BUSINESS OBJECTS

Crystal Reports 10 and .NET

Upgrading from Crystal Reports to RAS and Crystal Enterprise

Overview

The purpose of this document is to show the steps required to upgrade a Crystal Reports .NET WebForm Viewer application to use Report Application Server (RAS) and the Crystal Enterprise Framework. This document applies to version 10 and later.

Contents

AUTHORS Terry Penner – Program Manager, Enterprise Reporting SDKs
Ian Treleaven – Senior Developer, Client SDKs

Introduction

Beginning in version 10, with only a few lines of code per report, you can now easily migrate all your ReportDocument-based embedded reporting applications to Enterprise-managed Report Application Servers.

As a starting point, follow the walkthrough, “Viewing a Report in a Web Application”, referenced in the Crystal Reports 10 help that is installed with Visual Studio. The walkthrough can also be referenced through the following link:
http://www.businessobjects.com/products/dev_zone/net_walkthroughs.asp

For additional information and updates about Crystal Reports 10, please visit one of the following sites:

http://www.businessobjects.com/products/reporting/crystalreports/net/

http://www.businessobjects.com/products/dev_zone/net/

Merge module updates and hot fixes for all supported languages can be found at:

http://support.businessobjects.com/mergemodules/

Viewer Configuration for Easy Migration

Before migrating your application to RAS or Crystal Enterprise, you need to create an initial Crystal Reports for .NET application from Visual Studio for .NET

1. Add a ReportDocument object to your Visual Studio .NET project. To do this, select the Toolbox in Visual Studio and open the Components list. Drag-and-drop a ReportDocument on to the project.

2. In the ‘Choose a ReportDocument’ dialog box that is displayed, you will be prompted to make a choice between a typed or untyped ReportDocument class. Choose ‘Untyped ReportDocument’ and press OK. Note that the ReportDocument instance’s ReportAppServer property is set to “<EMBEDDED_REPORT_ENGINE>”. This means that the report engine will be running inside the same process as the rest of your application.

3. Set the FileName property of the ReportDocument instance to the file path to your report.

4. Select the associated viewer instance. Change its ReportSource property to the name of the newly added ReportDocument (the name is reportDocument1 by default). When you press enter, the viewer’s preview display will change to the new report source.

1. Upgrading Your .NET Application to Unmanaged RAS

The RAS server provides enhanced performance and scalability options for report viewing scenarios. It allows for separation of the report processing from the rendering and viewing of your report, as well as optimizing performance across multiple web servers that are connected to the same RAS server.

To install a RAS server, run the Crystal Enterprise Embedded Edition install on your deployed server.

NOTE	Your RAS server is licensed for deployment only if you have purchased Crystal Reports Advanced or have separately purchased Crystal Enterprise Embedded Edition.

After installation, ensure that the RAS Server has been given permission to open reports from the location where your reports are stored. To check this, go to Start->Programs->Crystal Enterprise 10->Crystal Configuration Manager, and open the properties page for Report Application Server. Select the Parameters tab, and in "Option Type" choose "Server" and change the "Report Directory" setting.

The directory setting specifies which folder the Report Application Server has access to. For example, "C:\" would allow access to the entire C drive and all sub-folders. However, "C:\Reports" would allow access to only the Reports folder and all direct sub-folders. For the samples in the default install to work, you would specify "C:\Program Files\Crystal Decisions\Report Application Server 10"

Once you have your RAS server configured, you can change or add ReportDocument class instances to your .NET application and make the following property changes to the Report:

1. Set the FileName property to rassdk://<path to your report> if the report is stored on the same machine as your web server. Set the FileName property to ras:// <path to your report> if the report is stored on the same machine as your RAS Server.
2. Set the ReportAppServer property to the name of the RAS server machine.

For better performance, it is best to avoid streaming the file from your client SDK machine to the RAS server. To achieve this, change the FileName property in your application to use the “ras://” prefix and copy your report file to a location accessible to RAS Server machine.

2. Upgrading Your .NET Application to Managed RAS

RAS provides the ability to modify managed reports through code before displaying the report to the end user. It combines the manageability and scalability of Crystal Enterprise with the customizability of RAS. Developers familiar with the RAS object model provided by the .NET embedded RAS can apply the same techniques to managed reports because the object model is identical regardless of the context RAS is running in.

To use managed reports, you first need to install Crystal Enterprise and publish your reports to your Enterprise system.

2.1 Connecting Reports using Visual Studio's Server Explorer

Once you have published your reports, the easiest way to access managed reports is to drag them into your project from the Server Explorer. Doing so automatically sets up the ReportDocument class instance with the required information.

In the Visual Studio Server Explorer, add your Enterprise server if it has not already been added to the explorer (right click and select 'Add Server'). Under the server's name, expand the 'Crystal Services' node, and then the 'Crystal Enterprise' node. You will be prompted to log on. Proceed by navigating your Enterprise system and finding the report or possibly the instance of the report that you want to view. Drag it to your project and change the viewer's ReportSource property to be set to the new ReportDocument instance.

2.2 Connecting Reports Programmatically

Reports can be connected to your viewer in code using a number of methods.

To use the Crystal Enterprise and RAS classes in a .NET project, include the following references:

```
using CrystalDecisions.Enterprise;
using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
using CrystalDecisions.ReportAppServer.ClientDoc;
```

For the purposes of the samples below, it is assumed that these statements have been added to your project. Additionally, add the following member variables to your form:

```
private ReportDocument      m_reportDocument = null;
private EnterpriseSession   m_enterpriseSession = null;
```

2.3 Using the Enterprise Session Manager to Retrieve Reports

The Enterprise Session Manager is able to query for a report and view the results. The Session Manager allows you to query for a report or set of reports. To use this method, follow these steps:

Add the following code to the Web Form class:

```
protected InfoObject GetInfoObject( EnterpriseSession enterpriseSession, string objName )
{
        InfoStore     iStore = (InfoStore) enterpriseSession.GetService("", "InfoStore");
        InfoObjects   infoObjects =
                iStore.Query( "SELECT * FROM CI_INFOOBJECTS WHERE SI_NAME LIKE '" + objName +
"%'" );
        if ( infoObjects == null || infoObjects.Count <= 0 )
        {
```

```
                return ( null );
        }

        // Note: The InfoObjects collection is 1-based
        InfoObject obj = (InfoObject) infoObjects[1];

        return ( obj );
}

private void setupManagedRASReport()
{
        SessionMgr enterpriseSessionMgr = new SessionMgr();

        try
        {
                m_enterpriseSession = enterpriseSessionMgr.Logon(
                        "Administrator",
                        "",                 // Password
                        "iantreleaven3",    // Server
                        "secEnterprise"
                        );

                m_reportDocument = new ReportDocument();

                // Fetch the InfoObject
                InfoObject  infoObj = GetInfoObject( m_enterpriseSession, "Charting" );

                // Load the report
                m_reportDocument.Load(infoObj, m_enterpriseSession);

                this.CrystalReportViewer1.ReportSource = m_reportDocument;
        }
        catch ( Exception )
        {
        }
        finally
        {
                enterpriseSessionMgr.Dispose();
                enterpriseSessionMgr = null;
        }
}
```

Add the following code to the end of the InitializeComponent method, which part of the Web Form Designer generated code:

```
// Set up the managed report document and set it into the viewer
setupManagedRASReport();
```

Now that you have opened a report, you can manipulate it as you would a report opened via the embedded .NET engine or unmanaged RAS server.

2.4 Using the CEIS URL

You can load a managed report directly from Crystal Enterprise by specifying a ceis:// URL. The report name or report number can be specified. The code written below is a sample method that loads the World Sales Report:

```
private void setReportUsingCEIS()
{
        m_reportDocument = new ReportDocument();
        m_reportDocument.EnterpriseLogonInfo.ApsServer = "iantreleaven3";
        m_reportDocument.EnterpriseLogonInfo.AuthenticationType = "secEnterprise";
        m_reportDocument.EnterpriseLogonInfo.Username = "Administrator";
        m_reportDocument.EnterpriseLogonInfo.Password = "";
        m_reportDocument.Load("ceis://@iantreleaven3/Report Samples/general business/World
Sales Report");
```

```
        this.CrystalReportViewer1.ReportSource = m_reportDocument;
}
```

Add the following code to the end of the InitializeComponent method, which is part of the Web Form Designer generated code:

```
// Set up the managed report document and set it into the viewer
setReportUsingCEIS ();
```

You can use the Crystal Enterprise report object syntax by replacing the report name with a string in the format, “#xxx”. For example:

```
m_reportDocument.Load("ceis://@iantreleaven3/Report Samples/general business/#195”;
```

3. Upgrading to Enterprise Page Server

Crystal Enterprise provides the ability to view managed reports. The Page Server, which is part of every Enterprise system, provides access to the greatest performance and scalability options with the trade-off of less runtime manipulation capability.

To use managed reports, you first need to install Crystal Enterprise and publish your reports to your Enterprise system.

To use the Crystal Enterprise classes in a .NET project, include the following references:

```
using CrystalDecisions.Enterprise;
using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
using CrystalDecisions.ReportAppServer.ClientDoc;
using CrystalDecisions.Enterprise.Viewing;
using CrystalDecisions.ReportAppServer.Controllers;
```

You also need to add a reference to the CrystalDecisions.Enterprise.Viewing.ReportSource library.

For the purpose of the samples below, it is assumed that these statements have been added to your project. Additionally, add the following member variables to your form:

```
private ReportDocument      m_reportDocument = null;
private EnterpriseSession   m_enterpriseSession = null;
```

3.1 Using the Page Server Report Factory

The Page Server Report Factory provides access to reports and report instances in an Enterprise system.

Add the following code to the Web Form class:

```
protected InfoObject GetInfoObject( EnterpriseSession enterpriseSession, string objName )
{
	InfoStore     iStore = (InfoStore) enterpriseSession.GetService("", "InfoStore");
	InfoObjects   infoObjects =
		iStore.Query( "SELECT * FROM CI_INFOOBJECTS WHERE SI_NAME LIKE '" + objName +
"%'" );
	if ( infoObjects == null || infoObjects.Count <= 0 )
	{
		return ( null );
	}

	// Note: The InfoObjects collection is 1-based
	InfoObject obj = (InfoObject) infoObjects[1];

	return ( obj );
}

private void setReportUsingPageServer()
{
	SessionMgr enterpriseSessionMgr = new SessionMgr();

	try
	{
		m_enterpriseSession = enterpriseSessionMgr.Logon(
			"Administrator",
			"",                  // Password
			"IANTRELEAVEN3",     // Server
			"enterprise"// "secEnterprise"
			);
```

```
                // Fetch the InfoObject
                InfoObject  infoObj = GetInfoObject( m_enterpriseSession, "Charting" );

                PSReportFactory factory = (PSReportFactory)
m_enterpriseSession.GetService("PSReportFactory").Interface;
                ISCRReportSource reportSource = factory.OpenReportSource(infoObj.ID);

                this.CrystalReportViewer1.ReportSource = reportSource;
        }
        catch ( Exception e )
        {
                Trace.Write(e.ToString());
        }
        finally
        {
                enterpriseSessionMgr.Dispose();
                enterpriseSessionMgr = null;
        }
}
```

Add the following code to the end of the InitializeComponent method, which part of the Web Form Designer generated code:

```
// Set up the viewer using the Page Server Report Factory
setReportUsingPageServer ();
```

Once you determine the InfoObject ID, you can improve the viewing performance by supplying the ID directly to the factory.OpenReportSource method call.

4. Setting Parameters and Logons in Code

An important part of most applications is to set report parameters and database logons before previewing. It is usually desirable to set parameters and logons before previewing the report thus the end users are not required to enter these values.

4.1 Using the ReportDocument

Applications using a ReportDocument object can set parameters and logons easily in code. Following are some examples of how to do this. In all the above scenarios except using the Enterprise PageFactory, the ReportDocument object is used.

To use the following code, add the following required statements to your code:

```
using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
```

To set a database logon, use the ReportDocument.SetDatabaseLogon method:

```
doc.SetDatabaseLogon("guest", "password");
```

To set a parameter value, use the ReportDocument.SetParameterValue method:

Boolean parameter example:

```
doc.SetParameterValue(“parameterName”, true);
```

Numeric parameter example:

```
doc.SetParameterValue(“parameterName”, 1234.56);
```

Multi-value parameter example:

```
string[] countries = new string[2];
countries[0] = “USA”;
countries[1] = “Canada”;
doc.SetParameterValue(“Choose Countries”, countries);
```

Ranged parameter example:

```
ParameterRangeValue salesRange = new ParameterRangeValue();
salesRange.Kind = DiscreteOrRangeKind.RangeValue;
salesRange.StartValue = 1000.0;
salesRange.EndValue = 15000.0;
doc.SetParameterValue("ValueRange", salesRange);
```

4.1 Using the Enterprise SDK

For applications accessing Crystal Enterprise, parameters and database logons can be set through code when using the Enterprise SDK. See the InfoObject documentation for more details.

5. Appendix - Complete Sample Code

To assist in your application upgrade, full C# sample code is provided for each scenario. This code can be copied and pasted into your application with minor modifications to the report name, server names, and authentication information specific to your scenario.

5.1 Crystal Reports for .NET Starter Application

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
using CrystalDecisions.Enterprise;
using CrystalDecisions.ReportAppServer.ClientDoc;
using CrystalDecisions.Enterprise.Viewing;
using CrystalDecisions.ReportAppServer.Controllers;


namespace EnterpriseMigration
{
	/// <summary>
	/// Summary description for WebForm1.
	/// </summary>
	public class WebForm1 : System.Web.UI.Page
	{
		protected CrystalDecisions.CrystalReports.Engine.ReportDocument
embeddedReportDocument;
		protected CrystalDecisions.Web.CrystalReportViewer CrystalReportViewer1;

		private void Page_Load(object sender, System.EventArgs e)
		{
			// Put user code to initialize the page here
		}

		#region Web Form Designer generated code
		override protected void OnInit(EventArgs e)
		{
			//
			// CODEGEN: This call is required by the ASP.NET Web Form Designer.
			//
			InitializeComponent();
			base.OnInit(e);

			CrystalReportViewer1.ReportSource = embeddedReportDocument;
		}

		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
			System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(WebForm1));
			this.embeddedReportDocument = new
CrystalDecisions.CrystalReports.Engine.ReportDocument();
			//
			// embeddedReportDocument - change to the actual name of your report
			//
			this.embeddedReportDocument.FileName =
"rassdk://c:\\ParameterLogonTest.rpt";
			this.embeddedReportDocument.InitReport += new
System.EventHandler(this.embeddedReportDocument_InitReport);
			this.Load += new System.EventHandler(this.Page_Load);

		}
		#endregion

		private void embeddedReportDocument_InitReport(object sender, System.EventArgs
e)
		{
		}
	}
}
```

5.2 Unmanaged RAS Application

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
using CrystalDecisions.Enterprise;
using CrystalDecisions.ReportAppServer.ClientDoc;
using CrystalDecisions.Enterprise.Viewing;
using CrystalDecisions.ReportAppServer.Controllers;


namespace EnterpriseMigration
{
        /// <summary>
        /// Summary description for WebForm1.
        /// </summary>
        public class WebForm1 : System.Web.UI.Page
        {
                protected CrystalDecisions.CrystalReports.Engine.ReportDocument
embeddedReportDocument;
                protected CrystalDecisions.Web.CrystalReportViewer CrystalReportViewer1;

                private void Page_Load(object sender, System.EventArgs e)
                {
                        // Put user code to initialize the page here
                }

                #region Web Form Designer generated code
                override protected void OnInit(EventArgs e)
                {
                        //
                        // CODEGEN: This call is required by the ASP.NET Web Form Designer.
                        //
                        InitializeComponent();
                        base.OnInit(e);

                        CrystalReportViewer1.ReportSource = embeddedReportDocument;
                }

                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                        System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(WebForm1));
                        this.embeddedReportDocument = new
CrystalDecisions.CrystalReports.Engine.ReportDocument();
                        //
                        // embeddedReportDocument - change to the actual name of your report
                        // Report is copied to RAS Server machine
                        this.embeddedReportDocument.FileName =
"ras//c:\\ParameterLogonTest.rpt";
                        this.embeddedReportDocument.ReportAppServer = "MyRASServer"
                        this.embeddedReportDocument.InitReport += new
System.EventHandler(this.embeddedReportDocument_InitReport);
                        this.Load += new System.EventHandler(this.Page_Load);
```

```
		}
		#endregion

		private void embeddedReportDocument_InitReport(object sender, System.EventArgs
e)
		{
		}
	}
}
```

5.3 Managed RAS Application

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
using CrystalDecisions.Enterprise;
using CrystalDecisions.ReportAppServer.ClientDoc;
using CrystalDecisions.Enterprise.Viewing;
using CrystalDecisions.ReportAppServer.Controllers;


namespace EnterpriseMigration
{
	/// <summary>
	/// Summary description for WebForm1.
	/// </summary>
	public class WebForm1 : System.Web.UI.Page
	{
		protected CrystalDecisions.CrystalReports.Engine.ReportDocument
managedRASReportDocument;
		protected CrystalDecisions.Web.CrystalReportViewer CrystalReportViewer1;

		private void Page_Load(object sender, System.EventArgs e)
		{
			// Put user code to initialize the page here
		}

		#region Web Form Designer generated code
		override protected void OnInit(EventArgs e)
		{
			//
			// CODEGEN: This call is required by the ASP.NET Web Form Designer.
			//
			InitializeComponent();
			base.OnInit(e);

			CrystalReportViewer1.ReportSource = managedRASReportDocument;
		}

		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
			this.managedRASReportDocument = new
CrystalDecisions.CrystalReports.Engine.ReportDocument();
			//
			// managedRASReportDocument
			//
```

```
                        this.managedRASReportDocument.EnterpriseLogonInfo.AuthenticationType =
"secEnterprise";
                        this.managedRASReportDocument.EnterpriseLogonInfo.CmsServer =
"iantreleaven";
                        this.managedRASReportDocument.EnterpriseLogonInfo.Username =
"Administrator";
                        this.managedRASReportDocument.FileName =
"ceis://@IANTRELEAVEN.MYDOMAIN.NET/#106850";
                        this.managedRASReportDocument.UriIsUserEditable = false;
                        this.Load += new System.EventHandler(this.Page_Load);

                }
                #endregion

                private void embeddedRASReportDocument_InitReport(object sender,
System.EventArgs e)
                {
                }
        }
}
```

5.4 Managed Page Server Application

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
using CrystalDecisions.Enterprise;
using CrystalDecisions.ReportAppServer.ClientDoc;
using CrystalDecisions.Enterprise.Viewing;
using CrystalDecisions.ReportAppServer.Controllers;


namespace EnterpriseMigration
{
        /// <summary>
        /// Summary description for WebForm1.
        /// </summary>
        public class WebForm1 : System.Web.UI.Page
        {
                protected CrystalDecisions.Web.CrystalReportViewer CrystalReportViewer1;

                private EnterpriseSession   m_enterpriseSession = null;

                private void setReportUsingPageServer()
                {
                        SessionMgr enterpriseSessionMgr = new SessionMgr();

                        try
                        {
                                m_enterpriseSession = enterpriseSessionMgr.Logon(
                                        "Administrator",    // User ID
                                        "",                 // Password
                                        "IANTRELEAVEN",     // Server
                                        "enterprise"        // Security type
                                        );

                                PSReportFactory factory = (PSReportFactory)
m_enterpriseSession.GetService("PSReportFactory").Interface;
                                ISCRReportSource reportSource =
factory.OpenReportSource(106850);//106850
```

```
                        this.CrystalReportViewer1.ReportSource = reportSource;
                }
                catch ( Exception e )
                {
                        Trace.Write(e.ToString());
                }
                finally
                {
                        enterpriseSessionMgr.Dispose();
                        enterpriseSessionMgr = null;
                }
        }

        private void Page_Load(object sender, System.EventArgs e)
        {
                // Put user code to initialize the page here
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
                //
                // CODEGEN: This call is required by the ASP.NET Web Form Designer.
                //
                InitializeComponent();
                base.OnInit(e);

                setReportUsingPageServer();
        }

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
                this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion

        private void embeddedRASReportDocument_InitReport(object sender,
System.EventArgs e)
        {
        }
    }
}
```

5.5 Setting Parameters in ReportDocument

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
using CrystalDecisions.Enterprise;
using CrystalDecisions.ReportAppServer.ClientDoc;
using CrystalDecisions.Enterprise.Viewing;
using CrystalDecisions.ReportAppServer.Controllers;
```

```
namespace EnterpriseMigration
{
	/// <summary>
	/// Summary description for WebForm1.
	/// </summary>
	public class WebForm1 : System.Web.UI.Page
	{
		protected CrystalDecisions.CrystalReports.Engine.ReportDocument
embeddedRASReportDocument;
		protected CrystalDecisions.Web.CrystalReportViewer CrystalReportViewer1;

		private void setParametersAndLogons(ReportDocument doc)
		{
			// Set the database logon
			doc.SetDatabaseLogon("guest", "password");

			// A boolean parameter
			doc.SetParameterValue("TrueFalse", true);

			// A string parameter
			doc.SetParameterValue("UserName", "Bob");

			// A multi-value parameter
			string[] countries = new string[2];
			countries[0] = "USA";
			countries[1] = "Canada";
			doc.SetParameterValue("Choose Country", countries);

			// A range parameter
		ParameterRangeValue salesRange = new ParameterRangeValue();
			salesRange.Kind = DiscreteOrRangeKind.RangeValue;
			salesRange.StartValue = 1000.0;
			salesRange.EndValue = 15000.0;
			doc.SetParameterValue("ValueRange", salesRange);
		}

		private void Page_Load(object sender, System.EventArgs e)
		{
			// Put user code to initialize the page here
		}

		#region Web Form Designer generated code
		override protected void OnInit(EventArgs e)
		{
			//
			// CODEGEN: This call is required by the ASP.NET Web Form Designer.
			//
			InitializeComponent();
			base.OnInit(e);

			// Establish the report document to use
			ReportDocument doc = embeddedRASReportDocument;

			// Pre-set the logon and parameters or leave them for the user to
enter.
			setParametersAndLogons(doc);

			// Set the report source for the viewer.
			CrystalReportViewer1.ReportSource = doc;
		}

		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
			System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(WebForm1));
			this.embeddedRASReportDocument = new
CrystalDecisions.CrystalReports.Engine.ReportDocument();
			//
```

```
                    // embeddedRASReportDocument
                    //
                    this.embeddedRASReportDocument.FileName =
"rassdk://c:\\ParameterLogonTest.rpt";
                    this.embeddedRASReportDocument.InitReport += new
System.EventHandler(this.embeddedRASReportDocument_InitReport);
                    this.Load += new System.EventHandler(this.Page_Load);

                }
                #endregion

                private void embeddedRASReportDocument_InitReport(object sender,
System.EventArgs e)
                {
                }
        }
}
```

Finding More Information

For more information and resources, refer to the product documentation and visit the support area of the web site at:
http://www.businessobjects.com/services/support/default.asp

- www.businessobjects.com

The Business Objects product and technology are protected by US patent numbers 5,555,403; 6,247,008; 6,578,027; 6,490,593; and 6,289,352. The Business Objects logo, the Business Objects tagline, BusinessObjects, BusinessObjects Broadcast Agent, BusinessQuery, Crystal Analysis, Crystal Analysis Holos, Crystal Applications, Crystal Enterprise, Crystal Info, Crystal Reports, Rapid Mart, and WebIntelligence are trademarks or registered trademarks of Business Objects SA in the United States and/or other countries. Various product and service names referenced herein may be trademarks of Business Objects SA. All other company, product, or brand names mentioned herein, may be the trademarks of their respective owners. Specifications subject to change without notice. Not responsible for errors or omissions. Copyright © 2004 Business Objects SA. All rights reserved.