Business Objects

Crystal Enterprise and BusinessObjects Enterprise

Upgrading Report Designer Component (RDC) SDK Applications to the Report Application Server (RAS) SDK

Overview

The Report Application Server has become the recommended embeddable solution since its release in Crystal Enterprise Report Application Server (CE RAS) 9. It is more powerful than the Report Designer Component (RDC) with respect to architecture, scalability, and performance. It is the best replacement for RDC because of its capabilities and features such as report creation and modification. With these reasons in mind, it is strongly recommended to upgrade existing RDC web applications to RAS or one of the other newer web reporting technologies available in the Business Objects product suite. The other web reporting solutions include the Java Reporting Component (JRC) and the Crystal Reports .NET API.

This document focuses on the upgrade path of RDC applications to RAS embeddable application but you can learn more about your other options by visiting the Developer Zone or by viewing our interactive Enterprise Reporting Object Model diagram. The Developer Zone is a good source of whitepapers, tutorials, and samples, while the Object Model is an interactive diagram that can help you decide which web reporting technology is the best solution for you.

Applicable to all versions of the Report Application Server SDK, the purpose of this document is to do the following:

- Highlight the advantages of upgrading to RAS
- Introduce the reader to the RAS architecture and the RAS SDK object model
- Review common RDC methods and discuss the equivalents within RAS

Contents

Introduction

What is the Report Designer Component?

The Report Designer Component (RDC) is a component-based technology that was first introduced in Crystal Reports 7. It was designed mainly for desktop COM applications but was later modified for small- to medium-sized web reporting deployments. It is based mainly on the craxdrt.dll which is an ActiveX designer object. With the release of Crystal Reports XI, it is now considered a legacy web technology and will be deprecated in the next version of Crystal Reports. The RDC is suitable for only thick client desktop applications.

What is the Report Application Server?

RAS in contrast, is a client-server technology composed of the RAS and Viewer SDKs, and the RAS server service. The goal of RAS is to provide an embeddable web reporting solution that is server-oriented instead of component-based. In version XI, new features and deployment options have been added to Crystal Reports Server Embedded XI to enable new capabilities such as **in-proc RAS**. **In-proc RAS** is a feature that allows you to use the RAS SDK within a single-tier .NET application environment. That is, the deployment package for the in-proc RAS .NET application requires only the .NET assemblies. Note that if you use **in-proc RAS**, you do not gain the performance and stability benefits of a server-oriented out-process system. The components of RAS are discussed in the following sections.

RAS SDK

The SDK component provides an interface to the RAS server component. In this sense, the SDK forms the client part of the client/server system. The RAS SDK enables you to access to the report object model, allowing you to create new reports or modify existing ones programmatically. The report creation and modification capabilities include adding and removing tables, changing data source location, adding and removing database fields, as well as many of the same capabilities found in the Crystal Reports Designer.

Viewer SDK

The Viewer SDK includes a number of libraries that allow you to display reports through the web in DHTML using standard web report viewers, or embed reports in web pages using customized web report viewers. The SDK also provides an interface for you to programmatically set the report's database, parameter, and filter information, as well as export the reports to a variety of third-party formats.

RAS server

The RAS server that comes packaged as a stand-alone product is called Crystal Reports Server Embedded and is known as "Standalone" RAS. *Crystal Reports Server XI Embedded is available only through our OEM Partner Program.* The RAS server is also included as part of BusinessObjects Enterprise and Crystal Reports Server.

RAS provides services for creating and modifying reports. The RAS SDK communicates with the RAS server and sends requests and reports for the server to process. The RAS server forms the server part of the client/server system (Figure 1).

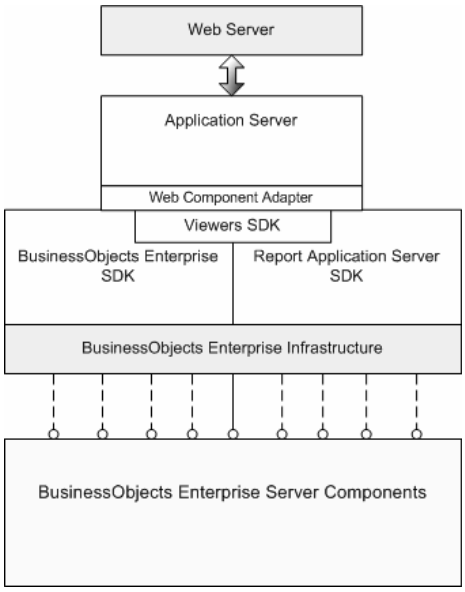

Figure 1: RAS within the BusinessObjects Enterprise and Crystal Reports Server framework

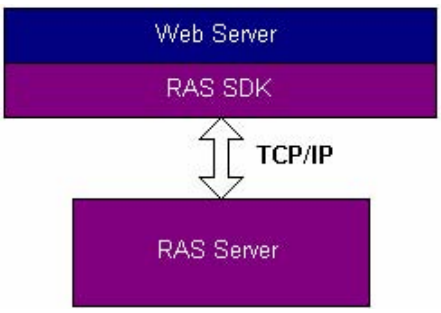

Figure 2: RAS within its standalone environment

NOTE	The same RAS service and SDK are part of the following server products: BusinessObjects Enterprise XI, Crystal Reports Server XI, and Crystal Reports Server XI Embedded.

Advantages of upgrading to the Report Application Server

Table 1 illustrates the advantages of upgrading an RDC application to the RAS application architecture.

Report Designer Component	Report Application Server
Component based technology	Service-Oriented Architecture (SOA)
Apartment threaded model	Runs as an .exe and is multi-threaded
"In process" - Lives entirely in the context of the host application (that is, IIS)	"Out process". Manages its own memory. Web server resources reserved for just the web application.
Designed for small workgroup-sized web applications	Designed for small- to medium-sized web application deployments
No scalability or fault tolerance	Limited scalability and fault tolerance
No application upgrade path for higher-end solutions – would require rewrite of application	Easy application upgrade path to higher-end server solutions – no code changes needed
Component based technology	Current technology that will continue to grow in capability and feature set

Table 1: Advantages of Upgrading to RAS

The Report Application Server SDK

The architecture

RAS is a client/server system with two primary components: the RAS server (Windows service) and RAS SDK. When you install RAS, a service called the Report Application Server is created. The RAS service executes the commands issued through the RAS SDK, processes the report and then delivers the report object back to the SDK for delivery to the end user.

NOTE	The object model and architecture of the RAS SDK is the same across all server products, platforms, as well as for the .NET, Java and COM versions of the SDK.

Model-View-Controller architecture

The RAS SDK contains a number of libraries that can be accessed in an ASP, JSP, or .NET development environment and is implemented using the Model-View-Controller (MVC) architecture.

The MVC architecture is a widely used architectural approach for interactive applications. It divides functionality among objects involved in maintaining and presenting data to minimize the degree of coupling between the objects. The architecture maps traditional application tasks--input, processing, and output -- to the graphical user interaction model. They also map to the domain of multi-tier web-based enterprise applications.

The MVC architecture divides applications into three layers -- model, view, and controller -- and decouples their respective responsibilities. Each layer handles specific tasks and has specific responsibilities to the other areas.

A **model** represents business data and business logic or operations that govern access and modification of this business data. Often, the model serves as a software approximation to real-world functionality. The model notifies views when it changes and provides the ability for the view to query the model about its state. It also provides the ability for the controller to access application functionality encapsulated by the model.

A **view** renders the contents of a model. It accesses data from the model and specifies how that data should be presented. It updates data presentation when the model changes. A view also forwards user input to a controller.

A **controller** defines application behavior. It dispatches user requests and selects views for presentation. It interprets user inputs and maps them to actions to be performed by the model. In a stand-alone GUI client, user inputs include button clicks and menu selections. In a web

application, they are HTTP GET and POST requests to the web tier. A controller selects the next view to display based on the user interactions and the outcome of the model operations. Typically, an application has one controller for each set of related functionality. Some applications use a separate controller for each client type, because view interaction and selection often vary between client types.

Figure 3 depicts the relationships between the model, view, and controller layers of an MVC application.

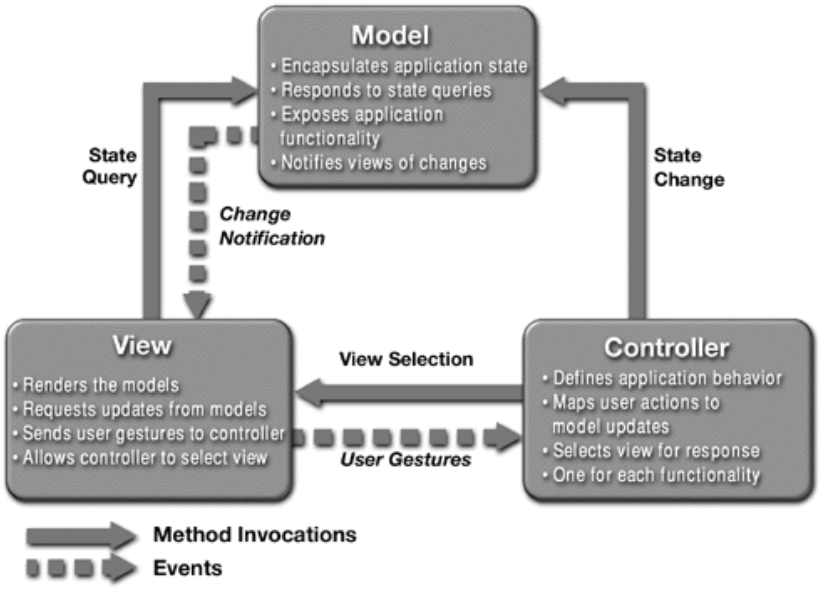

Figure 3: Model-View-Controller Architecture

Report Application Server SDK architecture

Given this overview of the MVC architecture, it is now possible to outline how the RAS SDK implements this architecture.

The **ReportClientDocument** object is a representation of a report (model). It enables you to access either the controllers to modify or extract data, or the object models to access data. Effectively, the **ReportClientDocument** object allows you to open, modify, and save reports.

The controllers possess the logic to modify the object models on the client side. Although the object models provide functionality to manipulate the data and report definition, they do not synchronize the data that the RAS client is manipulating with the data on the RAS server. To properly modify a report and ensure that the data on the RAS server is updated, you must use the appropriate controller.

The object models are used to define the content in a report. They contain the data structures necessary to determine the view of the objects in a report. The object models are implemented in two main libraries:

- **Crystal Reports Data Definition Model Library** - provides a definition for the report's database and data such as tables, fields, formulas, parameters, groups, and sorts.
- **Crystal Reports Report Definition Model Library** - determines the appearance and layout of the report objects that appear in a report document, such as sections, areas, charts, and text objects.

Finally, the RAS SDK is used to provide a particular view of the data in the report (model). There may many different views presenting a different picture of the data. Each view may access the data in the model directly by using a controller, which is necessary if you want to modify data.

The RAS SDK object model

Report Client Document

A **ReportClientDocument** object is the object representation of a report file. The **ReportClientDocument** object provides access to the report data definition and controllers that are required for modifying a report. A connection between the RAS server and RAS SDK must be setup before a report can be loaded into a **ReportClientDocument** object.

Report Application Server SDK models

A model represents structures of data. It is a software approximation of real world functionality such as a report file. The RAS SDK provides a number of different object models that define the layout and content of a report. These objects are also used to specify the changes that need to be made to a report. Examples of object models include sort fields, result fields, formula fields, groups, record filters, sections, tables, subreports, and database connections.

Crystal Reports Data Definition Model Library

The Crystal Reports Data Definition Model Library is used to provide a definition for the data that a report contains. The **Database** and **DataDefinition** objects are the two main objects in this library, and can be accessed as properties from the **ReportClientDocument** object.

Use the **Database** object to retrieve information about the following:

- Tables
- Table Links
- Database Fields
- Database Connections

Use the **DataDefinition** object to retrieve information about the following:

- Formula fields
- Groups
- Group filters
- Parameter fields
- Record filters
- Result fields (fields placed on a report)
- Sort fields
- Summary fields

Crystal Reports Report Definition Model Library

The Crystal Reports Report Definition Model Library contains objects that define the appearance and layout of a report. These objects can be accessed directly from the **ReportDefinition** property of the **ReportClientDocument** object.

Object models and context

It is important to understand that the RAS SDK object models can exist in two different contexts. First, an object model can exist as a property of the **ReportClientDocument** object. In this context, the purpose of the object model is to present report data. Directly modifying the property of a **ReportClientDocument** object does not change the report. To modify a report, the object model must be created outside the context of the **ReportClientDocument** object.

Suppose you want to change the text of a formula field inside a report. If you have experience programming with the Report Designer Component (RDC) object model, the following line of RAS code will look familiar:

```
(ReportClientDoc.getDataDefinition().getFormulaFields().get
Field(0)).setText("({Customer.Customer Last Year's
Sales}/12)")
```

With the RDC, any changes that you make directly to the report object's properties are automatically applied once a command is issued to view, export, save, or print a report file. However, with RAS, the above line of code neither changes the contents of an existing formula field, nor generates an error message. In other words, the line of code looks like it should work but it does not. When you preview the report in a DHTML viewer, the report contains the original formula field because the **FormulaField** object is a property of the **ReportClientDocument** object.

The sole role of the **ReportClientDocument** object is to represent the current properties of an existing formula field in the report file. To modify those properties, create a new **FormulaField** object as follows:

```
FormulaField newFormula = new FormulaField();
```

newFormula is an independent **FormulaField** object with properties that you can set. If you want to insert a new **FormulaField** object into a report, then set the values for the syntax, type, text, length, and name properties.

In this example, however, you only want to use *newFormula* as a template to modify the text property of an existing **FormulaField**. To transfer the property values from the existing **FormulaField** object to *newFormula*, use the **copyTo** or **clone** method. Conveniently, every object model and collection in the RAS SDK includes these methods. The only difference between these two methods is in the syntax that each requires. For example:

```
ReportClientDoc.getDataDefinition().getFormulaFields().getF
ield(0).copyTo (newFormula,true);
```

```
newFormula =
(FormulaField)(ReportClientDoc.getDataDefinition().getFormu
laFields().getField(0)).clone();
```

Most of the standard object models provide few other methods aside from the **clone** and **copyTo** methods. Collections and arrays of a specific object type have additional methods that allow you to insert, remove, or find member objects. Some SDK model objects such as the **Filter** and **ParameterFieldValue** objects have additional methods like **computeText**, which returns a string representation of the object or the data it contains. None of the object models provides methods that enable you to write data back to the **ReportClientDocument** object.

newFormula is now an exact replica of the **FormulaField** object that you want to change. Next, set the value for the text property of newFormula as follows:

```
newFormula.setText ("({Customer.Customer Last Year's
Sales}/12)");
```

You now have passed *newFormula* back to the report so that it replaces the existing **FormulaField** object. At this point, it might be tempting to remove the existing **FormulaField** object and add the new one using the

remove and **add** methods of the **FormulaFields** collection property of the **ReportClientDocument** object:

```
ReportClientDoc.getDataDefinition().getFormulaFields().remo
ve(0);
ReportClientDoc.getDataDefinition().getFormulaFields().add(
newFormula);
```

While these lines of code are syntactically correct, the properties of the collections, like those of the individual objects that they contain, cannot be modified directly in the context of the **ReportClientDocument** object. Therefore, the formula field is not replaced.

To complete the task of modifying the existing formula field, you need to use a controller.

Report Application Server SDK controllers

In an application based on MVC architecture, the role of the controller is to modify the data contained in the model. In the RAS SDK architecture, the controllers do more than just modify the data that is contained in the properties of the **ReportClientDocument** object.

Recall that the **ReportClientDocument** object is just an object representation of a physical report file opened by the RAS server. The objective is to apply the specified changes to the actual report file. The controllers send the new data to the RAS server component, which then commits the requested changes to the report. On submitting the request to the RAS server, the controller also updates the relevant properties of the **ReportClientDocument** object to ensure that it accurately reflects the state of the physical report.

The following line of code applies the changes specified in newFormula to the actual report:

```
ReportClientDoc.getDataDefController().getFormulaFieldContr
oller().modify(ReportClientDoc.getDataDefinition().getFormu
laFields().getField(0), newFormula);
```

The **modify** method of the **FormulaFieldController** takes two arguments. The first argument is the existing **FormulaField** object that you want to modify. You can also use the index number of the **FormulaField** object instead of the object itself. The second argument is the new **FormulaField** object that contains the specified changes. You can visualize this as a swapping of objects.

The **FormulaFieldController** sends this request to an intermediary queue that is managed by a communication object called a Report Agent.

The Report Agent buffers these requests and sends them as a group to the RAS server after it receives a command to view, save, print, or export the report file.

The controllers are accessed as properties of the **ReportClientDocument** object. In parallel with the RAS SDK object models, the controllers are also organized by common functionality. Because the object models and controllers mirror each other to some extent, it is simple to locate the controller for the corresponding object model that you want to change (see Table 2).

Controller	Object
CustomFunctionController Object	CustomFunction Object
DatabaseController Object	Database Object
DataDefController Object	DataDefinition Object
FilterController Object	Filter Object
FormulaFieldController Object	FormulaField Object
GroupController Object	Group Object
ParameterFieldController Object	ParameterField Object
PrintOutputController Object	ByteArray Object
ReportAreaController Object	Area Object
ReportObjectController Object	ReportObject Object
ReportSectionController Object	Section Object
ResultFieldController Object	All fields that are displayed in the report.
RowsetController Object	Rowset Object
SearchController Object	Rowset Object
SortController Object	Sort Object
SubreportController Object	Subreport Table Object
SummaryFieldController Object	SummaryField Object

Table 2: RAS Controllers and Objects

While controllers can have properties, the majority of them do not. Most of the functionality that the controllers provide is implemented as methods. Some controllers will have properties that hold other controllers and object models. As mentioned previously, these object models can be properties of the **ReportClientDocument** object. For example, a table in a report can be accessed using two different methods, as the following lines of code demonstrate:

```
ReportClientDoc.getDatabase().getTables().getTable(0);
```

- OR -

```
ReportClientDoc.getDatabaseController().getDatabase().getTables().getTable(0);
```

The two methods return the same table object, and therefore can be used interchangeably. While having these two methods to return the same table object provides flexibility and convenience, the resulting redundancy may also make the code more difficult to understand.

For consistency, it is recommended to retrieve the object models as properties of the **ReportClientDocument** object rather than from the controllers.

Fortunately, there is only one way of accessing each report controller per **ReportClientDocument** object. For example, you can retrieve the database controller as follows:

```
ReportClientDoc.getDatabaseController();
```

The Data Definition Controllers

The primary controller in the RAS SDK is the **DataDefController** object. It contains the controllers to manipulate the object models that are accessed through the **DataDefinition** property of the **ReportClientDocument** object. The **DataDefController** object is used to add, remove, and modify the data definition of the report. The data definition defines which data is being retrieved and how the data is grouped, summarized, and filtered. Using the **DataDefController**, you may modify any part of the report that acts on the data from the database. The **DataDefController** object provides access to the following controllers:

FormulaFieldController

The **FormulaFieldController** is used to add, remove, and modify report formulas.

GroupController

The **GroupController** is used to add, remove, and modify groups in a report.

ParameterFieldController

The **ParameterFieldController** adds, removes, and modifies parameter fields in a report.

FilterController

The **FilterController** is used to modify filters in a report. Filters are used in record selection and group selection formulas.

ResultFieldController

The **ResultFieldController** is used to add, remove, and move result fields in a report. Results fields are fields that are used in the report—for example, database fields, parameter fields, and formula fields. You must add any fields that you want to appear on the report to the collection of **ResultField** objects. Consequently, the **ResultFields** collection consists of many different field types.

SortController

The **SortController** is used to modify the way in which data in the report is sorted. If a report contains groups, then a sort order can be applied to these groups as well as to the details in each group.

SummaryFieldController

The **SummaryFieldController** is used to add, remove, and modify summary fields in a report.

The Database Controller

The **DatabaseController** object is used to manipulate database objects that are contained in a report. It allows you to add, modify, remove, and link tables.

The Report Definition Controllers

The **ReportDefController** object provides access to the **ReportAreaController**, **ReportSectionController**, and **ReportObjectController**. These controllers allow you to modify areas, sections, as well as format report objects such as fields, pictures, and charts. These object models are provided in the **Report Definition Model Library**.

ReportAreaController

The **ReportAreaController** is used to set the properties for areas in the report. Every report is broken up into a number of areas, each of which can be divided into a number of sections. Report objects can only be placed within a section. At a minimum, a report has the following areas: report header, page header, detail, page footer, report footer. If the report has groups, then group areas will also exist. Although you cannot manually add or delete areas in the report, you can format and rename them using the **SetProperty** method. For more information on areas and sections, see the Crystal Reports Online Help.

Area objects are defined in the Crystal Reports Report Definition Model Library, and can be accessed by using the **ReportDefinition** object. If you want to access a group area, you must indicate which group you want to access. To determine how many group areas are in the report, you must check how many groups there are; that is, check **Report.DataDefinition.Groups.Count.** Once you have retrieved the desired area, you must then access the area's sections. When adding a **ReportObject** to the report, you must indicate in which section you want it placed.

ReportSectionController

The **ReportSectionController** is used to add, remove, and set the properties for sections in the report. Each section belongs to a particular report area, and can contain different types of report objects. Report objects can only be placed in a section.

ReportObjectController

The **ReportObjectController** is used to add, remove, and modify the report objects that are defined in the Crystal Reports Report Definition Model Library. These objects include charts, pictures, lines, text objects, and field objects.

The Rowset Controller

The **RowsetController** is used to directly retrieve unformatted data in a report. In other words, it provides access to the rowsets in the **ReportClientDocument** object. When data is retrieved from the database, it is filtered, grouped, sorted, and then stored in a tree structure called the **Totaller**. The **RowsetController** object allows you to search for and access this data in the tree. Additionally, it allows control over how data is retrieved and cached by the server and the client.

The Search Controller

The **SearchController** object is used in conjunction with the **RowsetController** object to perform searches on the data in the report. Search results are stored in a **Rowset** object.

The Subreport Controller

The **SubreportController** object can retrieve the names of all the subreports in a report. You can then use a subreport name to retrieve a list of tables in the subreport, or to change a table's database. RAS provides limited functionality to modify subreports. Other than the database connection specified in a subreport, no other properties can be modified.

The Viewer SDK

The Viewer SDK contains the libraries that provide the web viewing capabilities for RAS applications. There are four zero-client object-based viewers (also known as the Crystal Reports Web Reporting Type Library) available in the Viewer SDK:

- Report Page Viewer
- Interactive Viewer
- Grid Viewer
- Report Part Viewer

The DHTML viewers are popular because they are zero-client viewers and therefore do not require any client software to be downloaded and installed. All four viewers are available in the .NET, COM, and Java SDKs. The only difference is that in .NET, these viewers are Web Controls that you can drag directly onto your Web Form. A brief description of each Viewer is given in Table 3.

Viewer	Description
Report Page Viewer	Provides basic web reporting viewer capabilities through a thin client.
Interactive Viewer	Provides all the capabilities of the COM Report Page Viewer, plus the ability to do a Boolean search. Boolean searches allow you to extract subsets of data from the report while viewing.
Grid Viewer	Provides the functionality to view report data in a grid format that is independent of the original format of the source report. This object implements the report grid viewer. *Available only in CE 10 Embedded and later
Report Part Viewer	Provides the capability to view individual segments of a report called Report Parts. Report parts include such items as charts, text, and fields. This allows you to show a specific portion of the report, and provides the ability to give the end user a very specific view and navigation of the report.

Table 3: RAS Viewers

NOTE	The ActiveX Viewer may also be used in RAS applications.

These viewers provide a great deal of control over viewer and report performance. For example, you can use the Viewer SDK to serve a report including information such as logon information, selection formula criteria, and parameter values. If any required view time information is missing from the report, the viewers will prompt the end user with a form requesting that information.

Upgrading from RDC to RAS SDK

This section compares how to perform the following tasks when using the RDC and RAS SDKs:

- Loading a report
- Passing parameters values
- Changing table location
- Filtering data
- Logging onto a database
- Setting a data set
- Exporting a report
- Viewing reports

For more details, refer to the samples listed in the Additional Resources section of this document.

Loading a report

New in BusinessObjects Enterprise XI and Crystal Reports Server XI is the ability to load reports into RAS applications from the local file system (that is, unmanaged reporting). In Crystal Enterprise 8, 8.5, 9, and 10, you had to first publish reports to the repository before you could load them into a RAS application. Now you can load them directly into your RAS application without the intermediary step of publishing them.

NOTE	With Crystal Reports Server Embedded XI, you can only load reports from the file system because it does not contain the repository.

The sections below contain code examples of how to load a report from the repository or from the local file system.

Loading a report from the repository

RAS SDK using Visual Basic .NET

```
Dim mySampleReportName As String = "World Sales Report"
Dim mySessionMgr As New SessionMgr()
Dim myEnterpriseSession As EnterpriseSession
Dim myReportAppFactory As ReportAppFactory
Dim myInfoStore As InfoStore
Dim myInfoObjects As InfoObjects
Dim myInfoObject As InfoObject
Dim myReportClientDocument As ReportClientDocument
Dim myEnterpriseService As EnterpriseService
Dim myObject As Object


‘ Logon to the Enterprise system
myEnterpriseSession = mySessionMgr.Logon("enterprise_user",
"enterprise_password", "cms_name", "secEnterprise")


‘ Get the InfoStore service (provides access to repository)
myEnterpriseService =
myEnterpriseSession.GetService("InfoStore")
myInfoStore = New InfoStore(myEnterpriseService)


‘ Query the repository database for the objects you need
ie. Report objects
myInfoObjects = myInfoStore.Query("Select SI_ID From
CI_INFOOBJECTS Where SI_NAME='" + mySampleReportName + "'
And SI_INSTANCE=0")
myInfoObject = myInfoObjects(1)


‘ Retrieve the RAS service to process report
myObject = myEnterpriseSession.GetService("",
"RASReportFactory").Interface
myReportAppFactory = CType(myObject, ReportAppFactory)


‘ Load the report into a ReportClientObject
myReportClientDocument =
myReportAppFactory.OpenDocument(myInfoObject.ID, 0)
```

RAS SDK using Java

```
String reportName ="Pubs Report";   /* Name of report */
String mUser = "Administrator";      /* User Name */
String mPassword = "";               /* Password */
String mCMSName = "cms_name";        /* CMS Name */
String mAuthType = "secEnterprise"; /* Authorization Type */


/* Connect to Crystal Enterprise and get SessionManager */
ISessionMgr sm = CrystalEnterprise.getSessionMgr();
IEnterpriseSession es = sm.logon(mUser, mPassword,
mAPSName, mAuthType);


/* Get the Report Application Factory service from Crystal
Enterprise */
IReportAppFactory rptAppFactory =
(IReportAppFactory)es.getService("", "RASReportService");


/* Get the InfoStore service from Crystal Enterprise */
IInfoStore infoStore = (IInfoStore)es.getService("",
"InfoStore");


/* Retrieve the report by name from Crystal Enterprise */
IInfoObjects oInfoObjects = infoStore.query("Select SI_ID
From CI_INFOOBJECTS Where SI_NAME = '" + reportName + "'");


/* Open and load report into a Report Document object */
ReportClientDocument clientDoc =
rptAppFactory.openDocument((IInfoObject)oInfoObjects.get(0)
, 0, Locale.ENGLISH);
```

Loading a report from the file system

RAS SDK using Visual Basic .NET

```
‘ NOTE: If your reports are located on the application
‘ server and not the RAS server,
‘ pre-append “rassdk://” to your report path.
‘ This directs the RAS server to search for the report
‘ on the SDK computer.


Dim mySampleReportPath As String = "C:\Program
Files\Business Objects\Crystal Reports
11\Samples\en\Reports\General Business\World Sales
Report.rpt"
```

```
Dim path As Object = CType(mySampleReportPath, String)
Dim RptClientDoc As New ReportClientDocumentClass()
‘ Specify location of RAS server
RptClientDoc.ReportAppServer = “name_of_reporting_server”

‘ Open the report and load into ReportClientDocument object
RptClientDoc.Open(path, 1)
```

RAS SDK using Java

```
/* NOTE: If your reports are located on the application
server and not the RAS server, add “rassdk://” to the front
of your report path to tell the RAS server to search for
the report on the SDK computer. */

String reportName =" C:\\Program Files\\Business
Objects\\Crystal Reports 11\\Samples\\en\\Reports\\General
Business\\World Sales Report.rpt";

/* Create the report client document object */
ReportClientDocument clientDoc = new
ReportClientDocument();

/* Set the RAS Server to be used for the Client Document */
clientDoc.setReportAppServer("<name_of_reporting_server");

/* Open the report, and set the open type to Read Only */
clientDoc.open(path + reportName,
OpenReportOptions._openAsReadOnly);
```

RDC SDK using Visual Basic

```
Dim reportPath = “C:\Program Files\Business Objects\Crystal
Reports 11\Samples\en\Reports\General Business\World Sales
Report.rpt”

‘ Create report object and load report
Set session("oRpt") =
session("oApp").OpenReport(reportPath, 1)
```

Passing parameter values

RAS SDK using Visual Basic .NET

…

```
' Pass string value to the parameter

' If we are passing params to the subreport, then the first
param would be the name of the subreport

myReportClientDocument.DataDefController.ParameterFieldCont
roller.SetCurrentValue("", "StringParam", "USA")


' Pass a numeric value to the parameter

myReportClientDocument.DataDefController.ParameterFieldCont
roller.SetCurrentValue("", "NumericParam", 5000)
```

RAS SDK using Visual Basic .NET in an unmanaged environment

This unmanaged RAS sample performs the following tasks:

- Sets the directory path to a sample Crystal report.
- Creates an instance of the ReportClientDocument.
- Creates an object which contains the directory path to the sample report.
- Uses the Open() method of ReportClientDocument to open the report.
- Binds the CrystalReportViewer to the ReportClientDocument

```
Private Sub SetParameters_unmanagedRAS()

Dim mySampleReportPath As String = "C:\Program
Files\Business Objects\Report Application Server
11\Samples\VB.NET\Simple Discrete
Parameters\odbc_customer_params.rpt"

Dim path As Object = CType(mySampleReportPath, String)


Dim rcd As New ReportClientDocumentClass()

rcd.Open(path, 1)


' The report named odbc_customer_params.rpt has two
parameters:

' Country and Sales. Use the SetCurrentValue property in

' the ParameterFieldController Class to set these

' parameters to 'USA' and '5000'.
```

```
rcd.DataDefController.ParameterFieldController.SetCurrentVa
lue("", "Country", "USA")

rcd.DataDefController.ParameterFieldController.SetCurrentVa
lue("", "Sales", "5000")

myCrystalReportViewer.ReportSource = rcd

End Sub
```

RAS SDK using Java

```
…

/* Pass string value to parameter */

/* If we are passing params to the subreport, then the
first param would be the name of the subreport */

clientDoc.getDataDefController().getParameterFieldControlle
r().setCurrentValue("", "StringParam", "Test");


/* Pass numeric value to parameter */

clientDoc.getDataDefController().getParameterFieldControlle
r().setCurrentValue("", "NumericParam",
Integer.valueOf("200"));
```

RDC SDK using Visual Basic

```
…

‘ Value of parameter we want to set

Dim param_value


‘ Retrieve the parameter field collection

Set paramFields = Session("oRpt").ParameterFields


‘ Retrieve the parameter from the collection by name

Set stringParam = paramFields.GetItemByName("StringParam”)



‘ Set the parameter value

stringParam.AddCurrentValue(“USA”)

‘ Retrieve the parameter from the collection by name

Set numberParam = paramFields.GetItemByName("NumberParam”)


‘ Set the parameter value

numberParam.AddCurrentValue(5000)
```

Changing table location

RAS SDK using Visual Basic

```
Imports  CrystalDecisions.ReportAppServer.DataDefModel

…

‘ Retrieve handle to DatabaseController

Dim dbController As DatabaseController =
RptClientDoc.DatabaseController


‘ Change the table location by passing in table alias,
servername, db name, uid and pwd

dbController.SetTableLocationByServerDatabaseName(“authors”
, “ServerA”, “pubs”, “sa”, “password”)
```

RAS SDK using Java

```
…

/* Get the database controller */

DatabaseController dbController =
rptClientDoc.getDatabaseController();


/* Change the table location by passing in table alias,
servername, database name, user ID, and password */

dbController.setTableLocationByServerDatabaseName(“authors”
, “ServerA”, “pubs”, “sa”, “password”);,
```

RDC SDK using Visual Basic

```
…

' Get the first table in the report.

Set DBTable = Report.Database.Tables(1)


' Get the collection of connection properties.

Set CPProperties = DBTable.ConnectionProperties


' Change the database DLL used by the report from Access to
ADO/OLEDB

DBTable.DllName = "crdb_ado.dll"


' Clear all the ConnectioProperty objects from the
collection.

CPProperties.DeleteAll
```

```
' Add the name value pair for the provider.
CPProperties.Add "Provider", "SQLOLEDB"
CPProperties.Add "Data Source", "ServerA"
CPProperties.Add "Initial Catalog", "pubs"
CPProperties.Add "User ID", "sa"
CPProperties.Add "Password", "password"

' Set the table name.
DBTable.Location = "Authors"
```

Filtering data using record selection formulas

RAS SDK using Visual Basic .NET

```
…
Dim myFilter As Filter

‘ Create new filter object
myFilter = New
CrystalDecisions.ReportAppServer.DataDefModel.Filter()

‘ Edit the formula text of the filter
myFilter.FreeEditingText = "{Employee.Employee Id} > 2 and
{Employee.Employee Id} < 8"

‘ Commit the change back to the report’s filter
myReportClientDocument.DataDefController.RecordFilterContro
ller.Modify(myFilter)
```

RAS SDK using Java

```
…
/* Set the filter string to be used as the Record Filter */
String freeEditingFilter = "{Employee.Employee Id} > 2 and
{Employee.Employee Id} < 8";

/* Retrieve the record filter for the Data Definition
Controller */
IFilter iFilter =
clientDoc.getDataDefController().getDataDefinition().getRec
ordFilter();

/* Set the filter to free editing text filter string */
iFilter.setFreeEditingText(freeEditingFilter);
```

```
/* Modify the filter through the Record Filter Controller
to the report */
clientDoc.getDataDefController().getRecordFilterController(
).modify(iFilter);
```

RDC SDK using Visual Basic

```
…
' Set the report filter
session("oRpt").RecordSelectionFormula =
CStr("{Employee.Employee Id} > 2 and {Employee.Employee Id}
< 8")
```

Logging on to a database

RAS Visual Basic .NET

```
…
' Log on to the database using the DatabaseController's
logon method
RptClientDoc.DatabaseController.logon("limitedPermissionAcc
ount", "test")
```

RAS SDK using Java

```
…
/* Log on to the database using the DatabaseController's
logon method */
clientDoc.getDatabaseController().logon("sa","password");
```

RDC using Visual Basic

```
…
' Log on to the database for each table
Set mainReportTableCollection =
Session("oRpt").Database.Tables

For Each mnTable in mainReportTableCollection
  With mnTable.ConnectionProperties
    ' Database user name
   .Item("user ID") = "sa"
    ' Database password
   .Item("Password") = "password"
    ' DSN name
   .Item("DSN") = "Pubs Sample Database"
```

```
    ' Database name
   .Item("Database") ="pubs"
  End With
Next
```

Setting a dataset

RAS SDK using Visual Studio .NET

```
…
Dim data As System.Data.DataSet


' Note: ASPNET_wp user needs read permission to mdb_path,
and both read/write permission to xsd_path


' Set the path to your xtreme.mdb file
Dim mdb_path As String = "c:\program files\crystal
decisions\crystal reports
9\samples\en\databases\xtreme.mdb"


' Set the path to where you want to create the customer
schema file
Dim xsd_path As String =
"c:\Crystal\RASNET\ras9_vbnet_web_DataSetReport\customer.xs
d"


' Create OLEDB connection
conn = New OleDbConnection()
conn.ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & mdb_path


' Create Data Adapter
adap = New OleDbDataAdapter("select * from Customer where
Country='Canada'", conn)
```

```
‘ Create dataset and fill
data = New System.Data.DataSet()
adap.Fill(data, "Customer")

‘ Create a schema file
data.WriteXmlSchema(xsd_path)

‘ Set the dataset to the report
RptClientDoc.DatabaseController.SetDataSource(DataSetConver
ter.Convert(data), "Customer", "Customer")
```

RAS SDK using Java

```
…
/* Create class for JDBC:ODBC bridge */
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

/* Construct connection */
java.sql.Connection connection =
DriverManager.getConnection("jdbc:odbc:Xtreme Sample
Database 9", "Admin", "");

/* Construct statement */
Statement statement = connection.createStatement();

/* Construct result set through query ( Select all
Customers and their Last Years Sales where Sales is greater
than 20000$ ) */
ResultSet results = statement.executeQuery("Select
Customer.`Customer Name`, Customer.`Last Year's Sales` from
Customer where Customer.`Last Year's Sales` > 20000");

/* Get name of table we’re going to set the dataset on */
String tableName =
clientDoc.getDatabase().getTables().getTable(0).getName();
/* Set the java.sql.ResultSet dataset */
clientDoc.getDatabaseController().setDataSource(results,
tableName, "Reports");
```

RDC SDK using Visual Basic

```
…
' This line creates an ADO Connection object
Set oADOConnection =
Server.CreateObject("ADODB.Connection")


' Open a connection to the Xtreme Sample Database
oADOConnection.Open ("Xtreme Sample Database 11")


' Creates an ADO Recordset object
Set oADORecordset = Server.CreateObject("ADODB.Recordset")
' Populate the recordset object by executing an SQL
statement against the database
Set oADORecordset = oADOConnection.Execute("Select
[Employee ID],[First Name], [Last Name] From Employee")
Set oRptTable = session("oRpt").Database.Tables.Item(1)


' Set the tables data source to be the recordset object.
oRptTable.SetDataSource oADORecordset, 3
```

Exporting a report

RAS SDK using Visual Basic .NET

```
…
‘ Export the report to the specified format
Dim myBytes As ByteArray =
rptClientDoc.PrintOutputController.Export(CrReportExportFor
matEnum.crReportExportFormatPDF)


‘ Clear the response stream
Response.Clear()


‘ Add header to response stream for displaying the PDF
inline
‘ Setting this header to “attachment” will prompt end user
to save or open file
Response.AddHeader("content-disposition",
"inline,filename=untitled.pdf")


‘ Set the proper context type
Response.ContentType = "application/pdf"
```

```
‘ Send binary export to the response stream
Response.BinaryWrite(myBytes.ByteArray)
```

RAS SDK using Java

```
…
/* Use the PrintOutputController to export the report to a
ByteArrayInputStream */
ByteArrayInputStream byteIS = (ByteArrayInputStream)
clientDoc.getPrintOutputController().export(ReportExportFor
mat.PDF);

/* Create a byte[] (same size as the exported
ByteArrayInputStream) */
byte[] buf = new byte[2000 * 1024];
int nRead = 0;

/* Set response headers to indicate pdf MIME type and
inline file */
response.reset();

/* Add header to response stream for displaying the PDF
inline */
response.setHeader("Content-disposition",
"inline;filename=untitled.pdf");
response.setContentType("application/pdf");

/* Send the Byte Array to the response stream for display
in browser */
while ((nRead = byteIS.read(buf)) != -1) {
  response.getOutputStream().write(buf, 0, nRead);
}

/* Flush the output stream */
response.getOutputStream().flush();

/* Close the output stream */
response.getOutputStream().close();
```

RDC SDK using Visual Basic

```
…
'Retrieve the ExportOptions from the report
Set CrystalExportOptions = Session("oRpt").ExportOptions


'Set format type
ExportType = "31" 'PDF


'Physical location and file name to give the export result
CrystalExportOptions.DiskFileName =
"c:\export_directory\exportedreport.pdf"


'This line of code specifies the export format
CrystalExportOptions.FormatType = CInt(ExportType)


'This line of code specifies that the export destinatin is
to be disk.
CrystalExportOptions.DestinationType = CInt(1)


'Export the report (false suppresses any prompts)
Session("oRpt").Export False
```

Viewing a report

RAS SDK using Visual Basic .NET

The web viewers in the RAS.NET SDK are .NET Web Controls that you can drag directly onto your Web Form. The code for the actual viewer is hidden in the background so we only need one line of code to specify a viewer's report source.

```
…
myCrystalReportViewer.ReportSource = myReportClientDocument
```

RAS SDK using Java

```
…
/* Create a Viewer object */
CrystalReportViewer viewer = new CrystalReportViewer();


/* Set the name for the viewer */
viewer.setName("Crystal_Report_Viewer");
```

```
/* Set the source for the  viewer to the client documents
report source */

viewer.setReportSource(clientDoc.getReportSource());


/* Process the http request to view the report */

viewer.processHttpRequest(request, response,
getServletConfig().getServletContext(), out);


/* Dispose of the viewer object */

viewer.dispose();
```

RDC SDK using Visual Basic

```
<BODY BGCOLOR=C6C6C6 ONUNLOAD="CallDestroy();" leftmargin=0
topmargin=0 rightmargin=0 bottommargin=0>

<OBJECT ID="CRViewer"

      CLASSID="CLSID:A1B8A30B-8AAA-4a3e-8869-1DA509E8A011"

      WIDTH=100% HEIGHT=99%

      CODEBASE="/crystalreportviewers10/ActiveXControls/Act
iveXViewer.cab#Version=10,0,0,280" VIEWASTEXT>

<PARAM NAME="EnableRefreshButton" VALUE=0>

<PARAM NAME="EnableGroupTree" VALUE=1>

<PARAM NAME="DisplayGroupTree" VALUE=1>

<PARAM NAME="EnablePrintButton" VALUE=1>

<PARAM NAME="EnableExportButton" VALUE=1>

<PARAM NAME="EnableDrillDown" VALUE=1>

<PARAM NAME="EnableSearchControl" VALUE=1>

<PARAM NAME="EnableAnimationControl" VALUE=1>

<PARAM NAME="EnableZoomControl" VALUE=1>

</OBJECT>


<SCRIPT LANGUAGE="VBScript">

<!--

Sub Window_Onload

      On Error Resume Next

      Dim webBroker

      Set webBroker =
CreateObject("CrystalReports10.WebReportBroker.1")

      if ScriptEngineMajorVersion < 2 then

            window.alert "IE 3.02 users on NT4 need to get
the latest version of VBScript or install IE 4.01 SP1. IE
3.02 users on Win95 need DCOM95 and latest version of
```

```
VBScript, or install IE 4.01 SP1. These files are available
at Microsoft's web site."
       else
              Dim webSource
              Set webSource =
CreateObject("CrystalReports10.WebReportSource.1")
              webSource.ReportSource = webBroker
              webSource.URL = "rptserver.asp"
              webSource.PromptOnRefresh = True
              CRViewer.ReportSource = webSource
       end if
       CRViewer.ViewReport
End Sub
-->
</SCRIPT>
```

Finding More Information

Report Application Server Resources

Introduction to Crystal Reports Server XI Embedded (OEM Only)

ftp://ftp1.businessobjects.com/outgoing/oem/documents/crsXI_embedded_quick_start_guide.pdf

Installing Crystal Reports Server XI Embedded (OEM Only)

http://support.businessobjects.com/library/docfiles/cps10/downloads/en/boxi_rasinstall_en.pdf

BusinessObjects Enterprise XI COM SDK Help (includes RAS SDK)

http://support.businessobjects.com/library/docfiles/cps10/downloads/en/boeXI_com_docs_en.zip

BusinessObjects Enterprise XI Java SDK Help (includes RAS SDK)

http://support.businessobjects.com/library/docfiles/cps10/downloads/en/boeXI_java_docs_en.zip

BusinessObjects Enterprise XI .NET SDK Help

http://support.businessobjects.com/library/docfiles/cps10/downloads/en/boeXI_net_docs_en.zip

RAS XI .NET SDK Help

http://support.businessobjects.com/library/docfiles/cps10/downloads/en/rasXI_net_docs_en.zip

Additional Resources

Sample Applications

http://support.businessobjects.com/fix/samplescr.asp

Online Technical Support Site - Knowledge Base Articles, Whitepapers, Product Updates

http://support.businessobjects.com/search/

Monthly Hot Fixes

http://support.businessobjects.com/fix/hot/mhf/default.asp

Service Packs

http://support.businessobjects.com/downloads/updates/service_packs/default.asp

www.businessobjects.com