crystal decisions™

Crystal Enterprise

Understanding Session Handling within Crystal Enterprise

Overview

This document is intended to provide background information about how sessions are handled within the Crystal Enterprise environment. As there are several different approaches when developing custom web interfaces to the Crystal Enterprise environment, it is important to understand how these different methodologies can be utilized and how they potentially can affect licensing. This document assumes the reader has a basic understanding of ASP/CSP programming and an understanding of the Crystal Enterprise architecture.

Contents

Introduction

Within the Crystal Enterprise environment, developers have the ability to create custom web interfaces to allow users to interact with objects held within the environment. The primary tool for the creation of these web pages is Crystal Server Pages (CSP). Crystal Server Pages, being very similar to Active Server Pagers (ASP), allow developers to create pages filled with dynamically created content – in this case, usually a mixture of reports and folders. By nature, a Crystal Server page is a mixture of HTML and scripting code such as VB Script or Java Script. Crystal Server Pages are evaluated by the Web Component Server (WCS) within the Crystal Enterprise environment. The Web Component Server may or may not be the same physical machine as the web server itself; however, it is important to note that WCS can be considered the gateway for requests being made within the Crystal Enterprise environment.

Within the Crystal Enterprise environment, the Automated Process Scheduler (APS) is the component responsible for managing objects and security for the system. When a user develops a web interface for the Crystal Enterprise environment, they will be logging on the system and returning the particular objects from the APS to satisfy the request of their code being processed on the Web Component Server. This result set is then presented to the client's browser based on the script that is processed on the Web Component Server. It is necessary to understand how, as a developer, you can control these interactions within the Crystal Enterprise environment to provide an efficient and seamless user experience while applying the administration controls that you may require.

Sessions Variables versus Enterprise Sessions

When discussing session handling within the Crystal Enterprise environment, it is necessary to differentiate between session management and Crystal Enterprise Sessions.

NOTE	To highlight text in a PDF document for copying and pasting code, click the **Text Select Tool** toolbar button in Adobe Acrobat. This procedure applies to Adobe Acrobat 4.0 and 5.0.

The Session Variable

When developing custom web applications, it is often necessary when working with multiple pages of an application to have a means by which information can be shared between the pages. Variables declared within script only maintain scope for the page in which they are created. Often, when developing applications, it is desirable to pass variable's values between pages of code. To do this, you would use a session variable. The particular variable will remain in affect for the duration of the users session with the system.

Crystal Server Pages support the use of session variables. To define a session variable within your .csp code, you would use the following syntax:

```
Session("VariableName") = value
```

Within the Crystal Enterprise environment, the Web Component Server evaluates .csp script. Therefore, the Web Component Server will store in memory the session variable(s) defined for users interaction with the system.

NOTE	Unlike ASP, CSP does not support the scope of Application.

The Enterprise Session

When developing .csp code, the ultimate goal will be to dynamically display objects contained within Crystal Enterprise. A typical scenario would have a user logon to CE from a web page and be presented with a listing of folders, reports and\or report instances that the user has permission to interact with.

In order to do this, the user will require an Enterprise session object. Once the user has a valid Enterprise session object they can then create an InfoStore object that can be used to query the APS for objects the user can interact with.

The Enterprise session object can be considered a valid ‘ticket’ to Crystal Enterprise. It can also be considered synonymous with a user license within the Enterprise environment. For the sake of user efficiency and license management, it is important to consider what we do with this valid ‘ticket’ once it’s obtained.

To create an Enterprise session we would use the following code:

```
<%
’DECARE VARIABLES
Dim username
Dim password
Dim apsname
Dim authentication
Dim sm
Dim es
Dim iStore

username = "TestUser"
password = ""
apsname = ""
authentication = "secEnterprise"

'CREATE A SESSION MANAGER SO THAT WE CAN LOGON
Set sm = CreateObject("CrystalEnterprise.SessionMgr")

'LOGON USER
Set es = sm.Logon(username, password, apsname,
authentication)

'STORE THE INFOSTORE SESSION
Set iStore = es.Service ("", "InfoStore")
%>
```

In the above code, once the logon is successful to the APS, the Enterprise session manager’s service function is used to create the InfoStore object. At this point, we could write additional code to return various objects from the APS for the user to interact with.

The variable iStore contains the InfoStore object and has the scope of the particular page that is being evaluated. When the script is finished processing, the iStore variable will no longer contain the InfoStore object. Therefore, if the user were to make another request, they would have to have to re-logon to the system in order to establish the Enterprise session and the InfoStore object.

Managing the Enterprise Session

As discussed in the previous section, concept of an Enterprise session is independent of the session scope of .csp programming.

Once we have established an Enterprise session and created an InfoStore object we can query the APS for objects. However, that connection is only available for the duration of the script being processed. If the user were to perform more queries, the user would have to logon to the system again to create a new Enterprise session and InfoStore object.

There are several approaches that can be utilized to manage the Enterprise session so that users don't have to keep logging into the system.

Storing the Enterprise Session in a session variable

The simplest way to keep the Enterprise session active for the user is to store the Enterprise session within a session variable. By doing so, the connection will be stored in memory on the WCS.

For the previous code sample, we could add the following line:

```
'STORE THE INFOSTORE SESSION
Set iStore = es.Service ("", "InfoStore")
Set Session("IStoreWCS")= iStore
```

By storing the iStore session in the variable IstoreWCS, the Enterprise session is then available from different .csp's for the duration of the user's session.

However, using session variables can be an expensive operation. As the Web Component Server will hold these session variables in memory, performance can be impacted when many users are connecting to the system – each storing session variables.

Another aspect of the above example that needs consideration is the fact that the Enterprise session is synonymous with a user license. Therefore, by storing the Enterprise session in the session variable, we are using a Crystal Enterprise license for the duration of the session.

To limit the lifespan of the session variable there are a couple of things that can be done. By default, a session will expire after 20 minutes of inactivity. You can change this default by setting the TimeOut property within your code. For example:

```
Session.TimeOut = 10
```

You can also destroy the session variables before the timeout setting by calling the abandon method that will destroy all session variables for the browseror you could just clear the specific session variable:

```
Session("IStoreWCS") = Nothing

Session.Abandon
```

Storing the Enterprise session in a session variable is the easiest method for managing Enterprise sessions. By doing so, the user will have a seamless experience while the developer can use multiple .csp pages to present the user with objects from the Enterprise environment.

However, this ease comes with a price in terms of performance on the Web Component Server, and possibly will create a scenario where user licenses are not used efficiently.

Using a Logon Token to reinstate the Enterprise session

One of the features of Crystal Enterprise is the ability to create a logon token to facilitate the re-creation of the Enterprise Session without the user constantly having to keep logging on to the Enterprise system. Logon tokens are easy to create within .csp script. Once a logon token is obtained, it is then stored in a cookie on the client's machine.

By using logon tokens, you can utilize the LogonWithToken method to reinstate the Enterprise session as required. While this strategy will take some extra calls within your .csp code, it will provide a scenario where user licenses are only used when information is being retrieved from the APS, instead of tying up the license for the duration of the session as discussed in the previous method.

To create a logon token, you would use the LogonTokenMgr property of the Enterprise session object. Once obtained, store the logon token in a cookie with the name "LogonToken". The syntax for creating the logon token is as follows:

```
Dim LogonTokenMgr
Set LogonTokenMgr = Sess.LogonTokenMgr
Response.Cookies("LogonToken") =
LogonTokenMgr.CreateLogonToken("", 1, 100)
```

It is important to note that the logon token can only be created after the logon has successfully occurred. In the above code, the logon token is created in the line:

```
LogonTokenMgr.CreateLogonToken("", 1, 100)
```

When requesting a logon token we have the ability to control the number of days the token is valid for and\or the number times the token can be used.

After the logon token has been written as a cookie on the client's machine, we can then use this token to logon to the Crystal Enterprise environment anytime a query is to be made to the APS to retrieve information.

To reinstate the client's enterprise session using the logon token we would use the LogonWithToken method. Once we have reinstated the enterprise session, we can then create the InfoStore object and query the APS.

To logon using the logon token:

```
'Retrieve the cookie
LogonToken = Request.Cookies("LogonToken")
'Create a SessionManager
Set SessionManager =
CreateObject("CrystalEnterprise.SessionMgr")
'Logon using the tokon. This may fail if the token is no
longer valid.
Set Sess = SessionManager.LogonWithToken(LogonToken)
'Create the IStore object
Set IStore = Sess.Service("", "InfoStore")
```

For the sake of the above sample, error checking has not been included; however, it would be a good idea to check that the cookie exists. As well, the LogonWithToken method could fail if the logon token has exceeded the number of days or number of uses specified when the token was initially created.

Using logon tokens provides for a mechanism where the Enterprise session does not have to be held open for the duration of the user's session with the Enterprise system; this will result in a more efficient use of user licenses. However, utilizing logon tokens will require some extra steps when developing your application. When developing using this strategy, another consideration to take into account is that your pages will be logging into the Enterprise system more often than if you stored the Enterprise session in a session variable. This will result in some additional overhead for your requests, as you cannot immediately perform an InfoStore query from the .csp page that the user is visiting as you will first have to:

1. Check for the presence of the cookie

2. Get the logon token

3. Logon

4. Create the InfoStore Object

5. Perform query against APS

Using a combination of Session Variables and Logon Tokens

Probably the most ideal method for managing Enterprise session is to use a combination of Session variables and logon tokens. By using a combination of both techniques, the .csp developer can create a user experience that is seamless while maintaining efficient use of user licenses and server performance.

In this scenario, the user would logon to the Enterprise system. The .csp code would establish the enterprise session and store it in a session variable. The .csp code would also create a logon token that would be written to a cookie.

From the user's perspective, they would log into the Enterprise system once. While they are actively navigating through different .csp pages and interacting with objects, the enterprise session is stored in a server variable. If, due to inactivity, the user's session is timed out, when they resume interacting with the .csp page, the logon token will be used to reinstate the enterprise session and it will occur seamlessly to the user.

This technique will require the most coding but will produce the most efficient user experience.

The various sample applications provided with Crystal Enterprise as well as the developer's tutorial found within the Crystal Enterprise Web Developers Guide use this technique. In the simplest form, a .csp page is created that handles checking the existence of a current enterprise session or reinstating the enterprise session by performing a logon utilizing the logon token. This generic page is then included as a server-side include to all .csp pages that the user will access.

An example of a generic page that performs these actions is as follows:

```
<%

Function RetrieveIStore(IStore)
'Precondition:
'None
'
'Postcondition:
'IStore - Returns the InfoStore object that allows us to
query the server
'
'The function returns TRUE if successful and FALSE
otherwise.
On Error Resume Next

Dim LogonToken
Dim SessionManager
Dim Sess

'Check to see if the cookie is there.
If( Request.Cookies("LogonToken") <> "" ) Then

    'Check to see if the InfoStore already exists
          If( TypeName(Session("IStore")) <> "ISInfoStore"
      ) Then

'Retrieve the cookie
LogonToken = Request.Cookies("LogonToken")

'Create a session manager
Set SessionManager =
Server.CreateObject("CrystalEnterprise.SessionMgr")

'Logon using the token. This may fail if the token is no
longer valid.
Set Sess = SessionManager.LogonWithToken(LogonToken)

'Create the InfoStore object.
Set IStore = Sess.Service("", "InfoStore")

'Save the InfoStore in the session.
Session("IStore") = IStore

RetrieveIStore = TRUE

    Else
            'The InfoStore already exists so simply
retrieve it from
            'the session.
            Set IStore = Session("IStore")
            RetrieveIStore = TRUE
    End If
Else
    RetrieveIStore = FALSE
End If
End Function

%>
```

The above code performs the following steps:

Checks to see if LogonToken exists in cookie

If logon token exists in cookie, checks to see InfoStore object exists in session variable

If InfoStore object doesn't exist, retrieves the logon token into LogonToken variable

Creates the Enterprise Session Manager

Performs Logon using LogonWithToken method

Once logon occurs, creates the InfoStore object

Stores the InfoStore object in session variable Istore

Sets flag that RetrieveIStore has been successful

If the InfoStore object does exist in session variable, sets Istore variable.

Sets flag that RetrieveIStore has been successful.

The above page can be referenced by adding a server-side include to the other .csp pages in your application by adding the following in each page:

```
<!-- #include file=RetrieveIStore.csp -->
```

Report Viewing and Sessions

The previous sections of this document have focused on the management of the Enterprise session, specifically how it relates to user licenses and the InfoStore object. When developing the application, it is also important to consider how sessions are handled once the user has a report object open in the report viewer.

When a user selects a link to view a report object, the report viewer is invoked and the page appears for the user to view. During this process, the viewer will create it's own session cookie for the duration that the report viewer is displaying the report. By default, the report viewing session will expire after 20 minutes. If a user were to leave a report open in a viewer and leave their desk, after 20 minutes the session (and user license) will be released. If the user were to come back and click on a page of the report, the session cookie would be used to reinstate the viewing session.

Getting More Information

For more information, look at the following documentation or contact Technical Support.

Product Documentation

Available in printed documentation or in electronic format on the Crystal Enterprise CD and in the Online Store at: http://store.crystaldecisions.net

- Crystal Enterprise Web Developer's Guide
- Crystal Enterprise Administrator's Guide

Technical Briefs:

Available from http://support.crystaldecisions.com/docs

- ASP vs CSP: Deployment Issues (CE8_ASP_vs_CSP.pdf)

Contacting Crystal Decisions for Technical Support

We recommend that you refer to the product documentation and that you visit our Technical Support web site for more resources.

Self-serve Support:

http://support.crystaldecisions.com/

Email Support:

http://support.crystaldecisions.com/support/answers.asp

Telephone Support:

http://www.crystaldecisions.com/contact/support.asp

****