

Single Sign-On Design and Implementation

By Citrix Consulting Services

Citrix Systems, Inc.



CITRIX®

Notice

The information in this publication is subject to change without notice.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. CITRIX SYSTEMS, INC. ("CITRIX"), SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR DIRECT, INCIDENTAL, CONSEQUENTIAL OR ANY OTHER DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS PUBLICATION, EVEN IF CITRIX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES IN ADVANCE.

This publication contains information protected by copyright. Except for internal distribution, no part of this publication may be photocopied or reproduced in any form without prior written consent from Citrix.

The exclusive warranty for Citrix products, if any, is stated in the product documentation accompanying such products. Citrix does not warrant products other than its own.

Product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

Copyright © 2001 Citrix Systems, Inc., 6400 NW 6th Way, Ft. Lauderdale, Florida 33309 U.S.A. All rights reserved.

Version History		
November 14, 2000	Katherine Schaefer	Version 1.0
November 21, 2000	Larry Sweeney, Katherine Schaefer	Version 1.1
December 1, 2000	Eric Kline, Everett Marshall	Version 1.2



Table of Contents

OVERVIEW	1
REQUIRED ARCHITECTURE	2
WEB PORTAL LAYOUT	2
ISSUES	4
SECURITY CONCERNS	4
CODE REFERENCES	5
<i>DetectBrowser()</i>	5
<i>getUserInfo()</i>	6
<i>Redirect.asp</i>	6
<i>Login.asp</i>	7
<i>ReAuthenticate.asp</i>	7
<i>NFuse Authentication</i>	7



Overview

Single sign-on allows users to navigate to multiple Web sites from dynamically created or static links in a portal, without having to provide user credentials more than once. With single sign-on, users need log in only once to be authenticated “behind the scenes” to external Web applications (EWAs), such as Outlook Web Access and NFuse applications, residing on remote servers.

Single sign-on can be achieved by dynamically creating URLs inside of the ASP pages and enabling IIS basic authentication to authenticate the user. The user credentials are “stuffed” into the http:// request for the external Web application.

This white paper explains how to implement single sign-on with MetaFrame and NFuse.

Required Architecture

Web portal layout

The following diagram illustrates a sample portal page layout and process flow that incorporates single sign-on and NFuse.

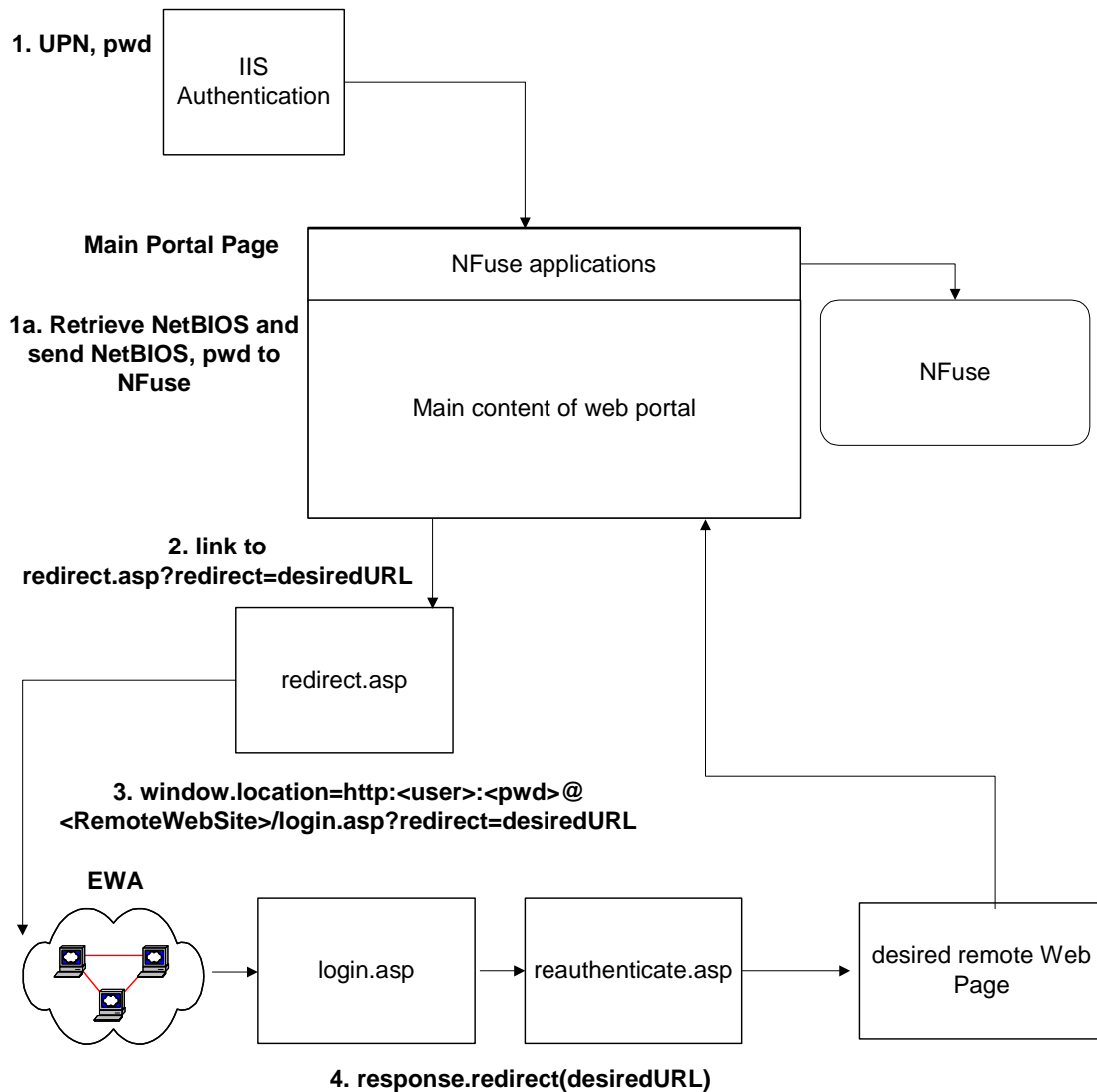


Figure 2: Web Portal / NFuse Integrated Process

The following process occurs when a user accesses the Web portal interface to launch an application published on a MetaFrame server:

1. When the user first attempts to access the Web portal, he is presented with the standard IIS authentication login box.
Typically, the user log in with the User Principal Name (UPN) credentials (i.e. john.doe@citrix.com). When the user is authenticated they are served the WebTop homepage. A function is called to retrieve the user's netBIOS name using ADSI. The netBIOS name is saved and passed to NFuse at a later time. [The netBIOS name is required because versions of NFuse released prior to version 1.5 do not support UPN authentication.]
2. Because EWAs are typically housed on different IIS servers than the Web portal interface, re-authentication is required when the user clicks a link for a specific EWA. To avoid re-authentication, links to EWAs should point to the redirect.asp page on the Web portal server, with the desired EWA URL passed in the query string as a parameter.
3. Redirect.asp causes a client-initiated redirection (window.location rather than Response.Redirect) to a custom login page that must sit on each IIS server that hosts an EWA. The redirection will be of the form:
`https://<user>:<pwd>@<RemoteLoginUrl/login.asp>?redirect=<urlUserDesires>`
4. The user name and password variables are substituted via ASP (they reside as session variables), and the URL that the user expects to be served is included at the end of the query string.
Note: The redirection must be client-initiated in order to force the authentication handshake to occur with the client and not the server.
5. Any remote EWA that requires single sign-on must have custom login.asp and reauthenticate.asp pages associated to it that reside on the EWA's Web server. The login.asp page exists to accept the credentials from the first communication with the originating IIS/EWA server via the URL's <username>:<pwd> values. Login.asp calls reauthenticate.asp which will Response.Redirect the browser to the URL desired by the user (listed in the query string by the "redirect" parameter.)

Issues

Common issues with implementing single sign-on are discussed below. See “Code References” later in this document for code samples you can use to handle these issues.

- **Checking browser type and version upon login to determine Web access capabilities**
The browser type is determined when the user initially logs in. If the user’s browser supports single sign-on, the user is allowed access. Otherwise, the user is sent back to the login page. The browsers that currently support single sign-on include Netscape 6.0 and Internet Explorer 4.0 and above.
- **ADSI Integration**
User groups and other user information can be retrieved using the UPN name of the user and querying Active Directory.
- **Passing user credentials to NFuse for automatic authentication**
The user credentials are passed to NFuse from the Web portal page through session variables. The session variables are then set equal to the transient cookies that NFuse expects. This eliminates the need for the user to log in again.
- **Ticketing**
Ticketing is used to eliminate the need for user credentials to be included in the ICA file. When you use ticketing, the user credentials, which are held on the server, are linked to a ticket that is included in the ICA file. When the user selects an application to run, the ticket validates the user’s credentials and allows access to the application. The ticket has an expiration time and can be used only once.
- **NFuse 1.5 ICA Client download and SecureICA**
The most updated SecureICA Client is compatible for use with NFuse 1.5. SecureICA is provided with the entire ICA Client.

Security Concerns

Common security concerns and their solutions are listed below.

1. **Encryption of user credentials when traversing the network**
 - Use SSL with NFuse 1.5
 - Use SecureICA with 128-bit encryption
 - Use ticketing to eliminate the user credentials from the ICA file and hold the credentials on the server
2. **Continued existence of cookie data**
 - The cookies being used by NFuse are transient cookies; therefore they are destroyed when the user closes the browser.
 - The user credentials are being sent from page to page in session variables for authentication to take place. The pages reside on the server and only the application information is supplied to the browser.
 - Ticketing eliminates the risk of user credentials being exposed on the client side or while traversing the network.

Code References

This section includes code samples you can use to implement single sign-on.

DetectBrowser()

Use the following code to detect a Web browser's properties and capabilities, including the browser name and version, and whether the browser supports frames, tables, VBScript, and/or JavaScript

Pseudo code:

```
Function DetectBrowser()  
  
    'Create variables to hold return values  
    Dim strBrowser  
    Dim strVersion  
    Dim blnIsFrames  
    Dim blnIsTables  
    Dim blnIsVBScript  
    Dim blnIsJavaScript  
  
    Set bc = Server.CreateObject("MSWC.BrowserType")  
  
    'Set variables  
    strBrowser = bc.browser  
    strVersion = bc.version  
    'Determine the properties that are supported by the browser.  
    'Support Frame  
    if (bc.frames = TRUE) then  
        blnIsFrames = "True"  
    else  
        blnIsFrames = "False"  
    end if  
    'Support Tables  
    if (bc.tables = TRUE) then  
        blnIsTables = "True"  
    else  
        blnIsTables = "False"  
    end if  
    'Support VBScript  
    if (bc.vbscript = TRUE) then  
        blnIsVBScript = "True"  
    else  
        blnIsVBScript = "False"  
    end if  
    'Support JavaScript  
    if (bc.javascript = TRUE) then  
        blnIsJavaScript = "True"  
    else  
        blnIsJavaScript = "False"
```

end if
End Function

getUserInfo()

Purpose: This function will retrieve required user information from Active Directory.

Inputs: The input shall be the UPN username.

Output: getUserInfo() shall return the user's NetBIOS name, full name, and the list of the groups of which the user is a member.

Pseudo code:

```
Public Function getUserInfo(ByVal username As Variant) as Variant
Create IADS variables that will be used to gain access to Active Directory.
Dimension and set variable that will hold the UPN that has been passed into the function.
Query Active directory to retrieve and set variables:
    Set NetBios Name
    Set Full name
    For each group user belongs
        Set group name
    Next
End Function
```

Redirect.asp

This page exists on the Web portal and accepts the following inputs from the main Web portal page when the user clicks a dynamic EWA link:

Virtual IP address (RemoteLoginURL) of the Web server where the EWA resides

User UPN name

User password

URL user desires

Login.asp for the EWA

When you use the inputs listed above in the following function, there is a client-initiated redirection to a custom login page (login.asp) that resides on the IIS server hosting the EWA.

redirectClientConnection()

```
StrURL = "https://<user>:<password>@RemoteLoginURL/<login.asp>?redirect=<URLUserDesires>"
Window.location = StrURL
```

Login.asp

Login.asp exists on the remote Web server to prevent credential information from being displayed in the URL in the event that the initial Web page for the EWA cannot be displayed correctly. This page must allow anonymous access. After receiving credential information from redirect.asp, this page sends the user to the ReAuthenticate.asp page on the EWA server.

```
StrURL =  
https://<%=Server.UrlEncode(Session("WebTopUsername"))%>:<%=Server.UrlEncode(Session("WebTopPassword"))%>@authenticate.asp
```

```
Window.location.href = StrURL
```

ReAuthenticate.asp

ReAuthenticate.asp authenticates the user to the EWA server. Consequently, anonymous access should not be allowed. When the user is transparently authenticated, he is redirected to the desired page on the EWA server based on information passed by the original WebTop server.

The desired URL is passed from WebTop's redirect.asp to the EWA's login.asp to the EWA's authenticate.asp. Using the desired URL, there is a server-initiated redirection (since the client has already been authenticated) to the desired page.

```
StrURL = Request("redirect")
```

```
Response.Redirect("strURL")
```

NFuse Authentication

The code in this section is found in the applist.asp file, which is installed during NFuse 1.5 installation. Below is a detailed description of the necessary modifications to the applist.asp file.

NFuse normally handles its own authentication through a standard login page. For single sign-on, however, the authentication takes place before NFuse is contacted to gather the user applications. Therefore, the user credentials will be harvested from session variables created in authenticate.asp. Using these credentials, the applist.asp file returns a list of applications that pertain to that user.

The code is changed to match the sample code below. Information is placed into a transient cookie that is used each time the authenticated user opens an application. Because the cookie is transient, it is not saved to the client machine and is eliminated when the browser is closed.

```
user = app.urlEncode(Session("WebTopNetBios"))
```


```
domain = app.urlEncode(Session("WebTopDomain"))
```

```
password = app.urlEncode(Session("WebTopPassword"))
```

```
Response.Cookies("NFuseData")("NFuse_User") =user
```

```
Response.Cookies("NFuseData")("NFuse_Domain") =domain
```

```
Response.Cookies("NFuseData")("NFuse_Password") =password
```



Applist.asp searches for the applications related to the specific user. The numtotal variable returns the number of applications received. If this variable returns 0, an error message is displayed. If applications are returned, they are displayed as normal.

```
numApps = appEnumerator.getNumApps(currentFolder)
```

```
numFolders = appEnumerator.getNumFolders(currentFolder)
```

```
numTotal = numApps + numFolders
```

```
If numTotal = 0 Then
```

```
    'Do nothing and display a blank page, no apps are published
```

```
Else
```

```
    'The apps are placed into a table and displayed vertically
```

```
End If
```



6400 NW 6th Way

Fort Lauderdale, FL 33309

954-267-3000

<http://www.citrix.com>



Copyright © 2001 Citrix Systems, Inc. All rights reserved. Citrix, WinFrame and ICA are registered trademarks, and MultiWin and MetaFrame are trademarks of Citrix Systems, Inc. All other products and services are trademarks or service marks of their respective companies. Technical specifications and availability are subject to change without prior notice.