.

allowParameterPrompting

Optional

Defaults to true

boolean

Sets whether to allow the user to be prompted for parameter values.

hyperlinkTarget

Optional

Defaults to `_self`

String

Sets the window or frame where hyperlinked documents are displayed.

Use the target `_self` to display the HTML document in the same frame, `_parent` to display HTML document in the same frame or window that contains the current frameset, `_top` to display HTML document in the entire browser window, and `_blank` to display HTML document in a new browser window.

isOwnPage

Optional

Defaults to `false`

boolean

Sets whether the viewer control owns the page. Set this attribute to false if the page is within a portal. If the server control owns the page, it provides the opening and closing HTML tags and it is able to get and set values for the entire page.

Note: If you set `isOwnPage` to false then you must set the charset and the content-type of the HTML page where the viewer is displayed.

displayToolbarViewList

Optional

Defaults to `true`

boolean

Sets whether to display the view list on the toolbar.

Note: The view list includes the Main Report and any views you have drilled-down into. This can include groups, charts and subreports.

displayToolbarExportButton

Optional

Defaults to `true`

boolean

Sets whether to display the export button on the toolbar.

Note: isOwnPage must be set to true or the button is not rendered, regardless of the value set in the setHasExportButton method.

displayToolbarFindPageButton

Optional

Defaults to true

boolean

Sets whether to display the Go To Page button on the toolbar.

Note: If true, the HTML returned includes a text box, into which the user can type a page number to navigate to a specific page in the report. If false, the HTML returned displays the current page, but does not allow the user to navigate by page number.

displayToolbarPageNavigationButtons

Optional

Defaults to true

boolean

Sets whether to display the page navigation buttons on the toolbar.

displayToolbarPrintButton

Optional

Defaults to true

boolean

Sets whether to display the print button on the toolbar. This is only available if isOwnPage is set to true.

Note: By default, printing is accomplished by automatically exporting the report to PDF and then printing the report from the PDF viewer. For systems with ActiveX support, ActiveX print mode, which uses ActiveX controls to print the report, can be used. To specify the print mode, use the printMode attribute.

displayToolBarRefreshButton

Optional

Defaults to false

boolean

Sets whether to display the refresh button on the toolbar.

displayToolBarFindButton

Optional

Defaults to true

boolean

Sets whether to display the search button on the toolbar.

displayToolBarToggleTreeButton

Optional

Defaults to true

boolean

Sets whether to display the Group Tree toggle button on the toolbar.

displayToolBarZoomList

Optional

Defaults to true

boolean

Sets whether to display a zoom factor drop down list on the toolbar.

displayToolBarCrystalLogo

Optional

Defaults to true

boolean

Sets whether to display the Crystal Decisions logo on the toolbar.

reportSourceType

Required

No default value.

String

Specifies the type of report source the viewer will use. For more information on report sources, see Report sources.

Possible values are:

- `reportingComponent`

Specifies that the Java Reporting Component will be used to render the report.

- `reportApplicationServer`

Specifies that a Report Application Server will be used as a report source.

- `pageServer`

Specifies that a Page Server from part of a Crystal Enterprise deployment will be used as a report source.

reportSourceVar

Optional

No default value.

String

Sets the name of the session variable where the report source will be cached. Caching the report source in the browser's session allows it to be reused, improving performance when the same report source will be used multiple times.

viewerName

Required

Defaults to `CrystalViewer`

String

Sets the name of the viewer. The name represents the HTML form name used to post back requests to the server. When there is more than one viewer on a page, each viewer must be given a unique name, otherwise, a post back in one viewer may be sent to the wrong viewer.

Example

The following example shows how to create a viewer with the Group Tree hidden and 80% zoom:

```
<crviewer:viewer displayGroupTree="false" zoomPercentage="80"
reportSourceType="reportingComponent">
```

...

```
</crviewer:viewer>
```

Report tag

The report tag is used to specify the report that is displayed by the viewer. It is required by the viewer tag, unless you are using a manually cached report source.

Nested tags

None

Attributes

Attribute	Description	Required
reportName	Sets the name of the report to be viewed when an unmanaged report source is used.	Yes

reportName

Required

No default value

String

Sets the name of the report to be viewed when an unmanaged report source is used. This applies to unmanaged RAS and Java Reporting Component report sources.

For Java Reporting Component report sources, the reportName attribute can be specified using absolute or relative file paths and URLs. For more information, see Java Reporting Component.

For unmanaged RAS report sources, the `reportName` attribute must specify a path that is accessible by Report Application Server. That is, the path is interpreted by the Report Application Server and so local paths refer to the machine on which the RAS server is running. If you would like to access reports running on a separate machine, you must make sure that the report is accessible through a network share.

Example

The following example shows how to view a report stored on the local file system:

```
<crviewer:viewer ...>  
  
<crviewer:report reportName="c:\\Reports\\World Sales.rpt" />  
  
</crviewer:viewer>
```

Samples

The Viewer Tag Library is designed around simplicity and ease-of-use. The following examples demonstrate how to use the functionality provided by the tag library to view reports.

Click the appropriate link to jump to that section:

- [Viewing unmanaged reports](#)
- [Viewing managed reports](#)

Viewing unmanaged reports

Unmanaged reports do not require authentication or other logon procedures in order to access them. Components that support unmanaged reports include the Java Reporting Component and unmanaged Report Application Servers. Viewing an unmanaged report using the Viewer Tag Library primarily relies on specifying the appropriate report source type and location of the report file.

Click the appropriate link to jump to that section:

- [Viewing a report using the Java Reporting Component](#)
- [Viewing a report using an unmanaged RAS server](#)

Viewing a report using the Java Reporting Component

This is a basic sample designed to show you how to use the Viewer Tag Library to display a report using the Java Reporting Component.

To view a report

1. Ensure that the appropriate reference to the Viewer Tag Library's descriptor file is added at the top of your JSP page.

You must also ensure that the appropriate JAR files and additional support files are present. If you are using a wizard, this is automatically done for you.

```
<%@ taglib uri="/crystal-tags-reportviewer.tld" prefix="crviewer" %>
```

Note: You can specify any name for the prefix attribute. This just determines what tag prefix you use to access the tag library's tags.

2. Create an opening viewer tag.

In this tag, you must specify the viewer name and the type report source that is being used.

```
<html>
```

```
<body>
```

```
<crviewer:viewer viewerName="CrystalViewer"
reportSourceType="reportingComponent" >
```

3. Create the required report tag.

This specifies the report that is displayed.

```
<crviewer:report reportName="/reports/sample.rpt"/>
```

4. Close the viewer tag.

The JSP page is now ready to view the sample.rpt report.

```
</crviewer:viewer>
```

```
</body>
```

```
</html>
```

Viewing a report using an unmanaged RAS server

This sample shows you how to use the Viewer Tag Library to display a report using an unmanaged Report Application Server.

To view a report

1. Ensure that the appropriate reference to the Viewer Tag Library's descriptor file is added at the top of your JSP page.

```
<%@ taglib uri="/crystal-tags-reportviewer.tld" prefix="crviewer" %>
```

Note: You can specify any name for the prefix attribute. This just determines what tag prefix you use to access the tag library's tags.

2. Set system property to specify the location of the clientSDKOptions.xml file.

This file is used to set the location of the RAS server.

```
<%  
  
    System.setProperty("ras.config","C:\\temp");  
  
%>
```

Note: You can also set the location of the clientSDKOptions.xml file in the classpath.

3. Create an opening viewer tag.

In this tag, you must specify the viewer name and the type report source that is being used. In this case, the reportApplicationServer report source is being used.

```
<html>  
  
<body>  
  
    <crviewer:viewer viewerName="CrystalViewer"  
    reportSourceType="reportApplicationServer" >
```

4. Create the required report tag, specifying the report to be displayed.

The reportName value is prefixed with rassdk:// so that the RAS server can correctly locate the file.

```
<crviewer:report reportName="rassdk://c:/reports/sample.rpt"/>
```

5. Close the viewer tag.

The JSP page is now ready to view the sample.rpt report.

```
</crviewer:viewer>  
  
</body>  
  
</html>
```

Viewing managed reports

Although there are no tags that provide direct support for viewing managed reports, a combination of tags and JSP code can provide the equivalent functionality. Viewing managed reports requires you to have either a Crystal Enterprise deployment or Report Application Server running.

Click the appropriate link to jump to that section:

- Viewing a report using a Page Server
- Viewing a report using a managed RAS server

Viewing a report using a Page Server

This sample shows you how to use the Viewer Tag Library to display a report using a Page Server report source.

To view a report using a Page Server

1. Log on to Crystal Enterprise and get an InfoStore object.

In this case, the default administrator account is used to log on to a Crystal Enterprise installation running on the same machine as the application server.

<%

```
    IEnterpriseSession es =
    CrystalEnterprise.getSessionMgr().logon("administrator", "",
    "localhost", "secEnterprise");
```

```
    IInfoStore infoStore = (IInfoStore)
    es.getService("", "InfoStore");
```

2. Query for the report you wish to view.

```
    IInfoObjects infoObjects = infoStore.query("SELECT * FROM
    CI_INFOOBJECTS WHERE SI_NAME='World Sales Report'");
```

```
    IInfoObject report = (IInfoObject) infoObjects.get(0);
```

3. Create Page Server report source for the report you retrieved.

```
    IReportSourceFactory2 rsFactory = (IReportSourceFactory2)
    es.getService("PSReportFactory");
```

```
    IReportSource rptSrc = rsFactory.createReportSource(report,
    request.getLocale());
```

4. Cache the ReportSource object in a session variable.

IDE Extensions

This allows it to be retrieved by the viewer.

```
session.setAttribute("ReportSource", rptSrc);

%>
```

5. Create a viewer tag, specifying the viewer name, report source type, and report source variable.

In this case, the pageServer report source is being used and the report source variable refers to the session variable where you stored the ReportSource object.

```
<html>

<body>

<crviewer:viewer viewerName="CrystalViewer"
reportSourceType="pageServer" reportSourceVar="ReportSource" />

</body>

</html>
```

Note: In this case, a report tag is not needed because the report source has already been cached in a session variable and does not to be retrieved by the viewer.

Viewing a report using a managed RAS server

This sample shows you how to use the Viewer Tag Library to display a report using a managed Report Application Server report source.

To view a report using a managed RAS server

1. Log on to Crystal Enterprise and get an InfoStore object.

In this case, the default administrator account is used to log on to a Crystal Enterprise installation running on the same machine as the application server.

```
<%

    IEnterpriseSession es =
CrystalEnterprise.getSessionMgr().logon("administrator", "",
"localhost", "secEnterprise");

    IInfoStore infoStore = (IInfoStore)
es.getService("", "InfoStore");
```

2. Query for the report you wish to view.

IDE Extensions

```
IInfoObjects infoObjects = infoStore.query("SELECT * FROM  
CI_INFOOBJECTS WHERE SI_NAME='World Sales Report'");
```

```
IInfoObject report = (IInfoObject) infoObjects.get(0);
```

3. Create a RAS report source for the report you retrieved.

```
IReportSourceFactory2 rsFactory = (IReportSourceFactory2)  
es.getService("RASReportFactory");
```

```
IReportSource rptSrc = rsFactory.createReportSource(report,  
request.getLocale());
```

4. Cache the ReportSource object in a session variable.

This allows it to be retrieved by the viewer.

```
session.setAttribute("ReportSource", rptSrc);
```

```
%>
```

5. Create a viewer tag, specifying the viewer name, report source type, and report source variable.

In this case, the reportApplicationServer report source is being used and the report source variable refers to the session variable where you stored the ReportSource object.

```
<html>
```

```
<body>
```

```
<crviewer:viewer viewerName="CrystalViewer"  
reportSourceType="pageServer" reportSourceVar="ReportSource" />
```

```
</body>
```

```
</html>
```

Note: In this case, a report tag is not needed because the report source has already been cached in a session variable and does not to be retrieved by the viewer.

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Viewer Tag Library

The Viewer Tag Library allows you to access the report viewing capabilities of the Crystal report viewers without needing to embed large amounts of Java code in your JSP pages.

This section first provides a brief overview of the Viewer Tag Library, including how to set up your JSP pages to use the tag library. The section then provides a list of all the tags available, their specific uses, and relevant examples.

Click the appropriate link to jump to that section:

- [Overview](#)
- [Tag reference](#)
- [Samples](#)

Overview

The Viewer Tag Library allows you to access the Crystal report viewers through modular and reusable tags. Using tags allows you to reduce the amount of embedded Java code in a JSP page by moving the functional implementation into the tag and its implementing class. Doing so effectively abstracts the details of the code from the presentation of the actual page, allowing the web page author to concentrate on developing JSP pages. This allows all the functionality of the Crystal report viewers to be accessed without knowing how to write Java code.

The Viewer Tag Library is made up of a primary tag, the viewer tag, and a collection of nested tags. The viewer tag supports attributes that relate to the way the Crystal report viewers appear and behave.

Using the Viewer Tag Library is equivalent to using the Viewer Java SDK to programmatically customize the viewer. It provides the same basic functionality that the Viewer Java SDK provides, supporting the same features and allowing the same level of customization over how the viewer is displayed. The advantage of using the Viewer Tag Library, however, is that it consistently generates efficient code, as the tag implementation has been carefully tested and optimized.

Using the Viewer Tag Library

To use the tags provided by the Viewer Tag Library in a JSP page, a reference to the tag library's descriptor file is required. The following code needs to be added to the top of each JSP page using the tag library:

```
<%@ taglib uri="/crystal-tags-reportviewer.tld" prefix="crviewer" %>
```

Tag reference

The Viewer Tag Library is comprised of several tags. The tag reference is designed to provide a concise description of the tags available, their supported attributes, and how the tags and attributes should be used. Each tag description follows the same format. First, a brief summary of the tag and its function are given. This is followed by a list of nested tags and a table containing a summary of all the attributes. The attributes are

then described in detail.

Each attribute description provides a quick summary that states whether the tag is optional or required, its default value, and the type of value it expects. This summary is followed by a more comprehensive description of the attribute, and, if required, what specific values the attribute accepts.

The following list provides a hierarchical view of the tags available:

Viewer tag

- Report tag

Viewer tag

The viewer tag is the main tag that instantiates the CrystalReportViewer. It handles the creation of the viewer and allows you to set viewer properties through supported attributes.

Each instance of the viewer tag requires a report tag to specify the report to be displayed. If multiple instances of the viewer will be used on a single HTML page, each instance must be given a unique name, specified through the viewerName attribute.

Nested tags

Report tag

Attributes

Appearance

Attribute	Description	Required
enablePageToGrow	Sets whether the viewer takes up as much space as it needs to properly display the report.	No
height	Sets the viewer height in pixels.	No
width	Sets the viewer width in pixels.	No
left	Sets the left position of the viewer.	No
top	Sets the top position of the viewer.	No
displayGroupTree	Sets whether to display the Group Tree on viewer start up.	No
displayPage		No

IDE Extensions

	Sets whether to display the report page.	
displayToolBar	Sets whether to display the toolbar.	No
groupTreeWidth	Sets the width of the Group Tree in pixels.	No
groupTreeWidthUnit	Sets whether to use pixels or percentage to determine the group tree width.	No
zoomPercentage	Sets the zoom factor used when displaying the report.	No
styleSheet	Sets the stylesheet that used to display the report content.	No

Behavior

Attribute	Description	Required
printMode	Sets whether to print using PDF or ActiveX print mode when the user clicks the print button.	No
allowDatabaseLogonPrompting	Sets whether to allow users to be prompted for database logon information.	No
allowDrillDown	Sets whether to allow the user to drill down on the report.	No
allowParameterPrompting	Sets whether to allow the user to be prompted for parameter values.	No
hyperlinkTarget	Sets the window or frame where hyperlinked documents are displayed.	No
isOwnPage	Sets whether the viewer control owns the page.	No

Toolbar

Attribute	Description	Required
displayToolBarViewList	Sets whether to display the view list on the toolbar.	No
displayToolBarExportButton	Sets whether to display the export button on the toolbar.	No
displayToolBarFindPageButton	Sets whether to display the Go To Page button on the toolbar.	No
displayToolBarPageNavigationButtons	Sets whether to display the page navigation buttons on the toolbar.	No
displayToolBarPrintButton	Sets whether to display the print button on the toolbar.	No
displayToolBarRefreshButton	Sets whether to display the refresh button on the toolbar.	No
displayToolBarFindButton	Sets whether to display the search button on the toolbar.	No
displayToolBarToggleTreeButton	Sets whether to display the Group Tree toggle button on the toolbar.	No
displayToolBarZoomList	Sets whether to display a zoom factor drop down list on the toolbar.	No

IDE Extensions

displayToolbarCrystalLogo	Sets whether to display the Crystal Decisions logo on the toolbar.	No
---------------------------	--	----

Report Source

Attribute	Description	Required
reportSourceType	Specifies the type of report source the viewer will use.	Yes
reportSourceVar	Sets the name of the variable where the report source will be cached.	No

Identification

Attribute	Description	Required
viewerName	Sets the name of the viewer.	Yes

enablePageToGrow

Optional

Defaults to true

boolean

Sets whether the viewer ignores the height and width values it is given and try to find the best fit for the report by taking up as much space as it needs to properly display the report.

height

Optional

Defaults to 600

int

Sets the viewer height in pixels. Together, the height and width attributes control the viewer's viewable size.

width

Optional

Defaults to 800

int

Sets the viewer width in pixels. Together, the height and width attributes control the viewer's viewable size.

left

Optional

Defaults to 0

int

Sets the left position of the viewer. The units used are browser dependent.

top

Optional

Defaults to 0

int

Sets the top position of the viewer. The units used are browser dependent.

displayGroupTree

Optional

Defaults to true

boolean

Sets whether to display the Group Tree on viewer start up.

displayPage

Optional

Defaults to true

boolean

Sets whether to display the report page.

displayToolbar

Optional

Defaults to true

boolean

Sets whether to display the toolbar.

groupTreeWidth

Optional

Defaults to 200

int

Sets the width of the Group Tree in pixels.

groupTreeWidthUnit

Optional

pixel

String

Sets whether to use pixels or percentage to determine the group tree width.

Use `pixel` to specify to use pixels and `percent` to specify to use percentage to determine the group tree width.

zoomPercentage

Optional

Defaults to 100

int

Sets the zoom factor used when displaying the report. Acceptable values for the zoom factor range from 10 to 400 percent.

styleSheet

Optional

Defaults to `default.css`

String

Sets the style sheet that used to display the report content. The cascading style sheet that you use needs to contain the same classes as the default style sheet. The default style sheet is located in the `/crystalreportviewers10/css` directory. When specifying a custom style sheet, the paths are always

relative to this directory.

printMode

Optional

Defaults to PDF

String

Sets whether to print using PDF or ActiveX print mode when the user clicks the print button. In PDF print mode, a PDF is displayed, allowing the user to then print it. In ActiveX print mode, an ActiveX control is downloaded to the client machine and is sent directly to the printer. If ActiveX print mode is selected on a system that does not support ActiveX controls, the print mode defaults to PDF printing.

Use PDF to specify PDF print mode and ActiveX to specify ActiveX print mode.

allowDatabaseLogonPrompting

Optional

Defaults to true

boolean

Sets whether to allow users to be prompted for database logon information.

allowDrillDown

Optional

Defaults to true

boolean

Sets whether to allow the user to drill down on the report.

allowParameterPrompting

Optional

Defaults to true

boolean

Sets whether to allow the user to be prompted for parameter values.

hyperlinkTarget

Optional

Defaults to `_self`

String

Sets the window or frame where hyperlinked documents are displayed.

Use the target `_self` to display the HTML document in the same frame, `_parent` to display HTML document in the same frame or window that contains the current frameset, `_top` to display HTML document in the entire browser window, and `_blank` to display HTML document in a new browser window.

isOwnPage

Optional

Defaults to `false`

boolean

Sets whether the viewer control owns the page. Set this attribute to false if the page is within a portal. If the server control owns the page, it provides the opening and closing HTML tags and it is able to get and set values for the entire page.

Note: If you set `isOwnPage` to false then you must set the charset and the content-type of the HTML page where the viewer is displayed.

displayToolbarViewList

Optional

Defaults to `true`

boolean

Sets whether to display the view list on the toolbar.

Note: The view list includes the Main Report and any views you have drilled-down into. This can include groups, charts and subreports.

displayToolbarExportButton

Optional

Defaults to `true`

boolean

Sets whether to display the export button on the toolbar.

Note: isOwnPage must be set to true or the button is not rendered, regardless of the value set in the setHasExportButton method.

displayToolbarFindPageButton

Optional

Defaults to true

boolean

Sets whether to display the Go To Page button on the toolbar.

Note: If true, the HTML returned includes a text box, into which the user can type a page number to navigate to a specific page in the report. If false, the HTML returned displays the current page, but does not allow the user to navigate by page number.

displayToolbarPageNavigationButtons

Optional

Defaults to true

boolean

Sets whether to display the page navigation buttons on the toolbar.

displayToolbarPrintButton

Optional

Defaults to true

boolean

Sets whether to display the print button on the toolbar. This is only available if isOwnPage is set to true.

Note: By default, printing is accomplished by automatically exporting the report to PDF and then printing the report from the PDF viewer. For systems with ActiveX support, ActiveX print mode, which uses ActiveX controls to print the report, can be used. To specify the print mode, use the printMode attribute.

displayToolBarRefreshButton

Optional

Defaults to false

boolean

Sets whether to display the refresh button on the toolbar.

displayToolBarFindButton

Optional

Defaults to true

boolean

Sets whether to display the search button on the toolbar.

displayToolBarToggleTreeButton

Optional

Defaults to true

boolean

Sets whether to display the Group Tree toggle button on the toolbar.

displayToolBarZoomList

Optional

Defaults to true

boolean

Sets whether to display a zoom factor drop down list on the toolbar.

displayToolBarCrystalLogo

Optional

Defaults to true

boolean

Sets whether to display the Crystal Decisions logo on the toolbar.

reportSourceType

Required

No default value.

String

Specifies the type of report source the viewer will use. For more information on report sources, see Report sources.

Possible values are:

- `reportingComponent`

Specifies that the Java Reporting Component will be used to render the report.

- `reportApplicationServer`

Specifies that a Report Application Server will be used as a report source.

- `pageServer`

Specifies that a Page Server from part of a Crystal Enterprise deployment will be used as a report source.

reportSourceVar

Optional

No default value.

String

Sets the name of the session variable where the report source will be cached. Caching the report source in the browser's session allows it to be reused, improving performance when the same report source will be used multiple times.

viewerName

Required

Defaults to `CrystalViewer`

String

Sets the name of the viewer. The name represents the HTML form name used to post back requests to the server. When there is more than one viewer on a page, each viewer must be given a unique name, otherwise, a post back in one viewer may be sent to the wrong viewer.

Example

The following example shows how to create a viewer with the Group Tree hidden and 80% zoom:

```
<crviewer:viewer displayGroupTree="false" zoomPercentage="80"
reportSourceType="reportingComponent">
```

...

```
</crviewer:viewer>
```

Report tag

The report tag is used to specify the report that is displayed by the viewer. It is required by the viewer tag, unless you are using a manually cached report source.

Nested tags

None

Attributes

Attribute	Description	Required
reportName	Sets the name of the report to be viewed when an unmanaged report source is used.	Yes

reportName

Required

No default value

String

Sets the name of the report to be viewed when an unmanaged report source is used. This applies to unmanaged RAS and Java Reporting Component report sources.

For Java Reporting Component report sources, the reportName attribute can be specified using absolute or relative file paths and URLs. For more information, see Java Reporting Component.

For unmanaged RAS report sources, the `reportName` attribute must specify a path that is accessible by Report Application Server. That is, the path is interpreted by the Report Application Server and so local paths refer to the machine on which the RAS server is running. If you would like to access reports running on a separate machine, you must make sure that the report is accessible through a network share.

Example

The following example shows how to view a report stored on the local file system:

```
<crviewer:viewer ...>  
  
<crviewer:report reportName="c:\\Reports\\World Sales.rpt" />  
  
</crviewer:viewer>
```

Samples

The Viewer Tag Library is designed around simplicity and ease-of-use. The following examples demonstrate how to use the functionality provided by the tag library to view reports.

Click the appropriate link to jump to that section:

- [Viewing unmanaged reports](#)
- [Viewing managed reports](#)

Viewing unmanaged reports

Unmanaged reports do not require authentication or other logon procedures in order to access them. Components that support unmanaged reports include the Java Reporting Component and unmanaged Report Application Servers. Viewing an unmanaged report using the Viewer Tag Library primarily relies on specifying the appropriate report source type and location of the report file.

Click the appropriate link to jump to that section:

- [Viewing a report using the Java Reporting Component](#)
- [Viewing a report using an unmanaged RAS server](#)

Viewing a report using the Java Reporting Component

This is a basic sample designed to show you how to use the Viewer Tag Library to display a report using the Java Reporting Component.

To view a report

1. Ensure that the appropriate reference to the Viewer Tag Library's descriptor file is added at the top of your JSP page.

You must also ensure that the appropriate JAR files and additional support files are present. If you are using a wizard, this is automatically done for you.

```
<%@ taglib uri="/crystal-tags-reportviewer.tld" prefix="crviewer" %>
```

Note: You can specify any name for the prefix attribute. This just determines what tag prefix you use to access the tag library's tags.

2. Create an opening viewer tag.

In this tag, you must specify the viewer name and the type report source that is being used.

```
<html>
```

```
<body>
```

```
<crviewer:viewer viewerName="CrystalViewer"
reportSourceType="reportingComponent" >
```

3. Create the required report tag.

This specifies the report that is displayed.

```
<crviewer:report reportName="/reports/sample.rpt"/>
```

4. Close the viewer tag.

The JSP page is now ready to view the sample.rpt report.

```
</crviewer:viewer>
```

```
</body>
```

```
</html>
```

Viewing a report using an unmanaged RAS server

This sample shows you how to use the Viewer Tag Library to display a report using an unmanaged Report Application Server.

To view a report

1. Ensure that the appropriate reference to the Viewer Tag Library's descriptor file is added at the top of your JSP page.

```
<%@ taglib uri="/crystal-tags-reportviewer.tld" prefix="crviewer" %>
```

Note: You can specify any name for the prefix attribute. This just determines what tag prefix you use to access the tag library's tags.

2. Set system property to specify the location of the clientSDKOptions.xml file.

This file is used to set the location of the RAS server.

```
<%
    System.setProperty("ras.config","C:\\temp");
%>
```

Note: You can also set the location of the clientSDKOptions.xml file in the classpath.

3. Create an opening viewer tag.

In this tag, you must specify the viewer name and the type report source that is being used. In this case, the reportApplicationServer report source is being used.

```
<html>
<body>
<crviewer:viewer viewerName="CrystalViewer"
reportSourceType="reportApplicationServer" >
```

4. Create the required report tag, specifying the report to be displayed.

The reportName value is prefixed with rassdk:// so that the RAS server can correctly locate the file.

```
<crviewer:report reportName="rassdk://c:/reports/sample.rpt"/>
```

5. Close the viewer tag.

The JSP page is now ready to view the sample.rpt report.

```
</crviewer:viewer>
</body>
</html>
```

Viewing managed reports

Although there are no tags that provide direct support for viewing managed reports, a combination of tags and JSP code can provide the equivalent functionality. Viewing managed reports requires you to have either a Crystal Enterprise deployment or Report Application Server running.

Click the appropriate link to jump to that section:

- Viewing a report using a Page Server
- Viewing a report using a managed RAS server

Viewing a report using a Page Server

This sample shows you how to use the Viewer Tag Library to display a report using a Page Server report source.

To view a report using a Page Server

1. Log on to Crystal Enterprise and get an InfoStore object.

In this case, the default administrator account is used to log on to a Crystal Enterprise installation running on the same machine as the application server.

<%

```
    IEnterpriseSession es =
    CrystalEnterprise.getSessionMgr().logon("administrator", "",
    "localhost", "secEnterprise");
```

```
    IInfoStore infoStore = (IInfoStore)
    es.getService("", "InfoStore");
```

2. Query for the report you wish to view.

```
    IInfoObjects infoObjects = infoStore.query("SELECT * FROM
    CI_INFOOBJECTS WHERE SI_NAME='World Sales Report'");
```

```
    IInfoObject report = (IInfoObject) infoObjects.get(0);
```

3. Create Page Server report source for the report you retrieved.

```
    IReportSourceFactory2 rsFactory = (IReportSourceFactory2)
    es.getService("PSReportFactory");
```

```
    IReportSource rptSrc = rsFactory.createReportSource(report,
    request.getLocale());
```

4. Cache the ReportSource object in a session variable.

IDE Extensions

This allows it to be retrieved by the viewer.

```
session.setAttribute("ReportSource", rptSrc);  
  
%>
```

5. Create a viewer tag, specifying the viewer name, report source type, and report source variable.

In this case, the pageServer report source is being used and the report source variable refers to the session variable where you stored the ReportSource object.

```
<html>  
  
<body>  
  
<crviewer:viewer viewerName="CrystalViewer"  
reportSourceType="pageServer" reportSourceVar="ReportSource" />  
  
</body>  
  
</html>
```

Note: In this case, a report tag is not needed because the report source has already been cached in a session variable and does not to be retrieved by the viewer.

Viewing a report using a managed RAS server

This sample shows you how to use the Viewer Tag Library to display a report using a managed Report Application Server report source.

To view a report using a managed RAS server

1. Log on to Crystal Enterprise and get an InfoStore object.

In this case, the default administrator account is used to log on to a Crystal Enterprise installation running on the same machine as the application server.

```
<%  
  
    IEnterpriseSession es =  
    CrystalEnterprise.getSessionMgr().logon("administrator", "",  
    "localhost", "secEnterprise");  
  
    IInfoStore infoStore = (IInfoStore)  
    es.getService("", "InfoStore");
```

2. Query for the report you wish to view.

IDE Extensions

```
IInfoObjects infoObjects = infoStore.query("SELECT * FROM  
CI_INFOOBJECTS WHERE SI_NAME='World Sales Report'");
```

```
IInfoObject report = (IInfoObject) infoObjects.get(0);
```

3. Create a RAS report source for the report you retrieved.

```
IReportSourceFactory2 rsFactory = (IReportSourceFactory2)  
es.getService("RASReportFactory");
```

```
IReportSource rptSrc = rsFactory.createReportSource(report,  
request.getLocale());
```

4. Cache the ReportSource object in a session variable.

This allows it to be retrieved by the viewer.

```
session.setAttribute("ReportSource", rptSrc);
```

```
%>
```

5. Create a viewer tag, specifying the viewer name, report source type, and report source variable.

In this case, the reportApplicationServer report source is being used and the report source variable refers to the session variable where you stored the ReportSource object.

```
<html>
```

```
<body>
```

```
<crviewer:viewer viewerName="CrystalViewer"  
reportSourceType="pageServer" reportSourceVar="ReportSource" />
```

```
</body>
```

```
</html>
```

Note: In this case, a report tag is not needed because the report source has already been cached in a session variable and does not to be retrieved by the viewer.

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Welcome to Crystal Reports for BEA WebLogic Workshop

This section briefly describes Crystal Reports for BEA WebLogic Workshop and outlines the contents and the intended audience of this online help. Product registration and technical support information is also included, along with a brief description of the document conventions used within this online help.

Click the appropriate link to jump to that section:

- What is Crystal Reports for BEA WebLogic Workshop?
- Who should use this online help?
- About this online help
- Product registration
- Crystal Care technical support
- Crystal Training
- Crystal Consulting
- Document conventions
- Trademark acknowledgements

What is Crystal Reports for BEA WebLogic Workshop?

Crystal Reports for BEA WebLogic Workshop integrates Crystal Decisions' Java report processing and rendering capabilities with BEA WebLogic Workshop's Java Server Pages development environment. It allows you to quickly and simply add Crystal report viewing functionality to your J2EE applications, minimizing hand coding of data-connectivity and presentation formatting.

Who should use this online help?

This online help is intended for BEA WebLogic Workshop users who want to include Crystal report viewing functionality in their projects. This online help is intended for J2EE developers who are familiar with BEA WebLogic Workshop and have a solid understanding of Java.

For more information about Crystal Reports for BEA WebLogic Workshop, consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide*.

About this online help

This online help provides you with procedures for ensuring Crystal Reports for BEA WebLogic Workshop is correctly installed and how to download and install the Crystal Reports for BEA WebLogic Workshop Designer. It also shows you how to use Crystal Reports for BEA WebLogic Workshop to integrate report viewing into BEA WebLogic Workshop projects.

Section contents

The following is a short description of each of the remaining sections in this online help.

Getting Started

This section shows you how to set up Crystal Reports for BEA WebLogic Workshop. It shows you how to ensure that Crystal Reports for BEA WebLogic Workshop is installed correctly, and how to obtain and install a copy of Crystal Reports. If Crystal Reports for BEA WebLogic Workshop is not installed, this section shows you how to update your BEA WebLogic Workshop installation to include it.

Using Crystal Reports for BEA WebLogic Workshop

This section shows you the various ways you can use Crystal Reports for BEA WebLogic Workshop to add Crystal report viewing functionality to your BEA WebLogic Workshop projects. It guides you through the process of adding new or existing reports to your projects, editing reports you've added to your projects, and provides important information on report migration.

Product registration

Product registration can be accessed at any time by following the on–screen registration prompts that appear when using Crystal Reports for BEA WebLogic Workshop components.

Product registration allows you to keep up to date with product advancements and gives you access to:

- A copy of the Crystal Reports for BEA WebLogic Workshop Designer, a customized desktop report designer.
- Two free technical support incidents.
- JSP and report samples.
- An upgrade utility to migrate legacy reports to the Crystal Reports 9 format.

Crystal Reports for BEA WebLogic Workshop is compatible with the Crystal Reports version 9 and newer report formats.

Once you have completed the online registration process, an email is sent to you. This email provides you with:

- A registration number.
- Keycode information.
- Instructions for downloading and registering the Crystal Reports for BEA WebLogic Workshop Designer.
- Technical support information.

This includes instructions on how to access technical support resources.

- Instructions on accessing the Developer Zone.

IDE Extensions

The Developer Zone gives you access to a variety of developer resources, including sample code, tutorials, and discussion forums.

Your product registration number and the keycode information provided by the registration process are needed in order to install and activate your copy of the Crystal Reports for BEA WebLogic Workshop Designer. Completing the activation process finalizes your product registration.

Crystal Care technical support

Your copy of Crystal Reports for BEA WebLogic Workshop entitles you to two free technical support incidents. In order to receive this support, your copy of Crystal Reports for BEA WebLogic Workshop must be registered. For more information about product registration, refer to Product registration.

To find out about the technical support programs available for Crystal Reports for BEA WebLogic Workshop:

- Go to our support web site at:

<http://support.crystaldecisions.com/crystalcare/>

- Contact your regional office. For details, go to:

<http://www.crystaldecisions.com/contact/offices.asp>

Crystal Training

Whether you're a developer, information technology professional, or business user, we offer a wide range of Crystal Reports training courses designed to build or enhance your existing skills. Courses are available online, at certified training centers, or at your own site:

- For a complete list of training courses and special offers, visit:

<http://www.crystaldecisions.com/training/>

- Or contact your regional office. For details, go to:

<http://www.crystaldecisions.com/offices/>

Crystal Consulting

Our global team of certified consultants and consulting partners can guide you through a corporate-wide solution strategy, design, integration and deployment the fastest results, maximum performance, and increased productivity.

- To learn more, visit:

<http://www.crystaldecisions.com/consulting/>

- Or contact your regional office. For details, go to:

<http://www.crystaldecisions.com/offices/>

Document conventions

This online help uses the following conventions:

- Commands and buttons

For easy recognition within procedures, User Interface (UI) features appear in bold type. For example: On the **File** menu, click **New**.

- Keyboard shortcuts

Delete means the **Delete** key, or the **Del** key on your numeric keypad. Enter means the **Enter**, **Return**, or **CR** key, depending on which of these keys appears on your keyboard.

- Key combinations

CTRL+KEY, **SHIFT+KEY**, and **ALT+KEY** are examples of key combinations. Hold down the first key in the combination and, at the same time, press the second key in the combination (designated above as KEY). For example: **CTRL+C** means hold the **Control** key down and press the letter **C** on your keyboard (**CTRL+C** is the Windows Copy command).

- Key terms are italicized when first defined.
- Monospaced font indicates file paths and names, lines of code, and data that you enter using your keyboard (for example: In the Formula Editor, type `If Sales > 1000 Then crRed`).
- Monospaced, italicized font indicates variable data that you must replace with data appropriate to your current settings, environment, or task. For example, in the following URL, you would replace *webserver*:

`http://webserver/crystal/enterprise/`

Trademark acknowledgements

Crystal Decisions, Crystal Reports, Crystal Enterprise, Crystal Analysis, Crystal Services, Crystal Care, Crystal Assist, Crystal Applications, Info and Holos are trademarks or registered trademarks of Crystal Decisions, Inc. in the U.S. and/or other countries. All other trademarks or registered trademarks referenced are the property of their respective owners.

All BEA brand and product names are trademarks or registered trademarks of BEA Systems, Inc. in the United States and other countries.

Getting Started

This section shows you how to set up Crystal Reports for BEA WebLogic Workshop. It shows you how to ensure that Crystal Reports for BEA WebLogic Workshop is installed correctly, and how to obtain and install a copy of the Crystal Reports for BEA WebLogic Workshop Designer. If Crystal Reports for BEA WebLogic Workshop is not installed, this section shows you how to update your BEA WebLogic Workshop installation to include it.

Click the appropriate link to jump to that section:

- [Setting up Crystal Reports for BEA WebLogic Workshop](#)
- [Setting up the Crystal Reports for BEA WebLogic Workshop Designer](#)

Setting up Crystal Reports for BEA WebLogic Workshop

Crystal Reports for BEA WebLogic Workshop allows you to quickly add support for viewing Crystal report files to your J2EE applications. The following components must be installed before you can use Crystal Reports for BEA WebLogic Workshop:

- **BEA WebLogic Workshop**

For more information on how to install BEA WebLogic Workshop, consult your BEA WebLogic Workshop documentation.

- **Crystal Reports for BEA WebLogic Workshop**

To learn how to determine whether Crystal Reports for BEA WebLogic Workshop is installed, see [Ensuring Crystal Reports for BEA WebLogic Workshop is installed](#).

- **Crystal Reports for BEA WebLogic Workshop Designer**

For help with obtaining and installing a copy of Crystal Reports for BEA WebLogic Workshop Designer, see [Setting up the Crystal Reports for BEA WebLogic Workshop Designer](#).

Ensuring Crystal Reports for BEA WebLogic Workshop is installed

Crystal Reports for BEA WebLogic Workshop is installed as part of the BEA WebLogic Workshop installation. You can verify that Crystal Reports for BEA WebLogic Workshop is installed by checking to see if you can access the additional functionality added to BEA WebLogic Workshop as part of the Crystal Reports for BEA WebLogic Workshop install process. If the functionality is not existent, you can run the Crystal Reports for BEA WebLogic Workshop installer.

To ensure Crystal Reports for BEA WebLogic Workshop is installed

1. Select a web project in BEA WebLogic Workshop.

Note: A web project has a small green globe icon.

2. Right-click the web project, and then select **Install**.

The menu displays a list of the items that you can install for this particular web project. The **Crystal Reports** option should appear.

If no option to install **Crystal Reports** appears, Crystal Reports for BEA WebLogic Workshop is not installed. You can add Crystal Reports for BEA WebLogic Workshop to an existing BEA WebLogic Workshop installation through the Crystal Reports for BEA WebLogic Workshop installer.

To add Crystal Reports for BEA WebLogic Workshop to an existing BEA WebLogic Workshop installation

1. Double-click on the Crystal Reports for BEA WebLogic Workshop installer executable.

Navigate through the prompts by clicking **Next**, until you are prompted for the **Product Key Code**.

2. Enter the **Product Key Code** into the field and click **Next**.
3. The installer prompts you for the BEAHOME directory. Make sure that the correct directory is entered and click **Next**.
4. You are asked to review your installation settings. Click **Install** to complete the installation.

Setting up the Crystal Reports for BEA WebLogic Workshop Designer

A customized version of the Crystal Reports for BEA WebLogic Workshop Designer is included with your copy of BEA WebLogic Workshop. You can obtain your copy through a simple web download process. Installing the Crystal Reports for BEA WebLogic Workshop Designer gives you the ability to create new reports and modify existing reports. On Windows systems, when editing a report in BEA WebLogic Workshop, the Crystal Reports for BEA WebLogic Workshop Designer is automatically launched, integrating the process of report creation, design, and deployment.

Click the appropriate link to jump to that section:

- Downloading Crystal Reports for BEA WebLogic Workshop Designer
- Installing Crystal Reports for BEA WebLogic Workshop Designer

Downloading Crystal Reports for BEA WebLogic Workshop Designer

To download the Crystal Reports for BEA WebLogic Workshop Designer, you must complete the Crystal Reports for BEA WebLogic Workshop product registration. You can register online at <http://www.crystaldecisions.com/beaweb/>. Registering gives you access to the Crystal Reports for BEA WebLogic Workshop Designer download location, and provides you with the registration number and keycode required by the installer.

For more information on registering Crystal Reports for BEA WebLogic Workshop, see Product registration.

Installing Crystal Reports for BEA WebLogic Workshop Designer

After you have downloaded your copy of Crystal Reports for BEA WebLogic Workshop Designer, you can install it by running the installation file and following the prompts. You need your Crystal registration number and keycode in order to complete the installation. Further information on how to use Crystal Reports for BEA WebLogic Workshop Designer can be found in the Crystal Reports for BEA WebLogic Workshop Designer's documentation which is installed with the Crystal Reports for BEA WebLogic Workshop Designer.

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Using Crystal Reports for BEA WebLogic Workshop

This section shows you how to add report viewing functionality to your BEA WebLogic Workshop projects. It shows you how to add new or existing reports to your projects. In addition, it shows you how to edit reports using the Crystal Reports for BEA WebLogic Workshop Designer. A migration section is included for reports created with Crystal Reports version 8.5 or earlier.

Click the appropriate link to jump to that section:

- Adding a report to your project
- Adding a viewer to your project
- Editing reports
- Migrating reports

Adding a report to your project

Crystal Reports for BEA WebLogic Workshop allows you to add report files to your web projects. Adding report files to your BEA WebLogic Workshop project enables them to be managed as part of your project, and deployed with your web applications. Both existing reports and new reports can be added.

Click the appropriate link to jump to that section:

- Adding a new report
- Adding an existing report

Adding a new report

BEA WebLogic Workshop integrates the process of creating a new report in your web projects. If you are working on a Windows-based system, you can launch the Crystal Reports for BEA WebLogic Workshop Designer so you can begin editing your report immediately.

To add a new report

1. Ensure that a web project is open.
2. On the **File** menu, select **New > Crystal report**.

The New File dialog box is displayed.

3. In the left pane of the **New File** dialog box, select **Web User Interface**.
4. In the right pane, select **Crystal report**.
5. Specify the name and location of the new report in the **File name** field.

Note: By default, the report is created in your BEA WebLogic Workshop project's root directory. Use the Browse button to designate another location.

6. Click **Create**.

The designer is now launched. While the designer is running, a message in the main window informs you that the report is being edited.

Adding an existing report

You can add your existing reports to your BEA WebLogic Workshop projects and use them without further modification.

If your report was created with Crystal Reports version 8.5 or earlier, it must be updated using the report migration tool. The report migration tool updates the report format to ensure it is compatible with the additional features provided in Crystal Reports 9 and above. Use the report migration tool to ensure that your existing reports display correctly. For more information about updating existing reports, see [Migrating reports](#).

To add an existing report

1. On the **File** menu, select **Import Files**.
2. Browse for the .rpt file you want to add.
3. Click **Import**.

The report is added to your project.

Adding a viewer to your project

The Java viewer allows reports to be displayed in a standard web browser. The Crystal Reports Viewer Wizard is the most efficient way to add a report viewer to your JSP page. The wizard uses the Viewer Tag Library to automatically insert and set all necessary tags. It also ensures that all dependencies are added correctly.

While you can use the wizard to simplify the process, you can also insert and set viewer tags manually.

Note that the tags encapsulate commonly used functions of the Viewer Java SDK. Using the Viewer Java SDK allows you to access the actual classes and methods that drive the report viewing process, giving you the greatest amount of flexibility and configurability.

Click the appropriate link to jump to that section:

- [Installing the Viewer for Your Web Project](#)
- [Viewer Tag Library](#)
- [Crystal Reports Viewer Wizard](#)
- [Viewer Java SDK](#)

Installing the Viewer for Your Web Project

The viewer is not installed by default for each web project. To use the Crystal Reports Viewer Wizard or Viewer Tag Library, you must first install the viewer for your particular project.

To install the viewer

- On the **Application** tab, right-click the web project folder and select **Install > Crystal Reports**.

Viewer Tag Library

The Viewer Tag Library allows you to access the report viewer through modular and reusable tags. Using tags allows you to reduce the amount of embedded Java code in a JSP page by moving the functional implementation into the tag and its implementing class. Doing so effectively abstracts the details of the code from the presentation of the actual page, allowing web page authors to concentrate on developing JSP pages. Using the Viewer Tag Library allows you to access the functionality of the Crystal report viewers without writing Java code. For more information on how to use the Viewer Tag Library, consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide*.

Crystal Reports Viewer Wizard

The Crystal Reports Viewer Wizard allows you to add a viewer to your JSP page. The wizard guides you through adding report functionality, and automatically inserts the viewer tag code.

Tag properties and values displayed in the wizard are the default values; if you do not explicitly set a value, the attribute is not set in the inserted code.

To insert a viewer using the wizard

1. Open the JSP page where you want to add the viewer.
2. On the **Palette** tab, drag the Viewer tag into the page where you want your report to be displayed.

This starts the Crystal Reports Viewer Wizard.

3. Click the **Browse (...)** button next to the **Existing Report** field.
4. Browse for the .rpt file you want to display.
5. Click **Open**.
6. Type a name into the **Viewer Name** field.

The name uniquely identifies this particular instance of the viewer. If you have multiple viewers on the same page, ensure that their names are unique.

7. Click **Next**.

You can customize the appearance and behavior of the viewer. Use the tabs to navigate between categories, and set values.

8. When you are finished customizing the viewer, click **Finish**.

The viewer code required to display the report you specified is inserted.

Viewer Java SDK

The Viewer Tag Library only encapsulates commonly used functions of the Viewer Java SDK. The Crystal Reports functionality provided by Crystal Reports for BEA WebLogic Workshop can also be accessed directly through the Viewer Java SDK. The SDK provides access to classes and methods that customize the report viewing functionality provided by the viewer. These classes and methods allow you to enable and disable viewer functionality, set default parameter values, specify database logons, and configure a wide range of different report viewing options. For more information on how to use the Viewer Java SDK, consult the *Crystal Reports for BEA WebLogic Workshop Developer's Guide*.

Editing reports

Crystal Reports for BEA WebLogic Workshop also allows you to edit reports you have added to your projects. Although report editing cannot be done directly in the BEA WebLogic Workshop environment, on Windows systems the reports can be edited through the Crystal Reports for BEA WebLogic Workshop Designer.

To edit a report

- Double-click the report file.

The Crystal Reports for BEA WebLogic Workshop Designer launches. While the designer is running, a message in the main window informs you that the report is being edited.

Migrating reports

Reports created in Crystal Reports version 8.5 or earlier require the report migration tool. The migration tool converts the reports to a format that is compatible with all versions of Crystal Reports 9 and above. You can convert the report to either version 9 or 10. Make sure to convert all pre-9 reports before deploying them with your web applications to ensure that they will be displayed correctly.

You can download Crystal Reports for BEA WebLogic Workshop Designer and sample reports when you register your copy of Crystal Reports for BEA WebLogic Workshop. For more information on registering your product, see Product registration.

To convert your reports

1. Run the Report Converter Wizard.
2. Select whether to convert a single report or multiple reports, then click **Next**.

You are then prompted to specify the location of the reports you wish to convert.

3. Select the report(s) you wish to convert. Click **Next**.

If you chose to convert multiple reports, you can add multiple directories containing reports that you wish to convert. Selecting Include Subdirectories before clicking Add Directory causes the Report

IDE Extensions

Converter Wizard to recursively search through subdirectories for reports.

4. Specify the output location for the converted report(s). Click **Next** to begin the conversion process.

You can choose to overwrite the existing reports or output the converted reports to a new location.

For multiple reports, prefixing or suffixing the report names with v9 allows them to be written to the same directory without overwriting the old reports.

5. Click **Finish** to close the Report Converter Wizard.

If there were any errors during the conversion process, click View Log to see them.

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

A

audience, intended

C

Crystal Reports for BEA WebLogic Workshop

installing

setup

Crystal Reports for BEA WebLogic Workshop Designer

downloading

installing

setup

Crystal Reports Viewer Wizard

D

document conventions

G

Getting Started

P

product registration

R

registration, of product

reports

converting

editing

migrating

reports, adding

existing reports

new reports

S

setup

Crystal Reports for BEA WebLogic Workshop

Crystal Reports for BEA WebLogic Workshop Designer

support, technical

T

technical support

Trademark acknowledgements

V

viewer

adding

sdk

tag library

wizard

installing

Viewer Java SDK

Viewer Tag Library

W

web sites

consulting

product registration

technical support

training

Welcome

This section briefly describes the Java viewer, and outlines the contents and intended audience of this online help. Product registration and technical support information is also included, along with a brief description of the document conventions used within this online help.

Click the appropriate link to jump to that section:

- [What is the Java Crystal report web viewer?](#)
- [Who should use this online help?](#)
- [About this online help](#)
- [Product registration](#)
- [Crystal Care technical support](#)
- [Crystal Training](#)
- [Crystal Consulting](#)
- [Document conventions](#)
- [Trademark acknowledgements](#)

What is the Java Crystal report web viewer?

The Java viewer is a collection of Java classes that are used primarily for generating HTML and Dynamic HTML (DHTML) pages that display Crystal reports.

As a web application developer, you can include these classes in your Java Server Pages (JSP), and programmatically manipulate them to send customized reports to your users' web browsers. The viewer is entirely thin—client technology of your web application don't need to install any additional software on their computers in order to view the reports.

Besides providing the web-based report viewer, the Viewer Java SDK also provides you with the ability to programmatically export Crystal reports to a variety of other formats. See [Export formats](#).

The Java viewer included with Crystal Reports for BEA WebLogic Workshop is complemented by other Java viewers available with other Crystal products. These viewers fit into the same framework and rely on the same classes, but support additional features such as boolean searches, report parts, and mapping.

The viewer has been specifically designed to work with the Java Reporting Component included with Crystal Reports for BEA WebLogic Workshop, as well as Crystal Enterprise and the Report Application Server.

Who should use this online help?

This online help is intended to help you develop web applications that use Java Server Pages (JSP) to access and display Crystal reports. It is, therefore, assumed that you have knowledge of the fundamentals of JSP, HTML, and DHTML.

About this online help

This online help provides the information necessary to help you to get started with the Viewers Java SDK. It includes code samples and a number of step-by-step examples that you can follow in order to help you integrate the Java viewer in your own web applications.

Section contents

The following is a short description of each of the remaining sections in this online help.

Architecture

This section introduces the Viewer Java SDK, and describes the components it interacts with in order to create HTML and DHTML report views. An overview of how the complete system fits together is followed by details of the Viewer Java SDK itself.

Viewer Functionality

This section first provides a brief description of the individual features available to the viewer. The section then provides information about the various formats to which the SDK can programmatically export Crystal reports.

Setting up the Development Environment

This section contains information that helps you set up your computer so that you can start using the Viewer Java SDK. It provides an overview of the software that must be installed before you can use the SDK, as well as which Java components are part of the SDK and where you can find them.

Getting Started

This section is designed to help you get started using the Viewer Java SDK to develop web-based applications. It provides a series of tutorials that guide you through the initial steps of creating the code to obtain a report source, then moves on to show you how to incorporate the viewer and the export control into your Java Server Pages (JSP).

SDK Reference

This section contains reference material for the viewer. This material supplements the API documentation provided by the *Crystal Reports for BEA WebLogic Workshop Java API Reference*. It contains essential sample code for ensuring the best performance from the viewer and some simple troubleshooting tips.

Viewer Tag Library

This section first provides a brief overview of the Viewer Tag Library, including how to set up your JSP pages to use the tag library. The section then provides a list of all the tags available, their specific uses, and relevant examples.

Product registration

Product registration can be accessed at any time by following the on–screen registration prompts that appear when using Crystal Reports for BEA WebLogic Workshop components.

Product registration allows you to keep up to date with product advancements and gives you access to:

- A copy of the Crystal Reports for BEA WebLogic Workshop Designer, a customized desktop report designer.
- Two free technical support incidents.
- JSP and report samples.
- An upgrade utility to migrate legacy reports to the Crystal Reports 9 format.

Crystal Reports for BEA WebLogic Workshop is compatible with the Crystal Reports version 9 and newer report formats.

Once you have completed the online registration process, an email is sent to you. This email provides you with:

- A registration number.
- Keycode information.
- Instructions for downloading and registering the Crystal Reports for BEA WebLogic Workshop Designer.
- Technical support information.

This includes instructions on how to access technical support resources.

- Instructions on accessing the Developer Zone.

The Developer Zone gives you access to a variety of developer resources, including sample code, tutorials, and discussion forums.

Your product registration number and the keycode information provided by the registration process are needed in order to install and activate your copy of the Crystal Reports for BEA WebLogic Workshop Designer. Completing the activation process finalizes your product registration.

Crystal Care technical support

Your copy of Crystal Reports for BEA WebLogic Workshop entitles you to two free technical support incidents. In order to receive this support, your copy of Crystal Reports for BEA WebLogic Workshop must be registered. For more information about product registration, refer to Product registration.

IDE Extensions

To find out about the technical support programs available for Crystal Reports for BEA WebLogic Workshop:

- Go to our support web site at:

<http://support.crystaldecisions.com/crystalcare/>

- Contact your regional office. For details, go to:

<http://www.crystaldecisions.com/contact/offices.asp>

Crystal Training

Whether you're a developer, information technology professional, or business user, we offer a wide range of Java Crystal report web viewers training courses designed to build or enhance your existing skills. Courses are available online, at certified training centers, or at your own site:

- For a complete list of training courses and special offers, visit:

<http://www.crystaldecisions.com/training/>

- Or contact your regional office. For details, go to:

<http://www.crystaldecisions.com/offices/>

Crystal Consulting

Our global team of certified consultants and consulting partners can guide you through a corporate-wide solution strategy, design, integration and deployment the fastest results, maximum performance, and increased productivity.

- To learn more, visit:

<http://www.crystaldecisions.com/consulting/>

- Or contact your regional office. For details, go to:

<http://www.crystaldecisions.com/offices/>

Document conventions

This online help uses the following conventions:

- Commands and buttons

For easy recognition within procedures, User Interface (UI) features appear in bold type. For example: On the **File** menu, click **New**.

IDE Extensions

- Keyboard shortcuts

Delete means the **Delete** key, or the **Del** key on your numeric keypad. Enter means the **Enter**, **Return**, or **CR** key, depending on which of these keys appears on your keyboard.

- Key combinations

CTRL+KEY, **SHIFT+KEY**, and **ALT+KEY** are examples of key combinations. Hold down the first key in the combination and, at the same time, press the second key in the combination (designated above as KEY). For example: **CTRL+C** means hold the **Control** key down and press the letter **C** on your keyboard (**CTRL+C** is the Windows Copy command).

- Key terms are italicized when first defined.
- Monospaced font indicates data that you enter using your keyboard. For example: In the Formula Editor, type `If Sales > 1000 Then crRed`
- Monospaced, italicized font indicates variable data that you must replace with data appropriate to your current settings, environment, or task. For example, in the following URL, you would replace *webserver*:

`http://webserver/crystal/enterprise/`

Trademark acknowledgements

Crystal Decisions, Crystal Reports, Crystal Enterprise, Crystal Analysis, Crystal Services, Crystal Care, Crystal Assist, Crystal Applications, Info and Holos are trademarks or registered trademarks of Crystal Decisions, Inc. in the U.S. and/or other countries. All other trademarks or registered trademarks referenced are the property of their respective owners.

All BEA brand and product names are trademarks or registered trademarks of BEA Systems, Inc. in the United States and other countries.

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Architecture

This section introduces the Viewer Java SDK, and describes the components it interacts with in order to create HTML and DHTML report views. An overview of how the complete system fits together is followed by details of the Viewer Java SDK itself.

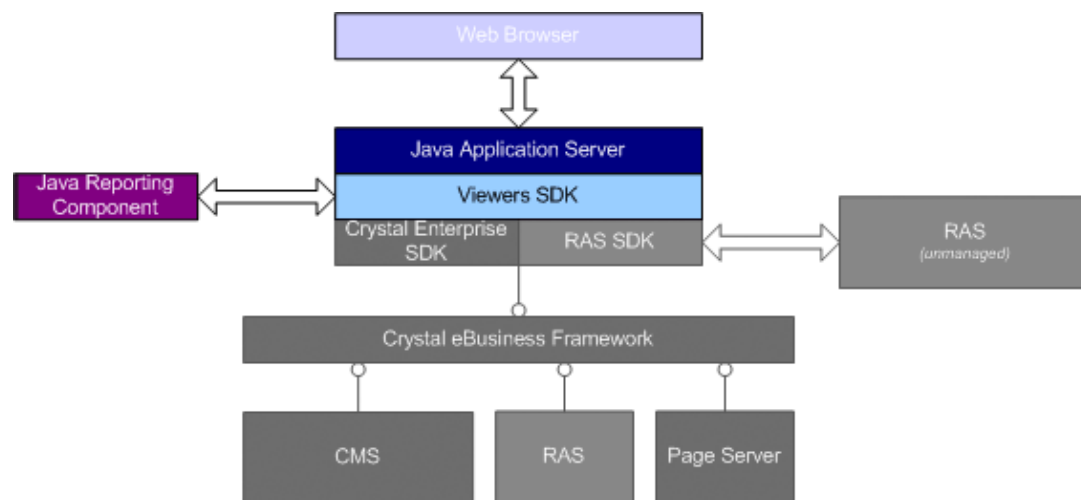
Click the appropriate link to jump to that section:

- [System overview](#)
- [Viewer Java SDK overview](#)

System overview

The Viewer Java SDK works together with the Java Reporting Component to allow reports to be viewed through a web browser. The Java Reporting Component is an all-Java component that enables the viewer to render Crystal Reports.

The diagram below illustrates the relationship between the Java Viewer SDK the Java Reporting Component, and other Crystal products:



Click the appropriate link to learn more about that component:

- [Java Application Server](#)
- [Viewer Java SDK](#)
- [Java Reporting Component](#)

Components

Java Application Server

The Java Application Server hosts the web applications that you write. These applications consist of one or more Java Server Pages (JSP) that contain code which uses the Java classes from Crystal Reports for BEA WebLogic Workshop and the Viewer Java SDK. In response to requests from a user's web browser, the server processes your JSPs to serve HTML and DHTML Crystal report views.

Viewer Java SDK

The Viewer Java SDK supplies the Java classes that enable you to programmatically generate HTML or DHTML for web delivery of Crystal report content. The viewer can be customized according to your needs, giving you full control over its display characteristics. Furthermore, the SDK allows you to programmatically export reports to a variety of formats.

See Viewer Java SDK overview for more information.

Java Reporting Component

The Java Reporting Component handles the report rendering process. It runs entirely on a Java Application Server and doesn't require any additional components, such as a Crystal Enterprise server or Report Application Server to run. It communicates with the viewer and data sources to produce HTML and DHTML Crystal report views.

Viewer Java SDK overview

The Viewer Java SDK allows you to build customized report viewers for displaying reports as HTML and Dynamic HTML (DHTML). It works together with the Java Reporting Component to allow Crystal reports to be viewed through a web browser using only a Java application server. As a web application developer, you can include the viewer in your Java Server Pages (JSP) and programmatically manipulate them to produce the output that you require. This allows you to minimize the time spent on presentation and formatting code by leveraging the viewer's ability to use existing reports to render formatted HTML.

The Viewer Java SDK allows you to control how a report is displayed and the features that are available to the user. It also allows you to connect to several report sources, allowing you to view reports included with your web application, as well as reports obtained from a Crystal Enterprise system or Report Application Server.

Besides providing a solution for report viewing, the Java Viewer SDK also allows you to add report exporting functionality to your web applications. This means that you can programmatically export Crystal reports to a range of different formats.

See Viewer Functionality for a complete description of the features that are available with the viewer, a list of the export formats supported, as well as typical usage scenarios for the viewer.

Click the appropriate link to jump to that section:

- [Report sources](#)
- [Java Reporting Component](#)

Report sources

A report source is an object used by the viewer to display the contents of the report. It is a pointer to the report instance that the viewer renders in HTML, and it provides the viewer with the means to query for report data, page information, and other internal report information.

The primary report source you will use with Crystal Reports for BEA WebLogic Workshop is the Java Reporting Component. This is an all-Java report source that is deployable with your web applications.

Although only the Java Reporting Component is provided with Crystal Reports for BEA WebLogic Workshop, other report sources can be used if you have Crystal Enterprise or a Report Application Server. With Crystal Enterprise and the Report Application Server, the Page Server and RAS server report sources are available. This allows the viewer to display reports that are accessible through a Crystal Enterprise Page Server or a RAS server. These report sources are not 100% Java-based solutions, as the underlying report rendering facilities are not written in Java. They do, however, provide additional features that are not available with the Java Reporting Component.

Some of the advantages of using a Page Server include support for caching, higher throughput, and greater scalability. Advantages of using a RAS server include support for report creation and modification, as well as additional viewers.

Java Reporting Component

When creating a Java Reporting Component report source, the report's path can be specified in several different ways:

- **File path**

This is an absolute path that specifies a report located on the machine running the Java Reporting Component.

For example, `C:\reports\sample.rpt`.

- **UNC address**

This specifies a report file that is accessible through a network resource.

For example, `\\hostname\sample.rpt`.

- **Relative file path**

For example, `reports\sample.rpt` refers to a directory called reports in the same directory as the Java Reporting Component.

- **Relative URL**

This is a URL that is specified relative to the location of the Java Reporting Component.

For example, `../reports/sample.rpt` resolves to `/WEB-INF/reports/sample.rpt`

where the Java Reporting Component JAR file is in `/WEB-INF/lib`.

Java Reporting Component configuration

The behavior of relative paths can be modified by specifying the value of the `reportlocation` tag in the `CrystalReportEngine-config.xml`, which is located in your `WEB-INF` directory. The value of `reportlocation` must specify a location as an absolute path or as a path relative to the location of the Java Reporting Component, which is by default `WEB-INF/lib`. The value of the `reportlocation` tag becomes the new root directory that is used to determine the location of a report.

The `CrystalReportEngine-config.xml` file also allows you to configure the time out interval that determines when inactive report sources are freed up. This is necessary as inactive report sources still consume system resources such as database connections, server memory, and disk space used by temporary files. The time-out interval can be specified by setting the value of the `timeout` tag in the `CrystalReportEngine-config.xml` file. By default, the timeout interval is 10 minutes. The timeout interval only applies to inactive reports that are being processed are not timed out as a result of exceeding this value. Each time a report source request is successfully completed, the timeout timer is reset. If a report source is not used within the timeout interval, it is freed up and its resources are made available to other processes.

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Viewer Functionality

The Viewer Java SDK allows you to create web applications that can both display Crystal reports and export them to other formats.

This section first provides a brief description of the individual features available to the viewer. The section then provides information about the various formats to which the SDK can programmatically export Crystal reports.

Click the appropriate link to jump to that section:

- [Viewer features](#)
- [Export formats](#)
- [Additional viewer features](#)

Viewer features

Crystal Reports provides a wealth of features for data analysis and presentation. Ideally, these features should be preserved when a report is viewed through a web browser. The viewer included with Crystal Reports for BEA WebLogic Workshop allows you to view reports through the web, while maintaining core report features and functionality. The viewer provides thin-client report viewing capabilities, allowing you to view graphs, drill down, perform text searches, and even export reports to other formats. These features can be selectively enabled or disabled using the Viewer Java SDK, allowing report viewing to be tailored to numerous applications.

The viewers included with Crystal Enterprise and Report Application Server provide additional support for the more advanced features provided by Crystal Reports. For more information on the features supported by these viewers, see [Additional viewer features](#).

Click the appropriate link to jump to that section:

- [Viewer feature descriptions](#)
- [Viewer uses](#)
- [Locale-specific viewing](#)

Viewer feature descriptions

The following tables describe both the basic and advanced features of the viewer. These features can be enabled or disabled through the SDK, allowing you to customize the viewer to meet your specific needs.

Basic features

Basic Features	Description
Graphs	Allows graphs to be viewed and drilled down upon.
Drill Down	Allows drilling down on fields or summarized data.

IDE Extensions

Page Navigation	Navigates to different pages in the report.
Text Search	Searches report fields for the specified string.
Toggle Group Tree	Shows or hides the group tree.
Zoom	Zooms to the desired magnification.
Highlight	Highlights the drill-down object or the first occurrence of the search text.
Logon Prompt	Prompts for log on.
Can Change View	Allows web application users to change the view of a report by picking from the view list.

Advanced features

Advanced Features	Description
Subreports	Allows on-demand subreports, or subreports with saved data, to be viewed. Also allows drilling down on these subreports.
Parameter Prompt	Prompts for parameters.
Export Button	Provides a button to export a report. Supported export formats are PDF and RTF. See Export formats for more information.
Print Button	Provides a button that enables a report to be printed.
Navigation via Hyperlink	Allows hyperlink navigation to another page in the same report, or to a page in another report.

Viewer uses

The following list presents several common usage scenarios for the viewer.

- View a basic report from a web page.
- View a report from a web page and search for a text string.
- View a report from a web page and drill down for further detail.
- Prompt users for authentication information or parameters and pass the information to the report viewer.
- Pass a selection formula or report source to the viewer.
- Capture an event and display information accordingly.

Locale-specific viewing

The Viewer Java SDK can be used to enable locale-specific viewing for other languages when Microsoft Internet Explorer is used to access the viewer. Locale-specific viewing enables reports to be viewed with settings that are specifically tailored to the user's geographic location. This allows date and time formatting, special font sets, and currency symbols to be preserved. To use the Viewer Java SDK to display reports in other languages, the international fonts must be installed on the following machines:

- The Java Application Server.
- The machine on which Internet Explorer is running.

- The RAS or Page Server.

This is only required when these components are used as report sources.

Although some of the report contents (such as currency symbols and DateTime formatting) also rely on a locale setting, this setting does not need to be the same as the locale setting on the RAS server or Page Server. In other words, client locale settings are used when locale-specific viewing is enabled with Page Server or RAS report sources.

Tip: It is possible for the web browser's default language setting to be different from the system regional setting. For example, the system regional setting may be Chinese, but if the browser default language setting is English, the viewer is displayed in English.

To set the proper locale for the browser, select Internet Options from the Tools menu of Microsoft Internet Explorer, and click the Languages button. Move the language you want to use to the top of the list. To set the system regional setting, consult the documentation for your operating system.

Export formats

A powerful feature that the viewer provides is the ability to export Crystal report files to a variety of different formats. Reports can be viewed before being exported to a different format for either viewing on a local machine or for distributing. The SDK provides the ability to programmatically export a report in addition to the ability to export the report through the HTML viewer.

This table shows the export formats that are supported by the SDK.

Format Type	Export Format
RTF	Rich Text Format
PDF	Adobe PDF

While you can export a Crystal report to another format, you cannot do the reverse. For example, you cannot use the viewers SDK to transform an RTF file into a Crystal report.

Additional viewer features

The viewer provided with Crystal Reports for BEA WebLogic Workshop is complimented by other feature-rich viewers available as part of Crystal Enterprise and the Report Application Server. These viewers provide thin-client support for additional features in Crystal reports. Some of these additional features include:

- **Boolean Search**

Searches report fields using Boolean and comparison operators.

- **Maps**

IDE Extensions

Allows embedded maps to be viewed, and drilled down upon. This feature requires the mapping DLLs to be installed on the RAS server.

- **Mobile Output**

Produces output for mobile devices.

- **PDA Output**

Produces output for PDA devices.

- **Report Parts**

Allows individual report parts, such as charts, text, and fields, to be viewed.

The viewers included with Crystal Enterprise and the Report Application Server also provide support for additional report export formats, including Microsoft Word, Microsoft Excel, plain text, and character separated values.

Finally, these viewers support a special feature tag parsing. Crystal Reports smart tags allow you to integrate Crystal reporting technology into Microsoft Office XP documents. Users can embed a link to a report from Office XP programs by copying and pasting a report part into an Office XP document.

Click the appropriate link to jump to that section:

- [Additional functionality](#)
- [More information](#)

Additional functionality

The following table provides a look at the additional functionality available with the Crystal Enterprise and the Report Application Server viewers.

Additional Functionality	Required Component
View a report from a web page and search report fields using Boolean and comparison operators.	Java Interactive Viewer
View a report from a web page and drill down for further detail.	Java Report Page Viewer
View a chart from a web page and drill down for further detail.	Java Report Part Viewer
View a chart from a mobile device and drill down for further detail.	Java Report Part Viewer
View a chart from a Microsoft XP document and drill down for further detail.	Java Report Part Viewer (via smart tags)

More information

For more information on Crystal Enterprise, the Report Application Server, and their respective viewers, please visit <http://www.crystaldecisions.com/products/>

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Setting up the Development Environment

This section contains information that helps you set up your computer so that you can start using the Viewer Java SDK. It provides an overview of the software that must be installed before you can use the SDK, as well as which Java components are part of the SDK and where you can find them.

It is assumed that you are familiar with the basics of developing, packaging, and deploying Java 2 Platform, Enterprise Edition (J2EE) web components.

Click the appropriate link to jump to that section:

- [Prerequisite software](#)
- [Locating the SDK components](#)
- [Configuring data sources](#)

Prerequisite software

Before you can use the viewer, you must have a Java Application Server installed and configured. This can be installed on the computer that you use to develop or deploy your JSP pages, or it can be on another computer accessible through your network. Refer to your application server's documentation for installation and configuration details.

You must also ensure that BEA WebLogic Workshop and Crystal Reports for BEA WebLogic Workshop have been installed. This is required, as the JAR files that make up the viewer and the Java Reporting Component are installed as part of the Crystal Reports for BEA WebLogic Workshop install process. For more information on installing Crystal Reports for BEA WebLogic Workshop, see the *Crystal Reports for BEA WebLogic Workshop User's Guide*.

What components must be running?

Because the viewer and the Java Reporting Component are written entirely in Java, only your Java Application Server needs to be running in order to use the Viewer Java SDK to view and export reports. If you wish to use the Viewer Java SDK with other Crystal products such as Crystal Enterprise or a Report Application Server, however, you must ensure that the appropriate components for those products are running in order to access the features they provide.

Locating the SDK components

The Java classes that are specific to the Viewer Java SDK are packaged into an archive file named `webreporting-core.jar`. These core SDK classes, however, are dependent upon classes from several other JAR files and support files that are installed as part of Crystal Reports for BEA WebLogic Workshop.

On Windows systems, the JAR files can be found in `C:\Program Files\Common Files\Crystal Decisions\2.5\java\lib`. On Unix systems, the JAR files can be found in `crystal-root/java/lib`, where `crystal-root` is the directory where Crystal Reports for BEA WebLogic Workshop is installed. If you would like to use the Viewer Tag Library, ensure that the `taglib` directory is included with your application

as well.

Once you have located the JAR files, you must add them to your web application's `WEB-INF/lib` folder. The `CrystalReportEngine-Config.xml` file contained in the directory containing the JAR files must also be copied over. It contains configuration details for the Java Reporting Component. For more information on how to configure these settings, see [Java Reporting Component configuration](#).

Support files are also located under the same directory structure outlined above. The `crystalreportviewers10` directory is needed by the viewer for images, style sheets, and general print and export handling. This directory should be copied to your web application's root directory. You can specify the location of this directory by setting the `crystal_image_uri` context parameter in your `web.xml` file.

Configuring data sources

The Java Reporting Component makes use of JNDI to establish connections to data sources. It relies on the JNDI server running on your application server to determine how to connect to the data sources specified in a report. Once the connection information has been retrieved from the JNDI server, the Java Reporting Component uses the information to establish a JDBC connection to the data source. To ensure that the Java Reporting Component can successfully establish a connection when retrieving report data, ensure that the JNDI entries for the required data sources are correctly configured. For more information on how to configure a JNDI data source entry, please consult your application server's and JDBC driver's documentation.

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Getting Started

This section is designed to help you get started using the Viewer Java SDK to develop web-based applications. It provides a series of tutorials that guide you through the initial steps of creating the code to obtain a report source, then moves on to show you how to incorporate the viewer and the export control into your Java Server Pages (JSP).

Before reading this section, it is recommended that you read *Setting up the Development Environment*.

After the tutorials, a general overview of recommended practices to follow when using the Viewer Java SDK is provided. This includes tips on optimizing reports for web application performance and general coding guidelines.

Click the appropriate link to jump to that section:

- [Tutorials](#)
- [Best practices](#)

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Tutorials

This section contains a series of tutorials that show you how to use the Viewer Java SDK. The tutorials assume that you are familiar with JSP programming. Since each tutorial builds on the previous ones, it is recommended that you work through the series sequentially.

The first tutorial shows you how to create a report source programmatically. You must be able to create a report source before you can use the viewer or the export control in your JSP pages. The remaining tutorials then show you how to use the report source in conjunction with the export control and the viewer.

To jump to a particular tutorial, click the appropriate link:

- [How to create a report source](#)
- [How to use the Java viewer](#)
- [How to export a report](#)
- [How to set a parameter field](#)
- [How to set a database logon](#)

How to create a report source

Before you can use the export control or the viewer in your JSP pages, you must obtain a report source.

A report source is an object that represents a single instance of a report. It implements the `IReportSource` interface, which contains a set of methods that are used by both the export control and the viewer. The report source included with Crystal Reports for BEA WebLogic Workshop is the Java Reporting Component.

The tutorial contains the following sections:

- Creating a report source object

Creating a report source object

Creating a Java Reporting Component report source only requires you to have the location of the report you wish to view or export. The report location can be specified using a file system path, UNC address, or URL. For more information on the Java Reporting Component report source and specifying paths, see [Java Reporting Component](#). Also, for the Java Reporting Component to correctly retrieve data for a report, the report's data sources must be correctly specified through JNDI. For more information on data source configuration, see [Configuring data sources](#).

To create a report source object

1. Ensure that you have imported the `JPEReportSourceFactory` class and the `IReportSource` and `IReportSourceFactory2` interfaces.

```
<%@ page
import="com.crystaldecisions.reports.reportengineinterface.JPEReportSource
com.crystaldecisions.sdk.occa.report.reportsource.IReportSourceFactory2,
com.crystaldecisions.reports.reportengineinterface.IReportSource"
%>
```

2. Create a new `JPEReportSourceFactory` object.

```
IReportSourceFactory2 rptSrcFactory = new JPEReportSourceFactory();
```

3. Call the `IReportSourceFactory2` object's `createReportSource` method, passing it the path to the desired report and the current locale settings.

The `IReportSource` object can now be used by the viewer or export control.

```
String report = "/reports/sample.rpt";

IReportSource reportSource = (IReportSource)
rptSrcFactory.createReportSource(report, request.getLocale());
```

How to use the Java viewer

This tutorial demonstrates how to use the Java viewer programmatically to display a report in a web browser.

It is assumed that you have already created a report source. If you haven't, see [Creating a report source object](#) for details.

The tutorial contains the following sections:

- Creating and initializing a Java viewer
- Launching the Java viewer
- Performing garbage collection
- Example

Creating and initializing a Java viewer

The viewer is simply an instance of a `CrystalReportViewer` object. A `CrystalReportViewer` object has many set methods that affect how it displays reports in a web browser. Some of them need to be called before the viewer can render a report. When using the Java Reporting Component, you need to set the viewer's report source.

To create and initialize a Java viewer

1. Instantiate a `CrystalReportViewer` object.

```
CrystalReportViewer viewer = new CrystalReportViewer();
```

2. Set the viewer's report source by calling its `setReportSource` method, passing it a reference to a report source object.

The viewer has now been created and initialized.

```
viewer.setReportSource(reportSource);
```

Note: Once the viewer is created and initialized, you can set a variety of properties related to its display characteristics, database logon handling, and parameter prompting. For more information, see the `CrystalReportViewer` documentation in the *Crystal Reports for BEA WebLogic Workshop Java API Reference*.

Launching the Java viewer

Once you have created and initialized a Java viewer, call its `processHttpRequest` method to launch it in a web browser.

This example assumes that you have set up the viewer properly, and that you have valid request, response, and session objects.

To launch the Java viewer

1. Call the `processHttpRequest` method to launch the viewer in the current browser window.

```
viewer.processHttpRequest(request, response,  
getServletConfig().getServletContext(), null);
```

Note: If the viewer's content is going to be displayed more than once, using the `getHtmlContent` method is more efficient. This is because the request is processed only once and the resulting HTML string can be used multiple times.

Performing garbage collection

In order to ensure the maximum performance for your web applications, you must ensure that the `dispose` method is called whenever you are finished with the viewer. This ensures that any resources being tied up are freed for use by other applications.

To perform garbage collection for the viewer

1. Call the viewer's `dispose` method.

```
viewer.dispose();
```

Example

The following example is a simple JSP page that demonstrates how to use the viewer to display a simple report.

viewreport.jsp

```
<%@ page
import="com.crystaldecisions.report.web.viewer.CrystalReportViewer" %>

<%@ page
import="com.crystaldecisions.reports.reportengineinterface.JPEReportSourceFactory"
import="com.crystaldecisions.sdk.occa.report.reportsource.IReportSourceFactory2,
com.crystaldecisions.reports.reportengineinterface.IReportSource"
%>

<%

String report = "c:/reports/sample.rpt";

IReportSourceFactory2 rptSrcFactory = new JPEReportSourceFactory();

IReportSource reportSource = (IReportSource)
rptSrcFactory.createReportSource(report, request.getLocale());
```

```

CrystalReportViewer viewer = new CrystalReportViewer();

viewer.setReportSource(reportSource);

viewer.processHttpRequest(request, response,
getServletConfig().getServletContext(), out);

viewer.dispose();

%>

```

How to export a report

This tutorial shows you how to use the Export Control to export a Crystal report to one of several available formats. For the complete list of possible export formats, see [Export formats](#).

It is assumed that you have already created a report source. If you haven't, see [Creating a report source object](#) for details.

The tutorial contains the following sections:

- [Creating and initializing an Export Control](#)
- [Exporting the report](#)
- [Performing garbage collection](#)
- [Example](#)

Creating and initializing an Export Control

The export control handles all aspects of exporting the report. It allows you to preview the exported report within the browser window or export it as an attachment, prompting the user with a download dialog. The export control is represented by the `ReportExportControl` class.

To create an Export Control

1. Instantiate a `ReportExportControl` object.

```
ReportExportControl exportControl = new ReportExportControl();
```

Once you have created the `ReportExportControl` object, you must specify the export format that you want. For the purpose of this example, RTF has been chosen as the export format. For a complete list of export

formats, see the Export formats section.

To specify the export format

1. Create an ExportOptions object.

```
ExportOptions exportOptions = new ExportOptions();
```

2. Specify the export format by calling the ExportOptions object's setExportFormatType method, passing it a ReportExportFormat constant representing the desired format.

```
exportOptions.setExportFormatType(ReportExportFormat.RTF);
```

Note: A list of the valid constants specifying export formats can be found in the ReportExportFormat class documentation.

Some formats contain additional options that can be configured to customize how the report is exported. This includes control over what page range is exported and so on.

To configure format specific options

1. Create the appropriate format options object.

In this case, because the export format is RTF, a RTFWordExportFormatOptions object is created.

```
RTFWordExportFormatOptions RTFExpOpts = new  
RTFWordExportFormatOptions();
```

2. Configure the options you wish to set.

In this example, the export options are configured so that only pages 1 to 3 are exported.

```
RTFExpOpts.setStartPageNumber(1);
```

```
RTFExpOpts.setEndPageNumber(3);
```

3. Call the ReportExportOptions object's setFormatOptions method, passing it the format options object.

```
exportOptions.setFormatOptions(RTFExpOpts);
```

To initialize the Export Control

1. Set the control's report source by calling its setReportSource method and passing the method a reference to the report source object that you created.

```
exportControl.setReportSource(reportSource);
```

IDE Extensions

2. Call the control's `setExportOptions` method, passing it the `ExportOptions` object that you created earlier.

```
exportControl.setExportOptions(exportOptions);
```

3. You may also want to call the `setExportAsAttachment` method.

Setting this method to true causes the Export Control to display a dialog box that allows users of your web application to save the exported report before they open it. Otherwise, the exported report is displayed in the browser window directly.

```
exportControl.setExportAsAttachment(true);
```

Exporting the report

Once you have created and initialized an export control, calling its `processHttpRequest` method completes the export.

This example assumes that you have set up the export control properly, and that you have valid request, response, and session objects.

To export a report

- Call the `processHttpRequest` method to export the report.

If `setExportAsAttachment` is set to true, the user is prompted with a download dialog, otherwise the exported report is displayed directly in the browser.

```
exportControl.processHttpRequest(request, response,
    getServletContext(), null);
```

Note: If you intend to export a report more than once, using the `getHtmlContent` method is more efficient, because the request is processed once and the resulting HTML string can be used multiple times.

Performing garbage collection

In order to ensure the maximum performance for your web applications, you must ensure that the `dispose` method is called whenever you are finished with the export control. This ensures that any resources being tied up are freed for use by other applications.

To perform garbage collection for the viewer

1. Call the export control's `dispose` method.

```
exportControl.dispose();
```

Example

The following example is a simple JSP page that demonstrates how to use the export control to export the first 3 pages of a report to RTF.

exportreport.jsp

```
<%@ page
import="com.crystaldecisions.report.web.viewer.ReportExportControl" %>

<%@ page
import="com.crystaldecisions.reports.reportengineinterface.JPEReportSourceFactory"
import="com.crystaldecisions.sdk.occa.report.reportsource.IReportSourceFactory2,"
import="com.crystaldecisions.reports.reportengineinterface.IReportSource"
%>

<%@ page
import="com.crystaldecisions.sdk.occa.report.exportoptions.ExportOptions"
%>

<%@ page
import="com.crystaldecisions.sdk.occa.report.exportoptions.ReportExportFormat"
%>

<%@ page
import="com.crystaldecisions.sdk.occa.report.exportoptions.IRTFWordExportFormat"
%>

<%

String report = "c:/reports/sample.rpt";

IReportSourceFactory2 rptSrcFactory = new JPEReportSourceFactory();

IReportSource reportSource = (IReportSource)
rptSrcFactory.createReportSource(report, request.getLocale());

ReportExportControl exportControl = new ReportExportControl();

ExportOptions exportOptions = new ExportOptions();

exportOptions.setExportFormatType(ReportExportFormat.RTF);
```

```

RTFWordExportFormatOptions RTFExpOpts = new
RTFWordExportFormatOptions();

RTFExpOpts.setStartPageNumber(1);

RTFExpOpts.setEndPageNumber(3);

exportOptions.setFormatOptions(RTFExpOpts);

exportControl.setReportSource(reportSource);

exportControl.setExportOptions(exportOptions);

exportControl.setExportAsAttachment(true);

exportControl.processHttpRequest(request, response,
getServletConfig().getServletContext(), null);

exportControl.dispose();

%>

```

How to set a parameter field

This tutorial shows you the basic steps required to programmatically set parameter fields for a report that is displayed using the viewer. Setting the parameter fields for a report allows a report that contains parameter prompts to always be displayed using the same values. Also, setting parameter fields can be used to change the default values or modify the values as needed. The report used in this example is created in the Crystal Reports Designer and has two parameter fields: Region and Country Code. The default values of the parameter fields and the new values that the code sets them to are as follows:

- **Region** (String)

Default value: Canada

Set value: Japan

- **Country Code** (Number)

Default value: 1

Set value: 81

Note: This report is not included with the tutorial, but a simple report containing similar parameter fields and information can be easily created in the Crystal Reports Designer.

It is assumed that you have already created a report source. If you haven't, see [Creating a report source object](#) for details.

The tutorial contains the following sections:

- Creating and initializing parameter fields
- Setting parameter fields
- Example

Creating and initializing parameter fields

Before a parameter field can be set in a report, the fields to be set must first be created, then initialized. Individual parameter fields are all stored in a Fields object. The Fields object is simply a collection of different fields that can be passed to the viewer.

To create parameter fields

1. Create a Fields object to store the parameter fields in.

```
Fields fields = new Fields();
```

2. Create a ParameterField object for each field that you wish to set.

```
ParameterField pfield1 = new ParameterField();
```

```
ParameterField pfield2 = new ParameterField();
```

3. Create a Values object and a ParameterFieldDiscreteValue object for each parameter field you wish to set.

If a ranged value is being set, a ParameterFieldRangeValue object should be used instead of the discrete value object.

```
Values vals1 = new Values();
```

```
Values vals2 = new Values();
```

```
ParameterFieldDiscreteValue pfieldDV1 = new  
ParameterFieldDiscreteValue();
```

```
ParameterFieldDiscreteValue pfieldDV2 = new  
ParameterFieldDiscreteValue();
```

Once all the required objects have been created, the values for the fields can be initialized.

To initialize parameter fields

1. Set the name and value for each parameter field that is added.

Values for parameter fields are represented by a `ParameterFieldDiscreteValue` or `ParameterFieldRangeValue` object.

```
pfield1.setName("Region");  
  
pfieldDV1.setValue("Japan");  
  
pfieldDV1.setDescription("The region is Japan");
```

```
Integer CountryCode = new Integer("81");  
  
pfield2.setName("Country Code");  
  
pfieldDV2.setValue(CountryCode);  
  
pfieldDV2.setDescription("The country code is 81");
```

2. Add the parameter field values to the Values collection object.

```
vals1.add(pfieldDV1);  
  
vals2.add(pfieldDV2);
```

3. Set the current Values collection for each parameter field.

```
pfield1.setCurrentValues(vals1);  
  
pfield2.setCurrentValues(vals2);
```

4. Add each parameter field to the Fields collection.

The Fields object is now ready to be used with the viewer.

```
fields.add(pfield1);  
  
fields.add(pfield2);
```

Setting parameter fields

After all the parameter fields have been initialized and added to the Fields object, the Fields object can be passed to the viewer.

To set parameter fields

1. Create an instance of the viewer, passing it a reference to a report source object.

```
CrystalReportViewer viewer = new CrystalReportViewer();  
viewer.setReportSource(reportSource);
```

2. Set the parameter fields for the viewer by passing in the initialized Fields object.

User prompting should be disabled to automatically use the set parameter field value.

```
viewer.setParameterFields(fields);  
viewer.setEnableParameterPrompt(false);
```

3. Call the viewer's refresh method to apply new parameters.

```
viewer.refresh();
```

4. Call the processHttpRequest method to launch the viewer in the current browser window.

```
viewer.processHttpRequest(request, response,  
getServletConfig().getServletContext(), null);
```

5. Call the dispose method to allow the viewer to perform the appropriate garbage collection and free system resources.

```
viewer.dispose();
```

Example

The following example is a JSP page that demonstrates how to set two parameter fields for a report containing Region and Country Code parameter. After the parameters have been set, the report is displayed.

parameterFieldsViewReport.jsp

```
<%@ page import= "com.crystaldecisions.report.web.viewer.*",  
com.crystaldecisions.sdk.occa.report.data.*" %>  
  
<%@ page  
import="com.crystaldecisions.reports.reportengineinterface.JPEReportSourceFactory,  
com.crystaldecisions.sdk.occa.report.reportsource.IReportSourceFactory2,  
com.crystaldecisions.reports.reportengineinterface.IReportSource"  
%>
```

<%

```

String report = "c:/reports/sample.rpt";

IReportSourceFactory2 rptSrcFactory = new JPEReportSourceFactory();

IReportSource reportSource = (IReportSource)
rptSrcFactory.createReportSource(report, request.getLocale());


Fields fields = new Fields();


ParameterField pfield1 = new ParameterField();
ParameterField pfield2 = new ParameterField();


Values vals1 = new Values();
Values vals2 = new Values();


ParameterFieldDiscreteValue pfieldDV1 = new
ParameterFieldDiscreteValue();

ParameterFieldDiscreteValue pfieldDV2 = new
ParameterFieldDiscreteValue();


pfield1.setName("Region");
pfieldDV1.setValue("Japan");
pfieldDV1.setDescription("The region is Japan");


Integer CountryCode = new Integer("81");
pfield2.setName("Country Code");

```

IDE Extensions

```
pfieldDV2.setValue(CountryCode);

pfieldDV2.setDescription("The country code is 81");


vals1.add(pfieldDV1);

vals2.add(pfieldDV2);


pfield1.setCurrentValues(vals1);

pfield2.setCurrentValues(vals2);


fields.add(pfield1);

fields.add(pfield2);


CrystalReportViewer viewer = new CrystalReportViewer();

viewer.setReportSource(reportSource);


viewer.setParameterFields(fields);

viewer.setEnableParameterPrompt(false);


viewer.refresh();


viewer.processHttpRequest(request, response,
getServletConfig().getServletContext(), out);


viewer.dispose();


%>
```

How to set a database logon

This tutorial shows you the basic steps required to programmatically set a database logon. Almost all reports rely on a database to retrieve information, with the large majority of database connections requiring a user logon. While the viewer prompts the user for database logon information when it is needed, it is often desirable to set the database logon information for a report, as it simplifies the user's viewing experience.

It is assumed that you have already created a report source. If you haven't, see [Creating a report source object](#) for details.

The tutorial contains the following sections:

- Creating and initializing database logon information
- Setting database logon information
- Example

Creating and initializing database logon information

Database logon information is stored in a `ConnectionInfo` object. The `ConnectionInfo` object is then added to a `ConnectionInfos` collection object. This allows more than one database logon to be added, providing support for subreports with different database connections.

To create and initialize database logon information

1. Create an `ConnectionInfos` object to store the database logon information in.

```
ConnectionInfos connInfos = new ConnectionInfos();
```

2. Create a `ConnectionInfo` object for each database logon you wish to set.

```
IConnectionInfo connInfo1 = new ConnectionInfo();
```

```
IConnectionInfo connInfo2 = new ConnectionInfo();
```

Note: The interface is used to manipulate the `ConnectionInfo` object, as it simplifies the methods available and allows for future support of different types of `ConnectionInfo` objects.

3. Set the database logon information for each `ConnectionInfo` object.

In this case, a generic guest account is used for each.

```
connInfo1.setUsername("guest");
```

```
connInfo1.setPassword("password");
```

```
connInfo2.setUsername("guest");
```

IDE Extensions

```
connInfo2.setPassword("password");
```

4. Add each ConnectionInfo object to the ConnectionInfos collection.

The ConnectionInfos object can now be used to set the database logon information for a report.

```
connInfos.add(connInfo1);
```

```
connInfos.add(connInfo2);
```

Note: If only one ConnectionInfo object is added to the ConnectionInfos collection, then the user name and password stored in that ConnectionInfo object is applied to all connections, including embedded subreports and on-demand subreports.

Setting database logon information

Once a properly initialized ConnectionInfos object has been created, the database logon information can be passed to the viewer. The viewer handles the process of passing this information to the report.

To set the database logon information

1. Create an instance of the viewer, passing it a reference to a report source object.

```
CrystalReportViewer viewer = new CrystalReportViewer();
```

```
viewer.setReportSource(reportSource);
```

2. Set the database logon information by passing the viewer the initialized ConnectionInfos object.

```
viewer.setDatabaseLogonInfos(connInfos);
```

```
viewer.setEnableLogonPrompt(false);
```

3. Call the processHttpRequest method to launch the viewer in the current browser window.

```
viewer.processHttpRequest(request, response,  
getServletConfig().getServletContext(), null);
```

4. Call the dispose method to allow the viewer to perform the appropriate garbage collection and free system resources.

```
viewer.dispose();
```

Example

The following example is a JSP page that demonstrates how to set the database logon information for a report requiring 2 database logons. After the logon information has been set, the report is displayed.

setDbLogonViewReport.jsp

```

<%@ page import= "com.crystaldecisions.report.web.viewer.*,"
        com.crystaldecisions.sdk.occa.report.data.*" %>

<%@ page
import="com.crystaldecisions.reports.reportengineinterface.JPEReportSourceFacto
com.crystaldecisions.sdk.occa.report.reportsource.IReportSourceFactory2,
com.crystaldecisions.reports.reportengineinterface.IReportSource"
%>

<%

String report = "c:/reports/sample.rpt";

IReportSourceFactory2 rptSrcFactory = new JPEReportSourceFactory();

IReportSource reportSource = (IReportSource)
rptSrcFactory.createReportSource(report, request.getLocale());

ConnectionInfos connInfos = new ConnectionInfos();

IConnectionInfo connInfo1 = new ConnectionInfo();
IConnectionInfo connInfo2 = new ConnectionInfo();

connInfo1.setUserName("guest");
connInfo1.setPassword("password");
connInfo2.setUserName("guest");
connInfo2.setPassword("password");

connInfos.add(connInfo1);

```



```
connInfos.add(connInfo2);

CrystalReportViewer viewer = new CrystalReportViewer();

viewer.setReportSource(reportSource);

viewer.setDatabaseLogonInfos(connInfos);

viewer.setEnableLogonPrompt(false);

viewer.processHttpRequest(request, response,
getServletConfig().getServletContext(), out);

viewer.dispose();

%>
```

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Best practices

You can do several things to improve the performance of web applications that make use of the viewer to display reports. These techniques range from optimization of the reports themselves to coding practices that improve the efficiency of your applications.

Click the appropriate link to jump to that section:

- [Report design](#)
- [Stateless viewing](#)
- [Caching report sources](#)
- [Viewer only pages](#)
- [Using the setOwnForm method](#)
- [Specifying the correct character set](#)

Report design

You can view potential performance bottlenecks in your report design by going to the Report menu in the Crystal Reports Designer and clicking Performance Information. The *Crystal Reports Online Help* contains a chapter with a detailed overview on how to create efficient reports titled "Designing Optimized Web Reports." Following the recommendations listed there helps you improve your report viewing efficiency considerably.

Stateless viewing

Beginning with version 9, the viewer can work effectively without being stored in a session variable on the web application server, providing stateless viewing. Stateless viewing only works, however, for reports that are read-only. If the user needs to modify the report using a Report Application Server or refresh the report's data source, stateless viewing cannot be used, and the viewer and report source instances must be stored in a session variable.

Caching report sources

Caching a report source in the session variable allows it to be used multiple times efficiently. When a report source is not cached, the process of creating a new report source multiple times becomes fairly expensive. Furthermore, caching a report source allows reports with or without saved data to be refreshed.

The following example shows how to cache a report source in the session variable:

```
String report = "/reports/sample.rpt";

JPEReportSource reportSource;

reportSource = new JPEReportSource(report);

session.putValue("reportSource", reportSource);
```

Note: If you are using a cached report source, the dispose methods for the viewer or report source should not be called until the report source is no longer being used.

Viewer only pages

If your JSP page contains only the viewer and nothing else, several things can be done ways to simplify the report viewing implementation.

Setting the setOwnPage property

The viewer is capable of generating complete HTML pages and can set the appropriate page properties depending on the viewing context. Setting the setOwnPage property to true provides several benefits, by allowing the viewer to completely handle the surrounding HTML content. Allowing the viewer to handle the surrounding HTML content reduces the amount of code you need to add to your JSP page and allows the viewer to automatically determine certain settings:

- It allows the viewer to choose which page start and end tags to use based on what device is being used to view the page.

For example, it writes out the `<html>` start tag for web browsers and `<wml>` start tag for mobile devices.

- It correctly sets the content-type and charset information for the page. This ensures that pages containing international characters is displayed correctly.

If `setOwnPage` is false, the surrounding HTML tags and content-type and charset directives need to be manually set.

Using the `processHttpRequest` method

When `setOwnPage` is set to true, you must use the `processHttpRequest` method to display the report instead of `getHtmlContent`. The `processHttpRequest` method must be used because using `getHtmlContent` has the same effect as setting `setOwnPage` to false, negating any of the benefits gained from setting `setOwnPage` to true.

Using the `setOwnForm` method

If your JSP page does not contain any controls that require post back, you should set the `setOwnForm` method to true. Doing so, allows the viewer to handle the view state information automatically. The view state is used to perform client-side caching of information about the current state of the report. If you have other controls on the page, you must `setOwnForm` is set to false and handle the view state information manually.

The following example shows how to set the view state information manually:

```
viewer.setOwnForm(false);

viewer.setViewState((String) session.getValue("viewState"));

viewer.processHttpRequest(request, response, getServletContext(),
pageContext.getOut());

session.putValue("viewState", viewer.getViewState());
```

Specifying the correct character set

To send characters from a JSP file to a web browser, you must use the correct encoding. Always specify the correct content-type and character set for all of your JSP pages.

If your JSP page returns content to a standard HTML browser, ensure that the correct character set is defined:

```
<%@ page contentType="text/html; charset=utf-8" %>
```

The `contentType` and `charset` directives let the browser know how the returned HTML page is encoded. UTF-8 is the recommended standard character set if it is available for your target client browser. For more

IDE Extensions

information, consult the Release Notes or the vendor of your target client browser.

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

SDK Reference

This section contains reference material for the viewer. This material supplements the API documentation provided by the *Crystal Reports for BEA WebLogic Workshop Java API Reference*. It contains essential sample code for ensuring the best performance from the viewer and some simple troubleshooting tips.

Click the appropriate link to jump to that section:

- Sample Code
- Netscape Troubleshooting

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Sample Code

This section contains sample code that provides useful examples not presented in the tutorials.

CrystalImageCleaner Object Sample Code

The CrystalImageCleaner is needed to ensure that temporary files used by the viewer are removed periodically. Adding a properly configured CrystalImageCleaner object to JSP pages that use the viewer helps improve the performance of your web application.

Note: The following line of code is necessary in your JSP pages in order to ensure that the CrystalImageCleaner object is available.

```
<% @ page import="com.crystaldecisions.report.web.viewer.CrystalImageCleaner" %>
```

Example 1

This sample code creates a CrystalImageCleaner object that scans for image files once every minute, but only deletes files that are at least 2 minutes old. Proper tweaking of these settings is necessary in order to ensure optimal performance of your application, as the optimal values for these settings are highly dependent on viewer usage and application design characteristics.

```
<%!  
  
public void jspInit(){  
  
    CrystalImageCleaner.start(getServletContext(), 60000, 12000);  
  
}  
  
%>
```

Example 2

This sample code stops the CrystalImageCleaner object once the JSP page is removed from service.

```
<%!  
  
public void jspDestroy(){  
  
    CrystalImageCleaner.stop(getServletContext());  
  
}  
  
%>
```

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

Netscape Troubleshooting

The section provides tips for using the report viewers with Netscape.

Click the appropriate link to jump to that section:

- [Viewing a report](#)
- [Tagging considerations](#)

Viewing a report

If Netscape continually crashes when you are attempting to view a report, try using one of these workarounds.

Click the appropriate link to jump to that section.

- [Adding a dummy style](#)
- [Adjust height and width](#)
- [Disable DHTML](#)
- [Using multiple reports](#)

Adding a dummy style

After the `</HEAD>` tag and before the `<BODY>` tag, add a dummy style. There is a known bug in Netscape that causes the browser to crash in some cases when styles are used in the body of a page.

Example

```
</HEAD>

<STYLE TYPE='text/css'><!-- .FakeStyle { color:blue; } --> </STYLE>

<BODY>
```

Adjust height and width

Ensure that any specified height and width in a <LAYER> or <DIV> tag is wide enough to handle the HTML version of your report. If Netscape is not able to fit the report into the container, it usually crashes.

Disable DHTML

As a last resort, set the RenderAsHTML32 property to True to return only pure HTML 3.2 tags. Although the output does not look as true to the original report compared to HTML 4.0 tags, this may be the only way for you to display the report in some browsers.

Using multiple reports

When using multiple reports in the same page, ensure that only one report is prompting for a database logon or parameter at any given time. The HTML returned by the automatic parameter prompting and automatic database logon routine is not designed to be displayed in multiple instances at the same time. See `TwoViewsWithParameters.asp` for an example of how to view multiple reports that have parameters.

Tagging considerations

This section contains guidelines to help you avoid tagging issues specific to Netscape.

Click the appropriate link to jump to that section.

- [Using closing tags](#)
- [Using <DIV> tags](#)
- [Applying a style](#)
- [Using hyperlinks](#)

Using closing tags

Ensure that all tags are properly matched up with a closing tag where appropriate. For example, if you put a <TABLE> tag without a corresponding </TABLE> tag later on in the web page, the table is not rendered correctly.

Using <DIV> tags

- When using nested <DIV> tags, even though the generated script is in the same <DIV> as the function calling it, Netscape sometimes considers all JavaScript functions to be out of scope. It appears that if <DIV> tags are used, the JavaScript function must be one level up for Netscape to consider the function in scope.
- When using complex nested <DIV> tags, Netscape often renders the report differently if you give the <DIV> tag an identifier. That is, <DIV ID=hello> causes Netscape to render the page differently in some cases than if the tag was simply <DIV>.
- A form which contains <DIV> tags does not work.

This code does not work:

```
<FORM>

<DIV ID="firstLayer">

<INPUT TYPE="Text" NAME="Input1" SIZE="20">

</DIV>

<DIV ID="secondLayer">

<INPUT TYPE="Text" NAME="Input2" SIZE="20">

</DIV>

</FORM>
```

The solution is to put a form inside each layer:

```
<DIV ID="firstLayer">

<FORM NAME="form1">

<INPUT TYPE="Text" NAME="Input1" SIZE="20">

</FORM>

</DIV>

<DIV ID="secondLayer2">

<FORM NAME="form2">

<INPUT TYPE="Text" NAME="Input2" SIZE="20">

</FORM>
```



```
</DIV>
```

Applying a style

- Styles set in linked style sheets do not always apply to classes inside your document. When using `<DIV>` tags, if Netscape is not applying a style set in your CSS page, you need to write the style inline in your HTML. The HTML returned by the Crystal Report Server Control does not use any external style sheets.
- Netscape sometimes crashes if there is a valid style applied to a `<SELECT>` object. Changing the order that the styles are placed in the style attribute of various HTML elements can potentially change the output as well. An example of this is the "position: relative;" style as applied to a "`<DIV>`" object. It is not uncommon for the HTML text "`<DIV style='position:relative; border-bottom-style: solid'>`" to render differently than "`<DIV style='border-bottom-style: solid; position:relative'>`", or for Netscape to crash in some of these cases.
- When working with tables, it is necessary to set the style in the containing object for Netscape in order to apply the style to the intended object. For example, when rendering the following HTML using Netscape, the style contained in "firstStyle" is applied to the contained `<SELECT>` object.

```
<TD class="firstStyle">

    <SELECT class="secondStyle"....>

</TD>
```

Using hyperlinks

URLs sent to a Netscape 4.x browser cannot contain spaces. Spaces in hyperlinks should be replaced by %20 to provide maximum compatibility.

Crystal Decisions
<http://www.crystaldecisions.com/>
 Support services
<http://support.crystaldecisions.com/>

Viewer Tag Library

The Viewer Tag Library allows you to access the report viewing capabilities of the Crystal report viewers without needing to embed large amounts of Java code in your JSP pages.

This section first provides a brief overview of the Viewer Tag Library, including how to set up your JSP pages to use the tag library. The section then provides a list of all the tags available, their specific uses, and relevant examples.

Click the appropriate link to jump to that section:

- [Overview](#)
- [Tag reference](#)
- [Samples](#)

Overview

The Viewer Tag Library allows you to access the Crystal report viewers through modular and reusable tags. Using tags allows you to reduce the amount of embedded Java code in a JSP page by moving the functional implementation into the tag and its implementing class. Doing so effectively abstracts the details of the code from the presentation of the actual page, allowing the web page author to concentrate on developing JSP pages. This allows all the functionality of the Crystal report viewers to be accessed without knowing how to write Java code.

The Viewer Tag Library is made up of a primary tag, the viewer tag, and a collection of nested tags. The viewer tag supports attributes that relate to the way the Crystal report viewers appear and behave.

Using the Viewer Tag Library is equivalent to using the Viewer Java SDK to programmatically customize the viewer. It provides the same basic functionality that the Viewer Java SDK provides, supporting the same features and allowing the same level of customization over how the viewer is displayed. The advantage of using the Viewer Tag Library, however, is that it consistently generates efficient code, as the tag implementation has been carefully tested and optimized.

Using the Viewer Tag Library

To use the tags provided by the Viewer Tag Library in a JSP page, a reference to the tag library's descriptor file is required. The following code needs to be added to the top of each JSP page using the tag library:

```
<%@ taglib uri="/crystal-tags-reportviewer.tld" prefix="crviewer" %>
```

Tag reference

The Viewer Tag Library is comprised of several tags. The tag reference is designed to provide a concise description of the tags available, their supported attributes, and how the tags and attributes should be used. Each tag description follows the same format. First, a brief summary of the tag and its function are given. This is followed by a list of nested tags and a table containing a summary of all the attributes. The attributes are

then described in detail.

Each attribute description provides a quick summary that states whether the tag is optional or required, its default value, and the type of value it expects. This summary is followed by a more comprehensive description of the attribute, and, if required, what specific values the attribute accepts.

The following list provides a hierarchical view of the tags available:

Viewer tag

- Report tag

Viewer tag

The viewer tag is the main tag that instantiates the CrystalReportViewer. It handles the creation of the viewer and allows you to set viewer properties through supported attributes.

Each instance of the viewer tag requires a report tag to specify the report to be displayed. If multiple instances of the viewer will be used on a single HTML page, each instance must be given a unique name, specified through the viewerName attribute.

Nested tags

Report tag

Attributes

Appearance

Attribute	Description	Required
enablePageToGrow	Sets whether the viewer takes up as much space as it needs to properly display the report.	No
height	Sets the viewer height in pixels.	No
width	Sets the viewer width in pixels.	No
left	Sets the left position of the viewer.	No
top	Sets the top position of the viewer.	No
displayGroupTree	Sets whether to display the Group Tree on viewer start up.	No
displayPage		No

IDE Extensions

	Sets whether to display the report page.	
displayToolBar	Sets whether to display the toolbar.	No
groupTreeWidth	Sets the width of the Group Tree in pixels.	No
groupTreeWidthUnit	Sets whether to use pixels or percentage to determine the group tree width.	No
zoomPercentage	Sets the zoom factor used when displaying the report.	No
styleSheet	Sets the stylesheet that used to display the report content.	No

Behavior

Attribute	Description	Required
printMode	Sets whether to print using PDF or ActiveX print mode when the user clicks the print button.	No
allowDatabaseLogonPrompting	Sets whether to allow users to be prompted for database logon information.	No
allowDrillDown	Sets whether to allow the user to drill down on the report.	No
allowParameterPrompting	Sets whether to allow the user to be prompted for parameter values.	No
hyperlinkTarget	Sets the window or frame where hyperlinked documents are displayed.	No
isOwnPage	Sets whether the viewer control owns the page.	No

Toolbar

Attribute	Description	Required
displayToolBarViewList	Sets whether to display the view list on the toolbar.	No
displayToolBarExportButton	Sets whether to display the export button on the toolbar.	No
displayToolBarFindPageButton	Sets whether to display the Go To Page button on the toolbar.	No
displayToolBarPageNavigationButtons	Sets whether to display the page navigation buttons on the toolbar.	No
displayToolBarPrintButton	Sets whether to display the print button on the toolbar.	No
displayToolBarRefreshButton	Sets whether to display the refresh button on the toolbar.	No
displayToolBarFindButton	Sets whether to display the search button on the toolbar.	No
displayToolBarToggleTreeButton	Sets whether to display the Group Tree toggle button on the toolbar.	No
displayToolBarZoomList	Sets whether to display a zoom factor drop down list on the toolbar.	No

IDE Extensions

displayToolBarCrystalLogo	Sets whether to display the Crystal Decisions logo on the toolbar.	No
---------------------------	--	----

Report Source

Attribute	Description	Required
reportSourceType	Specifies the type of report source the viewer will use.	Yes
reportSourceVar	Sets the name of the variable where the report source will be cached.	No

Identification

Attribute	Description	Required
viewerName	Sets the name of the viewer.	Yes

enablePageToGrow

Optional

Defaults to true

boolean

Sets whether the viewer ignores the height and width values it is given and try to find the best fit for the report by taking up as much space as it needs to properly display the report.

height

Optional

Defaults to 600

int

Sets the viewer height in pixels. Together, the height and width attributes control the viewer's viewable size.

width

Optional

Defaults to 800

int

Sets the viewer width in pixels. Together, the height and width attributes control the viewer's viewable size.

left

Optional

Defaults to 0

int

Sets the left position of the viewer. The units used are browser dependent.

top

Optional

Defaults to 0

int

Sets the top position of the viewer. The units used are browser dependent.

displayGroupTree

Optional

Defaults to true

boolean

Sets whether to display the Group Tree on viewer start up.

displayPage

Optional

Defaults to true

boolean

Sets whether to display the report page.

displayToolbar

Optional

Defaults to true

boolean

Sets whether to display the toolbar.

groupTreeWidth

Optional

Defaults to 200

int

Sets the width of the Group Tree in pixels.

groupTreeWidthUnit

Optional

pixel

String

Sets whether to use pixels or percentage to determine the group tree width.

Use `pixel` to specify to use pixels and `percent` to specify to use percentage to determine the group tree width.

zoomPercentage

Optional

Defaults to 100

int

Sets the zoom factor used when displaying the report. Acceptable values for the zoom factor range from 10 to 400 percent.

styleSheet

Optional

Defaults to `default.css`

String

Sets the style sheet that used to display the report content. The cascading style sheet that you use needs to contain the same classes as the default style sheet. The default style sheet is located in the `/crystalreportviewers10/css` directory. When specifying a custom style sheet, the paths are always

relative to this directory.

printMode

Optional

Defaults to PDF

String

Sets whether to print using PDF or ActiveX print mode when the user clicks the print button. In PDF print mode, a PDF is displayed, allowing the user to then print it. In ActiveX print mode, an ActiveX control is downloaded to the client machine and is sent directly to the printer. If ActiveX print mode is selected on a system that does not support ActiveX controls, the print mode defaults to PDF printing.

Use PDF to specify PDF print mode and ActiveX to specify ActiveX print mode.

allowDatabaseLogonPrompting

Optional

Defaults to true

boolean

Sets whether to allow users to be prompted for database logon information.

allowDrillDown

Optional

Defaults to true

boolean

Sets whether to allow the user to drill down on the report.

allowParameterPrompting

Optional

Defaults to true

boolean

Sets whether to allow the user to be prompted for parameter values.

hyperlinkTarget

Optional

Defaults to `_self`

String

Sets the window or frame where hyperlinked documents are displayed.

Use the target `_self` to display the HTML document in the same frame, `_parent` to display HTML document in the same frame or window that contains the current frameset, `_top` to display HTML document in the entire browser window, and `_blank` to display HTML document in a new browser window.

isOwnPage

Optional

Defaults to `false`

boolean

Sets whether the viewer control owns the page. Set this attribute to false if the page is within a portal. If the server control owns the page, it provides the opening and closing HTML tags and it is able to get and set values for the entire page.

Note: If you set `isOwnPage` to false then you must set the charset and the content-type of the HTML page where the viewer is displayed.

displayToolbarViewList

Optional

Defaults to `true`

boolean

Sets whether to display the view list on the toolbar.

Note: The view list includes the Main Report and any views you have drilled-down into. This can include groups, charts and subreports.

displayToolbarExportButton

Optional

Defaults to `true`

boolean

Sets whether to display the export button on the toolbar.

Note: `isOwnPage` must be set to true or the button is not rendered, regardless of the value set in the `setHasExportButton` method.

displayToolbarFindPageButton

Optional

Defaults to true

boolean

Sets whether to display the Go To Page button on the toolbar.

Note: If true, the HTML returned includes a text box, into which the user can type a page number to navigate to a specific page in the report. If false, the HTML returned displays the current page, but does not allow the user to navigate by page number.

displayToolbarPageNavigationButtons

Optional

Defaults to true

boolean

Sets whether to display the page navigation buttons on the toolbar.

displayToolbarPrintButton

Optional

Defaults to true

boolean

Sets whether to display the print button on the toolbar. This is only available if `isOwnPage` is set to true.

Note: By default, printing is accomplished by automatically exporting the report to PDF and then printing the report from the PDF viewer. For systems with ActiveX support, ActiveX print mode, which uses ActiveX controls to print the report, can be used. To specify the print mode, use the `printMode` attribute.

displayToolBarRefreshButton

Optional

Defaults to false

boolean

Sets whether to display the refresh button on the toolbar.

displayToolBarFindButton

Optional

Defaults to true

boolean

Sets whether to display the search button on the toolbar.

displayToolBarToggleTreeButton

Optional

Defaults to true

boolean

Sets whether to display the Group Tree toggle button on the toolbar.

displayToolBarZoomList

Optional

Defaults to true

boolean

Sets whether to display a zoom factor drop down list on the toolbar.

displayToolBarCrystalLogo

Optional

Defaults to true

boolean

Sets whether to display the Crystal Decisions logo on the toolbar.

reportSourceType

Required

No default value.

String

Specifies the type of report source the viewer will use. For more information on report sources, see Report sources.

Possible values are:

- `reportingComponent`

Specifies that the Java Reporting Component will be used to render the report.

- `reportApplicationServer`

Specifies that a Report Application Server will be used as a report source.

- `pageServer`

Specifies that a Page Server from part of a Crystal Enterprise deployment will be used as a report source.

reportSourceVar

Optional

No default value.

String

Sets the name of the session variable where the report source will be cached. Caching the report source in the browser's session allows it to be reused, improving performance when the same report source will be used multiple times.

viewerName

Required

Defaults to `CrystalViewer`

String

Sets the name of the viewer. The name represents the HTML form name used to post back requests to the server. When there is more than one viewer on a page, each viewer must be given a unique name, otherwise, a post back in one viewer may be sent to the wrong viewer.

Example

The following example shows how to create a viewer with the Group Tree hidden and 80% zoom:

```
<crviewer:viewer displayGroupTree="false" zoomPercentage="80"
reportSourceType="reportingComponent">
```

...

```
</crviewer:viewer>
```

Report tag

The report tag is used to specify the report that is displayed by the viewer. It is required by the viewer tag, unless you are using a manually cached report source.

Nested tags

None

Attributes

Attribute	Description	Required
reportName	Sets the name of the report to be viewed when an unmanaged report source is used.	Yes

reportName

Required

No default value

String

Sets the name of the report to be viewed when an unmanaged report source is used. This applies to unmanaged RAS and Java Reporting Component report sources.

For Java Reporting Component report sources, the reportName attribute can be specified using absolute or relative file paths and URLs. For more information, see Java Reporting Component.

For unmanaged RAS report sources, the `reportName` attribute must specify a path that is accessible by Report Application Server. That is, the path is interpreted by the Report Application Server and so local paths refer to the machine on which the RAS server is running. If you would like to access reports running on a separate machine, you must make sure that the report is accessible through a network share.

Example

The following example shows how to view a report stored on the local file system:

```
<crviewer:viewer ...>  
  
<crviewer:report reportName="c:\\Reports\\World Sales.rpt" />  
  
</crviewer:viewer>
```

Samples

The Viewer Tag Library is designed around simplicity and ease-of-use. The following examples demonstrate how to use the functionality provided by the tag library to view reports.

Click the appropriate link to jump to that section:

- [Viewing unmanaged reports](#)
- [Viewing managed reports](#)

Viewing unmanaged reports

Unmanaged reports do not require authentication or other logon procedures in order to access them. Components that support unmanaged reports include the Java Reporting Component and unmanaged Report Application Servers. Viewing an unmanaged report using the Viewer Tag Library primarily relies on specifying the appropriate report source type and location of the report file.

Click the appropriate link to jump to that section:

- [Viewing a report using the Java Reporting Component](#)
- [Viewing a report using an unmanaged RAS server](#)

Viewing a report using the Java Reporting Component

This is a basic sample designed to show you how to use the Viewer Tag Library to display a report using the Java Reporting Component.

To view a report

1. Ensure that the appropriate reference to the Viewer Tag Library's descriptor file is added at the top of your JSP page.

You must also ensure that the appropriate JAR files and additional support files are present. If you are using a wizard, this is automatically done for you.

```
<%@ taglib uri="/crystal-tags-reportviewer.tld" prefix="crviewer" %>
```

Note: You can specify any name for the prefix attribute. This just determines what tag prefix you use to access the tag library's tags.

2. Create an opening viewer tag.

In this tag, you must specify the viewer name and the type report source that is being used.

```
<html>
```

```
<body>
```

```
<crviewer:viewer viewerName="CrystalViewer"
reportSourceType="reportingComponent" >
```

3. Create the required report tag.

This specifies the report that is displayed.

```
<crviewer:report reportName="/reports/sample.rpt"/>
```

4. Close the viewer tag.

The JSP page is now ready to view the sample.rpt report.

```
</crviewer:viewer>
```

```
</body>
```

```
</html>
```

Viewing a report using an unmanaged RAS server

This sample shows you how to use the Viewer Tag Library to display a report using an unmanaged Report Application Server.

To view a report

1. Ensure that the appropriate reference to the Viewer Tag Library's descriptor file is added at the top of your JSP page.

```
<%@ taglib uri="/crystal-tags-reportviewer.tld" prefix="crviewer" %>
```

Note: You can specify any name for the prefix attribute. This just determines what tag prefix you use to access the tag library's tags.

2. Set system property to specify the location of the clientSDKOptions.xml file.

This file is used to set the location of the RAS server.

```
<%  
  
    System.setProperty("ras.config","C:\\temp");  
  
%>
```

Note: You can also set the location of the clientSDKOptions.xml file in the classpath.

3. Create an opening viewer tag.

In this tag, you must specify the viewer name and the type report source that is being used. In this case, the reportApplicationServer report source is being used.

```
<html>  
  
<body>  
  
    <crviewer:viewer viewerName="CrystalViewer"  
    reportSourceType="reportApplicationServer" >
```

4. Create the required report tag, specifying the report to be displayed.

The reportName value is prefixed with rassdk:// so that the RAS server can correctly locate the file.

```
<crviewer:report reportName="rassdk://c:/reports/sample.rpt"/>
```

5. Close the viewer tag.

The JSP page is now ready to view the sample.rpt report.

```
</crviewer:viewer>  
  
</body>  
  
</html>
```


Viewing managed reports

Although there are no tags that provide direct support for viewing managed reports, a combination of tags and JSP code can provide the equivalent functionality. Viewing managed reports requires you to have either a Crystal Enterprise deployment or Report Application Server running.

Click the appropriate link to jump to that section:

- Viewing a report using a Page Server
- Viewing a report using a managed RAS server

Viewing a report using a Page Server

This sample shows you how to use the Viewer Tag Library to display a report using a Page Server report source.

To view a report using a Page Server

1. Log on to Crystal Enterprise and get an InfoStore object.

In this case, the default administrator account is used to log on to a Crystal Enterprise installation running on the same machine as the application server.

<%

```
    IEnterpriseSession es =
    CrystalEnterprise.getSessionMgr().logon("administrator", "",
    "localhost", "secEnterprise");
```

```
    IInfoStore infoStore = (IInfoStore)
    es.getService("", "InfoStore");
```

2. Query for the report you wish to view.

```
    IInfoObjects infoObjects = infoStore.query("SELECT * FROM
    CI_INFOOBJECTS WHERE SI_NAME='World Sales Report'");
```

```
    IInfoObject report = (IInfoObject) infoObjects.get(0);
```

3. Create Page Server report source for the report you retrieved.

```
    IReportSourceFactory2 rsFactory = (IReportSourceFactory2)
    es.getService("PSReportFactory");
```

```
    IReportSource rptSrc = rsFactory.createReportSource(report,
    request.getLocale());
```

4. Cache the ReportSource object in a session variable.

IDE Extensions

This allows it to be retrieved by the viewer.

```
session.setAttribute("ReportSource", rptSrc);

%>
```

5. Create a viewer tag, specifying the viewer name, report source type, and report source variable.

In this case, the pageServer report source is being used and the report source variable refers to the session variable where you stored the ReportSource object.

```
<html>

<body>

<crviewer:viewer viewerName="CrystalViewer"
reportSourceType="pageServer" reportSourceVar="ReportSource" />

</body>

</html>
```

Note: In this case, a report tag is not needed because the report source has already been cached in a session variable and does not to be retrieved by the viewer.

Viewing a report using a managed RAS server

This sample shows you how to use the Viewer Tag Library to display a report using a managed Report Application Server report source.

To view a report using a managed RAS server

1. Log on to Crystal Enterprise and get an InfoStore object.

In this case, the default administrator account is used to log on to a Crystal Enterprise installation running on the same machine as the application server.

```
<%

    IEnterpriseSession es =
CrystalEnterprise.getSessionMgr().logon("administrator", "",
"localhost", "secEnterprise");

    IInfoStore infoStore = (IInfoStore)
es.getService("", "InfoStore");
```

2. Query for the report you wish to view.

IDE Extensions

```
IInfoObjects infoObjects = infoStore.query("SELECT * FROM  
CI_INFOOBJECTS WHERE SI_NAME='World Sales Report'");
```

```
IInfoObject report = (IInfoObject) infoObjects.get(0);
```

3. Create a RAS report source for the report you retrieved.

```
IReportSourceFactory2 rsFactory = (IReportSourceFactory2)  
es.getService("RASReportFactory");
```

```
IReportSource rptSrc = rsFactory.createReportSource(report,  
request.getLocale());
```

4. Cache the ReportSource object in a session variable.

This allows it to be retrieved by the viewer.

```
session.setAttribute("ReportSource", rptSrc);
```

```
%>
```

5. Create a viewer tag, specifying the viewer name, report source type, and report source variable.

In this case, the reportApplicationServer report source is being used and the report source variable refers to the session variable where you stored the ReportSource object.

```
<html>
```

```
<body>
```

```
<crviewer:viewer viewerName="CrystalViewer"  
reportSourceType="pageServer" reportSourceVar="ReportSource" />
```

```
</body>
```

```
</html>
```

Note: In this case, a report tag is not needed because the report source has already been cached in a session variable and does not to be retrieved by the viewer.

Crystal Decisions
<http://www.crystaldecisions.com/>
Support services
<http://support.crystaldecisions.com/>

A

additional viewer features

architecture

system overview

audience, intended

B

Best practices

caching report sources

report design

specifying the correct charset

stateless viewing

using the setOwnForm method

viewer only pages

C

character separated values, export to

components

locations

configuration

data sources

Java Reporting Component

Crystal report, export to

D

data sources, configuring

database logon, setting

Viewer Tag Library

development environment, setup

document conventions

E

Excel, export to

Export Control, formats

exporting, reports

F

formats, exporting to

G

Getting started

H

how to

create a report source

export a report

set a database logon

set a parameter field

use the java viewer

HTML, export to

J

Java Reporting Component

configuration

L

locale-specific viewing

M

managed reports, viewing using tags

N

Netscape troubleshooting

P

parameter fields, setting

PDF, export to

product registration

R

registration, of product

report sources

creating

Java Reporting Component

overview

reports, exporting

reports, viewing

tags

tutorial

required components, installing

RTF, export to

S

sample code

CrystalImageCleaner

setup, development environment

support, technical

system overview

Java Application Server

Java Reporting Component

Viewer Java SDK

T

tag libraries, viewer

technical support

Text, export to

Trademark acknowledgements

troubleshooting, Netscape

tutorials

U

unmanaged reports, viewing using tags

V

viewer

additional features

garbage collection

launching

tutorials

Viewer Tag Library

viewer features

descriptions

locale-specific viewing

Viewer Functionality

Viewer Java SDK

overview

Viewer Tag Library

report tag

samples

setting up

tag reference

viewer tag

viewer, using

W

web sites

consulting

product registration

technical support

training

Word, export to