

Administrator's Guide

NFuse

Version 1.51

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Citrix Systems, Inc.

© 2000-2001 Citrix Systems, Inc. All rights reserved.

Citrix, ICA (Independent Computing Architecture), MultiWin, DirectICA, SecureICA, Program Neighborhood, MetaFrame, NFuse, and *WINFRAME* are registered trademarks or trademarks of Citrix Systems, Inc. in the U.S.A. and other countries.

Microsoft, MS, MS-DOS, Windows, Windows NT, BackOffice, Active Server Pages, Internet Explorer, and Internet Information Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Netscape Navigator is a registered trademark of Netscape Communications Corporation in the United States and/or other countries. Netscape Enterprise Server is a trademark of Netscape Communications Corporation in the United States and/or other countries.

Apache is either a registered trademark or trademark of the Apache Software Foundation in the United States and/or other countries.

JavaServer Pages and iPlanet Web Server are either registered trademarks or trademarks of Sun Microsystems Corporation in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the U.S. and other countries. Solaris is a trademark or registered trademark of Sun Microsystems, Inc., in the United States and other countries. HP-UX is a registered trademark of Hewlett-Packard Company. AIX is a registered trademark of International Business Machines Corporation.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

RSA Encryption © 1996-1997 RSA Security Inc, All Rights Reserved.

This product incorporates IBM's XML Parser for C++ Edition and IBM's XML Parser for Java Edition.
© 1999, 2000 IBM Corporation.

All other Trade Names referred to are the Servicemark, Trademark, or Registered Trademark of the respective manufacturers.

Contents

Chapter 1 Before You Begin	9
Who Should Read this Guide	9
How to Use this Guide	9
Document Conventions	10
Finding More Information	11
Citrix Developer Network	12
Citrix on the World Wide Web	12
Reader Comments	13
Chapter 2 Welcome to Citrix NFuse	15
NFuse Features	16
New in NFuse 1.51	18
NFuse Components	18
Citrix Server Farm	18
Web Server	19
ICA Client Device	20
How NFuse Works	21
NFuse Programming Interface	22
NFuse Requirements	23
Citrix Server Requirements	23
Supported MetaFrame Versions	23
Additional Software Requirements	24
General Configuration Requirements	24
Backward Compatibility	24
Feature Limitations	25
Web Server Requirements	25
ICA Client Device Requirements	26
Overview of This Manual	28
What to Do Next	28

Chapter 3 Configuring Your Web Server.	29
Tasks to Complete	29
Web Server Extension Installation.	30
Installing the Web Server Extension on Microsoft IIS	30
Plugging the Citrix XML Service into IIS	32
Installing the Web Server Extension on Netscape Enterprise Server, iPlanet Web Server, and Apache Server	34
Configuring Netscape Enterprise Server and iPlanet Web Server	37
Configuring Apache Server	40
Installing the Web Server Extension Over a Previous Installation	43
Configuring Web Server Extension Properties	44
Introduction to the Citrix Web Site Wizard.	47
Installing the Citrix Web Site Wizard	47
Using the Citrix Web Site Wizard	48
Overriding the Default Citrix Server and Configuring SSL Support	48
Choosing a Scheme.	50
Choosing a Layout Model	50
Substitution-Tag-Based Layout	50
Scripting-Based Layout	51
Launching and Embedding.	52
Determining Launching and Embedding Capabilities	52
Launching Applications	53
Embedding Applications	53
Ticketing.	54
Authenticating Users.	55
Using User Principal Names and Active Directory Support	56
Authentication and MetaFrame for UNIX Operating Systems	56
Storing Login Information	57
What to Do Next.	57

Chapter 4	Configuring ICA Client Devices	59
Tasks to Complete		59
Web-Based ICA Client Installation		60
ICA Client Installation Files		60
ICA Win32 Client Installation Archives		61
How Web-Based ICA Client Installation Works		62
Configuring Web Browsers		65
Configuring the ICA Java Client		65
Configuring the ICA Macintosh Client		66
What to Do Next		67
Chapter 5	Using NFuse Tags	69
NFuse Substitution Tags		69
Simple Web Site Creation		70
Template ICA File Creation		71
NFuse Session Fields		73
NFuse Substitution Tag and Session Field Reference		74
General Tags		74
Application Property Tags		80
User Interface Tags		84
Conditional Tags		84
NFuse Substitution Tag and Session Field Tutorial		86
Creating a Substitution-Tag-Based Web Site		86
Files Included in a Substitution-Tag-Based Web Site		87
Application List Page		91
Using Session Fields		97
Setting Session Fields		98
Session Field Precedence		102
Chapter 6	NFuse Java Object Reference	103
NFuse Java Objects		104
CitrixWireGateway		104
ClearTextCredentials		108
GroupCredentials		110
AppEnumerator		113
App		119
AppSettings		130
AppDataList		135
AppListCache		137
TemplateParser		140

Chapter 7 ICA File Reference	145
ICA File Structure	146
[WFClient]	146
[ApplicationServers]	147
[ApplicationName]	147
ICA File Parameters	147
General Parameters	148
User Credential Parameters	149
Window Size and Color Parameters	150
Persistent Caching Parameters	151
TCP/IP Browsing Parameters	152
NFuse's Server Location Options	153
Encryption Parameters	155
Configuring Authentication Over Encrypted Connections	156
SOCKS Proxy Parameters	158
SpeedScreen Latency Reduction Parameters	160
Client Auto Update Parameters	160
Client Device Mapping Parameters	161
Chapter 8 Configuring NFuse Security	163
Client Device — Web Server Communication	164
Risks	164
Recommendations	165
Implement SSL-Capable Web Servers and Web Browsers	165
Encrypt Cookie Data	166
Use Ticketing	166
Web Server — Citrix Server Communication	169
Risks	169
Recommendations	170
Use the Citrix SSL Relay	170
Run a Web Server on Your Citrix Server	172
ICA Client — Citrix Server Communication	173
Risks	173
Recommendations	174
Use RC5 Encryption with Ticketing	174

Chapter 9 Example Web Sites	177
Getting Started	178
Improving NFuse Performance	179
Implementation Notes	179
Basic ASP/JSP and Per-User Application Caching	180
Using the Web Site	180
Implementing Per-User Application Caching	181
Application Caching and Filtering by Group	181
Using the Web Site	182
Implementing Filtering by Group	184
Application Caching and Filtering by Folder	184
Using the Web Site	185
Implementing Filtering by Folder	187
Securing NFuse	188
Implementation Notes	188
Ticketing	188
Using the Web Site	189
Implementing Ticketing	191
Cookie Data Encryption	191
Using the Web Site	192
Implementing Cookie Data Encryption	192
Improving Reliability and Usability	193
Backup MetaFrame Servers	193
Using the Web Site	193
Implementing Backup MetaFrame Servers	195
Multiple Server Farm Display	195
Using the Web Site	196
Implementing Multiple Server Farm Display	197
ICA Client Detection and Installation	198
Index	199

Before You Begin



Who Should Read this Guide

This guide is for Citrix server administrators and Web masters. Although in some cases a single person may be responsible for both Citrix server and Web server administration, this guide identifies these different administrative groups separately when describing specific NFuse tasks.

How to Use this Guide

This manual is both a task-oriented guide for quickly setting up NFuse and an in-depth reference for customizing NFuse functionality. Accordingly, initial chapters of the guide introduce NFuse and describe how to quickly deploy its components in a basic configuration.

Subsequent chapters describe NFuse's programming interface, security features, and example Web sites that you can use to enhance performance and secure NFuse transactions.

Document Conventions

The following conventional terms, text formats, and symbols are used throughout the printed documentation:

Convention	Meaning
Bold	Indicates column headings, command-line commands and options, dialog box titles, lists, menu names, tabs, and menu commands.
<i>Italic</i>	Indicates a placeholder for information or parameters that you must provide. For example, if the procedure asks you to type a filename, you must type the actual name of a file. Italics also indicate new terms and the titles of other books.
ALL UPPERCASE	Represents keyboard keys (for example, CTRL, ENTER, F2).
[brackets]	Encloses optional items in syntax statements. For example, <i>[password]</i> indicates that you can choose to type a password with the command. Type only the information within the brackets, not the brackets themselves.
...(ellipsis)	Indicates a command element can be repeated.
Monospace	Represents examples of screen text or entries that you might type at the command line or initialization files.
Code Sample	Example code appears in front of a gray background, as in the example below: <pre><html> <body></body> </html></pre>
➤	Indicates a procedure with sequential steps.
▪	Indicates a list of related information, not procedural steps.

Finding More Information

NFuse includes the following documentation:

- The “*Citrix NFuse Administrator’s Guide*” on *page Reference* (this manual) tells administrators how to install, configure, and customize NFuse.
- The online help for the Web Site Wizard gives procedural information on using NFuse’s Web page creation wizard. To access the help, click **Help** from any of the wizard’s screens.
- The NFuse Readme file contains last minute updates, corrections to the documentation, and a list of known problems. This file is located on the NFuse CD-ROM as well as on the Citrix download site (<http://www.citrix.com/download/>).
- The Web-based ICA Client installation Readme file contains information on using the standalone Web-based installation package that is included on the NFuse CD-ROM and in the downloadable Web-based installation image on the Citrix download site. See the file Readme.htm in the NFuse CD-ROM’s WebInst directory or in the downloaded CD image for information.
- The *Citrix MetaFrame XP Application Server for Windows, Version 1.0, Administrator’s Guide* tells administrators how to install and configure MetaFrame XP on Windows servers. Included in this documentation is information on publishing applications, configuring the Citrix XML Service, and configuring the Citrix SSL Relay. The *MetaFrame XP Administrator’s Guide* is available on the MetaFrame XP CD-ROM.
- The *Feature Release 1 and Service Pack 2 Installation Guide for Citrix MetaFrame for Windows Version 1.8* tells administrators how to install and configure Service Pack 2 and Feature Release 1 on MetaFrame 1.8 for Windows servers. Included in this documentation is information on configuring the Citrix XML Service and the Citrix SSL Relay. The Installation Guide is available on the Feature Release 1/Service Pack 2 CD-ROM and on the Citrix download site.
- The *Citrix MetaFrame for UNIX Operating Systems, Version 1.1, Administrator’s Guide* tells administrators how to install and configure MetaFrame for UNIX. Included in this documentation is information on publishing applications. The *MetaFrame for UNIX Administrator’s Guide* is available on the MetaFrame for UNIX CD-ROM.
- The *Citrix XML Service for UNIX Operating Systems Administrator’s Guide* explains how to install and configure the Citrix server component of NFuse that runs in a MetaFrame for UNIX server farm. This guide is available on the Citrix download site.

Citrix Developer Network

The Citrix Developer Network (CDN) is a Citrix program that extends the reach of Citrix application server technology to independent software vendors, independent hardware vendors, system integrators, ICA licensees, and corporate IT developers who want to incorporate Citrix server-based computing solutions into their products.

The Citrix Developer Network is a membership program with open enrollment. Through the new CDN Web site, Citrix provides access to developer tool kits, technical information, and test programs needed to successfully “design in” or add Citrix server-based computing compatibility to hardware and software. Today, the CDN program includes several software development kits (SDKs) and test kits, with an emphasis on delivering enabling technologies that promote technical relationships with Citrix.

Register for the Citrix Developer Network at the CDN Web site:

<http://www.citrix.com/cdn>

Citrix on the World Wide Web

The Citrix Web site, at <http://www.citrix.com>, offers a variety of information and services for Citrix customers and users. From the Citrix home page, you can access Citrix online Technical Support Services and other information designed to assist NFuse administrators, including the following:

- Downloadable Citrix ICA Clients (available at <http://download.citrix.com>)
- Program information on Citrix Preferred Support Services options
- An FTP server containing the latest service packs, hotfixes, utilities, and product literature for download
- An online Solution Knowledgebase containing an extensive collection of technical articles, troubleshooting tips, and white papers
- Interactive online Solution Forums for discussion of technical issues with other users
- Frequently Asked Questions pages with answers to common technical and troubleshooting questions
- Citrix Documentation Library containing the latest MetaFrame documentation
- Information about programs and courseware for Citrix training and certifications
- Contact information for Citrix headquarters, including worldwide, European, Asia Pacific, and Japan headquarters

Reader Comments

It is our goal to provide you with accurate, clear, complete, and usable documentation for Citrix products. If you have any comments, corrections, or suggestions for improving our documentation, we would be happy to hear from you.

You can send e-mail to the documentation authors at documentation@citrix.com. Please include the product name and version number, and the title of the document in your message.

Welcome to Citrix NFuse



Welcome to NFuse, Citrix's application management and deployment system. NFuse combines the centralized application management capabilities of Citrix server software with new techniques for Web application deployment into a highly customizable application delivery mechanism. Using NFuse, you can create standalone Web sites for application access or Web sites that can be integrated into your corporate portal.

NFuse brings a powerful user interface to the application deployment process. This interface uses Java object technology executed on a Web server to dynamically create an HTML-based presentation of the Citrix server farm for each of your users. Included in each user's presentation are all of the applications published in the Citrix server farm for that user.

NFuse is both a developer's tool and a Web master's application. NFuse includes an application programming interface and an easy-to-use wizard. The API lets you create customized Web server scripts from scratch to meet the requirements of your environment, while the wizard creates scripts for you that you can use as is or modify according to the NFuse API.

NFuse places complete control over the application deployment process in the hands of the administrator. Using NFuse's wizard and API, an administrator can configure on the Web server all ICA session options without ever visiting the user's desktop.

NFuse Features

A Web interface for Program Neighborhood. Now users of almost any ICA Client can benefit from the simplified application access provided by Program Neighborhood.

Dynamic user interface creation. NFuse lets you create template Web pages that Web servers can dynamically customize for each user.

Complete administrative control over application deployment. NFuse's use of Web server-side scripting lets you configure all ICA Client options in server-side scripts and ICA files.

Integration with popular Web technologies. NFuse's Java objects can be accessed from Web server scripts, such as Microsoft's Active Server Pages and Sun Microsystems' JavaServer Pages.

Simplified script writing. For those administrators not familiar with Web server scripting, NFuse includes simple HTML extensions. You can use these HTML extensions to write Web server scripts without having knowledge of scripting languages or scripting models such as VBScript, JavaScript, Active Server Pages, or JavaServer Pages.

Web Site Wizard. NFuse's Web Site Wizard creates complete NFuse-enabled Web sites. The Web Site Wizard contains configuration options for customizing your Web site.

Secure Sockets Layer (SSL) support. NFuse supports using SSL to secure communication between your Web server and Citrix server farm. *SSL* is an open, nonproprietary protocol that provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection. SSL support is provided by enhancements to the NFuse Java objects and requires use of the Citrix SSL Relay in your server farm. Implementing SSL along with Citrix RC5 encryption and a Web server and Web browsers that support SSL ensures the security of data as it travels your network.

Support for MetaFrame for UNIX Operating Systems. NFuse can deliver UNIX applications to your users. Support for MetaFrame for UNIX Operating Systems server farms allows NFuse to display and launch UNIX applications on your client devices. Use the Citrix Web Site Wizard to produce NFuse Web sites and the example sites to provide advanced functionality in your environment.

Note Some NFuse features are not available when using NFuse to access applications on MetaFrame for UNIX Operating Systems servers. See "Citrix Server Requirements" on page 23 for more information.

Ticketing. This feature provides enhanced authentication security. NFuse can create tickets that authenticate users to Citrix applications. Tickets have a configurable expiration period and are valid for a single login. After use, or after expiration, a ticket is invalid and cannot be used to access applications. Use of ticketing eliminates the explicit inclusion of credentials in the ICA files NFuse uses to launch applications.

Cookie data encryption. NFuse includes several example Web sites containing working code that you can integrate into your own NFuse Web pages. Among the example sites is one that displays how to encrypt credential data placed in NFuse cookies. Encrypting cookie data eliminates inclusion of clear text passwords in client-side cookies.

Application caching and filtering. NFuse Java objects provide the ability to cache published application information such as per-user application access lists on your NFuse Web server. Using cached application information speeds up enumeration of applications by eliminating the Web server's need to contact the Citrix server farm for each application request. Example Web sites demonstrate how to filter cached application information based on several criteria when displaying users' application sets.

Backup Citrix servers. This example Web site displays how to create a list of backup Citrix servers that the NFuse Web server can contact when generating a user's application set. Should the default Citrix server fail to respond to an NFuse request, the NFuse Web server sends the request to each of the servers in the backup list until one responds. Backing up your Citrix servers ensures that users have access to their applications in the event of server failure.

Web-based ICA Client installation. You can also use NFuse to deploy ICA Clients to your client devices. The NFuse Web Site Wizard and the example Web sites included in this release contain support for Web-based ICA Client installation. You can use this deployment method to deploy ICA Clients to any device that has a Web browser. When a client device user visits an NFuse Web site, the Web-based ICA Client installation code detects the device and Web browser types and prompts the user to install an appropriate ICA Client. In addition, this NFuse release includes support for a Web-based installation mechanism that is independent of NFuse. You can use this independent Web-based installation mechanism to perform general deployment outside of your NFuse sites.

Multiple server farm support in a single Web page. This example Web site demonstrates how to use a single NFuse Web site to contact multiple Citrix server farms and to display their applications in a single Web page. You can use this example to contact multiple MetaFrame for Windows farms, MetaFrame for UNIX Operating Systems farms, or a combination of both server farm types.

New in NFuse 1.51

Internet Server Application Program Interface (ISAPI) extension. The Citrix XML Service now contains an ISAPI extension that you can plug into Internet Information Server. Plugging the XML Service into IIS allows IIS to handle NFuse requests and serve NFuse Web pages on a single TCP/IP port. This configuration frees you from having to dedicate a port to the XML Service and is useful in environments that cannot permit multiple open TCP/IP ports on firewalls.

Active Directory and User Principal Name (UPN) support. All NFuse components are compatible with Microsoft's Active Directory. Users visiting NFuse Web pages can log into a Citrix server farm that is part of an Active Directory deployment and seamlessly access Citrix published applications. The login pages in NFuse Web sites are now compatible with Active Directory's use of User Principal Names.

NFuse Components

An NFuse deployment involves the interaction of three network components:

- A Citrix server farm
- A Web server
- ICA Client devices

Citrix Server Farm

A *Citrix server farm* is a group of Citrix servers managed as a single entity. A server farm is composed of a number of MetaFrame servers operating together to serve applications to ICA Client users. Citrix currently supports farms composed of MetaFrame for Windows servers and farms composed of MetaFrame for UNIX Operating Systems servers.

Important among a server farm's standard capabilities is application publishing. *Application publishing* is an administrative task that lets Citrix server administrators expose to users specific applications hosted by the server farm. When a Citrix server administrator publishes an application for a group of users, that application becomes available as an object to which ICA Clients can connect and initiate ICA sessions.

With MetaFrame Application Server for Windows Version 1.8 and *WINFRAME* 1.8, the Citrix server farm added Program Neighborhood to application publishing. *Program Neighborhood* automated the client-side configuration process by eliminating the need for administrators or ICA Client users to browse the network for published applications.

Using Program Neighborhood, any user can log in to the farm and receive a user-specific list of applications published for his or her user name. This list of applications is called an *application set*.

In an NFuse system, the Citrix server farm functions as an application-serving back end. In this role, the farm:

- Supplies application set information. The Citrix server responds to requests by NFuse for application set information. The server farm exports that information to an NFuse Web server for formatting into HTML pages that a user can view in a Web browser.
- Hosts applications. At a user's request (when the user clicks a link in a Web page), the server farm hosts an ICA session containing the requested application.

To communicate with the Windows or UNIX Citrix server farm, NFuse directs all requests to the Citrix XML Service. The *Citrix XML Service* is a MetaFrame component that provides published application information to ICA Clients and NFuse Web servers over the TCP/IP protocol. This service functions as the contact point between the server farm and NFuse's Web server component. The Citrix XML Service is installed with MetaFrame XP on MetaFrame XP for Windows systems, Citrix MetaFrame 1.8 Service Pack 2 on MetaFrame 1.8 for Windows systems, and Citrix XML Service for UNIX Operating Systems on UNIX systems.

Web Server

The Web server in an NFuse system hosts the NFuse Java objects and Web server-side scripts. You can call the NFuse Java objects from Web server-side scripts to perform several tasks that are important to NFuse. The NFuse Java objects provide the following services:

- Authenticate users to a Citrix server farm
- Retrieve application information, including a list of applications a user can access
- Give you the ability to modify the properties of individual applications before presenting them to users
- Create and send ICA files that users can access to start ICA sessions

The NFuse Java objects are added to your Web server as part of the *Citrix Web Server Extension*. This installation program also adds example Web sites that you can use to give your users access to their application sets and other example sites that demonstrate advanced NFuse features.

In addition, NFuse includes the *Web Site Wizard*, which creates customized Web sites that you can use as is or modify to fit your requirements. The wizard lets you create various types of Web sites, including sites based on Microsoft's Active Server Pages, Sun Microsystems' JavaServer Pages, and sites based on Citrix's own extensions to HTML (see "NFuse Programming Interface" on page 22 for more information).

ICA Client Device

In the context of NFuse, an *ICA Client device* is any computing appliance capable of executing an ICA Client and a Web browser. ICA Client devices include desktop PCs and network computers, among others.

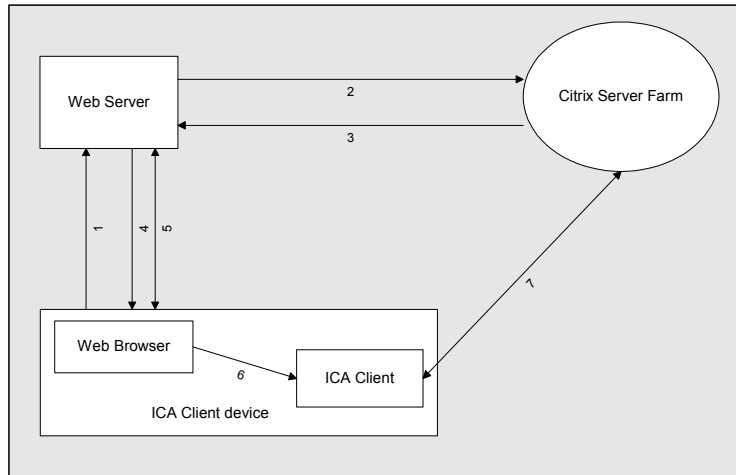
In an ICA Client device, the Web browser and ICA Client work together as a viewer and engine. The Web browser lets users view application sets (created by server-side scripting in an NFuse Web site) while the ICA Client acts as the engine that launches published applications.

NFuse is integrated with Web-based ICA Client installation. *Web-based ICA Client installation* is a Web browser-based method of deploying ICA Clients. The NFuse example Web sites and all sites produced by the Web Site Wizard include Web-based ICA Client installation HTML code. When a client device user visits an NFuse Web site, the Web-based ICA Client installation code detects the device and Web browser types and prompts the user to install an appropriate ICA Client. In the case of 16- and 32-bit Windows devices, Web-based ICA Client installation can also detect the presence or absence of an installed ICA Client and prompt the user only if necessary.

NFuse supports many Web browser/ICA Client combinations. For a complete list of browser/client combinations, see "ICA Client Device Requirements" on page 26.

How NFuse Works

The following diagram describes the interaction between the Citrix server farm, Web server, and ICA Client device:



1. An ICA Client device user visits a login page and enters user credentials. The Web browser sends an HTTP request containing the credentials to the Web server.
2. The Web server reads the user's information and uses the NFuse Java objects to forward that information to the Citrix XML Service on a designated Citrix server in the server farm. This designated server acts as a broker between the Web server and the Citrix server farm.
3. The Citrix XML Service on the designated server then retrieves from the farm a list of applications that the user can access. These applications comprise the user's *application set*. In MetaFrame XP and MetaFrame 1.8 server farms, the XML Service retrieves the application set from the Independent Management Architecture (IMA) system and Program Neighborhood Service, respectively. In a MetaFrame for UNIX Operating Systems farm, the Citrix XML Service on the designated Citrix server uses information gathered from the ICA Browser and the local NFuse configuration file to determine which applications the user can access.

The Citrix XML Service then forwards the user's application set information to the NFuse Java objects running on the Web server.

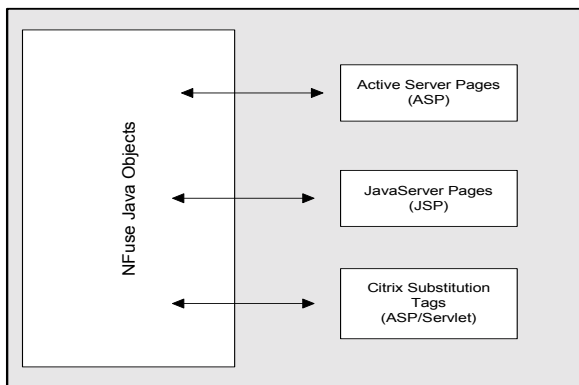
4. The Web server uses the NFuse Java objects to generate an HTML page containing links to the applications in the user's application set. Each hyperlink in the HTML page points to a template file stored on the Web server. This file serves as a template from which NFuse can dynamically generate ICA files. *ICA files* are text files containing parameters that configure ICA session properties such as the application to run in the session, the address of the server that will execute the application, and the properties of the window to display the application in. ICA files are written in .Ini file format and have an .Ica extension.
5. The user initiates the next step by clicking one of the hyperlinks in the HTML page. The Web browser sends a request to the Web server to retrieve an ICA file for the selected application.

The Web server passes this request to the NFuse Java objects, which retrieve the template ICA file. The template file contains substitution tags. The Java objects replace the substitution tags in the template ICA file with information specific to the user and desired application. The Java objects then send the customized ICA file to the Web browser.

6. The Web browser receives the ICA file and passes it to the client device's ICA Client.
7. The ICA Client receives the ICA file and initiates an ICA session with a Citrix server according to the ICA file's connection information.

NFuse Programming Interface

NFuse's Java objects provide functionality that you can access using NFuse's application programming interface. The programming interface lets you customize the Web sites created by the Web Site Wizard or build your own sites according to the requirements of your environment.



Web masters can use the following to access the NFuse programming interface:

- **Microsoft's Active Server Pages.** Using Active Server Pages, Web masters can write Web server scripts that implement NFuse Java object functionality. Such scripts call the Java objects to perform various functions and then place the results of those functions in plain HTML documents or ICA files.
- **Sun Microsystems' JavaServer Pages.** Like Active Server Pages, JavaServer Pages provide Web masters with a Web server scripting environment. JavaServer Pages support makes NFuse compatible with many Java Web servers.
- **Citrix substitution tags with support files.** Citrix substitution tags provide Web masters who are unfamiliar with Web server scripting a simplified method of accessing the NFuse Java objects. Substitution tags are proprietary HTML extensions that Web masters can write into plain HTML documents to create simple NFuse Web sites. Substitution-tag-based sites use a Java servlet or Active Server Page support files that you can create with the Web Site Wizard to perform necessary NFuse tasks.

In addition to the methods listed above, you can also write your own Java servlets using the NFuse Java objects.

NFuse Requirements

The following topics describe the NFuse requirements for each network component in an NFuse system.

Citrix Server Requirements

To participate in the NFuse system, your Citrix servers must meet the following requirements.

Supported MetaFrame Versions

NFuse requires one of the following Citrix platforms:

- MetaFrame XP Application Server for Windows Version 1.0
- MetaFrame Application Server for Windows Version 1.8
- Citrix MetaFrame for UNIX Operating Systems Version 1.1

NFuse operates with these MetaFrame versions on all of their supported platforms. For a list of supported platforms, see your MetaFrame documentation.

Additional Software Requirements

MetaFrame Application Server for Windows Version 1.8 servers must have:

- Citrix MetaFrame 1.8 Service Pack 2 installed on each server
- Citrix MetaFrame 1.8 Feature Release 1 license installed and activated on each server

Citrix MetaFrame for UNIX Operating Systems Version 1.1 servers must have:

- The Citrix XML Service for UNIX Operating Systems installed on at least one server in the farm. This primary server functions as the contact point between the NFuse Web server and the farm.
- Additional servers running the Citrix XML Service for UNIX Operating Systems are optional for primary server back up.

MetaFrame XP Application Server for Windows Version 1.0 servers have no additional software requirements.

General Configuration Requirements

MetaFrame for Windows servers must be members of a server farm. The servers in the farm must have applications published. Additionally, if your Citrix servers are running MetaFrame 1.8 for Windows, you must make sure they have applications published under the server farm management scope. For information about server farm membership and publishing applications in a server farm, see your *MetaFrame Administrator's Guide*.

MetaFrame 1.1 for UNIX Operating Systems servers also must have applications published. In addition, these applications must be configured for use with NFuse. See the *Citrix XML Service for UNIX Operating Systems Administrator's Guide* for information on installing the Citrix XML Service for UNIX and configuring published applications for use with NFuse.

Backward Compatibility

Compatibility issues depend upon the type of server farm in use. Server farms composed of MetaFrame XP 1.0 or MetaFrame 1.8 servers are backward compatible with:

- Web Server Extension 1.0
- Web Server Extension 1.5

Note When using Web Server Extension 1.0 with MetaFrame XP or MetaFrame 1.8 server farms, the components interoperate by using the down-level NFuse 1.0 XML protocol instead of the updated protocol included in NFuse 1.51 Web Server Extension. Use of the old protocol limits functionality to NFuse 1.0-level features and can create some performance overhead during protocol negotiation.

Server farms composed of MetaFrame for UNIX Operating Systems servers are backward compatible with Web Server Extension 1.5 only.

Feature Limitations

The following NFuse features are not available when using NFuse to access applications on MetaFrame for UNIX Operating Systems servers:

- Ticketing
- SSL communication between the Web server and server farm
- Filtering of applications based upon groups

See the Citrix download sites for updates to MetaFrame for UNIX Operating Systems.

Web Server Requirements

You can use NFuse on the following Web server/platform combinations:

- Microsoft Internet Information Server Version 4.0 running on Windows NT 4.0 Server
- Microsoft Internet Information Server Version 4.0 running on Windows NT 4.0, Terminal Server Edition
- Microsoft Internet Information Server Version 5.0 running on the Windows 2000 Server family
- Netscape Enterprise Server Version 3.6 on Solaris 7 and 8
- iPlanet Web Server 4.0 (formerly Netscape Enterprise Server 4.0) with Service Pack 4 on Solaris 7 and 8
- iPlanet Web Server 4.1 on Solaris 7 and 8
- Apache Server 1.3.9 on Redhat Linux 6.0 using Sun JDK 1.2.2, Apache JServ 1.1, and GNUJSP 1.0
- Apache Server 1.3.9 on Solaris 7 and 8 using Sun JDK 1.2.2, Apache JServ 1.1, and GNUJSP 1.0

Important Windows NT 4.0 (Server and Terminal Server Edition) ships with Microsoft IIS Version 3.0. Microsoft provides a free upgrade to Microsoft IIS 4.0 in its Windows NT Server 4.0 Option Pack.

Note also that during Microsoft IIS 4.0 installation, the setup program prompts you to install Internet Explorer Version 4 or 5. By default, when you install Internet Explorer Version 4, its setup program installs a Java Virtual Machine on your system. Internet Explorer Version 5 gives you the option to install the JVM instead of placing the JVM on your system by default. Make sure you install the JVM during Internet Explorer Version 5 setup. NFuse requires this JVM for execution of its Web Server Extension software.

When setup completes, make sure your system has the file Msjava.dll.

The above list contains all tested and supported Web server and platform combinations; however, you may be able to use NFuse on other Web servers that support Java servlets and/or JavaServer Pages.

In addition to the platform requirements above, NFuse installation on your Web server requires an ICA Client CD to populate the Web server's ICA Client file store. The Web server uses the ICA Client files for ICA Client installation and ICA session embedding. See "ICA Client Device Requirements" below for information on supported ICA Client CD versions.

ICA Client Device Requirements

To operate with NFuse, your ICA Client devices must have a supported ICA Client and a supported Web browser. With the exception of ICA Clients that cannot be executed with a browser (for example, the ICA DOS Clients), all ICA Clients that ship on the ICA Client CD are NFuse-compliant. The ICA Client CD is available in your MetaFrame XP media, Feature Release 1/Service Pack 2 media, Citrix MetaFrame for UNIX Operating Systems Version 1.1 media, or for free download from the Citrix download site.

Important The ICA Client CD shipping with the Solaris-only version of MetaFrame for UNIX Operating Systems 1.1 is not compatible with NFuse installation. Users of these systems must download the latest ICA Client CD from the Citrix download site before beginning NFuse deployment.

In addition to the ICA Clients on the ICA Client CD, some previously shipped ICA Clients are NFuse-compliant as well. The following table lists minimum ICA Client version levels.

ICA Client	Version	Supported browsers
Win32	4.21.779 and above	Netscape Navigator 4.01 and above Internet Explorer 4.0 and above
UNIX	3.0 and above	Netscape Navigator 4.01 and above Netscape Communicator 4.61 and above
Linux	3.0 and above	Netscape Navigator 4.01 and above Netscape Communicator 4.61 and above
ActiveX Control	4.21.779 and above	Netscape Navigator 4.01 and above Internet Explorer 4.0 and above
Netscape Plug-in	4.21.779 and above	Netscape Navigator 4.01 and above Internet Explorer 4.0 and above
Win16	4.21.779 and above	Netscape Navigator 4.08 and above Internet Explorer 4.01 and above
Java - Applet	4.11 and above	Netscape Navigator 4.01 and above Internet Explorer 4.0 and above
Java - Application	4.11 and above	Netscape Navigator 4.01 and above Internet Explorer 4.0 and above
Macintosh	4.10.23 and above	Netscape Navigator 4.01 and above Netscape Communicator 4.61 and above

The features and capabilities of each ICA Client differ. For information about supported ICA Client features, see the *Citrix ICA Client Administrator's Guide* for the ICA Client in question.

Overview of This Manual

This manual contains the following chapters:

Chapter 3: “Configuring Your Web Server” on page 29

Describes how to prepare your Web server to participate in an NFuse system. Topics covered include installing the Citrix Web Server Extension on your Web server and using the Citrix Web Site Wizard to create an NFuse-enabled Web site.

Chapter 4: “Configuring ICA Client Devices” on page 59

Describes steps required to prepare your ICA Client devices. Includes information on using Web-based ICA Client installation to deploy and install ICA Clients. Explains additional configuration required by some ICA Clients to work with NFuse.

Chapter 5: “Using NFuse Tags” on page 69

Lists Citrix substitution tags and explains their usage. This chapter includes a tutorial that describes how to make modifications to substitution-tag-based Web sites.

Chapter 6: “NFuse Java Object Reference” on page 103

Explains the NFuse application programming interface. This chapter lists NFuse’s objects and methods and gives examples of how to access them from Active Server Page and JavaServer Page-based Web documents.

Chapter 7: “ICA File Reference” on page 145

Explains the ICA file format and lists ICA file parameters you can add to the template ICA files used by NFuse.

Chapter 8: “Configuring NFuse Security” on page 163

Describes security considerations and lists measures you can take to secure your NFuse system.

Chapter 9: “Example Web Sites” on page 177

Describes implementation of example code included in the NFuse package. Example code provides advanced functionality such as server redundancy and performance enhancements.

What to Do Next

NFuse deployment begins with configuration of your Web server. For information, see Chapter 3, “Configuring Your Web Server” on page 29.

Configuring Your Web Server



This chapter explains how to configure your Web server to participate in an NFuse system. Web server configuration involves installing the Citrix Web Server Extension on your Web server and creating Web pages that users can visit to access their application sets.

Tasks to Complete

In this chapter Web server administrators will:

- Install the Citrix Web Server Extension on a Web server and configure the Web server if necessary
- Configure the Web Server Extension properties
- Use the Citrix Web Site Wizard to create Web pages that users can visit to access their application sets

Web Server Extension Installation

NFuse includes separate setup programs for installing the Web Server Extension on various Web servers. The topics that follow explain how to use these setup programs to install the Web Server Extension on:

- Microsoft Internet Information Server (IIS)
- Netscape Enterprise Server, iPlanet Web Server, and Apache Server

If you are upgrading a previous installation of the Web Server Extension, see “Installing the Web Server Extension Over a Previous Installation” on page 43 before proceeding.

Note that MetaFrame XP Setup offers administrators the option of installing NFuse during MetaFrame installation. This option installs the Web Server Extension on the server and places a single NFuse Web site in the Web document root directory. The Web site provides a fully-functional example of NFuse use and can be used as-is with no additional configuration. Unlike the Web Server Extension installation described in this chapter though, the NFuse installation offered in MetaFrame XP Setup does not include the full complement of NFuse example Web sites. If you want to use any of the functionality included in the example Web sites, you must install the Web Server Extension as described in this chapter.

Installing the Web Server Extension on Microsoft IIS

During Web Server Extension installation, you must identify a Citrix server in your farm that will act as a contact point between the server farm and your Web server. The name you specify can be a Windows NT server name, IP address, or fully-qualified DNS name. If your server farm is composed of MetaFrame for Windows servers, you can specify the name of any server in the farm. If your server farm is composed of MetaFrame for UNIX Operating Systems servers, you must specify the name of a server running the Citrix XML Service for UNIX Operating Systems.

In addition, you must specify the TCP/IP port on which the specified server is running the Citrix XML Service. If you do not know this port number, you can determine it by checking the MetaFrame server's port information.

➤ **To view the XML Service port assignment**

- On MetaFrame XP servers, open the Citrix Management Console. In the left pane, right-click the server and select **Properties**. In the **Properties** dialog box, select the **MetaFrame Settings** tab to view the port assignment.

Note that if during MetaFrame XP installation the administrator chose the option to share Internet Information Server's TCP/IP port with NFuse, the Citrix Management Console displays **Sharing with IIS** as the port in use. In this case, to determine the XML Service port you must locate the port used by Internet Information Server's WWW Service. By default the WWW Service uses port 80.

- On MetaFrame 1.8 servers, the port number is specified in the following registry key:

HKLM\SYSTEM\CurrentControlSet\Services\CtxHttp\TcpPort

- On MetaFrame for UNIX Operating Systems servers, type **ctxnfusesrv -l** at a command prompt to view port information.

Note If necessary, you can change the port in use on the Citrix server. For MetaFrame 1.8 servers, see the *Feature Release 1 and Service Pack 2 Installation Guide for Citrix MetaFrame for Windows Version 1.8*. For MetaFrame XP servers, see the installation chapter of the *MetaFrame XP Administrator's Guide*. For MetaFrame for UNIX servers, see the *Citrix XML Service for UNIX Operating Systems Administrator's Guide*.

Towards the end of Web Server Extension installation, the setup program prompts you to supply an ICA Client CD or CD image. Setup copies the contents of the CD's ICAWEB directory to a directory called NFuseClients that it creates off the Web server's Web publishing root. All example Web sites and sites created by the Web Site Wizard assume that the Web server contains the ICA Client files in this directory structure. If you do not want to copy the ICA Clients to the Web server during Web Server Extension installation, you can copy them to the server later. Make sure you create the required directory structure; for example, in an English installation: `<webroot>/NFuseClients/en/<icaclientversion>`

Important During installation or uninstallation of Web Server Extension on Microsoft IIS, the setup program stops and then restarts your Web server and all of its associated services. This stoppage causes a disruption of service to connected users for the duration of the installation.

➤ **To install the Web Server Extension on Microsoft IIS**

1. Make sure you are logged in as a user with administrator privileges.
2. If you are installing the Web Server Extension from an NFuse CD-ROM included in your MetaFrame XP or Feature Release 1/Service Pack 2 media, insert the CD-ROM in your Web server's CD-ROM drive. Locate the file named NFuseWebExt-IIS.exe. Double-click the file.

If you downloaded the Web Server Extension from a download site, copy the file NFuseWebExt-IIS.exe to your Web server. Double-click the file.
3. The installation wizard guides you through installation.
4. The Web Server Extension setup program installs example NFuse Web sites in a directory called Citrix\NFuse151 in your Web server's Web publishing root. When installation is complete, you can examine the example Web sites or create your own Web sites using the Citrix Web Site Wizard. To examine the example sites, point a Web browser at the Default.htm file in the Citrix\NFuse151 directory. For explanations of example sites, see "Example Web Sites" on page 177.

See "Introduction to the Citrix Web Site Wizard" on page 47 for information on using the Web Site Wizard to create NFuse Web sites.

See Chapter 8, "Configuring NFuse Security" on page 163 for information on securing your Web server.

Plugging the Citrix XML Service into IIS

This topic explains how to use Internet Information Server to perform some tasks associated with the Citrix XML Service. This procedure is not a required step in Web server configuration and is intended for those administrators who want to integrate the functionality of the Citrix XML Service with a Web server's HTTP service. Plugging the XML Service into IIS lets you use a single TCP/IP port for both Web server and NFuse traffic and eliminates the need for opening additional ports for NFuse in network firewalls.

Note The procedure described in this topic is a manual version of the automatic Internet Information Server extension offered by MetaFrame XP Setup. If you are using a MetaFrame XP server as your Web server and did not choose to extend IIS during MetaFrame XP installation, you can use this procedure to manually plug the XML Service into your Web server.

The Citrix XML Service contains an ISAPI extension that you can plug into Internet Information Server. The XML Service is composed of the following components:

- **Ctxmlss.exe.** This network component is a relay for NFuse data requests. The NFuse Java objects send application set requests to Ctxmlss.exe, which passes the requests to Wpnbr.dll (described below) for processing. Ctxmlss.exe then forwards the information returned by Wpnbr.dll to the Java objects that requested the data.
- **Wpnbr.dll.** This .Dll file communicates with the Program Neighborhood Service or Independent Management Architecture (IMA) systems in the farm to determine application set information for a user or user group.

Plugging the Citrix XML Service into IIS involves using IIS as the relay for NFuse data requests. In this configuration, IIS replaces Ctxmlss.exe as the network listener and Wpnbr.dll executes as an ISAPI extension to IIS.

The following procedure describes how to plug the Citrix XML Service into your IIS Web server.

➤ **To plug the Citrix XML Service into IIS**

1. Locate the following files on a Citrix server that has Service Pack 2 or MetaFrame XP installed: Wpnbr.dll and Ctxmlss.txt. The default installation location is %SystemRoot%\System32\.
- After copying the files, stop the Citrix XML Service on the server so that it is not available for NFuse requests. To stop the XML Service, use the Services applet in Control Panel to change the service's properties. In the Services applet, stop the Citrix XML Service. Then click **Startup** and in the **Startup Type** field, select **Disabled**.
2. Copy Wpnbr.dll and Ctxmlss.txt into the IIS scripts directory on your Web server. For example, the default scripts directory on IIS Version 5.0 is <RootDrive>:\inetpub\scripts\.
3. Use Internet Service Manager to give these two files Read and Write access.
4. Stop and restart the Web server.
5. Make sure the Web server that hosts the NFuse Java objects points to the server hosting the XML Service plug-in and that the port it sends requests to is the IIS port. If you have NFuse Web sites that use a non-default Citrix server/port for NFuse information, you must modify the Web sites themselves to contact the new server/port.

Installing the Web Server Extension on Netscape Enterprise Server, iPlanet Web Server, and Apache Server

The Setup program for installing the Web Server Extension on Netscape Enterprise Server, iPlanet Web Server, and Apache Server prompts you for locations in which to place various NFuse files. The following table lists these files by type. Use this table as a reference when installing the Web Server Extension and configuring your Web server.

File Type	Description	Directory
NFuse Java objects: nfuse.jar ctxxml4j.jar jsafeObf.jar sslplus3.1.7.jar	Java objects including the base NFuse Class files, IBM XML parser, and SSL/SOCKS provider Classes and cryptographic libraries.	You can copy these files to any directory.
Properties files: NFuse.properties NFuseErrorsResource.properties NFuseClientDetectStrings.properties	Text files containing NFuse configuration information, error message strings, and Web-based ICA Client installation strings, respectively.	The setup program copies these files to the directory you specify for the NFuse Java objects.
Web pages	Example NFuse Web sites.	Place these files in any directory from which your Web server can serve Web pages. The setup program defaults to the directory <code><webroot>/Citrix/NFuse151</code> .
Icon files	The setup program creates an icon cache directory that the NFuse Java objects use to store application icons (.Gif files).	Place this directory in any location from which your Web server can serve Web pages. The setup program defaults to the <code><webroot>/NFuseicons</code> directory.

File Type	Description	Directory
ICA Clients	Citrix ICA Client installation files used by NFuse Web sites to install ICA Clients on client devices.	The setup program prompts the administrator to supply an ICA Client CD or CD image and then copies the contents of the CD's ICAWEB directory to an NFuseClients directory off the Web server's Web publishing root. The example Web sites and any sites produced by the Web Site Wizard assume the ICA Client files are stored in this directory structure. If the ICA Client CD is not available, you can copy the contents of the CD's ICAWEB directory to your Web server after setup completes.

During Web Server Extension installation, you must identify a Citrix server in your farm that will act as a contact point between the server farm and your Web server. The name you specify can be a Windows NT server name, IP address, or fully-qualified DNS name. If your server farm is composed of MetaFrame for Windows servers, you can specify the name of any server in the farm. If your server farm is composed of MetaFrame for UNIX Operating Systems servers, you must specify the name of a server running the Citrix XML Service for UNIX Operating Systems.

In addition, you must specify the TCP/IP port on which the specified server is running the Citrix XML Service. If you do not know this port number, you can determine it by checking the MetaFrame server's port information.

➤ **To view the XML Service port assignment**

- On MetaFrame XP servers, open the Citrix Management Console. In the left pane, right-click the server and select **Properties**. In the **Properties** dialog box, select the **MetaFrame Settings** tab to view the port assignment.

Note that if during MetaFrame XP installation the administrator chose the option to share Internet Information Server's TCP/IP port with NFuse, the Citrix Management Console displays **Sharing with IIS** as the port in use. In this case, to determine the XML Service port you must locate the port used by Internet Information Server's WWW Service. By default the WWW Service uses port 80.

- On MetaFrame 1.8 servers, the port number is specified in the following registry key:
HKLM\SYSTEM\CurrentControlSet\Services\CtxHttp\TcpPort
- On MetaFrame for UNIX Operating Systems servers, type **ctxnfusesrv -l** at a command prompt to view port information.

Note If necessary, you can change the port in use on the Citrix server. For MetaFrame 1.8 servers, see the *Feature Release 1 and Service Pack 2 Installation Guide for Citrix MetaFrame for Windows Version 1.8*. For MetaFrame XP servers, see the installation chapter of the *MetaFrame XP Administrator's Guide*. For MetaFrame for UNIX servers, see the *Citrix XML Service for UNIX Operating Systems Administrator's Guide*.

Web Server Extension installation prompts you to enter a virtual URL for servlets. The value you enter must match the virtual URL you specify later when configuring your Web server to run NFuse. See “Configuring Netscape Enterprise Server and iPlanet Web Server” on page 37 and “Configuring Apache Server” on page 40 for more information on NFuse’s use of virtual URLs for servlets.

Towards the end of Web Server Extension installation, the setup program prompts you to supply an ICA Client CD or CD image. Setup copies the contents of the CD’s ICASWEB directory to a directory called NFuseClients that it creates off the Web server’s Web publishing root. All example Web sites and sites created by the Web Site Wizard assume that the Web server contains the ICA Client files in this directory structure. If you do not want to copy the ICA Clients to the Web server during Web Server Extension installation, you can copy them to the server later. Make sure you create the required directory structure; for example, in an English installation: `<webroot>/NFuseClients/en/<icaclientversion>`

➤ **To install the Web Server Extension on Netscape Enterprise Server, iPlanet Web Server, and Apache Server**

1. Log on as root at the server on which you want to install the Web Server Extension.
2. Copy the Web Server Extension file for UNIX (NFuseWebExt-UNIX.tar.gz) from your NFuse CD-ROM or the Citrix download site to an install directory on your Web server.
3. Unzip the NFuseWebExt-UNIX.tar.gz file. Unzipping the file produces NFuseWebExt-UNIX.tar, an archive containing the setup files for NFuse.
4. To extract the archived files from NFuseWebExt-UNIX.tar into the install directory, type **tar xvf NFuseWebExt-UNIX.tar** and press ENTER.
5. Stop your Web server.
6. Type **./setupNFuse** to begin the installation.

7. Follow the instructions on the screen to install the NFuse files in the appropriate directories. See the table at the beginning of this section for a detailed description of NFuse's files and directories in which you must place them.
8. Restart your Web server.

After you complete the installation, you must configure your Web server. For information about configuring Netscape Enterprise Server and iPlanet Web Server, see “Configuring Netscape Enterprise Server and iPlanet Web Server” below. For information on configuring Apache Server, see “Configuring Apache Server” on page 40.

See Chapter 8, “Configuring NFuse Security” on page 163 for information on securing your Web server.

Configuring Netscape Enterprise Server and iPlanet Web Server

The following procedures explain how to configure Netscape Enterprise Server and iPlanet Web Server for NFuse.

➤ To configure Netscape Enterprise Server 3.6 for NFuse

Note The following procedure assumes you are using the Bourne Shell.

1. To enable servlets, make sure the Web server's Java interpreter is activated and that the Java servlets directory points to the correct location for servlets. Note that Netscape Enterprise Server 3.6 does not support JavaServer Pages and cannot be used to serve NFuse Web sites based on the JavaServer Pages model.
2. Stop the Web server from the command line.
3. At the command prompt, type:
CLASSPATH=<full path to the file nfuse.jar : full path to the file ctxxml4j.jar : full path to the file jsafeObf.jar : full path to the file sslplus3.1.7.jar : full path to the parent directory for NFuse.properties and NFuseErrorsResource.properties>:\$CLASSPATH
4. Type **export CLASSPATH** to set the classpath as a global variable for the session.
5. Restart the Web server from the command line.

Important You **must** stop and start the service from the command line. Otherwise, the classpath information you entered will be lost.

➤ **To configure iPlanet Web Server 4.0 (formerly Netscape Enterprise Server 4.0) for NFuse**

1. Log in to Adminserv. Select the server you want to configure and click **Manage**.
2. On the **Servlets** page, enable the Servlet Engine and JSP if they are not enabled already.
3. In the left pane, click **Configure JRE/JDK Paths** and select **JDK**. In the **Path** field, make sure that the path to the JDK is correct.
4. In the **Classpath** field, add the following:
 - The full path to the file nfuse.jar.
 - The full path to the file ctxxml4j.jar.
 - The full path to the file jsafeObf.jar.
 - The full path to the file sslplus3.1.7.jar.
 - The full path to the directory containing the NFuse.properties and NFuseErrorsResource.properties files.
5. To use the HTML for Servlets model on iPlanet Web Server 4.0, you must configure the Web server to run the NFuse servlets:
com.citrix.nfuse.PNServletTemplate and com.citrix.nfuse.PNClientDetection.

In the left pane, click **Configure Servlet Attributes**. In the **Servlet Name** field, type a name for the NFuse servlet com.citrix.nfuse.PNServletTemplate. The name can be any you choose; for example NFuseTemplate.

In the **Servlet Code** field, type **com.citrix.nfuse.PNServletTemplate**. Click **OK** and save your changes when prompted.

Repeat this step for the com.citrix.nfuse.PNClientDetection servlet, entering a name for the servlet (for example NFuseClientDetection) and com.citrix.nfuse.PNClientDetection as the servlet code.
6. You must now configure virtual path translations for both NFuse servlets. In the left pane, click **Configure Servlet Virtual Path Translation**. In the **Virtual Path** field, enter the virtual path to servlets specified during Web Server Extension installation followed by
com.citrix.nfuse.PNServletTemplate. For example, if you specified /servlet during Web Server Extension installation, enter
/servlet/com.citrix.nfuse.PNServletTemplate

In the **Servlet** field, type the servlet name you specified for the com.citrix.nfuse.PNServletTemplate servlet in Step 5 (for example, NFuseTemplate). Click **OK** and save your changes when prompted.

Repeat this step for the `com.citrix.nfuse.PNClientDetection` servlet. In the **Virtual Path** field, enter the virtual path to servlets specified during Web Server Extension installation followed by `com.citrix.nfuse.PNClientDetection`. For example, if you specified `/servlet` during Web Server Extension installation, enter `/servlet/com.citrix.nfuse.PNClientDetection`

In the **Servlet** field, type the servlet name you specified for the `com.citrix.nfuse.PNClientDetection` servlet in Step 5 (for example, `NFuseClientDetection`). Click **OK** and save your changes when prompted.

7. Restart the Web server.

➤ **To configure iPlanet Web Server 4.1 for NFuse**

1. Log in to Adminserv. Click the **Global Settings** tab. In the left pane, click **Configure JRE/JDK Paths** and select **JDK**. In the **JDK Path** field, make sure that the path to the JDK is correct.
2. Click the **Servers** tab. Make sure your server is selected and click **Manage**.
3. Click the **Servlets** tab. Enable the Servlet Engine and JSP if they are not enabled already.
4. In the left pane, click **Configure JVM Attributes**. In the **Classpath** field, add the following:

The full path to the file `nfuse.jar`.

The full path to the file `ctxxml4j.jar`.

The full path to the file `jsafeObf.jar`.

The full path to the file `sslplus3.1.7.jar`.

The full path to the directory containing the `NFuse.properties` and `NFuseErrorsResource.properties` files.

5. To use the HTML for Servlets model on iPlanet Web Server 4.1, you must configure the Web server to run the NFuse servlets:
`com.citrix.nfuse.PNServletTemplate` and `com.citrix.nfuse.PNClientDetection`.

In the left pane, click **Configure Servlet Attributes**. In the **Servlet Name** field, type a name for the NFuse servlet `com.citrix.nfuse.PNServletTemplate`. The name can be any you choose; for example `NFuseTemplate`.

In the **Servlet Code** field, type `com.citrix.nfuse.PNServletTemplate`. Click **OK** and save your changes when prompted.

Repeat this step for the `com.citrix.nfuse.PNClientDetection` servlet, entering a name for the servlet (for example `NFuseClientDetection`) and `com.citrix.nfuse.PNClientDetection` as the servlet code.

6. You must now configure virtual path translations for both NFuse servlets. In the left pane, click **Configure Servlet Virtual Path Translation**. In the **Virtual Path** field, enter the virtual path to servlets specified during Web Server Extension installation followed by `com.citrix.nfuse.PNServletTemplate`. For example, if you specified `/servlet` during Web Server Extension installation, enter `/servlet/com.citrix.nfuse.PNServletTemplate`

In the **Servlet Name** field, type the servlet name you specified for the `com.citrix.nfuse.PNServletTemplate` servlet in Step 5 (for example, `NFuseTemplate`). Click **OK** and save your changes when prompted.

Repeat this step for the `com.citrix.nfuse.PNClientDetection` servlet. In the **Virtual Path** field, enter the virtual path to servlets specified during Web Server Extension installation followed by `com.citrix.nfuse.PNClientDetection`. For example, if you specified `/servlet` during Web Server Extension installation, enter `/servlet/com.citrix.nfuse.PNClientDetection`

In the **Servlet Name** field, type the servlet name you specified for the `com.citrix.nfuse.PNClientDetection` servlet in Step 5 (for example, `NFuseClientDetection`). Click **OK** and save your changes when prompted.

7. Restart the Web server.

Configuring Apache Server

The following instructions explain how to set up NFuse on an Apache Server running JServ and GNUJSP. These instructions assume that you have already installed Apache, JServ, and GNUJSP and have verified that basic “Hello World” examples for both Java Servlets and JavaServer Pages are working. If you want to use Java servlets only and do not intend to deploy any NFuse JavaServer Pages-based sites, you can ignore the instructions regarding GNUJSP.

Note The following instructions use a single servlet zone for NFuse Web sites. NFuse may not work properly if invoked from more than one servlet zone. It is important that all your NFuse pages, whether JavaServer Page or HTML for Servlets-based, use the same zone.

➤ **To configure Apache Server**

1. Open the configuration file `http.conf` located in your Apache installation's configuration directory; for example, `/usr/local/apache/conf/httpd.conf`
2. If you have installed and tested JServ, the following lines will appear somewhere in `httpd.conf`:

```
<IfModule mod_jserv.c>
.
.
.
</IfModule>
```

These lines configure Apache to send certain HTTP requests to the JServ servlet engine. Add or modify the following lines:

```
<IfModule mod_jserv.c>
.
.
.
    ApJServMount /servleturl /NFuseJServZone
    ApJServMount /jspurl /NFuseJServZone
    ApJServAction .jsp /jspurl/gnujsp
.
.
.
</IfModule>
```

where:

/servleturl is a virtual URL for which Apache will redirect HTTP requests to the JServ servlet engine when serving NFuse HTML for Servlets pages. This virtual URL must be the same virtual URL as entered during NFuse setup.

/NfuseJServZone is the name of the JServ zone for NFuse. For information on JServ zones, please refer to the JServ documentation. The name for this zone appears in the JServ configuration file described later.

/jspurl is the virtual URL for which Apache will redirect HTTP requests to the JServ servlet engine when serving NFuse .Jsp pages. Apache uses this virtual URL internally; the name you specify can be any name that doesn't conflict with other URLs.

/gnujsp is the name of the GNUJSP servlet alias as it appears in the GNUJSP zone configuration file described later. For default GNUJSP installations, this alias is `"/gnujsp"`.

Note The names specified above are placeholders. You do not have to specify the same names for your deployment.

3. Now you must modify the JServ configuration files. Open the master JServ configuration file. This file is usually named `jserv.properties` and is often located in the `conf` subdirectory of the JServ installation; for example, `/usr/local/jserv/conf/jserv.properties`
4. If it is not already there, add your NFuse JServ zone to the list of zones. The file should contain a line such as:
`zones=otherzone1, otherzone2`
Modify such a line so that it reads:
`zones=otherzone1, otherzone2, NfuseJServZone`
where *NfuseJServZone* is the name of the NFuse JServ zone specified in Step 2 above.
5. Now you must specify the NFuse JServ zone configuration file in the master JServ configuration file. Somewhere in the master JServ configuration file, add the following line:
`NfuseJServZone.properties=path-to-JServ-conf-directory/
NfuseJServZone.properties`
where *NfuseJServZone* is the name of the NFuse JServ zone specified in Step 2 above and *path-to-JServ-conf-directory* is the directory that contains `jserv.properties`
6. Now create the NFuse JServ zone configuration file. Create this file in the directory specified in Step 5 as *path-to-Jserv-conf-directory/
NfuseJServZone.properties*.
If you are *not* using GNUJSP, you can create this file by copying the default JServ zone configuration file called `zone.properties` from your JServ distribution.
If you are using GNUJSP, follow the installation instructions in your GNUJSP distribution to create your NFuse JServ configuration file. Make sure the name of the alias for the GNUJSP servlet is the same as specified in Step 2 above.

7. Add the NFuse jar files (nfuse.jar, ctxxml4j.jar, jsafeObf.jar, and sslplus3.1.7.jar) to the list of repositories in your NFuse JServ zone configuration file.

For example, if when you installed NFuse you chose to install the NFuse objects in /usr/local/jserv/NFuse, you would add or modify the following lines in your NFuse JServ zone configuration file:

```
repositories=/usr/local/jserv/NFuse
repositories=/usr/local/jserv/NFuse/nfuse.jar
repositories=/usr/local/jserv/NFuse/ctxxml4j.jar
repositories=/usr/local/jserv/NFuse/jsafeObf.jar
repositories=/usr/local/jserv/NFuse/sslplus3.1.7.jar
```

8. Stop and restart both Apache and JServ.

Installing the Web Server Extension Over a Previous Installation

This release of the Web Server Extension supports the following upgrade paths:

- Web Server Extension 1.0 or 1.5 to Web Server Extension 1.51 on all supported Web servers. When upgrading a previous installation, Web Server Extension Setup replaces the NFuse Java objects, NFuse.dtd, and the NFuse.properties, NFuseErrorsResource.properties, and NFuseClientDetectStrings.properties files. When replacing the properties and error resource files, Setup creates backup copies of existing files so that you can migrate any customizations into your new installation. When placing example Web sites in your Web server's Web publishing root directory, Setup creates a new Citrix/NFuse151 directory for the files and does not remove or overwrite existing NFuse Web sites.
- Web Server Extension 1.51 (MetaFrame XP) to full Web Server Extension 1.51. If you installed the Web Server Extension during MetaFrame XP installation, you can upgrade the installation to the full version of the Web Server Extension. Upgrading the installation adds full NFuse example Web site support to your installation. This upgrade path performs the same Java object and properties file replacement as described above and creates a new Citrix/NFuse151 directory for the files and does not remove or overwrite existing NFuse Web sites. Note, though, that if you do upgrade your installation and then decide to uninstall the full Web Server Extension, your previous installation of NFuse will not be functional.

Configuring Web Server Extension Properties

The Web Server Extension includes a configuration file that lets you change several of NFuse's properties. This file, called NFuse.properties, is located by default in the same directory as the NFuse Java objects (nfuse.jar, ctxml4j.jar, jsafeObj.jar, and sslplus3.1.7.jar).

The settings in NFuse.properties are global: all Web pages generated by the Web Server Extension draw from the file's values. Changes made to NFuse.properties affect all Web pages served by the Web Server Extension.

If necessary, you can override this file's values on a per-page basis in your Web server scripts; for example, if you want to use multiple Citrix servers acting as communication links to the Web server.

Important For changes made to NFuse.properties to take effect, you must stop and restart your Web server. For Microsoft Internet Information Server, use Control Panel to stop and restart IIS Admin Service and all of its dependent services. Restarting IIS Admin Service does not restart the dependent services; you must restart the dependent services manually.

A sample NFuse.properties file is included below. This copy includes values placed in the file during installation on a Microsoft IIS Web server.

```
ConfigurationFileEncoding=8859_1
ConfigurationFilePath=C:\\WINNT\\Java\\TrustLib
NFuse_ContentType=text/html
SessionFieldLocations=Script,Template,Url,Post,Cookie,Properties
Timeout=60
Version=1.51
SessionField.NFuse_CitrixServer=someServerName
SessionField.NFuse_CitrixServerPort=somePortNumber
SessionField.NFuse_IconCache=/NFuseIcons/
SessionField.NFuse_TemplatesDir=C:\\InetPub\\wwwroot\\Citrix\\NFuse151
URLMapping./=C:\\InetPub\\wwwroot
HttpInputEncoding=8859_1
HttpOutputEncoding=8859_1
TemplateFileEncoding=8859_1
CacheExpireTime=3600
SessionField.NFuse_TicketTimeToLive=200
SslKeystore=C:\\WTSRV\\keystore\\cacerts
DTDDirectory=C:\\WINNT\\system32
```

NFuse.properties can contain the following fields:

- **ConfigurationFileEncoding=8859_1.** Specifies the encoding used for NFuse.properties. For example, specify “8859_1” for basic Latin or “SJIS” for Shift-JIS Japanese on Windows Web servers (“ISO8859_1” and “SJIS,” respectively, on UNIX Web servers).
- **ConfigurationFilePath=C:\\WINNT\\Java\\TrustLib.** Specifies the directory in which NFuse.properties is saved. If you move NFuse.properties, you must modify this value.
- **SessionField.NFuse_ContentType.** Sets the MIME type of pages produced by the NFuse Java objects to the specified value. The default value is text/html. You must override this value in any pages that you use to produce ICA files. For an example of this override, see the file Template.ica produced when you use the Web Site Wizard to create a site.
- **SessionFieldLocations.** Specifies the valid locations for setting session fields. The default value is Script, Template, Url, Post, Cookie, Properties.
 - **Script.** Session field set in a Web page by a TemplateParser’s setSessionField() method.
 - **Template.** Session field set using the [NFuse_SetSessionField] session field in a template file.
 - **URL.** Session field set using the Get method in an HTML form.
 - **Post.** Session field set using the Post method in an HTML form.
 - **Cookie.** Session field set in a cookie.
 - **Properties.** Session field set in NFuse.properties; for example, NFuse_ContentType=text/html is set in the first line of the example NFuse.properties file above.
- **Timeout.** Specifies a communication timeout value, in seconds. Once the Java objects establish communication with a Citrix server, each subsequent Java object query of the Citrix server is subject to the specified timeout value. If the server does not respond to a Java object request within the allotted time, the operation times out.
- **Version.** Do not edit this field.
- **SessionField.NFuse_CitrixServer.** Specifies the name of a Citrix server in the farm. This server is the communication link between the server farm and the Web server. The default value is the server name entered during Web Server Extension installation. The server name can be a Windows NT server name, IP address, or fully-qualified DNS name.

- **SessionField.NFuse_CitrixServerPort.** Specifies the TCP/IP port used by the Citrix server specified in NFuse_CitrixServer for NFuse communication. The default value is the port number entered during Web Server Extension installation. This port number must match the port number used by the Citrix server.
- **SessionField.NFuse_IconCache.** Specifies the directory used to store NFuse-generated application icon files (.Gif). The default value is /NFuseIcons/ on Internet Information Server and /NFuseicons/ on UNIX Web servers. On Internet Information Server, the Internet guest account must have Read, Write, List, and Delete access to this directory. On UNIX Web servers, the files must be World readable and the directory must be World readable and executable.
- **SessionField.NFuse_TemplatesDir.** Specifies the directory where a TemplateParser object looks when a template file is specified with the NFuse_Template session field.
- **URLMapping./.** Specifies the path to your Web server's Web publishing root directory; for example, in many Microsoft Internet Information Server systems, the URLMapping./ entry specifies C:\inetpub\WWWRoot.
- **HTTPInputEncoding.** Specifies the encoding used for incoming HTTP such as form data. For example, specify "8859_1" for basic Latin or "SJIS" for Shift-JIS Japanese on Windows Web servers ("ISO8859_1" and "SJIS," respectively, on UNIX Web servers).
- **HTTPOutputEncoding.** Specifies the encoding used for outgoing HTTP such as HTML pages that display application sets. For example, specify "8859_1" for basic Latin or "SJIS" for Shift-JIS Japanese on Windows Web servers ("ISO8859_1" and "SJIS," respectively, on UNIX Web servers).
- **TemplateFileEncoding.** Specifies the encoding used for Citrix HTML template files. For example, specify "8859_1" for basic Latin or "SJIS" for Shift-JIS Japanese on Windows Web servers.
- **CacheExpireTime.** Specifies the default expiration timeout value in seconds for the AppDataList objects stored in the AppListCache object. Used for caching of application set information on the Web server. See the descriptions of example NFuse Web sites in "Improving NFuse Performance" on page 179 for more information on application caching.
- **SessionField.NFuse_TicketTimeToLive.** Specifies the amount of time in seconds an authentication ticket is valid for. A ticket that is older than the specified duration cannot successfully authenticate a user to the Citrix server farm.
- **SSLKeystore.** Specifies the directory containing the certificate authority root certificates. NFuse uses root certificates when authenticating a Citrix SSL Relay server.

- **DTDDirectory.** Specifies the directory containing the NFuse DTD file. If this parameter is unspecified, NFuse defaults to the Java Virtual Machine's working directory.
- **PooledSockets.** Specifies whether or not to use socket pooling, and if yes, specifies the minimum number of sockets to keep in the pool. Socket pooling allows the Web server to keep a socket open for communication with the Citrix server. Keeping a socket available eliminates the need to open a new TCP/IP connection for each NFuse information request.

Set to Off to disable socket pooling. Otherwise, set to some integer to specify a minimum number of sockets to keep open. When a minimum is specified, NFuse opens the minimum value at first user connection and keeps at least the minimum number of sockets available at all times.

If this parameter is not specified, NFuse defaults to enabled socket pooling with no minimum.

Introduction to the Citrix Web Site Wizard

The Web Server Extension setup places example Web sites in an NFuse subdirectory of your Web server's Web publishing root directory. The Web sites are a collection of Web pages and .Gif files that work with the Web Server Extension to produce the NFuse front-end to published applications. These example Web sites demonstrate NFuse capabilities and can often be used as is.

To fit NFuse into your own Web deployment scheme, you can use the Citrix Web Site Wizard to create a customized Web site.

Note It is important to note that all Web sites created using the Web Site Wizard require that the Web browsers on client devices support HTML frames.

Installing the Citrix Web Site Wizard

The Web Site Wizard is a 32-bit Windows application that you can install on any 32-bit Windows machine on your network.

➤ To install the Citrix Web Site Wizard

1. If you are installing the Web Site Wizard from an NFuse CD-ROM included in your MetaFrame XP or Feature Release 1/Service Pack 2 media, insert the CD-ROM in your machine's CD-ROM drive. Locate the file named NFuseWizard.exe. Double-click the file.

If you downloaded the Web Site Wizard installation program from the Citrix download site, copy the file to your machine. Double-click the file.

2. The installation wizard guides you through installation.

Using the Citrix Web Site Wizard

To start the wizard, click **Start**, and then select **Programs|Citrix|NFuse|Web Site Wizard** to begin using the wizard.

The Web Site Wizard presents you with a series of options for creating an NFuse-enabled Web site. The options you choose determine:

- How you access the NFuse Java objects on the Web server
- How users view applications
- How users authenticate themselves
- How the Citrix server and the Web server communicate
- How your Web pages appear

The following topics describe Web Site Wizard concepts and follow the order of appearance of these concepts in the wizard's screens.

Overriding the Default Citrix Server and Configuring SSL Support

When you install the Web Server Extension on your Web server, you must specify the name of a Citrix server in your server farm. The Citrix XML Service on this server acts as the communication link between the server farm and Web server. The Citrix Web Site Wizard allows you to override this default server so that individual NFuse Web sites can access various Citrix servers running the XML Service.

If you choose to override the default server, the wizard inserts code in your pages that overrides the default server specified in the `NFuse.properties` file on the Web server. This override applies to the current set of pages only. All other pages that do not themselves specify a server override use the default server specified in `NFuse.properties`.

If you choose to override the default Citrix server, specify a Windows NT server name, IP address, or fully-qualified DNS name for the non-default server. You must also supply a port number on which the server is running the Citrix XML Service. By default, the Citrix XML Service port is 80. See "To view the XML Service port assignment" on page 31 for information on determining the port used by the XML Service.

For information about the scripting code used to set which server the Web pages contact, see the description of the NFuse Java object called `CitrixWireGateway` on page 104.

Enable Secure Sockets Layer (SSL) support if you want to secure communication between the Web server and Citrix server farm. Communication between these components includes the passing of user credentials and application set information. Enabling SSL support provides authentication of the Citrix server, encryption of the data stream, and message integrity checks.

See “Configuring NFuse to Use the Citrix SSL Relay” on page 171 for information on configuring the SSL Relay.

Note Support for the Citrix SSL Relay is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems.

To enable SSL, you must specify the name of a Citrix server running the Citrix SSL Relay and the port on which the SSL Relay is listening for SSL traffic. The name you specify can be a Windows NT server name, IP address, or fully-qualified DNS name. Make sure the naming format you specify is consistent with the name specified in your SSL Relay server’s certificate; for example, specify a DNS name if the certificate contains a DNS name. By default, the Citrix SSL Relay runs on TCP/IP port 443.

When specifying an SSL Relay server, note that you must also consider the Citrix server you want to handle the NFuse requests. If you override the default Citrix server in this wizard screen, the resulting Web pages direct NFuse traffic to the non-default server via the SSL Relay server. If you don’t override the default Citrix server in this wizard screen, the NFuse Web pages direct NFuse traffic to the default Citrix server via the SSL Relay server (the default Citrix server is specified in the NFuse.properties file).

Important By default, the SSL Relay forwards traffic only to the server on which it is installed. However, you can configure the SSL Relay to forward traffic to other servers. If the SSL Relay in your deployment is on a machine other than the machine to which you want to send NFuse data, make sure the SSL Relay’s server list contains the server to which you want to forward NFuse data.

Choosing a Scheme

You can choose a predefined scheme that determines the look and feel of your Web pages. Choose the scheme that best matches your requirements. All schemes can be modified or used as is.

Choosing a Layout Model

The Web Site Wizard can create pages using one of two layout models: substitution-tag-based or scripting-based. These models are described below.

Substitution-Tag-Based Layout

There are two types of substitution-tag-based layout models, *HTML for IIS* and *HTML for Servlets*. Both types use substitution tags, which are Citrix proprietary HTML extensions that allow your Web pages to dynamically retrieve information from the Citrix server farm.

Choose a substitution-tag-based layout model if:

- You want to modify your Web pages but are not familiar with scripting languages or writing servlets
- You want to quickly create very basic pages

HTML for IIS

This substitution-tag-based layout allows your Web pages to have many of the features of server-side scripting without requiring the Web master to know a scripting language. Like fully scripted pages, these pages allow Web masters to authenticate users and retrieve application lists by accessing the NFuse Java objects on the Web server. The Java objects translate the substitution tags into standard HTML before sending the pages to the user. For a list of the substitution tags and an explanation of how the tags work, see Chapter 5, “Using NFuse Tags” on page 69. For a list of the files created by the Web Site Wizard for a substitution-tag-based site, see “Files Included in a Substitution-Tag-Based Web Site” on page 87.

Choose this layout model if:

- You are using Microsoft IIS as your Web server
- You want to quickly create very basic pages
- You want to modify your Web pages but are not familiar with scripting languages

Important Although the HTML for IIS layout does not require the Web master to write any server-side scripts, this layout does use Active Server Pages server-side scripting in support files to perform tag substitution.

HTML for Servlets

This substitution-tag-based layout model uses substitution tags to access the servlets installed with NFuse. These servlets retrieve application information and login credentials from the MetaFrame server. This information is then passed to the Web server, which translates the information into standard HTML.

Choose this layout model if:

- You are using Netscape Enterprise Server, iPlanet Web Server, or Apache Server as your Web server
- You want to quickly create very basic pages
- You want to modify your Web pages but are not familiar with scripting languages

Specifying Servlet and Web Page Directories

If you choose the HTML for Servlets layout model, the next two wizard screens contain fields where you must specify:

- The URL for the virtual directory where the NFuse servlets are located. This directory corresponds to the directory where you saved `nfuse.jar` during Web Server Extension installation.
- The URL and the path to the directory on the Web server where you will publish your NFuse Web pages.

Scripting-Based Layout

There are two scripting-based layout models: *Active Server Pages* and *JavaServer Pages*. If you use one of these models and want to modify the resulting pages, you must be familiar with a scripting language such as VBScript or JavaScript.

Active Server Pages

An Active Server Page is an HTML page that includes server-side scripting that the Web server processes before sending the page to a user. The scripting code accesses the NFuse Java objects that run on the Web server.

Choose this layout model if:

- You are using Microsoft IIS as your Web server
- You want to modify your Web pages and you are familiar with a scripting language such as VBScript
- You want to create pages capable of performing complex tasks

JavaServer Pages

A JavaServer Page uses Java code embedded in the Web page to access the NFuse Java objects that run on the Web server.

Choose this layout model if:

- You are using Netscape Enterprise Server, iPlanet Web Server, or Apache Server
- You want to modify your Web pages and you are familiar with a scripting language such as JavaScript
- You want to create pages capable of performing complex tasks

Launching and Embedding

NFuse supports:

- Launching applications in a separate window
- Embedding applications in a Web page

Determining Launching and Embedding Capabilities

Before choosing to launch or embed your applications, make sure that the ICA Clients deployed on your network support the method you choose.

You can use the following ICA Clients to launch applications:

- Win32
- Win16
- UNIX (including Linux)
- Macintosh
- Java (run in application mode)

You can use the following ICA Clients to embed applications:

- ActiveX Control for Microsoft Internet Explorer on 32-bit Windows systems
- Plug-in for Netscape on 32- and 16-bit Windows systems and Internet Explorer on 16-bit Windows systems
- Java applet run in applet mode on any client device or browser with a Java Virtual Machine

Note The 6.00 and later versions of the ICA Win32 and ICA Win16 Clients include the ActiveX Control and Netscape Plug-in by default. Client devices with these ICA Clients installed are fully capable of embedding ICA sessions.

Launching Applications

When a user clicks a link for a launched application, the application runs in a separate window on the local desktop, independent of the browser. Create Web pages that launch applications if:

- You want the applications to appear in a separate window
- Your users' ICA Clients do not support application embedding
- You want users to be able to browse to different Web pages without disconnecting from their applications

Web sites created by the Web Site Wizard that use ICA session launching include a hidden frame. This frame displays the standard ICA Session Launch message displayed by the Web browser during session launching. This frame also displays any error messages generated at session initialization. To view these error messages, right-click a malfunctioning link and view the link in a new window or save the target as a file on the client device's hard disk. Use a text editor to open the file and view the error message.

Embedding Applications

An embedded application runs within the Web page. Create a Web page that embeds an application if:

- You want the applications to appear in the browser window
- You require precise placement of the application in relation to the surrounding content on the Web page

When creating a Web site that embeds ICA sessions, the embedding tags written by the wizard into the pages reference ICA Client files. In the event that a client device without an installed ICA Client accesses the embedding page, the page can inform the device where to obtain an ICA Client.

In the case of embedding pages that use the ActiveX Control, the embedding tags reference the ICA Client installation files copied to the Web server during Web Server Extension installation. By default, the embedding page references the installation archive Ica32t.exe (in the CODEBASE variable in the page Appembed.x, where *x* is either jsp, asp, or htm). The file Ica32t.exe installs the ICA Win32 Web Client. You can modify this variable to reference the ICA Win32 Program Neighborhood Client installation file Ica32.exe. Both Ica32t.exe and Ica32.exe install the ActiveX Control.

For pages that use the Netscape Plug-in, the embedding tags refer the user to the Citrix download site. You can modify this behavior by editing the PLUGINSOURCE variable in the page Appembed.x, where *x* is either jsp, asp, or htm; for example, to redirect the user to an internal Web server containing your own download page and an ICA Client installation file. As described above for

the ActiveX Control, you can use either Ica32t.exe or Ica32.exe to install the Plug-in on 32-bit Windows systems. Use Ica16.exe for 16-bit Windows systems.

Note For more information on Ica32t.exe and Ica32.exe, see “ICA Win32 Client Installation Archives” on page 61.

Web sites that use the ICA Java Client to embed sessions must download the ICA Client archive file each time the user accesses the embedding page.

If you did not copy the ICA Client installation files to your Web server during Web Server Extension installation, see “ICA Client Installation Files” on page 60 for information on copying the files to your Web server. If you do not copy the ICA Client files to the Web server, application embedding may not function properly. Additionally, when a new version of an ICA Client becomes available, you must replace the ICA Clients in your client files directory.

For more information about the ActiveX Control, the Plug-in, or the Java applet, see the *Citrix ICA Client Administrator's Guides* for the ICA Win32 Client and the ICA Java Client.

Ticketing

Ticketing provides authentication security by eliminating user credentials from the ICA files sent from the Web server to client devices. Tickets have a configurable expiration period and are valid for a single ICA session. After use, or after expiration, the ticket is invalid and cannot be used to access applications.

Note Support for ticketing is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems. Disable ticketing for these deployments.

Ticketing uses Citrix servers to store credentials entered by users in NFuse login forms. When a user selects an application from the Web page, the NFuse Java objects on the Web server request from the farm a ticket for that user. The server farm generates a 30 character string that correlates the user to the user's credentials but does not contain the credentials themselves. The farm forwards this ticket to the Web server, which places the ticket in the ICA file sent to the client device. When the ICA Client uses the ticket to authenticate itself to the server farm, the server farm matches the ticket to the user's actual credentials and logs the user in.

See “Configuring Web Server Extension Properties” on page 44 for information on configuring the expiration period for tickets.

Authenticating Users

The Web Site Wizard generates a login page that lets users enter user name, password, and domain information. This information is passed to the Citrix server, which then determines what applications the user is authorized to access.

There are three possible configurations for the login page:

Explicit logins. If you allow explicit logins, a login page is generated with fields for user name, password, and domain. Users must enter their login credentials to access applications.

Tip If you select the **Force Domain** check box in the wizard and specify a domain name, only the user name and password fields appear on the Web page. Only users from the specified domain can access applications. This feature is useful when you want to hide the concept of domains from users who are accustomed to entering only a user name and password to access network resources. This feature must be used if your Citrix server farm is composed of MetaFrame for UNIX Operating Systems server. See “Authentication and MetaFrame for UNIX Operating Systems” below for more information.

Explicit and Guest logins. If you allow both explicit and guest logins, the login page includes a **Guest** button. When a user clicks the **Guest** button, only the applications that do not require a specific set of login credentials appear.

Guest logins only. If you choose to allow only guest logins, the wizard does not generate a login page. The application set appears as soon as the user opens the Web page.

If you choose to allow guest logins, you must specify a user name, domain, and password in the fields provided. The guest password appears as a hidden field in the login page source code. The password is visible to any user who views the login page source.

Using User Principal Names and Active Directory Support

The login page generated by the Web Site Wizard is compatible with User Principal Names (UPN), such as those used in an Active Directory deployment. When using UPN, consider the following methods of user credential entry:

- **Separate entry of UPN name and suffix components.** In this scenario, a user with the name *userx@domain.com* enters *userx* in the login form's **Username** field and *domain.com* in the **Domain** field. To allow this type of login, select **Allow Explicit Logins** in the wizard page. Select **Force Domain** and enter a domain name if you want to enter the domain for the users, for example, if all users accessing this site are from the same domain. In this case, the **Domain** field does not appear visible on the page (though it is included as hidden text in the HTML) and users need not worry about domain entry at all.
- **Complete UPN entry.** In this scenario, a user named *userx@domain.com* enters the entire name, *userx@domain.com*, in the form's **Username** field. The presence of the @ symbol in the name causes NFuse to not require entry of information in the **Domain** field in the NFuse login form, and for this reason the user can leave the form's **Domain** field blank. To allow this type of login, select **Allow Explicit Logins** in the wizard page. Select **Force Domain** and specify no domain name in the wizard if you do not want the **Domain** field to appear on the login form.

Note Note that entry of a complete UPN containing the domain suffix in the form's **Username** field and a domain in the **Domain** field, either explicitly by the user or by the administrator entering a forced domain, causes an authentication error.

Authentication and MetaFrame for UNIX Operating Systems

The NFuse Web sites and Java objects require domain information for user authentication. If the user login process does not provide domain information, NFuse generates errors.

If your server farm is composed of MetaFrame for UNIX Operating Systems servers, you must configure your Web sites to pass a non-null domain name to the XML Service running on the MetaFrame server. Passing a non-null domain name satisfies the NFuse Web server's requirement for domain information and is ignored by the XML Service on the MetaFrame for UNIX Operating Systems server.

When creating a Web site for a MetaFrame for UNIX Operating Systems server farm, make sure you check the **Force Domain** option in the Web Site Wizard's authentication options and enter some text in the field. You can specify any text; for example, FakeDomain.

When you use the force domain option, the Wizard replaces the following standard NFuse form input:

```
<TR>
  <TD ALIGN="right" VALIGN="middle">Domain:</TD>
  <TD><INPUT NAME="domain"></TD>
</TR>
```

with a hidden field entry containing the domain name you specify:

```
<TR>
  <TD COLSPAN="2"><INPUT TYPE="hidden" NAME="Domain" VALUE="FakeDomain">
</TD>
</TR>
```

The resulting login page does not prompt the user to specify a domain name and sends the hidden name instead. This change has the same effect as entering some domain text in an NFuse form that does not hide the domain field.

Make sure you modify existing Web sites and example Web sites to pass non-null domain information to your MetaFrame for UNIX Operating Systems servers.

Storing Login Information

By default, a user's login credentials are stored in a transient cookie on the client device, allowing this information to persist across Web pages. When users exit a browser, the cookie is expired. You can modify the Web pages to use persistent cookies if necessary.

For additional security, you can implement encryption in your cookies. Cookie encryption protects user credential information. For information on cookie encryption, see "Encrypt Cookie Data" on page 166.

If the Web browsers or JVMs on your client devices do not support cookies, you must create your own sites that do not rely on cookies for information storage. See your Active Server Pages or JavaServer Pages documentation for information on using other methods of information storage such as session variables.

What to Do Next

Once you have installed the Web Server Extension and created Web pages, administrators or users must configure ICA Client devices as described in Chapter 4, "Configuring ICA Client Devices."

Configuring ICA Client Devices



This chapter explains how to configure your ICA Client devices. To use an ICA Client device with NFuse, the device must have a supported Web browser and ICA Client installed. NFuse includes Web-based ICA Client installation to help you deploy and install ICA Clients. Web-based ICA Client installation uses HTML documents and ICA Client installation files stored on a Web server to determine the type of ICA Client supported by a device and to supply an appropriate ICA Client for installation.

Most ICA Clients require no configuration after installation to work with NFuse. Two ICA Clients, the ICA Java Client and the ICA Macintosh Client, do however require Web browser configuration. This chapter includes instructions for configuring the Web browsers of these ICA Clients.

Tasks to Complete

In this chapter Web and Citrix server administrators will:

- Learn about using Web-based ICA Client installation to deploy ICA Clients
- If necessary, configure the Web browsers of ICA Client devices

Web-Based ICA Client Installation

Web-based ICA Client installation is a default component of the Web sites produced by the Web Site Wizard. Included in each site are several files that aid in determining client device requirements and presenting installation files to users. In addition to automatic installation recommendations, the Web sites produced by the wizard also include installation links that users can click to manually invoke an ICA Client installation.

Note Citrix also provides a stand-alone version of Web-based ICA Client installation. You can use this version to deploy ICA Clients independent of your NFuse Web sites. For more information, see the file Readme.htm in the NFuse CD-ROM's WebInst directory or in the downloaded CD image.

To use Web-based ICA Client installation you must:

- Make sure your Web server contains the ICA Client installation files
- Use the Web site wizard to produce an NFuse Web site and publish this site on your Web server

ICA Client Installation Files

During Web Server Extension installation, the setup program prompts you to supply an ICA Client CD or CD image. Setup copies the contents of the CD's ICAWEB directory to a directory called NFuseClients that it creates off the Web server's Web publishing root directory. All example Web sites and sites created by the Web Site Wizard assume that the Web server contains the ICA Client files in the directory structure created by the Web Server Extension setup program:

`<webroot>/NFuseClients/<language>/<icaclientversion>`

where `<webroot>` is your Web server's Web publishing root directory, `<language>` is the language version of the ICA Clients you want to deploy (en for English, de for German, fr for French, es for Spanish, or ja for Japanese), and `<icaclientversion>` is the type of ICA Client (ica16, ica32, icajava, icamac, icaunix, and icawince).

For example, in an English installation of the Web Server Extension on a typical IIS Web server, the ICA Win16 Client is contained in the following directory: `C:\Inetpub\WWWRoot\NFuseClients\en\ICA16`.

If you did not copy the ICA Client installation files to your Web server during Web Server Extension installation, make sure you copy the files to your Web server before using Web-based ICA Client installation.

➤ **To copy the ICA Client files to your Web server**

1. Create a directory called NFuseClients in your Web server's Web publishing root directory.
2. Insert an ICA Client CD in your Web server's CD-ROM drive or browse your network for a shared ICA Client CD image.
3. Change directories to the CD's ICAWEB directory. Copy the contents of the ICAWEB directory into the NFuseClients directory. Make sure you copy the contents of the directory and not the ICAWEB directory itself.

ICA Win32 Client Installation Archives

By default, Web-based ICA Client installation offers 32-bit Windows client devices the ICA Win32 Web Client installation archive. You can modify this behavior so that Web-based ICA Client installation offers the ICA Win32 Program Neighborhood Client for installation. The following list differentiates the two installation archives:

- **ICA Win32 Web Client installation archive (Ica32t.exe).** The default archive for Web-based ICA Client installation, ICA32t.exe installs all files necessary for the ICA Win32 Client to launch and embed ICA sessions in Web browsers. This archive does not install the Program Neighborhood user interface and various other ICA Client components and is therefore smaller than the full archive and easier to download.
- **ICA Win32 Program Neighborhood Client installation archive (Ica32.exe).** This archive installs all components of the ICA Win32 Client including the Program Neighborhood user interface. You must use this archive to install the ICA Win32 Client if your users require full ICA Client functionality.

Modifying the default behavior of Web-based ICA Client installation requires editing a file in your NFuse Web site. For Active Server Page and HTML for IIS-based Web sites, you must modify the file Icaclientinfo.asp. For JavaServer Page-based Web sites, modify the file Icaclient.jsp. The following excerpt from Icaclientinfo.asp displays the section of code to modify (similar code exists in Icaclient.jsp):

```
' [01] ### Win32 ###  
clientData(1,1) = "Win32"  
clientData(1,2) = "en/ica32/ica32t.exe"  
clientData(1,3) = "4,2,779,0"  
clientData(1,4) = "en/ica32/ica32t.exe"
```

To force Web-based ICA Client installation to use the full ICA Win32 Client installation archive, modify the bold lines in the above code so that they reference Ica32.exe instead of Ica32t.exe (remove the letter "t").

For Web sites based on the HTML for Servlets model, references to installation archives are contained within the compiled NFuse servlet `com.citrix.nfuse.PNClientDetection`. To force Web-based ICA Client installation to use the full ICA Win32 Client installation archive in an HTML for Servlets model site, you must rename the ICA Win32 Client archives themselves. For example, rename `Ica32.exe` to `Ica32t.exe`.

How Web-Based ICA Client Installation Works

Each site produced by the Web Site Wizard includes several files that provide ICA Client/Web browser/platform detection and download links for ICA Client installation files. For each of NFuse's four Web site layout models, the following table lists the files involved in Web-based ICA Client installation:

Layout model	Files included
Active Server Pages and HTML for IIS	default.htm, icaclient.asp, icaclientinfo.asp, clientdet.htm
JavaServer Pages	index.html, icaclient.jsp, clientdet.htm
HTML for Servlets	index.html, com.citrix.nfuse.PNClientDetection servlet, clientdet.htm

The following procedure describes the Web-based ICA Client installation functionality of an Active Server Page or HTML for IIS Web site.

➤ How Web-based ICA Client installation works

1. When a client device user visits the NFuse Web site, the `default.htm` file loads. This file first checks for the presence of a `noClientDetect` entry in a cookie on the device. This entry exists if the user logged into the site previously and clicked the **Do not show this window at login** check box. If the entry exists, `default.htm` redirects the user to the login page.

Note To reverse this setting after a user has checked it, log into an NFuse Web site and click the **Install Client** button. In the pop-up window that appears, deselect the **Do not show this window at login** check box. Client detection will then work again at login.

2. If the `noClientDetect` entry does not exist, `default.htm` executes client-side scripting that detects the platform of the client device and the type of browser in use. Based on this information, the page determines the type of ICA Client the device should use and stores this information in a cookie.

The platform and browser detection code can successfully identify an appropriate ICA Client for the following platforms: 32-bit Windows, 16-bit Windows, Macintosh, Solaris/SPARC, Solaris/x86, SunOS, SGI, HP/UX, IBM/AIX, SCO, DEC/Tru64, and Linux.

For other platforms that the detection code cannot identify, Web-based ICA Client installation writes into the cookie a recommendation to offer the user the ICA Java Client application.

3. The default.htm file then redirects the user to `icaclient.asp`. This file displays a pop-up window to the user. For example, on a 32-bit Windows system, the following pop-up appears:



4. This window either immediately closes or remains open depending upon the following conditions:
 - **For 16- and 32-bit Windows devices**, `icaclient.asp` detects the presence or absence of an installed ICA Client. If an ICA Client is installed, `icaclient.asp` stops executing and the user is presented with the login page. If no ICA Client is installed, `icaclient.asp`'s pop-up window remains open to present the user with an installation link.
 - **For all other client devices**, Web-based ICA Client installation does not have access to information about installed ICA Clients and must always offer the user the opportunity to install an ICA Client. For these devices, the `icaclient.asp` pop-up window remains open.

5. The **Install Citrix ICA Client for Win32** in the example link above points to the installation executable for the ICA Win32 Client file on the Web server. When creating the link, `icaclient.asp` checks the file `icaclientinfo.asp`, which contains a list that correlates ICA Client types to the locations of the installation executables stored on the Web server.
6. When the user clicks **Install Citrix ICA Client for Win32**, `icaclient.asp` retrieves the file and offers the user the option to install the ICA Client or save the installation file to disk. Some ICA Clients, such as the ICA Win16 and ICA Win32 Clients, include Web installer packaging that automates installation from within the Web browser. Users can click the link and the installer program unpacks the ICA Client and invokes Setup. Other ICA Clients, such as the ICA Java and ICA Macintosh Clients require that the user download and save the installation file to disk before manually invoking the setup program.

Note Users can invoke an installation at any time by clicking the **Install Client** button that appears in the application list page, `applist.htm`. This file invokes the file `clientdet.htm`, which contains the same browser and platform detection logic contained in `default.htm`. After determining which ICA Client should be installed on the client device, `clientdet.htm` invokes `icaclient.asp` to provide the user with an installation link.

JavaServer Page sites perform similar Web-based ICA Client installation activities. In these sites, the file `index.html` provides functionality identical to that of `default.htm` described above. The file `icaclient.jsp` combines the functionality provided by `icaclient.asp` and `icaclientinfo.asp`. HTML for Servlets sites provide similar functionality, except the servlet `com.citrix.nfuse.PNClientDetection` replaces the file `icaclient.jsp`.

Configuring Web Browsers

Before accessing published applications, some client devices must be configured for use with NFuse. Most supported ICA Clients require no additional configuration; however, the ICA Java Client (run in application mode) and the ICA Macintosh Client require configuration of the client device's Web browser before the ICA Client can be used with NFuse. When using these ICA Clients, you must manually register application/x-ica as a MIME type in the client device's browser.

Although the method differs slightly for each ICA Client/Web browser combination, in general, all browsers require the following information about the application/x-ica MIME type:

Field	Setting
File type	ICA
MIME type	application/x-ica
Description	ICA File
Extension	.ica
Helper Application	The location and name of the client device's ICA Client

The following procedures describe how to register application/x-ica as a MIME type on four ICA Client/Web browser combinations. For information about how to register MIME types on other supported browsers, see your browser's documentation.

Configuring the ICA Java Client

The following procedures describe how to configure the browser on an ICA Client device to associate the ICA Java Client with the application/x-ica MIME type. If you are running the ICA Java Client in applet mode, you do not need to configure the browser.

Note The following instructions describe how to configure Netscape Navigator Version 4.72 and Internet Explorer Version 4.5 for use with the ICA Java Client Version 4.11. For specific details about how to register a MIME type on other versions, see your browser's documentation.

- **To register application/x-ica as a Netscape Navigator MIME type**
 1. Install the ICA Java Client on the client device.
 2. Start Netscape Navigator.
 3. From the toolbar, select **Edit**, then **Preferences**. Under **Navigator**, select **Application**.
 4. Click the **New Type** button.
 5. In the **Description** field, type **ICA file**.
 6. In the **File extension** field, type **.ica**.
 7. In the **MIME type** field, type **application/x-ica**.
 8. In the **Application to use** field, type **x:\jicasession.bat %1**, where *x* is the full path to jicasession.bat.
 9. Click **OK**.
- **To register application/x-ica as an Internet Explorer MIME type**
 1. Install the ICA Java Client on the client device.
 2. On the client device, open a text editor and create the following .Reg file.

```
REGEDIT4
[HKEY_CLASSES_ROOT\MIME\Database\Content Type\application/x-ica]
"Extension"=".ica"
```

 3. Save this file as *yourfile.reg*.
 4. Double click the .Reg file to update the client device's registry.

Configuring the ICA Macintosh Client

The following procedures describe how to configure the browser on a Macintosh to associate the ICA Macintosh Client with the application/x-ica MIME type.

Note The following instructions describe how to configure Netscape Navigator Version 4.72 and Internet Explorer Version 4.5 for use with the ICA Macintosh Client Version 3.0. For specific details on how to register a MIME type on other versions, see your browser's documentation.

- **To register application/x-ica as a Netscape Navigator MIME type**
 1. Install the ICA Macintosh Client on the client device.
 2. Start Netscape Navigator.
 3. From the toolbar, select **Edit**, then **Preferences**. Under **Navigator**, select **Applications**.

4. Click the **New** button.
 5. In the **Description** field, type **ICA file**.
 6. In the **MIME Type** field, type **application/x-ica**.
 7. In the **Suffixes** field, type **.ica**.
 8. Select the **Application** radio button and click **Choose** to browse to the location of the ICA Macintosh Client.
 9. Click **OK**.
- **To register application/x-ica as an Internet Explorer MIME type**
1. Install the ICA Macintosh Client on the client device.
 2. Select **Edit**, then **Preferences**.
 3. In the left panel of the **Preferences** dialog box, under **Receiving Files**, select **File Helpers**.
 4. Click **Add** to display the **Edit File Helper** dialog box.
 5. In the **Description** field, type **ICA file**.
 6. In the **Extension** field, type **.ica**.
 7. In the **MIME type** field, type **application/x-ica**.
 8. In the **File Type** section of the dialog box, click **Browse**.
 9. Browse to the location of the ICA Macintosh Client, select the ICA Client application, and click **Open**.
 10. Select the **Binary data** radio button and **Use for Incoming** check box.
 11. Select **Post process with Application** from the **How to Handle** drop down list.
 12. Click **OK**.

What to Do Next

Once you have installed and (if necessary) configured your ICA Client devices, NFuse setup is complete. For users to access the NFuse Web pages and their own application sets, provide them with the URL where your Web pages are saved.

For information on securing ICA Client devices, see Chapter 8, “Configuring NFuse Security” on page 163.

Using NFuse Tags



This chapter describes NFuse substitution tags and session fields. The NFuse substitution tags and session fields provide an interface to the NFuse Java objects. You can use this interface to:

- Write HTML pages and ICA file templates that serve published applications to ICA Client users
- Modify NFuse properties and, when used with HTTP cookies and URLs, create cross-page state in your NFuse Web sites

This chapter includes a reference that lists substitution tag and session field syntax. The chapter ends with a tutorial that steps through a typical Web site built using substitution tags and session fields.

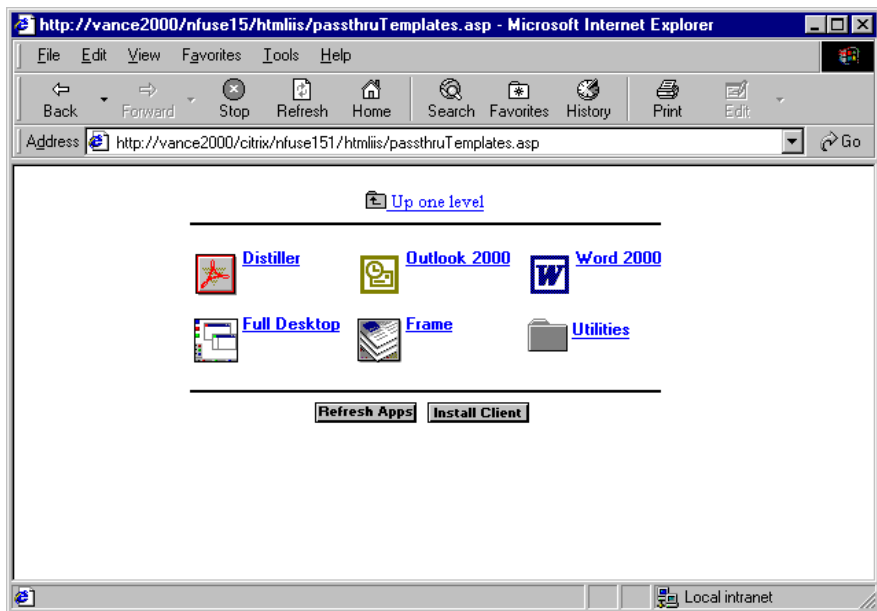
NFuse Substitution Tags

The NFuse substitution tags comprise a mark-up language that the NFuse Java objects can parse and deploy to Web browsers. Web masters can place substitution tags in template HTML and ICA files. When serving a template HTML or ICA file containing substitution tags, the Java object called `TemplateParser` scans the file for substitution tags and replaces them with relevant data. In the case of HTML pages, `TemplateParser` replaces the substitution tags with hyperlinks to published applications that the current user can access. When parsing a template ICA file, `TemplateParser` replaces the substitution tags with information that can be placed in an ICA file and sent to the client device's ICA Client for session initialization.

Simple Web Site Creation

NFuse includes substitution tags for those Web masters who want to create customized NFuse Web pages but are unfamiliar with Web server scripting. Much like scripting-based Web pages, such as pages that make use of Microsoft's Active Server Pages or Sun's JavaServer Pages technologies, substitution-tag-based pages can dynamically present published application information to Web browsers on ICA Client devices.

Substitution tags are best used in simple pages. Unlike scripting-based pages, which give you the ability to precisely place and order published application links in a Web page, substitution-tag-based pages are intended for drawing published application links in a simple HTML table. The following picture illustrates a typical substitution-tag-based display of published application links:



Important For more advanced deployment scenarios, including Web page layouts that require exact placement and ordering of applications within a page, use scripting-based Web pages. See "NFuse Java Object Reference" on page 103.

When responding to a request from a Web browser for a substitution-tag-based Web page, the NFuse Java objects on the Web server perform symbol substitution on all the substitution tags contained in the page. This symbol substitution replaces the tags with user-specific application information and sends the browser a plain HTML page containing links to the applications in the user's application set.

To use substitution-tag-based Web sites with Microsoft Internet Information Server, you must have a Web server that supports Active Server Pages. Substitution-tag-based pages for Microsoft IIS use Active Server Page support files to perform the symbol substitution and various other NFuse-related tasks such as calling the Java objects to pass user credentials to the server farm and creating ICA files for the applications users select to run.

Substitution-tag-based pages for Apache, Netscape, and iPlanet Web servers use Java servlets to perform symbol substitution. To use substitution-tag-based Web sites with these servers, make sure the servers support Java servlets.

Template ICA File Creation

An *ICA file* is a text file containing information about a published application. ICA files are written in Ini file format and organize published application information in a standardized way that ICA Clients can interpret. When an ICA Client receives an ICA file, it initializes a session containing the desired application on the Citrix server specified in the file. All application links in NFuse Web pages generate ICA files.

NFuse uses a template ICA file as the basis for the ICA files referenced in its hyperlinks. A *template ICA file* is a text file that adheres to the ICA file format and contains substitution tags that can be replaced with information about a specific user and the desired application.

When a user clicks an NFuse hyperlink, the NFuse Java objects retrieve the template ICA file and replace its substitution tags with information about the requested application and the user's credentials before sending the file to the client device.

The following is an example template ICA file:

```
<[NFuse_setSessionField NFuse_ContentType=application/x-ica]>
<[NFuse_setSessionField NFuse_WindowType=seamless]>

[WFClient]
Version=2
ClientName=[NFuse_ClientName]

[ApplicationServers]
[NFuse_AppName]=

[[NFuse_AppName]]
Address=[NFuse_IPV4Address]
InitialProgram=#[NFuse_AppName]
DesiredColor=[NFuse_WindowColors]
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0
<[NFuse_IFSESSIONFIELD sessionfield="NFUSE_ENCRYPTIONLEVEL" value="basic"]>
Username=[NFuse_User]
Domain=[NFuse_Domain]
Password=[NFuse_PasswordScrambled]
<[/NFuse_IFSESSIONFIELD]>
<[NFuse_IFSESSIONFIELD sessionfield="NFUSE_SOUNDTYPE" value="basic"]>
ClientAudio=On
<[/NFuse_IFSESSIONFIELD]>
[NFuse_IcaWindow]

[NFuse_IcaEncryption]
```

Bracketed items such as [NFuse_AppName] and [NFuse_IPV4Address] indicate substitution tags that the TemplateParser object will replace when processing this template. Bracketed section headings such as [ApplicationServers] are not substitution tags; these bracketed items are standard ICA file section names and are always enclosed in brackets.

Note also the presence of parameter/value pairs that do not contain substitution tags, such as the entry TransportDriver=TCP/IP. A template ICA file can contain any valid ICA file parameter/value pair specified along with the substitution-tag-based parameter/value pairs. For a list of valid ICA file entries, see “ICA File Reference” on page 145.

NFuse Session Fields

Session fields allow you to set NFuse properties and, when used with HTTP cookies and URLs, to maintain state information between Web pages.

By using session fields in your Web pages, you can set and modify the Citrix server and TCP/IP port used for published application requests, the location of NFuse's icon cache, and the directory where NFuse stores template files, among other properties.

Session fields also allow you to maintain state information between NFuse Web pages. State information can include the current user's password and user name or the properties of a published application, among others. You can use session fields to make these properties persist between pages so that they are available to the NFuse Java objects from different pages in your site.

For example, by placing user credential session fields in an HTTP cookie, you can make the login information entered by a user in a login form available to the NFuse Java objects parsing a template ICA file for a specific application. When the TemplateParser object performs symbol substitution on the template ICA file, it replaces user credential substitution tags in the template ICA file with the value of the user credential session fields specified in the cookie.

Note You can use session fields in substitution-tag-based pages and in scripting-based pages.

You can set session fields from various places in your Web site including in server script, URLs, cookies, and the NFuse.properties file. Session fields obey an override precedence order. This precedence order lets you set a session field to different values at different points in the execution of your site. See "Using Session Fields" on page 97 for more information.

NFuse Substitution Tag and Session Field Reference

All substitution tags and session fields use the delimiters [...] or <[...]>. All substitution tags begin with *NFuse_*, such as <[NFuse_Domain]>. White space can appear inside the delimiters or between the delimiters and the name of the tag.

The following topics list NFuse's substitution tags. Unless otherwise indicated, each tag functions as both a substitution tag and a session field. When processing a tag, the TemplateParser object replaces the tag with the current value of the corresponding session field. For example, when the TemplateParser object encounters the [NFuse_User] tag in a template ICA file, it replaces the tag with the current value of the [NFuse_User] session field. This session field must be set earlier in the execution of the site and made available to the template ICA file through a cookie or in a URL.

General Tags

These tags perform basic NFuse functions such as supplying the NFuse Java objects with Citrix server contact information and packaging of user credentials.

NFuse_Application	Specifies an instance of a published application.
NFuse_CitrixServer	Specifies which Citrix server provides the published application information. This value can be a fully-qualified DNS name, NT server name, or IP address.
NFuse_CitrixServerPort	Specifies the TCP/IP port used by the Citrix server specified in NFuse_CitrixServer for NFuse communication. The default value is the port number entered during Web Server Extension installation. This port number must match the port number used by the Citrix server.

NFuse_Transport	Specifies the protocol used to transport NFuse data between the Web server and Citrix server specified in NFuse_CitrixServer. Values include “HTTP” and “SSL.” Use HTTP to send the NFuse data over a standard HTTP connection to the server and port specified in NFuse_CitrixServer and NFuse_CitrixServerPort. Use SSL to send data over a secure connection that uses a Citrix server running the Citrix SSL Relay to perform host authentication and data encryption. This protocol sends the data to the server and port specified in NFuse_CitrixServer and NFuse_CitrixServerPort through a Citrix SSL Relay server specified in NFuse_RelayServer and NFuse_RelayServerPort. If NFuse_Transport is not present, the connection uses HTTP.
NFuse_RelayServer	Specifies the fully-qualified DNS name, NT server name, or IP address of a Citrix server running the Citrix SSL Relay. Used with SSL connections. Support for the Citrix SSL Relay is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems.
NFuse_RelayServerPort	Specifies the port on which NFuse_RelayServer is listening for SSL requests. Used with SSL connections.
NFuse_CurrentFolder	Determines what Program Neighborhood folder to enumerate when processing the [NFuse_DrawPN] and [/NFuse_DrawPN] tags. You can also use this tag to determine the value of the [NFuse_SubFolder] and [NFuse_ParentFolder] tags. If this session field is not set, the root folder is assumed.
NFuse_CurrentFolderUrl Encoded	Same as NFuse_CurrentFolder except this tag returns data in a URL-encoded string. This format is useful when the returned string must be placed in a URL, cookie, or other location where certain characters, if left unencoded, can cause errors in a Web browser or Web server.
NFuse_SubFolder	Replaced with the currently enumerated subfolder in a [NFuse_DrawPN]...[/NFuse_DrawPN] block.

NFuse_SubFolderUrl Encoded	Same as NFuse_SubFolder except this tag returns data in a URL-encoded string. This format is useful when the returned string must be placed in a URL, cookie, or other location where certain characters, if left unencoded, can cause errors in a Web browser or Web server.
NFuse_ParentFolder	Replaced with the parent folder of the folder specified by the NFuse_CurrentFolder session field. If the NFuse_CurrentFolder session field is the root, this tag is replaced by the root folder (an empty string).
NFuse_ParentFolderUrl Encoded	Same as NFuse_ParentFolder except this tag returns data in a URL-encoded string. This format is useful when the returned string must be placed in a URL, cookie, or other location where certain characters, if left unencoded, can cause errors in a Web browser or Web server.
NFuse_Template	Specifies a template ICA or HTML file to parse. You can specify a filename or sub-path located in the templates directory as specified by the NFuse_TemplatesDir session field.
NFuse_User	Specifies the client device user's user name.
NFuse_Domain	Specifies the client device user's domain.
NFuse_Password	Specifies the client device user's password.
NFuse_Password Scrambled	Specifies an encoded form of the value of the NFuse_Password session field. The encoding used is that which is expected by the ICA Client in ICA files. Using this tag without first setting NFuse_Password causes an error.

NFuse_Ticket	Retrieves authentication ticket for placement in ICA files. The TemplateParser object replaces this tag with all required credential information in the following format: User=actual user name Domain="\ " character followed by the last 16 characters of the ticket ClearPassword=first 14 characters of ticket For example: User=RCCollins Domain=\81C3B3041E6A703E ClearPassword=70AADC7DC5926E Support for ticketing is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems.
NFuse_TicketUpper	Retrieves first 14 characters of an authentication ticket. Used to replace password information in an ICA file. Can be used to place a ticket in an ICA file that contains a static user name or user name supplied by another NFuse tag. For example: User=RCCollins Domain=[NFuse_TicketLower] ClearPassword=[NFuse_TicketUpper]
NFuse_TicketLower	Retrieves last 16 characters of an authentication ticket preceded by a backslash. Can be used to place a ticket in an ICA file that contains a static user name or user name supplied by another NFuse tag. See NFuse_TicketUpper for example usage.
NFuse_TicketTimeToLive	Specifies the amount of time during which a ticket is valid. During this period the ticket can be used once. After the time period passes, the ticket is no longer valid. Values are specified in seconds.
NFuse_GroupNames	Retrieves a colon (:) delimited list of group names. For example: Domain Users:Administrators:Public. When using group names to filter retrieved applications, the TemplateParser obtains the list of group names from this session field.

NFuse_useCredentialType	Specifies the type of user credentials to use to retrieve applications. Possible values include: UseActualCredentials: Use credentials composed of a user name, domain, and password. UseGroupCredentials: Use a group ID. See “GroupCredentials” on page 110. UseNullCredentials: Use a credential set composed of a null user name, null domain, and null password. Use this method if you want to retrieve all published applications in a server farm. The default value is UseActualCredentials. The TemplateParser always checks this session field when passing credentials to the CitrixWireGateway object.
NFuse_ContentType	Specifies the MIME content type that the Web server reports for the response.
NFuse_MIMEExtension	Included to support older browsers that use file extensions for MIME type mapping. Use if your browser requires that your URLs end in .ica in order for the URL to be associated with an ICA Client. For example: <pre><a href="launch.asp?NFuse_Application=Excel&NFuse_MIMEExtension=.ica".</pre> This tag does not function as a session field. When using this tag, you must always set it equal to the value “.ica”.
NFuse_TemplatesDir	Specifies the directory in which the template ICA or HTML file (as specified by the NFuse_Template session field) can be found. This path must be absolute to the machine on which the Web server is running; for example: C:\InetPub\wwwroot\NFuse. Do not use Web server relative paths.
NFuse_TemplatesURL	Like NFuse_TemplatesDir, this tag specifies the directory in which the template HTML or ICA file (as specified by the NFuse_Template session field) can be found. This path is relative to your Web server's Web root directory. For example: /NFuse

NFuse_SetSessionField	Sets a session field from within a template ICA or HTML file. The NFuse_SetSessionField session field uses the following syntax: [NFuse_SetSessionField <i>SessionField=Value</i>] where <i>SessionField</i> is the name of the session field you want to set and <i>Value</i> is the new value for the session field. When parsing an NFuse_SetSessionField tag, the TemplateParser object removes the tag itself after setting the specified session field. For information about commands that let you set session fields, see “Setting Session Fields” on page 98.
NFuse_Cache	Controls caching of application information on the Web server. Possible values include: NoCache: Do not cache App object data. UseCache: Use cached App data first before checking the Citrix server for application information. If the data does not exist in the cache or the data is expired, NFuse contacts the Citrix server for application information. This value is the default value. RefreshCache: Contact the Citrix server for application information and update the App object data in the cache.

Application Property Tags

Application property tags represent properties of a published application. The application whose properties replace these tags is determined by the NFuse_Application, NFuse_CitrixServer, NFuse_User, NFuse_Domain, and NFuse_Password session fields.

All of the following properties are initially set by a Citrix administrator at the time of application publishing. You can explicitly override these properties by using a set session field command, in which case the overriding value is used. See “Using Session Fields” on page 97 for more information.

NFuse_WindowType	Specifies the window type of the ICA session window for the referenced application. Possible values include: pixels: A size in pixels percent: A size as a percentage of the client desktop seamless: Seamless window fullscreen: Full screen
NFuse_WindowHeight	Retrieves the height in pixels of the ICA session window for the referenced application.
NFuse_WindowWidth	Retrieves the width in pixels of the ICA session window for the referenced application.
NFuse_WindowScale	Retrieves the percentage of the client device's desktop that the ICA session window should occupy. This tag returns an integer from 0 to 100. If no percentage size is specified for this application, the tag returns 0.
NFuse_WindowColors	Retrieves an integer representing the number of colors used in the ICA session window to display the referenced application. This tag returns one of the following: 1: 16 colors 2: 256 colors
NFuse_IcaWindow	Retrieves ICA session window information for placement in ICA files. The TemplateParser object replaces this tag with all window information required in an ICA file.

NFuse_EncryptionLevel	Retrieves a string representing the level of encryption to use for the referenced application. To enable encryption levels higher than Basic, the Citrix server must support RC5 encryption. Support for RC5 encryption is included in MetaFrame XP, Feature Release 1 for MetaFrame 1.8, and SecureICA Services. MetaFrame for UNIX Operating Systems servers do not support RC5 encryption. This tag returns one of the following: - basic: Basic encryption (XOR) - rc5-login: 128-bit for login only - rc5-40: 40-bit - rc5-56: 56-bit - rc5-128: 128-bit
NFuse_IcaEncryption	Retrieves encryption information for placement in ICA files. The TemplateParser object replaces this tag with all encryption information required in an ICA file.
NFuse_SoundType	Retrieves a string representing the level of audio support for the referenced application. This tag returns one of the following: - none: No sound support - basic: Sound 1.0
NFuse_VideoType	Retrieves an integer representing the level of video support to use for the referenced application. This tag returns one of the following: - none: No video support - basic: Video 1.0
NFuse_AppFriendlyName	Retrieves the friendly name (also called Display Name or External Name) of an application published in a Citrix server farm. Friendly names identify applications to client device users. Use friendly names to display application names to users; for example in an application list page. This property is not settable as a session field.
NFuse_AppFriendlyName UrlEncoded	Same as NFuse_AppFriendlyName except this tag returns data in a URL-encoded string. This format is useful when the returned string must be placed in a URL, cookie, or other location where certain characters, if left unencoded, can cause errors in a Web browser or Web server.

NFuse_AppName	Retrieves the internal application name (also called Application Name or Application ID) of an application published in a Citrix server farm. Citrix servers use these application names internally to identify applications. A single internal name cannot be used by more than one application. Use internal names when identifying an application to run; for example, in an ICA file initial program entry such as InitialProgram=#[NFuse_AppName]. This property is not settable as a session field.
NFuse_AppNameUrl Encoded	Same as NFuse_AppName except this tag returns data in a URL-encoded string. This format is useful when the returned string must be placed in a URL, cookie, or other location where certain characters, if left unencoded, can cause errors in a Web browser or Web server.
NFuse_IPv4Address	Retrieves the IP address of the Citrix server hosting the published application. The use of this tag relates to the NFuse_ClientName tag described below. If you do not use NFuse_ClientName to specify a client name for the client connecting to the published application, NFuse_IPv4Address generates a unique client name based on the credentials contained in the NFuse_User, NFuse_Password, and NFuse_Domain session fields.
NFuse_IPv4Address Alternate	Retrieves the external (or public) IP address of the Citrix server hosting the published application. Use this tag when accessing a Citrix server through a firewall. The use of this tag relates to the NFuse_ClientName tag described below. If you do not use NFuse_ClientName to specify a client name for the client connecting to the published application, NFuse_IPv4AddressAlternate generates a unique client name based on the credentials contained in the NFuse_User, NFuse_Password, and NFuse_Domain session fields. To use alternate addressing, you must also configure the Citrix server. If your Citrix server is a MetaFrame for Windows server, see your server documentation for information on using the ALTADDR utility. For MetaFrame for UNIX Operating Systems servers, see your server documentation for information on using the CTXALT utility.

NFuse_ClientName	Retrieves a unique client name for the ICA Client based on the credentials contained in the NFuse_User, NFuse_Password, and NFuse_Domain session fields. A client name is required for session reconnect and client device mapping.
NFuse_AppDescription	Retrieves the description for the referenced application.
NFuse_AppIcon	Retrieves the URL of the .Gif file for the referenced application. By default, when you use NFuse to access applications, the NFuse Java Objects create a .Gif file for each application. NFuse saves these files in a directory on your Web server.
NFuse_AppIconUrlEncoded	Same as NFuse_AppIcon except this tag returns data in a URL-encoded string. This format is useful when the returned string must be placed in a URL, cookie, or other location where certain characters, if left unencoded, can cause errors in a Web browser or Web server.

User Interface Tags

The *user interface tags* create the application lists users see in their Web pages. User interface tags occur in pairs with the first tag opening the draw function and the second tag closing it. Text occurring between interface tags is parsed repeatedly for every application and folder in the user's view of the Program Neighborhood. After parsing of the interface tags completes, the TemplateParser object removes the interface tags themselves so that they are not seen by client Web browsers.

User interface tags are not settable as session fields.

NFuse_DrawPN

Opens the draw Program Neighborhood process. This tag takes a number of arguments that determine the look of what is drawn:

NumCols=*n* Draw application list in *n* columns.

NumRows=*n* Draw application list in *n* rows.

Flat=[Yes | No] Yes: Draw all the applications contained in the current folder and its subfolders. Draw no subfolders. No: Draw only those applications and subfolders contained in the current folder.

For example:

```
[NFuse_DrawPN NumCols="3" NumRows="3"
flat="no"]
```

/NFuse_DrawPN

Closes the draw Program Neighborhood process.

Conditional Tags

Interface tags often appear with conditional tags. *Conditional tags* envelop text that is processed only if a particular condition is met. If the condition is met, the conditional tags themselves are removed and the TemplateParser object processes all text as normal. If the conditions are not met, the conditional tags themselves and all the text within them are removed and not processed. It is possible to nest one conditional tag within another.

Important All conditional tags, with the exception of the NFuse_IfSessionField and /NFuse_IfSessionField tags, must be used only during the draw Program Neighborhood process (between the NFuse_DrawPN and /NFuse_DrawPN tags). You can use the NFuse_IfSessionField and /NFuse_IfSessionField tags in any location in your pages.

Like the user interface tags, conditional tags are not settable as session fields.

NFuse_IfApp	Opening conditional tag. The text between this tag and the corresponding /NFuse_IfApp is processed and passed through only if the current item being enumerated between the NFuse_DrawPN and /NFuse_DrawPN tags is an App object, not a folder.
/NFuse_IfApp	Closing conditional tag.
NFuse_IfFolder	Opening conditional tag. The text between this and the corresponding /NFuse_IfFolder is processed and passed through only if the current item being enumerated between the NFuse_DrawPN and /NFuse_DrawPN tags is a folder, not an App object.
/NFuse_IfFolder	Closing conditional tag.
NFuse_IfRowStart	Opening conditional tag. The text between this tag and the corresponding /NFuse_IfRowStart is processed and passed through only if the current iteration of the DrawPN text is the first iteration in a row as determined by the numRows or numColumns arguments to the NFuse_DrawPN tag.
/NFuse_IfRowStart	Closing conditional tag.
NFuse_IfRowEnd	Opening conditional tag. The text between this tag and the corresponding /NFuse_IfRowEnd is processed and passed through only if the current iteration of the DrawPN text is the last iteration in a row as determined by the numRows or numColumns arguments to the NFuse_DrawPN tag.
/NFuse_IfRowEnd	Closing conditional tag.

NFuse_IfSessionField	Opening conditional tag. The text between this tag and the corresponding /NFuse_IfSessionField is processed and passed through only if the specified session field is currently set to the specified value. The NFuse_IfSessionField conditional tag uses the following syntax: <pre><[NFuse_IfSessionField sessionfield=X value=Y]> <i>Code to process</i> <[/NFuse_IfSessionField]></pre> where <i>X</i> is the name of a session field, <i>Y</i> is the value you are testing for, and <i>Code to process</i> is some HTML, script, or substitution tags to include if the session field is currently set to <i>Y</i>. For an example of this tag's usage, see a template.ica file produced by the Web Site Wizard. An explanation of template.ica's use of this tag appears in "Encryption Parameters" on page 155.
/NFuse_IfSessionField	Closing conditional tag.

NFuse Substitution Tag and Session Field Tutorial

The following topics give practical examples of substitution tag and session field use. These topics describe:

- Typical substitution-tag-based site layout
- How to use session fields to set properties and create cross-page state in your Web pages

Creating a Substitution-Tag-Based Web Site

Use the Web Site Wizard to create a substitution-tag-based Web site. The Web Site Wizard creates a complete site that includes a login page, a main application list page, back-end support files that handle tag substitution and ICA Client installation, and graphic images used for navigation. In many cases, you can use these files unmodified. In case you do require functionality that differs from that of the standard sites produced by the wizard, use the wizard to create a standard site and then modify the necessary files. Often, you can use the login, support, and graphic image files as is and simply modify the application list page.

Files Included in a Substitution-Tag-Based Web Site

The following lists describe files found in substitution-tag-based sites produced by the wizard. The first site described is the substitution-tag-based site for Microsoft IIS (called **HTML for IIS** in the wizard).

Note The actual file list for your site depends upon the options you choose during site creation. For example, sites that allow anonymous logins or embedded ICA sessions will have a file list different from the sites described below.

A typical substitution-tag-based Web site created by the Web Site Wizard for Microsoft IIS contains the following files:

- **default.htm.** This page contains client-side scripting that detects the platform of the client device and the type of browser in use. Based on this information, the page determines the type of ICA Client the device should use and stores this information in a cookie. The default.htm file then redirects the user to the page icaclient.asp.
- **icaclient.asp.** When invoked by default.htm, this file presents the user with a link to an ICA Client to install. The page uses the browser and platform information in the cookie to present the user with the appropriate ICA Client. If, however, the user logged into the site in the past and chose to not be prompted to install an ICA Client, icaclient.asp redirects the user immediately to the login page, login.htm.

When evoked by clientdet.htm, icaclient.asp always presents the user with a link to an ICA Client to install and does not redirect the user even if the user previously chose to not be prompted to install an ICA Client.

- **login.htm.** A page that lets users enter credentials. This HTML form uses the Post method to send the user's user name, password, and domain to the file passthru templates.asp. (When authenticating users to MetaFrame for UNIX Operating Systems server farms, the login form must submit placeholder domain information. See "Authentication and MetaFrame for UNIX Operating Systems" on page 56 for more information.)
- **passthru templates.asp.** This scripted support file places the user's user name, password, and domain information in a cookie. The information is formatted in substitution tag/value pairs so that the other support files can mine the information as session fields.

This file also contains HTML code that redirects the user to a new page composed of a frameset containing two documents: boilerplate.asp and blank.htm.

- **boilerplate.asp.** This scripted support file initiates and controls the processing of pages such as applist.htm and template.ica that contain substitution tags. When parsing applist.htm, boilerplate.asp displays the data to the user in the form of an HTML page containing application links. When parsing template.ica, boilerplate.asp creates an ICA file that the ICA Client receives and uses to initiate an ICA session.

- **blank.htm.** This blank HTML document contains no viewable text or commands to process. It exists in the Web site to catch redirected messages created when an ICA Client user launches an application. If not redirected, these messages would otherwise obscure the user's application set.

This frame also displays any error messages generated at session initialization. To view these error messages, right-click a malfunctioning link and save the target as a file on the client device's hard disk. Use a text editor to open the file and view the error message.

- **applist.htm.** This template HTML file is the main application list page viewed by users. NFuse uses the TemplateParser object to parse this page. Parsing removes all substitution tags and replaces them with user-specific application information.

Each hyperlink in the application list seen by users points to the boilerplate.asp file. Each hyperlink contains information about a specific application. When a user clicks a link, boilerplate.asp receives the user's request, parses the URL for information about the requested application, and then feeds that information to the file template.ica.

In addition, applist.htm contains an **Install Client** button that users can click to manually invoke an ICA Client installation. When a user clicks this link, the page invokes the file clientdet.htm.

- **template.ica.** This file is the template ICA file used by NFuse to send an ICA session request to the ICA Client. The template.ica file, like applist.htm, is a template file containing substitution tags. When the boilerplate.asp file parses template.ica, it removes all substitution tags and replaces them with application information before sending the ICA file to the client device.
- **NFuseMedia.** This directory contains graphic image files used in the Web site for navigation. The other files in the site assume this directory exists as named.
- **clientdet.htm.** This file is included in the site to aid in manually invoked ICA Client installations (as opposed to the immediate client installation offered at first login). This page is invoked when a user clicks **Install Client** in the applist.htm file. The clientdet.htm file contains the same browser and platform detection logic contained in default.htm. Using the browser and platform information, the page determines the type of ICA Client the device should use and stores this information in a cookie. The clientdet.htm file then redirects the user to the file icaclient.asp.

- **icaclientinfo.asp.** This file contains a list that correlates ICA Client types to locations of the actual installation executables stored on the Web server. The icaclient.asp file uses this information when creating a link to the ICA Client installation executable.
- **log.txt.** This text file lists options chosen in the wizard during creation of the site.

A typical substitution-tag-based Web site created by the Web Site Wizard for Apache, Netscape, and iPlanet Web servers contains the following files:

- **index.html.** This page contains client-side scripting that detects the platform of the client device and the type of browser in use. Using this information, the page determines the type of ICA Client the device should use and stores this information in a cookie. The index.html file then invokes the servlet called com.citrix.nfuse.PNClientDetection. This servlet performs ICA Client installation.

If the user logged into the site in the past and chose to not be prompted to install an ICA Client, com.citrix.nfuse.PNClientDetection immediately redirects the user to the login page, login.htm.

Otherwise, after com.citrix.nfuse.PNClientDetection completes client installation, the user is redirected to login.htm.

- **login.htm.** A page that lets users enter credentials. This HTML form uses the Post method to send the user's user name, password, and domain to the com.citrix.nfuse.PNServletTemplate Java servlet class. This servlet takes the form data and processes it when parsing the next page, passthru templates.htm. (When authenticating users to MetaFrame for UNIX Operating Systems server farms, the login form must submit placeholder domain information. See "Authentication and MetaFrame for UNIX Operating Systems" on page 56 for more information.)
- **passthru templates.htm.** When parsing passthru templates.htm, the PNServletTemplate class takes the user credential form data and places it in a cookie. The information is formatted in substitution tag/value pairs so that the other files can mine the information as session fields.

This file also contains HTML code that displays a new page composed of a frameset containing two documents. The first frame contains a command that redirects the user to the PNServletTemplate class with the file applist.htm specified as a template for the servlet to parse. The second frame contains blank.htm, which is described below.
- **applist.htm.** This page is the main application list page viewed by users. NFuse uses the PNServletTemplate class to parse this page. Parsing removes all substitution tags and replaces them with user-specific application information.

Each hyperlink in the page contains information about a specific application. When a user clicks a link, the browser calls the `PNServletTemplate` class to parse `template.ica` with information about the selected application and user.

In addition, `applist.htm` contains an **Install Client** button that users can click to manually invoke an ICA Client installation. When a user clicks this link, the page invokes the file `clientdet.htm`.

- **blank.htm.** This blank HTML document contains no viewable text or commands to process. It exists in the Web site to catch redirected messages created when an ICA Client user launches an application. If not redirected, these messages would otherwise obscure the user's application set.

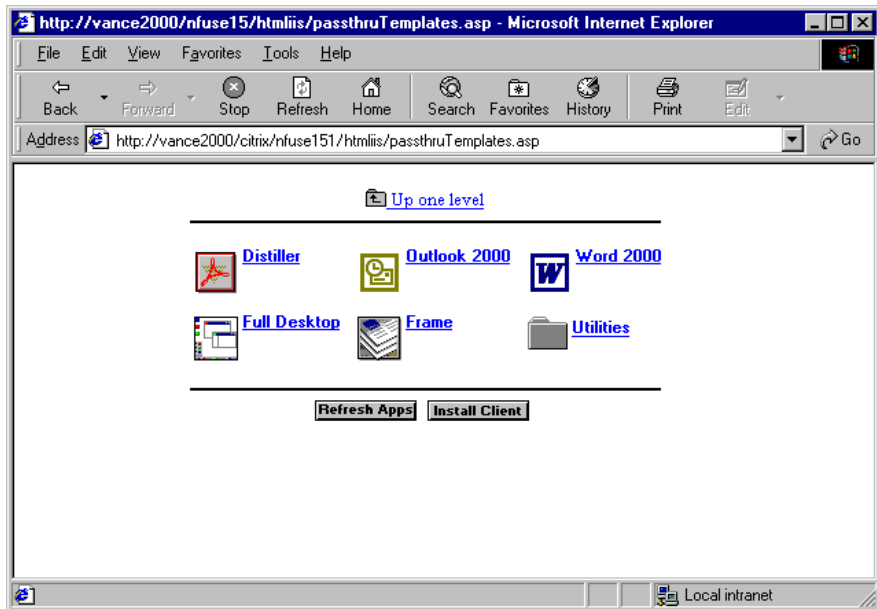
This frame also displays any error messages generated at session initialization. To view these error messages, right-click a malfunctioning link and save the target as a file on the client device's hard disk. Use a text editor to open the file and view the error message.
- **template.ica.** This file is the template ICA file used by NFuse to send an ICA session request to the ICA Client. The `template.ica` file, like `applist.htm`, is a template file containing substitution tags. When the `PNServletTemplate` class parses `template.ica`, it removes all substitution tags and replaces them with application information before sending the ICA file to the client device.
- **NFuseMedia.** This directory contains graphic image files used in the Web site for navigation. The other files in the site assume this directory exists as named.
- **clientdet.htm.** This file is included in the site to aid in manually invoked ICA Client installations (as opposed to the immediate client installation offered at first login). This page is invoked when a user clicks **Install Client** in the `applist.htm` file. The `clientdet.htm` file contains the same browser and platform detection logic contained in `index.html`. Using the browser and platform information, the page determines the type of ICA Client the device should use and stores this information in a cookie. The `clientdet.htm` file then invokes the servlet `com.citrix.nfuse.PNClientDetection`. This servlet presents the user with a link to an ICA Client to install.

The following topic explains the application list file `applist.htm`. You can edit `applist.htm` to change the way in which it displays applications, the amount of application information it includes in the page, or the properties of the actual applications.

Application List Page

The applist.htm file described below is the default file included in an HTML for IIS Web site created by the Web Site Wizard. To create the same page, use the wizard and accept all default options with the exception of the Seamless setting. Uncheck the **Use Seamless if available** option in the wizard screen that lets you select to launch or embed the applications.

By default, this applist.htm page displays each application's name and icon as hyperlinks. When a user launches an application, the ICA session appears in a window of whatever size the Citrix server administrator specified at the time of application publishing. In this exercise, you will learn how to change the default window size behavior to allow users to select a window size. The page edits required to make this modification can be duplicated to change other application properties. The following graphic depicts the default look of applist.htm:



The following code example shows the HTML used to create this page. The applist.htm file includes a section of code delimited by the `<[/NFuse_DrawPN numCols="3" flat="no"]>` and `<[/NFuse_DrawPN]>` tags. This section is responsible for drawing the application icons and links into the HTML page.

Other sections of applist.htm's code are omitted to display only the part of the file that requires editing.

```
<[NFuse_DrawPN numCols="3" flat="no"]>
<[NFuse_IfRowStart]><TR><[/NFuse_IfRowStart]>
  <[NFuse_IfApp]>
    <TD ALIGN="LEFT" VALIGN="TOP">
      <A HREF="boilerplate.asp?NFuse_Template=template.ica&
        NFuse_Application=[NFuse_AppNameUrlEncoded]&
        NFuse_MIMEExtension=.ica" TARGET="hiddenwindow">

      <IMG SRC="[NFuse_AppIconUrlEncoded]" border=0
        alt="[NFuse_AppFriendlyName]" align="LEFT"
        HSPACE=3 VSPACE=3>

      <B STYLE="font-family:MS Sans Serif,Arial,geneva,Helvetica,sans-serif;
        font-size:9pt">[NFuse_AppFriendlyName]</B><BR>
      </A>
    </TD>
  <[/NFuse_IfApp]>
  <!-- code omitted here -->
<[/NFuse_IfRowEnd]></TR><[/NFuse_IfRowEnd]>
<[/NFuse_DrawPN]>
```

The actual HTML that creates the hyperlinks for each application occurs between the <[NFuse_IfApp]> and <[/NFuse_IfApp]> tags. This code creates links to the boilerplate.asp file. Included in the URL for each link are the following session fields:

- **NFuse_Template=template.ica.** This session field identifies the template ICA file boilerplate.asp will parse to create a user- and application-specific ICA file.
- **NFuse_Application=[NFuse_AppNameUrlEncoded].** This session field identifies the published application the user has selected.
- **NFuse_MIMEExtension=.ica.** Included to support older browsers that use file extensions for MIME type mapping. This tag does not function as a session field.

When a user selects a hyperlink for an application, the Web server retrieves `boilerplate.asp`, which parses the template file specified in the URL. The following `template.ica` file is the default ICA file template included for this site:

```
<[NFuse_setSessionField NFuse_ContentType=application/x-ica]>

[WFClient]
Version=2
ClientName=[NFuse_ClientName]

[ApplicationServers]
[NFuse_AppName]=

[[NFuse_AppName]]
Address=[NFuse_IPV4Address]
InitialProgram=#[NFuse_AppName]
DesiredColor=[NFuse_WindowColors]
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0

AutologonAllowed=ON
[NFuse_Ticket]

<[/NFuse_IFSESSIONFIELD]>
<[NFuse_IFSESSIONFIELD sessionfield="NFUSE_SOUNDTYPE" value="basic"]>
ClientAudio=On
<[/NFuse_IFSESSIONFIELD]>
[NFuse_IcaWindow]

[NFuse_IcaEncryption]
```

When parsing this template, `boilerplate.asp` replaces all substitution tags (such as `[NFuse_AppName]` and `[NFuse_WindowColors]`) with the current value of their relative session fields. For example, the application links in `applist.htm` contain the session field `NFuse_Application=[NFuse_AppNameUrlEncoded]`. When a user clicks this link, this session field is set to some application name. When `boilerplate.asp` replaces `[NFuse_AppName]` in `template.ica`, it replaces the tag with the session field value set when the user clicked the link.

All window size settings are specified in `template.ica` by the `[NFuse_IcaWindow]` substitution tag. When replacing this tag, `boilerplate.asp` retrieves all current information concerning the ICA Client window size and type and places the information in the ICA file.

The remainder of this exercise describes the modifications to applist.htm and template.ica required to change default window size behavior. Often, modifying an application property is a two step process:

1. Modify the URL in applist.htm that launches the application. To this URL you must append the desired information concerning the application.
2. Modify template.ica so that it calls for the new property information to be placed in the resulting ICA file. In some cases, this step is not necessary because, by default, template.ica includes many application property tags. These tags pick up the current values of various session fields that you can include in your URLs.

The following HTML code shows the modifications to applist.htm that you can perform to cause the application hyperlinks to include window size information:

```
<[NFuse_IfApp]>
  <TD ALIGN="LEFT" VALIGN="TOP">

<!-- this first code block places the application icon and name as an unlinked image and text -->

      <IMG SRC="[NFuse_AppIconUrlEncoded]" border=0 alt="[NFuse_AppFriendlyName]"
      align="LEFT" HSPACE=3 VSPACE=3>
      <B STYLE="font-family:MS Sans Serif,Arial,geneva,Helvetica,sans-serif;font-
      size:9pt">[NFuse_AppFriendlyName]</B><BR>

<!-- this second code block creates a link for the application to run at 640x480 -->

      <A HREF="boilerplate.asp?NFuse_Template=template.ica&
      NFuse_Application=[NFuse_AppNameUrlEncoded]&NFuse_MIMEExtension=.ica&
      NFuse_WindowWidth=640&NFuse_WindowHeight=480"
      TARGET="hiddenwindow">640x480</a><br>

<!-- this third code block creates a link for the application to run at 800x600 -->

      <A HREF="boilerplate.asp?NFuse_Template=template.ica&
      NFuse_Application=[NFuse_AppNameUrlEncoded]&NFuse_MIMEExtension=.ica&
      NFuse_WindowWidth=800&NFuse_WindowHeight=600"
      TARGET="hiddenwindow">800x600</a>
    </TD>
<[/NFuse_IfApp]>
```

The first modification places the application icon and name as an unlinked image and text, whereas before the image and text were hyperlinked.

The second modification takes the hyperlink for the application that originally appeared in `applist.htm` and appends two session fields to the URL. Whereas before, the URL contained only `NFuse_Template`, `NFuse_Application`, and `NFuse_MIMEExtension`, the new link contains the following additional session fields:

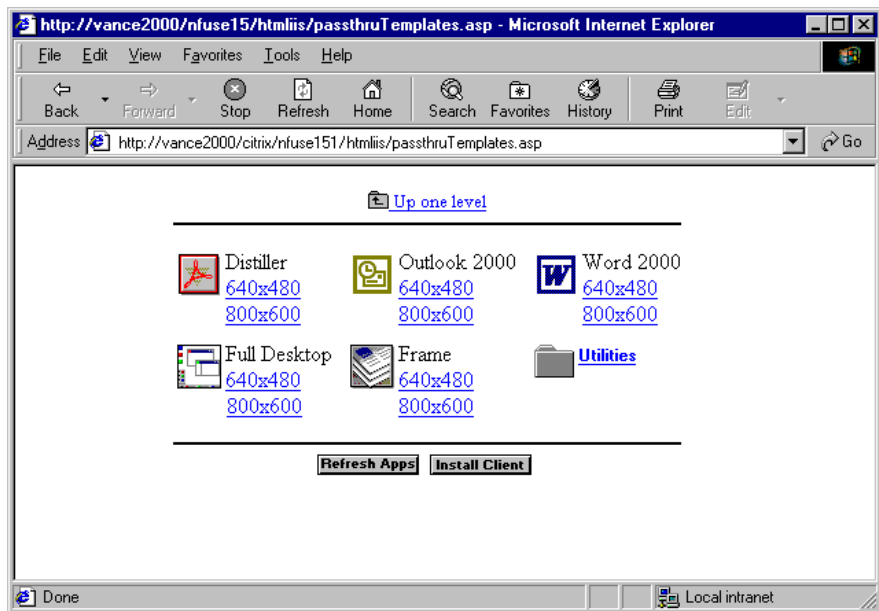
- **NFuse_WindowWidth=640**. This parameter/value pair specifies the `NFuse_WindowWidth` session field with a value of 640 pixels.
- **NFuse_WindowHeight=480**. This parameter/value pair specifies the `NFuse_WindowHeight` session field with a value of 480 pixels.

The third modification is like the second, except it specifies a width and height of 800 and 600 pixels respectively so that users have a second window size option.

When a user clicks either of these links, `boilerplate.asp` retrieves `template.ica` and begins to parse it for substitution tags. Part of the parsing sequence calls for `boilerplate.asp` to search the URL for specified session fields. Placing `NFuse_WindowWidth=x` and `NFuse_WindowHeight=y` in the URL makes this information available to `template.ica`.

Now you must determine if `template.ica` requires modifications so that it calls for the new property information to be placed in the ICA file. In the case of window size information, `template.ica` contains the `[NFuse_IcaWindow]` substitution tag by default. When parsing a page containing this tag, `boilerplate.asp` replaces the tag with the current window size and type. In this example, the current window size is set in session fields in the URL. When replacing the `[NFuse_IcaWindow]` tag, `boilerplate.asp` takes the values specified in the URL and places them in the ICA file to be sent to the user. No modifications of `template.ica` are necessary.

The following graphic depicts applist.htm after the window size modifications described above:



Each link launches the application in the specified window size.

In some cases, you must modify template.ica so that it explicitly calls for the new property information to be included in the ICA file. In such cases, specify the application property as a parameter/value pair composed of a standard ICA file parameter and a value represented by an NFuse substitution tag.

Note Information specified in ICA files must adhere to the ICA file format and supported parameter list. For information about ICA files, see Chapter 7, "ICA File Reference."

Using Session Fields

The use of session fields is controlled by a setting in your NFuse.properties file on the Web server (see “Configuring Web Server Extension Properties” on page 44 for a description of NFuse.properties). By default, the properties file on an Internet Information Server contains the following entries:

```
ConfigurationFileEncoding=8859_1
ConfigurationFilePath=C:\\WINNT\\Java\\TrustLib
NFuse_ContentType=text/html
SessionFieldLocations=Script,Template,Url,Post,Cookie,Properties
Timeout=60
Version=1.51
SessionField.NFuse_CitrixServer=someServerName
SessionField.NFuse_CitrixServerPort=somePortNumber
SessionField.NFuse_IconCache=/NFuseIcons/
SessionField.NFuse_TemplatesDir=C:\\InetPub\\wwwroot\\Citrix\\NFuse151
URLMapping./=C:\\InetPub\\wwwroot
HttpInputEncoding=8859_1
HttpOutputEncoding=8859_1
TemplateFileEncoding=8859_1
CacheExpireTime=3600
SessionField.NFuse_TicketTimeToLive=200
SslKeystore=C:\\WTSRV\\keystore\\cacerts
DTDDirectory=C:\\WINNT\\system32
```

The entry for SessionFieldLocations defines the valid locations for setting session fields. By default, NFuse lets you set session fields from six locations:

- **Script.** Within a Web server script such as an Active Server Page or JavaServer Page.
- **Template.** Within a template HTML or ICA file. A template HTML or ICA file is any text file that contains substitution tags. You use the TemplateParser object to parse such a template.
- **URL.** Within an HTTP URL.
- **Post.** Using the Post method of an HTML form.
- **Cookie.** Within an HTTP cookie.
- **Properties.** Within the NFuse properties file itself.

For information about each of these locations, see “Setting Session Fields” below.

In addition to specifying the valid locations for setting session fields, the `SessionFieldLocations` entry of the properties file also determines a precedence order for session field override. This precedence order lets you set a session field multiple times in your Web site, with the most weighted location in the precedence order taking effect. Precedence is useful when you want to set a default property for a site and then want to override that property at various points in the execution of the site. See “Session Field Precedence” on page 102 for more information.

Setting Session Fields

The following topics provide command syntax for setting session fields from within scripts, HTML or ICA templates, URLs, HTTP cookies, and the `NFuse.properties` file.

Setting Session Fields in a Script

To set a session field from within an Active Server Page or JavaServer Page, you use the `TemplateParser` object's `setSessionFields()` or `setSingleSessionField()` method. For example, the following excerpted Active Server Page sets the `NFuse_TemplatesDir` session field:

```
Parser.setSingleSessionField "NFuse_TemplatesDir",  
Left(Request.ServerVariables("PATH_TRANSLATED"),InStrRev(Request.ServerVariables  
("PATH_TRANSLATED"),"\"))
```

To set a single session field from within an Active Server Page, place the following command in the script: `Parser.setSingleSessionField “X”,Y`

where *Parser* is a previously created `TemplateParser` object, *X* is the name of a session field, and *Y* is the value you want to set the session field to.

When calling this method from a JavaServer Pages script or a servlet, enclose the session field name and value in parentheses:

```
Parser.setSingleSessionField (“X”,Y)
```

To set multiple session fields from within a script file, call `setSingleSessionField()` multiple times or place the following command in the script: `Parser.setSessionFields “X”`

where *Parser* is a previously created `TemplateParser` object, *X* is a string representing the session fields to set, in the form

“Name1=Value1&Name2=Value2&...” or
“&Name1=Value1&Name2=Value2&...”.

Setting Session Fields in a Template

To set a session field from within an NFuse template, you use the `NFuse_setSessionField` command. For example, the first line of the following template ICA file sets an NFuse session field called `NFuse_ContentType`:

```
<[NFuse_setSessionField NFuse_ContentType=application/x-ica]>
<[NFuse_setSessionField NFuse_WindowType=seamless]>

[WFClient]
Version=2
ClientName=[NFuse_ClientName]

[ApplicationServers]
[NFuse_AppName]=

[[NFuse_AppName]]
Address=[NFuse_IPV4Address]
InitialProgram=#[NFuse_AppName]
DesiredColor=[NFuse_WindowColors]
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0
<[NFuse_IFSESSIONFIELD sessionfield="NFUSE_ENCRYPTIONLEVEL" value="basic"]>
Username=[NFuse_User]
Domain=[NFuse_Domain]
Password=[NFuse_PasswordScrambled]
<[/NFuse_IFSESSIONFIELD]>
<[NFuse_IFSESSIONFIELD sessionfield="NFUSE_SOUNDTYPE" value="basic"]>
ClientAudio=On
<[/NFuse_IFSESSIONFIELD]>
[NFuse_IcaWindow]

[NFuse_IcaEncryption]
```

The `NFuse_ContentType` session field specifies the MIME content type that the Web server reports for the response. When parsing a template ICA file for delivery to the client device's Web browser, you must set the content type to `application/x-ica`.

To set a session field from within a template, place the following command in the template file:

```
<[NFuse_setSessionField X=Y]>
```

where *X* is the name of a session field and *Y* is the value you want to set the session field to.

Setting Session Fields in a URL

Setting a session field from a URL is a two step process. First, you must place the session fields you want to set in a URL. The following example URL contains two session fields named NFuse_Template and NFuse_Application:

```
<A HREF="boilerplate.asp?NFuse_Template=template.ica&
NFuse_Application=[NFuse_AppName]">
```

In the URL string, NFuse_Template and NFuse_Application are set to template.ica and [NFuse_AppName] respectively.

The second step involves retrieving the session fields from the URL and then using the TemplateParser object's setURLSessionFields() or setSingleURLSessionField() method to set the session field(s).

When a user clicks the URL, the server retrieves a file such as boilerplate.asp. This file contains the following code:

```
MyUrlSessionFields = Request.ServerVariables("QUERY_STRING")

TemplateParser.setUrlSessionFields(MyUrlSessionFields)
```

The first command uses the Request object's ServerVariables() method to retrieve the URL string from the server and place the string in the variable named MyUrlSessionFields.

The second command uses the TemplateParser object's setUrlSessionFields() method. This method causes the TemplateParser object to parse the URL string. While parsing, the setUrlSessionFields() method sets all session fields specified in the string; in this case, setUrlSessionFields() sets the NFuse_Template and NFuse_Application session fields to the specified values.

Setting Session Fields in a Cookie

Setting a session field in a cookie is a two step process. First, you must place in a cookie the session fields you want to set. The following code places three session fields for user credentials (NFuse_User, NFuse_Domain, and NFuse_Password) and their values (retrieved from the HTML form used to log users into the NFuse site) in a cookie named NFuseData:

```
<%
NFuseCookie="NFuse_User=" + Request.Form("user")
NFuseCookie = NFuseCookie + "&NFuse_Domain=" + Request.Form("domain")
NFuseCookie = NFuseCookie + "&NFuse_Password=" + Request.Form("password")
Response.Cookies("NFuseData") = NFuseCookie
%>
```

The second step involves retrieving the session fields from the cookie and then using the `TemplateParser` object's `setCookieSessionFields()` or `setSingleCookieSessionField()` method to set the session field(s). In the following example, a page uses two script commands to retrieve the user credential session fields and then set them:

```
CookStr = Request.Cookies("NFuseData")
TemplateParser.setCookieSessionFields(CookStr)
```

The first command retrieves the cookie from the server. The second command uses the `setCookieSessionFields()` method to set all session fields contained in the cookie to their specified values.

Setting Session Fields in the Properties File

To set a session field from within the main properties file for NFuse, you specify a session field setting command followed by the name and value of the session field you want to set. For example, the following example `NFuse.properties` file sets multiple session fields:

```
ConfigurationFileEncoding=8859_1
ConfigurationFilePath=C:\\WINNT\\Java\\TrustLib
NFuse_ContentType=text/html
SessionFieldLocations=Script,Template,Url,Post,Cookie,Properties
Timeout=60
Version=1.51
SessionField.NFuse_CitrixServer=someServerName
SessionField.NFuse_CitrixServerPort=somePortNumber
SessionField.NFuse_IconCache=/NFuseIcons/
SessionField.NFuse_TemplatesDir=C:\\InetPub\\wwwroot\\Citrix\\NFuse151
URLMapping./=C:\\InetPub\\wwwroot
```

To set a session field from within the `NFuse.properties` file, place the following command in the parameter list: `SessionField.X=Y`

where *X* is the name of a session field and *Y* is the value you want to set the session field to. For example:

```
SessionField.NFuse_CitrixServer=MyServer
```

You can use the `NFuse.properties` file to set any session field. A session field set in this file acts as a default session field value for all NFuse Web sites you create. You can, of course, override these defaults on a site-by-site or page-by-page basis by using any of the other session field setting methods (script, template, URL, post, and cookie). See the next section, “Session Field Precedence,” for information.

Session Field Precedence

Session field precedence is determined by the order of the values entered for the SessionFieldLocations entry in the NFuse.properties file. By default, the properties file specifies the following:

SessionFieldLocations=Script,Template,Url,Post,Cookie,Properties

Entries to the left in the SessionFieldLocations string take precedence over entries to the right. For example, in an NFuse Web site operating with the above precedence order, if a session field is set in the properties file and in an HTTP cookie, the value of the session field specified in the cookie takes precedence over the value specified in the properties file.

To modify the precedence order, change the ordering of the values in the SessionFieldLocations entry of the properties file.

Session field precedence is useful when you want to change the value of a session field at various points in the execution of your site. For example, by default the properties file sets several session fields including NFuse_CitrixServer and NFuse_CitrixServerPort. These session fields determine the Citrix server used as the communication link between the Web server and Citrix server farm and the TCP/IP port used by that Citrix server for NFuse communication. Because these session fields are set in the properties file, they act as defaults for all NFuse Web sites served by the Web server. When the Web server requires information from the server farm, it checks the current values of the NFuse_CitrixServer and NFuse_CitrixServerPort session fields and makes requests according to their values.

In the case that your implementation requires communication with multiple farms, you must communicate with a Citrix server in each farm. To do this, change the value of the NFuse_CitrixServer and NFuse_CitrixServerPort session fields in the Web site that must communicate with the second farm. For example, you can specify the following in a Web server script:

```
Parser.setSessionField "NFuse_CitrixServer", SomeOtherServer  
Parser.setSessionField "NFuse_CitrixServerPort", SomePort
```

When the Web server executes a site containing the above code, it overrides the values of NFuse_CitrixServer and NFuse_CitrixServerPort specified in the properties file with the values set in the script.

NFuse Java Object Reference



The NFuse objects are Java objects that you can access from Web server scripts or custom-written Java servlets to perform NFuse-related tasks. The NFuse Java objects perform the following services:

- Authenticate users to a Citrix server farm
- Retrieve per-user application sets from a server farm
- Give you the ability to modify the properties of individual applications before presenting them to users
- Parse template HTML and ICA files that display application sets to users and provide them with links to initiate ICA sessions

The Java objects responsible for performing these tasks are:

- **CitrixWireGateway.** Creates a communication link between the Web page requesting a user's application information and the server farm containing that information.
- **ClearTextCredentials.** Encapsulates user authentication information for presentation to the server farm.
- **GroupCredentials.** Contains a list of group names and an associated domain for use in retrieving applications for user groups.
- **AppEnumerator.** Provides an interface for accessing a user's application set.
- **App.** Represents a single application in an application set. App objects contain the properties of an application.
- **AppSettings.** Contains application properties you can modify.
- **AppDataList.** Contains a list of App objects that you can access to quickly determine application set lists for users.

- **AppListCache.** Caches AppDataList objects on a Web server so that you can retrieve application set information without repeatedly contacting the Citrix server farm.
- **TemplateParser.** Performs substitution tag processing on text files. This symbol substitution allows you to create template HTML and ICA files that TemplateParser modifies for each user.

When beginning to work with the NFuse Java objects, create and examine a scripted Web site produced by the Web Site Wizard. The wizard's output includes working examples of Java object usage.

NFuse Java Objects

The following sections include descriptions of NFuse's Java objects. Each section concludes with code examples describing how to create the objects and call their methods.

CitrixWireGateway

A CitrixWireGateway object establishes a communication link between a Web server script and a Citrix server farm. You use a CitrixWireGateway object to create a communication channel through which you can send the server farm a user's credentials and receive in return application information for that user.

CitrixWireGateway objects have the following methods:

void initialize(ClearTextCredentials *credentials*)

void initialize(ClearTextCredentials *credentials*, String *citrixServer*, String *transport*, int *citrixServerPort*)

void initialize(ClearTextCredentials *credentials*, String *citrixServer*, String *transport*, int *citrixServerPort*, String *relayServer*, int *relayServerPort*)

void initialize(ClearTextCredentials *credentials*, String *relayServer*, int *relayServerPort*, String *transport*, int *reserved*)

Initializes the CitrixWireGateway. This method must be called prior to calling GetEnumerator() on the CitrixWireGateway object.

The first version of this method directs the initialization request to the default Citrix server, on the default TCP/IP port, over the default protocol specified in the NFuse.properties file. If the default protocol is not specified in the properties file or if it is set to "HTTP," the method creates an HTTP connection to the default server on the default port. If the protocol specified in the properties file is "SSL," the method determines the SSL Relay server and port specified in the properties file and creates an SSL connection through the SSL Relay server. (Note that by default, the properties file does

not contain entries for SSL Relay server, relay port, or protocol. You can add these values.)

Use the second version of the `Initialize()` method if you want to direct your initialization request to a Citrix server or TCP/IP port that differs from the default server and port specified in the properties file.

Use the third version of the `Initialize()` method if you want to direct your initialization request to a non-default Citrix server through a Citrix SSL Relay server or SSL port that differs from the default SSL Relay server and port specified in the properties file. (Note that by default, the properties file does not contain entries for SSL Relay server, relay port, or protocol. You can add these values.)

Use the fourth version of the `Initialize()` method if you want to direct your initialization request to a default Citrix server through a Citrix SSL Relay server or SSL port that differs from the default SSL Relay server and port specified in the properties file. (Note that by default, the properties file does not contain entries for SSL Relay server, relay port, or protocol. You can add these values.)

Support for the Citrix SSL Relay is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems.

Parameters

credentials: A `ClearTextCredentials` object. A `CitrixWireGateway` object uses these credentials to authenticate the user with the Citrix server maintaining the application information and to filter the applications returned via the `getAppEnumerator()` method (see “`ClearTextCredentials`” on page 108).

citrixServer: The DNS name, NetBIOS name, or IP address of the Citrix server from which to retrieve the application information.

transport: The protocol over which to transport the NFuse data. NFuse supports two protocols: HTTP (Hypertext Transport Protocol) and SSL (Secure Socket Layer). Use HTTP to send the NFuse data over a standard HTTP connection. Specify SSL to send data over a secure connection that uses a Citrix server running the Citrix SSL Relay to perform host authentication and data encryption.

citrixServerPort: The TCP/IP port on which *citrixServer* is listening for NFuse requests.

relayServer: The DNS name, NetBIOS name, or IP address of a Citrix server running the Citrix SSL Relay.

relayServerPort: The port on which *relayServer* is listening for SSL requests.

reserved: Always set to 0.

Return

None

AppEnumerator getAppEnumerator()

Retrieves an AppEnumerator object for a CitrixWireGateway object. You can then use the returned AppEnumerator object to enumerate the App objects retrieved via the CitrixWireGateway object (see “AppEnumerator” on page 113 for information about AppEnumerator objects).

Before calling this method, you must call the initialize() method on the CitrixWireGateway object.

Parameters

None

Return

An AppEnumerator object or null if there was an error in communication with the Citrix server.

App getApp(String *desiredAppName*)

Retrieves a single specific App object from a CitrixWireGateway object. Use this method instead of getAppEnumerator() if you want to retrieve a single application instead of all applications for a specific user.

Before calling this method, you must call the initialize() method on the CitrixWireGateway object.

Parameters

desiredAppName: The name of the published application you want to retrieve.

Return

An App object or null if there was an error in communication with the Citrix server.

AppDataList getAppDataList()

Retrieves an AppDataList object from a CitrixWireGateway object. You can save the AppDataList object in an AppListCache object and reference it in other Web server sessions.

Parameters

None

Return

An AppDataList object or null if there was an error in communication with the Citrix server.

String getLastError()

If the user of the CitrixWireGateway object detects that an error has occurred, getLastError() should be called to return a description of the error.

Parameters

None

Return

A description of the last error on this object.

Example Usage: CitrixWireGateway Object

CitrixWireGateway object creation is a two-step process: first create a CitrixWireGateway object and then call the gateway's initialize() method.

The following Active Server Page/VBScript example illustrates how to create and initialize a CitrixWireGateway object. In this example, myCredentials is a previously created ClearTextCredentials object.

```
'create a CitrixWireGateway object
set myGat = Server.CreateObject("com.citrix.nfuse.CitrixWireGateway")
'initialize the gateway object
myGat.initialize myCredentials
```

The above example initializes the gateway with a ClearTextCredentials object and unspecified citrixServer, transport, and citrixServerPort parameters, thus causing the NFuse Java objects to look in the NFuse.properties file for defaults for the unspecified parameters. The following example illustrates how to override the defaults in NFuse.properties. Once again, myCredentials is a previously created ClearTextCredentials object.

```
'initialize the gateway object so that the Web server
'contacts a Citrix server named ServerX on TCP port 9999
myGat.initialize myCredentials, ServerX, "Http", 9999
```

The next example illustrates how to create a secure connection through a Citrix server running the Citrix SSL Relay. Once again, `myCredentials` is a previously created `ClearTextCredentials` object.

```
'initialize the gateway object so that the Web server  
'contacts a Citrix server named ServerX on TCP port 9999 through an SSL Relay server named  
'ServerR on port 443  
myGat.initialize myCredentials, ServerX, "SSL", 9999, ServerR, 443
```

Note that the above example explicitly specifies the name of the Citrix server running the Citrix SSL Relay and the port on which it is listening for requests. If these parameters are not specified, the `Initialize()` method searches the properties file for default values.

ClearTextCredentials

A `ClearTextCredentials` object is a container that holds an ICA Client user's credentials. User credentials include the user's user name and Windows NT domain in plain text and a password encrypted using basic encryption. You use a `ClearTextCredentials` object to package a user's credentials before sending those credentials via a `CitrixWireGateway` object to the server farm for authentication.

`ClearTextCredentials` objects have the following methods:

`void initialize(String user, String domain, String password)`
`void initialize()`

The first version of this method uses user name, domain, and password information to initialize a `ClearTextCredentials` object. You must call this method before you use the `ClearTextCredentials` object or pass it to another object (such as `CitrixWireGateway`). When authenticating to MetaFrame for UNIX Operating Systems servers, you must include a domain name to use during initialization. See "Authentication and MetaFrame for UNIX Operating Systems" on page 56 for more information.

Use the second version of the `Initialize()` method if you want to initialize a `ClearTextCredentials` object without user name, domain, and password information (also called a null credential authentication request). This form of initialization lets you retrieve all published applications for all users in the server farm.

Parameters

user: The plain-text user name.

domain: The plain-text domain.

password: The plain-text password.

Return

None

String getLastError()

If the user of the ClearTextCredentials object detects that an error has occurred, getLastError() should be called to return a description of the error.

Parameters

None

Return

A description of the last error on this object.

Example Usage: ClearTextCredentials Object

Like CitrixWireGateway object creation, ClearTextCredentials object creation is a two-step process: first create a ClearTextCredentials object and then call the credentials object's initialize() method.

The following Active Server Page/VBScript example illustrates how to create and initialize a ClearTextCredentials object. In this example, user, domain, and password are variables representing previously collected user authentication information.

```
'create a ClearTextCredentials object
Set myC = Server.CreateObject("com.citrix.nfuse.ClearTextCredentials")
'initialize the credentials object
myC.initialize user, domain, password
```

GroupCredentials

A GroupCredentials object is a container that holds a list of group names and an associated domain. You use a GroupCredentials object to retrieve a list of applications for a user group via a CitrixWireGateway object. Note that a GroupCredentials object provides a viewable application list only. You cannot use a GroupCredentials object to authenticate users or groups to a farm.

For an example of GroupCredentials object use, examine the example Web site that filters applications by user group. For more information see, “Application Caching and Filtering by Group” on page 181.

GroupCredentials objects have the following methods:

void setDomain(String *domain*)

Populates a GroupCredential object's domain field.

Parameters

domain: The domain name associated with the set of group names.

Return

None

void addGroupName(String *newGroupName*)

Adds the specified group name to the list of group names associated with the groupCredentials object.

Parameters

newGroupName: The group name to add to the group names list.

Return

None

void setGroupNames(String[] *newGroupNamesArray*)

Populates the GroupCredentials object with a new set of group names contained in an array. This method replaces all previously added group names (if any).

Parameters

newGroupNamesArray: Array containing a list of names to use to populate the GroupCredentials object.

Return

None

void setGroupNames(Vector *newGroupNamesVector*)

Like setGroupNames(), this method populates the GroupCredentials object but uses vectors instead of an array of strings.

Parameters

newGroupNamesVector: Vector containing a list of names to use to populate the GroupCredentials object.

Return

None

String getFirstGroupName()

Returns the first group name associated with the GroupCredentials object. Use to obtain a list of all groupnames associated with the GroupCredentials object. Subsequent call can be getNextGroupName().

Parameters

None

Return

First group name associated with the GroupCredentials object.

String getNextGroupName()

Returns the next available group name associated with the GroupCredentials object. Called after getFirstGroupName(). Returns null if no more group names are found.

Parameters

None

Return

Next group name in the group name list.

int getGroupNamesListSize()

Returns the number of group names associated with the GroupCredentials object.

Parameters

None

Return

The total number of group names associated with the GroupCredentials object.

String getGroupNameAt(int *index*)

Returns a group name at a particular index in the list of group names associated with the GroupCredentials object.

Parameters

index: Position in the group name list.

Return

Group name at the specified index.

void setDomainType(String *domainType*)

Sets the domain type for the GroupCredentials object. Possible types include NT and UNIX.

Parameters

domainType: Type of domain used for group name association.

Return

None

String getDomainType()

Returns the current domain type associated with the GroupCredentials object. Possible types include NT and UNIX.

Parameters

None

Return

Type of domain currently associated with the GroupCredentials object.

String getDomain()

Returns the domain currently associated with the GroupCredentials object.

Parameters

None

Return

Domain currently associated with the GroupCredentials object.

String getLastError()

If the user of the GroupCredentials object detects that an error has occurred, getLastError() should be called to return a description of the error.

Parameters

None

Return

A description of the last error on this object.

Example Usage: GroupCredentials Object

The following Active Server Page/VBScript example illustrates how to create a GroupCredentials object. The example then uses the addGroupName() method to add a group named myGroup to the object.

```
'create a GroupCredentials object
Set myCredentials = Server.CreateObject("com.citrix.nfuse.GroupCredentials")
'add a group name to the GroupCredentials object
myCredentials.addGroupName("myGroup")
```

AppEnumerator

An AppEnumerator object is a class that your Web pages can use to access applications published in a server farm. AppEnumerator returns App objects and Program Neighborhood folders that you can then use to manipulate applications. AppEnumerator is returned by a CitrixWireGateway object (see “CitrixWireGateway” on page 104).

Citrix Management Console and Published Application Manager (the tools used by Citrix administrators to publish applications on MetaFrame XP and 1.8 servers, respectively) organizes applications in a folder hierarchy to give Citrix administrators the ability to present users with applications grouped in folders. When enumerating applications or subfolders for inclusion in your Web pages, you must specify a base folder from which to start the enumeration. Applications can either be enumerated flatly or normally. A *flat* enumeration causes all the applications contained in the base folder and any of its subfolders to be returned in the enumeration. A *normal* enumeration causes only those applications directly contained in the base folder to be returned in the enumeration.

AppEnumerator also has methods for enumerating folders. All folder enumeration methods are normal.

AppEnumerator objects have the following methods:

App nextAppFlat(String *folder*)

Returns the next application in the current flat enumeration. When using this method to cycle through applications, all applications contained in the specified folder and its subfolders are returned.

It is important to note that a new flat app enumeration is started when a folder name is passed to this method or `hasMoreAppsFlat()` that is different than the previous folder name passed to this method or `hasMoreAppsFlat()`.

Parameters

folder: The folder in which to enumerate applications.

Return

An App object for an application contained in the specified folder (or any of its subfolders) or null if there are no more applications to be enumerated in the specified folder or its subfolders.

boolean hasMoreAppsFlat(String *folder*)

Determines if there are more applications to enumerate in the current flat app enumeration. That is, this method reports whether or not there are more applications to enumerate in the specified folder or any of its subfolders.

A new flat enumeration is started when a folder name is passed to this method or `nextAppFlat()` that is different than the previous folder name passed to this method or `nextAppFlat()`.

Parameters

folder: The folder in which to determine if there are more applications to enumerate.

Return

True: There are more application(s) in the specified folder (or any of its subfolders) to enumerate.

False: There are no more applications in the specified folder (or any of its subfolders) to enumerate.

int getNumAppsFlat(String *folder*)

Returns the total number of applications contained in the specified folder and its subfolders.

Does not affect the current flat app enumeration.

Parameters

folder: The folder in which to determine the number of applications.

Return

The total number of applications contained in the specified folder and its subfolders. This number does not include folders contained in the specified folder; for example, if the specified folder contains one application and a subfolder that contains an application, this method returns 2 (the subfolder is not counted as an application).

App nextApp(String *folder*)

Returns the next application in the current normal enumeration. When using this method to cycle through applications, only those applications directly contained in the specified folder are returned.

A new normal enumeration is started when a folder name is passed to this method or `hasMoreApps()` that is different than the previous folder name passed to this method or `hasMoreApps()`.

Parameters

folder: The folder in which to enumerate applications.

Return

An App object for an application contained in the specified folder or null if there are no more applications to enumerate in the specified folder.

boolean hasMoreApps(String *folder*)

Determines if there are more applications to enumerate in the current normal enumeration; that is, this method reports whether or not there are more applications to enumerate in the specified folder (but not any of its subfolders).

A new normal enumeration is started when a folder name is passed to this method or `nextApp()` that is different than the previous folder name passed to this method or `nextApp()`.

Parameters

folder: The folder in which to determine if there are more applications to enumerate.

Return

True: There are more application(s) to enumerate in the specified folder.

False: There are no more applications to enumerate in the specified folder.

int getNumApps(String *folder*)

Returns the number of applications immediately contained in the specified folder.

Does not affect the current normal app enumeration.

Parameters

folder: The folder in which to determine the number of applications.

Return

The number of applications immediately contained in the specified folder. This number includes neither the folders in the specified folder nor applications contained in subfolders of the specified folder; for example, if the specified folder contains one application and a subfolder that contains an application, this method returns 1.

String nextFolder(String *folder*)

Returns the next folder in the current folder enumeration. Only those folders directly contained in the specified folder are returned.

A new folder enumeration is started when a folder name is passed to this method or `hasMoreFolders()` that is different than the previous folder name passed to this method or `hasMoreFolders()`.

Parameters

folder: The folder in which to enumerate subfolders.

Return

A folder immediately contained in the specified folder in the form “aaa\bbb\ccc\...” or null if there are no more folders to enumerate in the specified folder.

String nextFolderUrlEncoded(String *folder*)

Returns the next folder in the current folder enumeration. Returned data is in a URL-encoded string. This format is useful when the returned string must be placed in a URL, cookie, or other location where certain characters, if left unencoded, can cause errors in a Web browser or Web server.

Only those folders directly contained in the specified folder are returned.

A new folder enumeration is started when a folder name is passed to this method or `hasMoreFolders()` that is different than the previous folder name passed to this method or `hasMoreFolders()`.

Parameters

folder: The folder in which to enumerate subfolders.

Return

A folder immediately contained in the specified folder in the form “%5Caaa%5Cbbb%5Cccc%5C...” or null if there are no more folders to enumerate in the specified folder.

boolean hasMoreFolders(String *folder*)

Determines if there are more folders to enumerate in the current folder enumeration; that is, this method reports whether or not there are more subfolders immediately contained in the specified folder.

A new folder enumeration is started when a folder name is passed to this method or `nextFolder()` that is different than the previous folder name passed to this method or `nextFolder()`.

Parameters

folder: The folder in which to enumerate folders.

Return

True: There are more folder(s) to enumerate in the specified folder.

False: There are no more folders to enumerate in the specified folder.

int getNumFolders(String *folder*)

Returns the number of folders immediately contained in the specified folder.

Does not affect the current folder enumeration.

Parameters

folder: The folder in which to determine the number of subfolders.

Return

The number of folders immediately contained in the specified folder. This number does not include applications in the specified folder; for example, if the specified folder contains one application and a subfolder, this method returns 1.

String getLastError()

If the user of the AppEnumerator object detects that an error has occurred, getLastError() can be called to return a description of the error.

Parameters

None

Return

A description of the last error on this object.

Example Usage: AppEnumerator Object

The following Active Server Page/VBScript example illustrates how to create an AppEnumerator object and then how to retrieve a single App object from AppEnumerator. In this example, myGat is a previously created CitrixWireGateway object.

```
'create an AppEnumerator object
Set myEnumerator = myGat.getAppEnumerator
'retrieve the first App object from the AppEnumerator object
Set myApp = myEnumerator.nextAppFlat("")
```

App

An App object is a container that holds the properties of a single application published in a Citrix server farm. You use an App object to access the results of a query of a server farm for information such as a published application's name, the size and color depth of its ICA session window, and its supported encryption level, among other properties set by a Citrix server administrator at the time of application publishing.

An App object gives you query access to both an application's non-settable and settable properties. *Non-settable properties* include properties that define an application (such as the application name) and properties that can be determined only by the publisher of the application (such as if the application is published only on TCP/IP machines or if the application is disabled).

Settable properties are application properties that Web masters can modify in their Web server scripts (see "AppSettings" on page 130 for information about modifying settable properties).

You create App objects by retrieving them from an AppEnumerator object (see "AppEnumerator" on page 113).

App objects have the following methods:

String getFriendlyName()

Retrieves the friendly name (also called Display Name or External Name) of an application published in a Citrix server farm. Friendly names identify applications to Program Neighborhood users.

Use friendly names to display application names to users; for example in an application list page.

This property is non-settable.

Parameters

None

Return

The friendly name of the application.

String getFriendlyNameUrlEncoded()

Retrieves the friendly name (also called Display Name or External Name) of an application published in a Citrix server farm. Return data is in a URL-encoded string. This format is useful when the returned string must be placed in a URL, cookie, or other location where certain characters, if left unencoded, can cause errors in a Web browser or Web server.

This property is non-settable.

Parameters

None

Return

The friendly name of the application in URL-encoded format.

String getName()

Retrieves the internal application name (also called Application Name or Application ID) of an application published in a Citrix server farm. Citrix servers use these application names internally to identify applications. A single internal name cannot be used by more than one application.

Use internal names when identifying an application to run; for example, in an ICA file initial program entry such as InitialProgram=#[NFuse_AppName].

This property is non-settable.

Parameters

None

Return

The internal name of the application.

String getNameUrlEncoded()

Retrieves the internal application name (also called Application Name or Application ID) of an application published in a Citrix server farm. Return data is in a URL-encoded string. This format is useful when the returned string must be placed in a URL, cookie, or other location where certain characters, if left unencoded, can cause errors in a Web browser or Web server.

This property is non-settable.

Parameters

None

Return

The internal name in URL-encoded format.

String getWindowType()

Returns a string representing the type of window in which the referenced application is set to display.

Parameters

None

Return

pixels: Size specified as height and width in pixels

percent: Size specified as percentage of client desktop

seamless: ICA Client window is seamless

fullscreen: ICA Client window is full-screen mode

int getWindowPercentage()

Retrieves the percentage of the client device's desktop that the ICA session window should occupy. You can set this property via an AppSettings object.

Parameters

None

Return

An integer from 0 to 100. If no percentage size is specified for this application, the method returns 0.

int getWindowWidth()

Retrieves the width in pixels of the ICA session window for the referenced application. You can set this property via an AppSettings object.

Parameters

None

Return

A positive integer representing the width in pixels. If no height and width size are specified for this application, the method returns 0.

int getWindowHeight()

Retrieves the height in pixels of the ICA session window for the referenced application. You can set this property via an AppSettings object.

Parameters

None

Return

A positive integer representing the height in pixels. If no height and width size are specified for this application, the method returns 0.

int getColorDepth()

Retrieves an integer representing the number of colors used in the ICA session window to display the referenced application. You can set this property via an AppSettings object.

Parameters

None

Return

1: 16 colors

2: 256 colors

String getSoundType()

Retrieves a string representing the level of audio support for the referenced application. You can set this property via an AppSettings object.

Parameters

None

Return

none: No sound support

basic: Sound 1.0

String getEncryptionLevel()

Retrieves a string representing the level of encryption to use for the referenced application. You can set this property via an AppSettings object. To enable encryption levels higher than Basic, the Citrix server must support RC5 encryption. Support for RC5 encryption is included in MetaFrame XP, Feature Release 1 for MetaFrame 1.8, and SecureICA Services. MetaFrame for UNIX Operating Systems servers do not support RC5 encryption.

Parameters

None

Return

basic: Basic encryption (XOR)

rc5-login: 128-bit for login only

rc5-40: 40-bit

rc5-56: 56-bit

rc5-128: 128-bit

String getVideoType()

Retrieves a string representing the level of video support to use for the referenced application. You can set this property via an AppSettings object. Use of Video 1.0 requires Citrix VideoFrame.

Parameters

None

Return

none: No Video support

basic: Video 1.0

boolean getStartMenu()

Retrieves a boolean representing whether the referenced application has the property set for placing a shortcut in the Windows Start menu of ICA Win32 Client devices. You can set this property via an AppSettings object.

Parameters

None

Return

True: This application has the Start menu shortcut creation property set to on.

False: This application has the Start menu shortcut creation property set to off.

boolean getDesktop()

Retrieves a boolean representing whether the referenced application has the property set for placing a shortcut on the Windows desktops of ICA Win32 Client devices. You can set this property via an AppSettings object.

Parameters

None

Return

True: This application has the desktop shortcut creation property set to on.

False: This application has the desktop shortcut creation property set to off.

String getFolder()

Retrieves the Program Neighborhood folder to which the referenced application belongs. You can set this property via an AppSettings object.

Parameters

None

Return

The folder in the form “\aaa\bbb\...”

String getDescription()

Retrieves the description for the referenced application. You can set this property via an AppSettings object.

Parameters

None

Return

The application's description.

String getIconFile()

Retrieves the URL of the .Gif file for the referenced application. By default, when you use NFuse to access applications, the NFuse Java objects create a .Gif file for each application. NFuse saves these files in a directory on your Web server. You can set this property via an AppSettings object.

Parameters

None

Return

The location of the .Gif file relative to the Web server's root directory. That is, this method returns what can be placed to the right of SRC in the HTML tag `<img SRC="path">`.

String getIconFileUrlEncoded()

Retrieves the URL of the .Gif file for the referenced application. Return data is in a URL-encoded string. This format is useful when the returned string must be placed in a URL, cookie, or other location where certain characters, if left unencoded, can cause errors in a Web browser or Web server.

Parameters

None

Return

A URL-encoded string representing the location of the .Gif file relative to the Web server's root directory. That is, this method returns what can be placed to the right of SRC in the HTML tag `<img SRC="path">`.

void applySettings(AppSettings *newSettings*)

Overrides a current property or properties of an application with new properties set using an AppSettings object. You must use this method to initialize any property changes for an application.

Note that you can use this method to override application properties set at the time of application publishing or properties set via a previous instance of the applySettings method.

See “Example Usage: AppSettings Object” on page 135 for an example illustrating how to use the applySettings() method.

Parameters

newSettings: the new settings to be applied over the application’s existing settings.

Return

None

String getIPv4Address(CitrixWireGateway *gateway*)**String getIPv4Address(CitrixWireGateway *gateway*, String *clientname*)**

Using a passed-in CitrixWireGateway object, this method retrieves the IP address of the Citrix server hosting the published application. If a client name is not specified, the method generates a unique client name based on the credentials contained in the gateway object. The client name is used to identify the client device connecting to the application.

By using this method instead of getName() to place an address in an ICA file, you can eliminate ICA Client-side UDP browsing used in name resolution of published applications.

Parameters

gateway: A CitrixWireGateway object.

clientname: A unique name to identify the ICA Client connecting to the application. Leave unspecified to cause the getIPv4Address() method to generate a client name.

Return

The IP address of the Citrix server hosting the application.

String getIPv4AddressAlternate(CitrixWireGateway *gateway*)**String getIPv4AddressAlternate(CitrixWireGateway *gateway*, String *clientname*)**

Use this method to access a Citrix server across a firewall. Using a passed-in CitrixWireGateway object, this method retrieves the external (or public) IP address of the Citrix server hosting the published application. If a client name is not specified, the method generates a unique client name based on the credentials contained in the gateway object. The client name is used to identify the ICA Client connecting to the application.

By using this method instead of getName() to place an address in an ICA file, you can eliminate ICA Client-side UDP browsing for published application name resolution.

See your Citrix server documentation for information about server-side configuration of ICA connections using alternate addresses.

To use alternate addressing, you must also configure the Citrix server. If your Citrix server is a MetaFrame for Windows server, see your server documentation for information on using the ALTADDR utility. For MetaFrame for UNIX Operating Systems servers, see your server documentation for information on using the CTXALT utility.

Parameters

gateway: A CitrixWireGateway object.

clientname: A unique name to identify the ICA Client connecting to the application. Leave unspecified to cause the getIPv4AddressAlternate() method to generate a client name.

Return

The external IP address of the Citrix server hosting the application.

String generateClientName(Credentials *credentials*)

Use this method to generate a unique client name that you can use to identify the ICA Client connecting to the application. This method uses the same client-name-generating algorithm as getIPv4Address(*gateway*) and getIPv4AddressAlternate(*gateway*).

Parameters

credentials: A ClearTextCredentials object.

Return

A unique name to identify the ICA Client connecting to the application.

String getTicket(CitrixWireGateway wireGateway, ClearTextCredentials credentials, String ticketType)

This method retrieves an authentication ticket for an application. Before calling this method, you must call either the `getIPv4Address()` or `getIPv4AddressAlternate()` method on the App object.

Support for ticketing is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems.

Parameters

wireGateway: The CitrixWireGateway object from which to retrieve the ticket.

credentials: A ClearTextCredentials object. A CitrixWireGateway object uses these credentials to authenticate the user during ticket retrieval.

ticketType: Identifies the type of ticket to retrieve. Must be "CtxLogon."

Return

A 30 character ticket string.

String getTicketUpper(CitrixWireGateway wireGateway, ClearTextCredentials credentials, String ticketType)

Retrieves first 14 characters of an authentication ticket. The first 14 characters of a ticket correspond to information you can place in an ICA file as the value for a ClearPassword parameter. See the description of [NFuse_TicketUpper] in "General Tags" on page 74.

Before calling this method, you must call either the `getIPv4Address()` or `getIPv4AddressAlternate()` method on the App object.

Parameters

wireGateway: The CitrixWireGateway object from which to retrieve the ticket.

credentials: A ClearTextCredentials object. A CitrixWireGateway object uses these credentials to authenticate the user during ticket retrieval.

ticketType: Identifies the type of ticket to retrieve. Must be "CtxLogon."

Return

A string composed of the first 14 characters of the ticket; for example, 7456C5E8F56EBE.

**String getTicketLower(CitrixWireGateway wireGateway,
ClearTextCredentials credentials, String ticketType)**

Retrieves the last 16 characters of an authentication ticket preceded by a backslash. The last 16 characters of a ticket correspond to information you can place in an ICA file as the value for a domain parameter. See the description of [NFuse_TicketLower] in “General Tags” on page 74.

Before calling this method, you must call either the `getIPv4Address()` or `getIPv4AddressAlternate()` method on the App object.

Parameters

wireGateway: The CitrixWireGateway object from which to retrieve ticket.

credentials: A ClearTextCredentials object. A CitrixWireGateway object uses these credentials to authenticate the user during ticket retrieval.

ticketType: Identifies the type of ticket to retrieve. Must be “CtxLogon.”

Return

A string composed of “\” followed by the last 16 characters of the ticket; for example, \9AC643FBAA919ADC.

**void setTicketTimeToLive(CitrixWireGateway wireGateway, int
ticketTimeToLive)**

This method sets the duration for which the ticket is valid. When this time period passes or the ticket is used, the ticket is invalid. Specify values in seconds.

Parameters

wireGateway: The CitrixWireGateway object to use to set the timeout period. Must be same object used to retrieve the ticket.

ticketTimeToLive: Amount of time for which the ticket is valid.

Return

None

boolean equals(Object app)

Determines whether the passed-in object is equal to this App object. Equivalence means the two App objects represent the same published application.

Parameters

app: A Java object.

Return

True: The passed-in object is equivalent to this App object.

False: The passed-in object is not equivalent to this App object.

String getLastError

If the user of an App object detects that an error has occurred, `getLastError()` can be called to return a description of the error.

Parameters

None

Return

A description of the last error on this object.

Example Usage: App Object

To retrieve a property of an application, you must first create an App object. Next, call one of App's methods for the property you want to query.

The following Active Server Page/VBScript example illustrates how to query the Web server for the URL of an application's icon (.Gif file) stored on the server. Next, the example describes how to query a server farm for the application's description. The example code writes both of these pieces of information to a Web page.

In this example, `myEnumerator` is a previously created `AppEnumerator` object.

```
'create an App object
set myApp = myEnumerator.nextAppFlat("")
'write an HTML IMG tag that specifies the URL of the icon
'file on the Web server
Response.Write "<img src="" & myApp.getIconfile & "">"
'write the application's description in the Web page
Response.Write myApp.getDescription & "<br>"
```

AppSettings

An AppSettings object is a container that holds the settable properties of a single application published in a Citrix server farm. Unlike an App object, which allows you to query application properties, an AppSettings object lets you modify application properties. At any given time, an AppSettings object can have none, some, or all of the settable application properties specified for an application.

Note A single AppSettings object can be applied to multiple App objects. Conversely, multiple App objects can draw from the settings specified in a single AppSettings object.

In addition to modifying an application's current properties, you can use an AppSettings object to handle application properties retrieved from a number of sources including an ODBC database or an "application settings" Web page. An AppSettings object could exist for each of these sources and be applied in turn to a given App object.

For the settings specified in an AppSettings object to actually be applied to an application, you must call App.applySettings on an App object for that application (see the description of the App object's applySettings() method on page 125 for more information).

AppSettings objects have the following methods:

void setWindowType(String *windowType*)

Sets the window type of the ICA session window for the referenced application.

Parameters

windowType

pixels: A size in pixels (when specifying size in pixels, you must also call the setWindowPixels() method)

percent: A size as a percentage of the client desktop (when specifying size as a percentage, you must also call the setWindowPercentage() method)

seamless: Seamless window

fullscreen: Full screen

Return

None

void setWindowPixels(int *windowWidth*, int *windowHeight*)

Sets the width and height of the ICA session window for the referenced application. Specify values in pixels.

Parameters

windowWidth: The width of the client window in pixels. Value must be an integer.

windowHeight: The height of the client window in pixels. Value must be an integer.

Return

None

void setWindowPercentage(int *windowScale*)

Specifies what percentage of the client desktop the ICA session should occupy.

Parameters

windowScale: The percentage of the client desktop. The value must be an integer between 0 and 100.

Return

None

void setColorDepth(int *colorDepth*)

Specifies the number of colors used in the ICA session window to display the referenced application.

Parameters

colorDepth

1: 16 colors

2: 256 colors

Return

None

void setSoundType(String *soundType*)

Specifies a level of sound support to apply to the referenced application.

Parameters

soundType

none: No sound support

basic: Sound 1.0

Return

None

void setEncryptionLevel(String *encryptionLevel*)

Specifies a level of encryption to use for the referenced application. To enable encryption levels higher than Basic, the Citrix server must support RC5 encryption. Support for RC5 encryption is included in MetaFrame XP, Feature Release 1 for MetaFrame 1.8, and SecureICA Services. MetaFrame for UNIX Operating Systems servers do not support RC5 encryption.

Parameters

encryptionLevel

basic: Basic encryption (XOR)

rc5-login: 128-bit for login only

rc5-40: 40-bit

rc5-56: 56-bit

rc5-128: 128-bit

Return

None

void setVideoType(String *videoType*)

Specifies a level of video support to use for the referenced application. Use of Video 1.0 requires Citrix VideoFrame.

Parameters

videoType

none: No video

basic: Video 1.0

Return

None

void setStartmenu(boolean *place*)

Allows you to set the Windows Start menu shortcut creation property to **On** or **Off** for the referenced application. Note that this method does not allow you to actually place shortcuts on client devices.

Parameters

place

True: Set the referenced application's Start menu shortcut creation property to on.

False: Set the referenced application's Start menu shortcut creation property to off.

Return

None

void setDesktop(boolean *place*)

Allows you to set the Windows desktop shortcut creation property to **On** or **Off** for the referenced application. Note that this method does not allow you to actually place shortcuts on client devices.

Parameters

place

True: Set the referenced application's desktop shortcut creation property to on.

False: Set the referenced application's desktop shortcut creation property to off.

Return

None

void setDescription(String *description*)

Specifies a description string for the referenced application.

Parameters

description

The description string you want to appear on the Web page.

Return

None

void setFolder(String *folder*)

Specifies the folder in which the referenced application should be displayed.

Parameters

folder

The folder in which you want the application to appear.

Return

None

void setIconFile(String *iconFile*)

Specifies the URL of an icon file (Gif) for the referenced application. Use this method to override the referenced application's default icon file URL. See the description of the App object's getIconFile method on page 124 for more information about icon files.

Parameters

iconFile

The location of the icon file (Gif) file relative to the Web server's root directory. When using this method, specify a string that can be placed to the right of SRC in the HTML tag .

Return

None

String getLastError()

If the user of an AppSettings object detects that an error has occurred, getLastError() can be called to return a description of the error.

Parameters

None

Return

A description of the last error on this object.

Example Usage: AppSettings Object

To override an application's settings, you must first create an AppSettings object. Next, call one of AppSetting's methods for the property you want to change and specify as its parameter the new value you want to apply to the application. To apply the new setting, you must call the App object's applySettings method with the AppSettings object specified as the parameter.

The following Active Server Page/VBScript example illustrates overriding an application's current description. In this example, myApp is a previously created App object.

```
'create an AppSettings object
Set NewSettings = Server.CreateObject("com.citrix.nfuse.AppSettings")
'call the setDescription method on the AppSettings object
NewSettings.setDescription("Application Description")
'call the App object's applySettings method with
'the parameter NewSettings
myApp.applySettings(NewSettings)
```

AppDataList

An AppDataList object contains a list of App objects. AppDataList objects are returned by a CitrixWireGateway object. You can save the AppDataList object in an AppListCache object for referencing by other sessions on the Web server.

You can use an AppDataList object to cache application set lists on the NFuse Web server. When users log into the server, you can retrieve their application set information from the AppDataList quickly instead of querying the Citrix server repeatedly for information that may change infrequently.

AppDataList objects have the following methods:

AppEnumerator getAppEnumerator()

Retrieves an AppEnumerator object for an AppDataList object. You can then use the returned AppEnumerator object to enumerate the App objects in the AppDataList object.

Parameters

None

Return

An AppEnumerator object or null if there was an error in communication with the Citrix server.

boolean isExpired()

Determines whether or not the AppDataList object is expired. The AppDataList timeout value (in seconds) is specified in the CacheExpireTime entry in NFuse.properties. You can call setExpireTime() to set the timeout value for the AppDataList object.

Parameters

None

Return

True: The AppDataList object has expired.

False: The AppDataList object has not expired.

void addAppList(AppDataList apps)

Combines multiple AppDataList objects into a single object. It is recommended that only AppDataList objects from the same server farm be merged.

Parameters

apps: An AppDataList object to be merged into this AppDataList object.

Return

None

void setExpireTime(long t)

Specifies the timeout value for the AppDataList object. After this time period the object is no longer valid.

Parameters

t: Timeout value in milliseconds for this AppDataList object. Specify a negative number to cause the object to never expire.

Return

None

long getCreateDate()

Returns the creation date of the object in milliseconds since January 1, 1970, 00:00:00 GMT.

Parameters

None

Return

The creation date of the object in milliseconds.

long getExpireTime()

Returns the AppDataList object's expiration timeout value in milliseconds.

Parameters

None

Return

The timeout value for this AppDataList object.

Example Usage: AppDataList Object

The following Active Server Page/VBScript example illustrates how to create an AppDataList object. The example then uses the `isExpired()` method to determine if the AppDataList object is expired. In this example, `myGateway` is a previously created CitrixWireGateway object.

```
'create an AppDataList object
Set myApps = myGateway.getAppDataList()
'determine if the object is expired
expired = myApps.isExpired()
```

AppListCache

An AppListCache object caches AppDataList objects on your Web server so that you can reference them later to quickly retrieve application set information without contacting the server farm. To give all Web server sessions access to an AppListCache object, create the AppListCache object in the Web server's application scope. The AppListCache object saves data as key/AppDataList pairs. Each AppDataList object must have a unique name.

AppListCache objects have the following methods:

void addToCache(String *key*, Object *object*)

Adds an AppDataList object to the AppListCache object.

Parameters

key: The key by which to identify the AppDataList object.

object: The name of the AppDataList object to add.

Return

None

AppDataList removeFromCache(String key)

Removes an AppDataList object from the AppListCache object.

Parameters

key: The key associated with the AppDataList object to remove.

Return

This method returns the object to be removed. It returns null if the object could not be found.

boolean contain(String key)

Determines whether or not the AppListCache object contains a specific AppDataList object.

Parameters

key: The key associated with a specific AppDataList object.

Return

True: The AppListCache object contains the specified AppDataList object.

False: The AppListCache object does not contain the specified AppDataList object.

AppDataList retrieveFromCache(String key)

Retrieves the specified AppDataList object from the AppListCache object.

Parameters

key: The key associated with the AppDataList object to retrieve.

Return

The specified AppDataList object or null if the object can not be found.

int size()

Retrieves the number of AppDataList objects stored in the AppListCache object.

Parameters

None

Return

An integer representing the number of AppDataList objects contained in the AppListCache object.

void compactCache()

Removes expired AppDataList objects from this AppListCache object.

Parameters

None

Return

None

void setOptimalNumber(int *totalNumber*)

Specifies the optimal number of objects to store in the AppListCache object. The addToCache() method calls compactCache() automatically if the total number of objects stored in this AppListCache object has reached this number.

The AppListCache object does not enforce the maximum number of objects that can be stored. It is possible to have more than this number of objects stored even after compactCache() is called.

Parameters

totalNumber: The optimal number of objects to store in this AppListCache object. When an AppListCache object is created, this optimal number is set to 100 by default.

Return

None

Example Usage: AppListCache Object

The following Active Server Page/VBScript example illustrates how to create an AppListCache object. The example then uses the size() method to determine the number of AppDataList objects the AppListCache object contains.

```
'create an AppListCache object
Set myCitrixCache = Server.CreateObject("com.citrix.nfuse.AppListCache")
'determine if the object is expired
size = myCitrixCache.size()
```

TemplateParser

A TemplateParser object performs symbol substitution on a template text file. During processing, a TemplateParser object searches for and replaces Citrix substitution tags. One use of a TemplateParser object involves calling the object from a Web interface to take a template ICA file and replace all the application-specific parts of the file with the appropriate properties of an App object.

For a list of Citrix substitution tags, see Chapter 5, “Using NFuse Tags” on page 69.

TemplateParser objects have the following methods:

void setCookieSessionFields(String *cookie*)

Sets the session fields of the TemplateParser object to the values specified in the passed-in cookie string. Whether a given session field is set or not depends on its current state and the override order specified in the SessionFieldLocations entry in the configuration file NFuse.properties (see “Using Session Fields” on page 97 for more information).

Parameters

cookie: A string representing the cookies passed to the Web server in the current HTTP request, in the form “Name1=Value1&Name2=Value2&...” or “&Name1=Value1&Name2=Value2&...”

Return

None

void setSingleCookieSessionField(String *sessionField*, String *value*)

Sets a single session field of the TemplateParser object to the value specified. Whether a given session field is set or not depends on its current state and the override order specified in the SessionFieldLocations entry in the configuration file NFuse.properties (see “Using Session Fields” on page 97 for more information).

Parameters

sessionField: The name of the session field to set.

value: The value to set the session field to.

Return

None

void setUrlSessionFields(String url)

Sets the session fields of the TemplateParser object to the values specified in the passed-in URL query string. Whether a given session field is set or not depends on its current state and the override order specified in the SessionFieldLocations entry in the configuration file NFuse.properties (see “Using Session Fields” on page 97 for more information).

Parameters

url: A string representing the URL query string received by the Web server in the current HTTP request, in the form “Name1=Value1&Name2=Value2&...” or “&Name1=Value1&Name2=Value2&...”

Return

None

void setSingleUrlSessionField(String sessionField, String value)

Sets a single session field of the TemplateParser object to the value specified. Whether a given session field is set or not depends on its current state and the override order specified in the SessionFieldLocations entry in the configuration file NFuse.properties (see “Using Session Fields” on page 97 for more information).

Parameters

sessionField: The name of the session field to set.

value: The value to set the session field to.

Return

None

void setPostSessionFields(String post)

Sets the session fields of the TemplateParser object to the values specified in the passed-in HTTP Post. Whether a given session field is set or not depends on its current state and the override order specified in the SessionFieldLocations entry in the configuration file NFuse.properties (see “Using Session Fields” on page 97 for more information).

Parameters

post: A string representing the Post received by the Web server in the current HTTP request, in the form “Name1=Value1&Name2=Value2&...” or “&Name1=Value1&Name2=Value2&...”

Return

None

void setSinglePostSessionField(String *sessionField*, String *value*)

Sets a single session field of the TemplateParser object to the value specified in the passed-in HTTP Post. Whether a given session field is set or not depends on its current state and the override order specified in the SessionFieldLocations entry in the configuration file NFuse.properties (see “Using Session Fields” on page 97 for more information).

Parameters

sessionField: The name of the session field to set.

value: The value to set the session field to.

Return

None

void setSessionFields(String *input*)

Sets the values of session fields from a Web server (script file) interface. Whether a given session field is set or not depends on its current state and the override order specified in the SessionFieldLocations entry in the configuration file NFuse.properties (see “Using Session Fields” on page 97 for more information).

Parameters

input: A string representing the session fields to set, in the form “Name1=Value1&Name2=Value2&...” or “&Name1=Value1&Name2=Value2&...”

Return

None

void setSingleSessionField(String *sessionField*, String *value*)

Sets the value of a single session field from a Web server (script file) interface. Whether the session field is actually set or not depends on its current state and the override order specified in the SessionFieldLocations entry in the configuration file NFuse.properties (see “Using Session Fields” on page 97 for more information).

Parameters

sessionField: The name of the session field to set.

value: The value to set the session field to.

Return

None

boolean Parse()

Parses a template text file. The template to parse must be specified in a session field (NFuse_Template). The result of the parsing (with the substitution tags processed and replaced) is accessed via the getNextDataBlock() method.

Parameters

None

Return

True: The parsing succeeded. You can now call getNextDataBlock() to get the result of the parse.

False: The parsing failed. Do not call getNextDataBlock(). Call getLastError() to get the error.

String getNextDataBlock()

Returns a portion of the parsed template. To fully retrieve the parsed template, call this method repeatedly until it returns the empty string.

Parameters

None

Return

A string of some maximum length. If there is no more parsed template to return, this method returns the empty string ("").

String getContentType()

Returns the MIME "Content-Type" for the result of this template parse. For instance, when parsing a template for an ICA file, this might be "application/x-ica."

Parameters

None

Return

The MIME type.

String getLastError()

If the user of the TemplateParser object detects that an error has occurred, getLastError() can be called to return a description of the error.

Parameters

None

Return

A description of the last error on this object.

Example Usage: TemplateParser Object

The following Active Server Page/VBScript example illustrates how to create a TemplateParser object. The example then retrieves a cookie containing parameter/value pairs, where the parameters are written in the syntax of NFuse session fields. Finally, the example sets the session fields retrieved from the cookie so that they are available to the TemplateParser as it parses an NFuse template. In this example, NFuseData is a previously created cookie.

```
'create a TemplateParser object
Set myParser = Server.CreateObject("com.citrix.nfuse.TemplateParser")
'retrieve a cookie containing session field/value pairs
CookStr = Request.Cookies("NFuseData")
'set the session fields contained in the cookie
myParser.setCookieSessionFields(CookStr)
```

ICA File Reference



The template ICA files included in NFuse allow you to use a single template as the basis for delivering a customized ICA file to each user for each requested application. The presence of substitution tags in the template file makes customization possible. When creating an ICA file from the template, the NFuse Java objects replace each substitution tag with the corresponding information about the user or desired application.

In addition to using NFuse's substitution tags to customize a template file, you can also directly edit a template by hard coding ICA file parameters. For example, if you have a group of users who always require ICA sessions of a certain resolution, you can use window size parameters to hard code that resolution into the template ICA file instead of using a substitution tag to query the Citrix server for window size information.

Hard coded ICA file parameters also give you access to various ICA session properties not configurable using substitution tags or standard application publishing. For example, you can use client device mapping parameters to enable or disable client devices such as printers and COM ports.

This chapter contains the following:

- Information about the structure and contents of ICA files
- Information about using ICA files with firewalls

ICA File Structure

An *ICA file* is a text file containing information about a published application. ICA files are written in Ini file format and organize published application information in a standardized way that ICA Clients can interpret. The following example depicts basic ICA file layout:

```
<[NFuse_setSessionField NFuse_ContentType=application/x-ica]>
[WFClient]
Version=2

[ApplicationServers]
ApplicationName=

[ApplicationName]
Parameter1=Value
Parameter2=Value
Parameter3=Value
```

[WFClient]

The [WFClient] section is the first section in an ICA file and must contain at least the parameter/value pair Version=2. The version number is for Citrix internal use and should not be modified.

In addition to the version parameter, the template ICA files created by the Web Site Wizard specify a second parameter/value pair. NFuse uses this pair (ClientName=[NFuse_ClientName]) to place a generated, unique ICA Client name in the ICA file. This client name identifies the ICA Client connecting to the published application and is a requirement of various ICA Client functions including printing and session reconnect.

The template ICA files created by the Web Site Wizard contain at least a single entry previous to the [WFClient] section. This first entry is a session field setting command used by NFuse to set the MIME type of the ICA file to application/x-ica. This entry is required for NFuse, but is not a requirement in standard ICA files.

[ApplicationServers]

The [ApplicationServers] section contains a single parameter. This parameter specifies the name of a Citrix published application. Following the published application name is an equals sign (=).

For example, in an ICA file for a published application named “Notepad,” the [ApplicationServers] section contains the following:

```
Notepad=
```

In an NFuse template ICA file, you use a substitution tag to enter the published application name in the [ApplicationServers] section. For example:

```
[NFuse_AppName]=
```

[ApplicationName]

The final required section in an ICA file is [*ApplicationName*], where *ApplicationName* is the name of the published application specified in the [ApplicationServers] section. In an NFuse template ICA file, the published application name can be specified using the substitution tag [NFuse_AppName]; for example, [[NFuse_AppName]].

The [*ApplicationName*] section contains configuration information for the specified published application. This information is in the form of a parameter/value pair list. The following topics describe some parameters you can use to customize your template ICA files.

ICA File Parameters

The following topics list parameters that configure:

- General ICA session properties
- User credentials
- Window size and number of colors
- Persistent bitmap caching
- ICA Client TCP/IP browsing
- Encryption
- SOCKS proxy servers

- SpeedScreen Latency Reduction
- Client Auto Update
- Client device mapping

Important Not all parameters are supported by all ICA Clients or all Citrix server products.

General Parameters

The following parameters configure basic ICA session properties.

InitialProgram	The name of the published application prefixed with a pound sign (#). For example, for a published application named “Notepad,” specify: InitialProgram=#Notepad
WorkDirectory	Specifies a working directory for the application to use.
TransportDriver	Transport protocol used to connect to the Citrix server. Always set to TCP/IP.
WinStationDriver	Version of the ICA protocol to use for the connection. Always set to ICA 3.0.
MouseTimer	Specifies a time interval in milliseconds during which mouse input is collected before being sent to the Citrix server. The default value of 100 milliseconds is optimized for WANs. In a Dial-In or LAN environment, reducing this value can give better responsiveness. Using too low a value in a LAN environment can generate a large number of small packets, which can affect network performance.
KeyboardTimer	Specifies a time interval in milliseconds during which keyboard input is collected before being sent to the Citrix server. The default value of 100 milliseconds is optimized for WANs. In a Dial-In or LAN environment, reducing this value can give better responsiveness. Using too low a value in a LAN environment can generate a large number of small packets, which can affect network performance.
SwapButtons	Specifies whether or not to switch the function of the client device’s left and right mouse buttons. Specify Yes to switch button functions.

WindowsCache	Specifies the size of the ICA Client's Thinwire memory cache. The default for the ICA Win32 Client is 3.5MB.
ICAPortNumber	By default, Citrix servers and ICA Clients use TCP/IP port 1494 to pass ICA traffic. Add this parameter to cause the ICA Client to use some other TCP/IP port. To use this parameter, you must also configure the Citrix server to use a non-default port. If your Citrix server is a MetaFrame for Windows server, see your server documentation for information on using the ICAPORT utility. For MetaFrame for UNIX servers, see your server documentation for information on using the CTXCFG utility. If your ICA Client does not support the ICAPORT parameter, you can specify the desired port by appending <i>:port#</i> to the ICA file's address parameter. For example, to use port 80: Address=[NFuse_IPv4Address]:80

User Credential Parameters

User credential parameters identify the user attempting to connect to the published application.

Username	A user name supported by your Citrix server's account authority. For example, a Windows NT user name if your Citrix server is a Windows Terminal Server or Windows 2000 Server Family system.
Domain	A Windows NT domain name.
Password	A valid password for the specified user account. ICA Clients expect password information entered for the Password parameter to be in scrambled format. Use the [NFuse_PasswordScrambled] substitution tag to enter passwords in scrambled format.
ClearPassword	Used to specify a password in clear text. In some cases, to use a clear text password, the Password field must also be included in the ICA file and set to a null value. For example: Password=

AutoLogonAllowed

By default, ICA connections that use encryption levels greater than Basic do not accept the user name, domain, and password information specified in an ICA file. Such connections force users to manually log in to each application they connect to, regardless of whether the ICA file already contains their credentials. Use this parameter to force the server to accept user credentials specified in an ICA file.

For an example of this tag's usage, see "Encryption Parameters" on page 155.

Specify On to allow automatic logon.

Window Size and Color Parameters

Window size and color parameters control display properties of the ICA session window.

DesiredColor

Number of colors used to display the ICA session window.

1=16 colors

2=256 colors

4=High Color (16-bit)

8=True Color (24-bit)

DesiredHRES

Specifies the width of the ICA session window in pixels. For example, 640.

DesiredVRES

Specifies the height of the ICA session window in pixels. For example, 480.

ScreenPercent

Specifies the horizontal and vertical pixel resolution as a percentage of the client desktop. If the ScreenPercent field is present, DesiredHRES and DesiredVRES fields are ignored.

TWIMode

If supported, this parameter causes the ICA Client to create a seamless ICA session window. Seamless session windows place the published application in a resizable window. Specify On for a seamless connection. In some cases, when enabling a seamless connection you must additionally specify the following two parameter/value pairs:

DesiredHRES=0xffffffff

DesiredVRES=0xffffffff

DesiredWinType	Specifies desired window type for a custom Program Neighborhood connection. 1=640x480 2=800x600 3=1024x768 4=1280x1024 5=Custom 6=Percent 7=Full Screen 8=Seamless 0=None. The connection inherits the default setting from the ICA Client.
----------------	--

Persistent Caching Parameters

Persistent caching parameters control the storing of commonly-used graphical objects such as bitmaps in a local cache on the client device's hard disk.

PersistentCacheEnabled	Enables and disables persistent bitmap caching. Specify On to enable caching.
PersistentCacheSize	Specifies the amount of disk space in bytes to use for bitmap caching.
PersistentCacheMinBitmap	Specifies the smallest bitmap in bytes that can be cached to disk.
PersistentCachePath	Specifies the location of the directory containing the cached image data. For example: D:\WTSRV\Profiles\User1\Application Data\ICAClient\Cache

TCP/IP Browsing Parameters

TCP/IP browsing parameters configure ICA Client server location. Server location provides a method for ICA Clients to resolve published application names into Citrix server IP addresses.

TcpBrowserAddress	Specifies the IP address of a Citrix server used for server location and published application name resolution. Specify up to 15 TCP browser addresses by entering: TcpBrowserAddress2=x.x.x.x TcpBrowserAddress3=x.x.x.x
BrowserTimeout	Specifies the number of milliseconds the ICA Client waits for a response after making a request to the master browser. The master browser request is an initial step required by server location and published application name resolution. This setting is useful in environments where the ICA Client's master browser request must pass through various impediments to quick response such as a WAN connection.
BrowserRetry	Specifies the number of times an ICA Client resubmits a master browser request that has timed out. The master browser request is an initial step required by server location and published application name resolution.
UseAlternateAddress	Defines whether to use a server's alternate address for ICA connectivity across a firewall or a router. Specify 1 to cause the ICA Client to use the Citrix server's alternate address. To use this parameter, you must also configure the Citrix server. If your Citrix server is a MetaFrame for Windows server, see your server documentation for information on using the ALTADDR utility. For MetaFrame for UNIX servers, see your server documentation for information on using the CTXALT utility.

NFuse's Server Location Options

NFuse's substitution tags allow you several methods of specifying a published application name/address in a template ICA file. The method you use depends upon where in the NFuse system you want NFuse to perform Citrix server location.

The following three options explain which substitution tag to use in your template ICA files to correctly resolve published application names:

- **Web-server-side server location.** Some network configurations using routers and multiple subnets may require you to specify the published application name as the IP address of the Citrix server hosting the application. Specifying the name as an address in the ICA file forces the Java objects on the Web server to resolve the published application name instead of relying upon the ICA Client to perform the translation. The NFuse Java objects, unlike the ICA Clients, do not use an initial UDP broadcast to locate a Citrix server and therefore eliminate the multiple subnet complications of ICA-Client-to-Citrix-server UDP broadcasting. (The Citrix-server-to-Citrix-server communication that occurs after the initial server location is, however, UDP-based.)

To perform Web-server-side server location, specify [NFuse_IPv4Address] as the published application address in your template ICA files. For example:

```
[ApplicationServers]
[NFuse_AppName]=

[[NFuse_AppName]]
Address=[NFuse_IPv4Address]
```

When using the [NFuse_IPv4Address] tag, it is recommended that your ICA file be configured for client name. Please see the end of this topic for more information. The [NFuse_IPv4Address] tag is the default entry used in the NFuse template ICA files.

- **ICA-Client-side server location.** This method causes the ICA Client to perform published application name resolution in the typical UDP-based fashion of ICA Clients. This method is useful if you want to place the address resolution load on individual ICA Clients instead of on the Java objects on the Web server.

To perform ICA-Client-side server location, specify [NFuse_AppName] as the published application address in your template ICA files. For example:

```
[ApplicationServers]
[NFuse_AppName]=

[[NFuse_AppName]]
Address=[NFuse_AppName]
```

In a multiple subnet environment, in which the ICA Client and Citrix server farm are on different subnets, you must also specify a TCPBrowserAddress entry so that the ICA Client can locate the master ICA Browser on the other subnet. For example:

```
[ApplicationServers]
[NFuse_AppName]=

[[NFuse_AppName]]
Address=[NFuse_AppName]
TCPBrowserAddress=x.x.x.x.
```

- **Server location through firewalls.** To perform published application name resolution across a firewall that uses network address translation, use the [NFuse_IPv4AddressAlternate] substitution tag to specify the published application name. This tag identifies the published application with the external server address of the Citrix server hosting the application. For example:

```
[ApplicationServers]
[NFuse_AppName]=

[[NFuse_AppName]]
Address=[NFuse_IPv4AddressAlternate]
```

Like Web-server-side server location, this method performs name resolution on the Web server. To use this method, your Citrix server must be configured using the ALTADDR (MetaFrame for Windows) or CTXALT (MetaFrame for UNIX) utility. See your server documentation for more information.

When performing Web-server-side server location and server location through firewalls, it is recommended that your template ICA file's [WFClient] section contain the following parameter/value pair: Clientname=[NFuse_ClientName]. Placing this pair in the ICA file causes the client name used by the Java objects during resolution and the client name used by the ICA Client for the ICA session

to be identical. This method works for all supported ICA Clients except the ICA Java Client.

When using the ICA Java Client to embed ICA sessions in a Web page, you must use the client name parameters of the Java Client's Applet tag to include client name information for the ICA session. For example, to use the Java Client's client name parameters in a substitution-tag-based site, the Applet tag must include the following:

```
<param name=client.wfclient.usehostname value='no'>  
<param name=client.wfclient.clientname value='[NFuse_ClientName]'>
```

In a scripted site, your Applet tag might look like this:

```
<param name=client.wfclient.usehostname value='no'>  
<param name=client.wfclient.clientname value='<%=app.generateClientName(credentials)%>'>
```

For an example of the use of the Applet tag's client name parameters, examine the Applet tag placed in a Java Client embedded Web site created by the Web Site Wizard. Locate the file Appembed.x, where *x* is either *jsp*, *asp*, or *htm* depending upon the Web site model you choose. This file contains the Applet tag with client name information added.

Encryption Parameters

Encryption parameters configure the level of encryption to use when sending data between the ICA Client and Citrix server. To enable encryption levels higher than Basic, the Citrix server must support RC5 encryption. Support for RC5 encryption is included in MetaFrame XP, Feature Release 1 for MetaFrame 1.8, and SecureICA Services.

EncryptionLevelSession	Selects the level of encryption for the ICA connection. Possible values include: EncRC5-40=40-bit encryption EncRC5-56=56-bit encryption EncRC5-128=128-bit encryption EncRC5-0=128-bit encryption (login only) If this parameter is left unspecified, the ICA session uses the default level (Basic).
------------------------	---

Note Support for RC5 encryption is not available if your server farm is composed of MetaFrame for UNIX Operating Systems servers.

When specifying an encryption level other than Basic, you must include in the ICA file an additional section that lists drivers the ICA Client must load to support the specified level of encryption. The section takes the name of the value specified for EncryptionLevelSession. For example, to use 56-bit encryption, create a section called [EncRC5-56]. In this section, list the driver to load for each ICA Client:

```
[EncRC5-56]
DriverNameWin32=PDC56N.DLL
DriverNameWin16=PDC56W.DLL
```

For an example of an encryption driver list, examine a template ICA file created by the Web Site Wizard.

Configuring Authentication Over Encrypted Connections

In its most secure configuration, RC5 encryption uses 128-bit encryption to encrypt user authentication to the Citrix server. To establish this secure connection, the ICA Client and Citrix server must negotiate the connection prior to the ICA Client passing user credentials to the Citrix server for user log in. For this reason, Citrix servers require that ICA connections using RC5 encryption do not allow automatic login. (ICA connections using the least strong encryption level Basic do allow automatic login.)

By default, any user accessing an application configured for greater than Basic encryption must manually log in to the application regardless of whether or not the ICA file used to start the session contains the user's credentials. For this reason, the Template.ica files created by the wizard place user credentials in a conditional block:

```
<[NFuse_IFSESSIONFIELD sessionfield="NFUSE_ENCRYPTIONLEVEL" value="basic"]>
Username=[NFuse_User]
Domain=[NFuse_Domain]
Password=[NFuse_PasswordScrambled]
<[/NFuse_IFSESSIONFIELD]>
```

The NFuse_IfSessionField and /NFuse_IfSessionField tags create a conditional block that causes the Java objects to execute the enclosed code only if a certain condition is met. When processing the example above, the Java objects place the user's credentials in the finished ICA file only if the session's encryption level is Basic. For all other levels, the Java objects omit all credential information.

In some deployment scenarios, you may want to simultaneously support automatic login (sending user credentials in the ICA file) and encryption levels greater than Basic. To do so, you must include the `AutoLogonAllowed` parameter in the section of the template ICA file that contains user credentials. For example:

```
[[NFuse_AppName]]
Address=[NFuse_IPV4Address]
InitialProgram=#[NFuse_AppName]
DesiredColor=[NFuse_WindowColors]
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0
AutoLogonAllowed=On
Username=[NFuse_User]
Domain=[NFuse_Domain]
Password=[NFuse_PasswordScrambled]
[NFuse_IcaWindow]
```

Note When enabling automatic login, you do not need to specify the `NFuse_IfSessionField` and `/NFuse_IfSessionField` tags in the template ICA file.

Additionally, make sure your template ICA file contains a `[Compress]` section with all required compression drivers listed. For example:

```
[Compress]
DriverName=PDCOMP.DLL
DriverNameWin16=PDCOMPW.DLL
DriverNameWin32=PDCOMP.N.DLL
```

Using Ticketing Over Encrypted Connections

Ticketing provides the most convenient and secure method of authentication by allowing automatic logon without explicitly placing user credentials in ICA files.

Note Support for ticketing is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems.

When you create sites with ticketing enabled, the NFuse Java objects place an authentication ticket instead of actual user credentials in the ICA file sent to a client device. A user executing an ICA file containing a ticket automatically logs into the application without entering actual credentials. Instead, the server farm stores the user's credentials within the farm and retrieves the credentials when presented with the ticket. The Citrix server then passes the actual credentials to the Citrix server that will host the desired application.

A template ICA file that supports ticketing can look like the following:

```
[[NFuse_AppName]]
Address=[NFuse_IPV4Address]
InitialProgram=#[NFuse_AppName]
DesiredColor=[NFuse_WindowColors]
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0
AutoLogonAllowed=On
[NFuse_Ticket]
[NFuse_IcaWindow]
```

To use ticketing with RC5 encryption, make sure your template ICA files contain the `AutoLogonAllowed=On` parameter.

For more information on ticketing, see “Use Ticketing” on page 166.

SOCKS Proxy Parameters

SOCKS parameters configure the ICA Client to work with SOCKS proxy servers. For information about SOCKS proxy servers, see the *Citrix ICA Client Administrator's Guide* for your ICA Client.

ICASOCKSProtocolVersion	Indicates which version of the SOCKS protocol to use for the connection. Possible values include: -1: None. Do not use SOCKS for this connection. 0: Autodetect. Client determines which version the proxy is using. 4: Use SOCKS version 4. 5: Use SOCKS version 5.
ICASOCKSProxyHost	Specifies the DNS name or IP address of the SOCKS proxy to use for this connection.
ICASOCKSProxyPortNumber	Port number of the SOCKS proxy server (usually 1080).

To use a SOCKS proxy server with NFuse, you must add the above parameters to both the [WFClient] and [[NFuse_AppName]] sections of the template ICA file. For example:

```
<[NFuse_setSessionField NFuse_ContentType=application/x-ica]>
<[NFuse_setSessionField NFuse_WindowType=seamless]>

[WFClient]
Version=2
ClientName=[NFuse_ClientName]
ICASOCKSProtocolVersion=0
ICASOCKSProxyHost=someServerAddress
ICASOCKSProxyPortNumber=somePortNumber

[ApplicationServers]
[NFuse_AppName]=

[[NFuse_AppName]]
Address=[NFuse_IPV4Address]
InitialProgram=#[NFuse_AppName]
DesiredColor=[NFuse_WindowColors]
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0
<[NFuse_IFSESSIONFIELD sessionfield="NFUSE_ENCRYPTIONLEVEL" value="basic"]>
Username=[NFuse_User]
Domain=[NFuse_Domain]
Password=[NFuse_PasswordScrambled]
<[/NFuse_IFSESSIONFIELD]>
<[NFuse_IFSESSIONFIELD sessionfield="NFUSE_SOUNDTYPE" value="basic"]>
ClientAudio=On
<[/NFuse_IFSESSIONFIELD]>
[NFuse_IcaWindow]

ICASOCKSProtocolVersion=0
ICASOCKSProxyHost=someServerAddress
ICASOCKSProxyPortNumber=somePortNumber
```

SpeedScreen Latency Reduction Parameters

The following parameters configure support for SpeedScreen's local text echo and mouse feedback features. On slow networks, ICA Client users can experience delays between an action such as keyboard input and the appearance, or echoing, of the text on screen. Similarly, a mouse click and visual confirmation of the click, such as the appearance of an hourglass, can also be separated by network latency. SpeedScreen enhances user experience by providing users with immediate keyboard and mouse feedback. For more information on SpeedScreen, including instructions on configuring your server's applications to use SpeedScreen features, see your server documentation.

ZLKeyboardMode	Specifies support for local text echoing. 0: Disable text echoing. 1: Enable text echoing. 2: Automatically enable or disable text echoing depending upon a check of the connection's latency. This is the default value.
ZLMouseMode	Specifies support for mouse feedback. 0: Disable mouse feedback. 1: Enable mouse feedback. 2: Automatically enable or disable mouse feedback depending upon a check of the connection's latency. This is the default value.

Client Auto Update Parameters

Client Auto Update parameters configure whether or not the ICA Client accepts client program updates from the Citrix server. See your server documentation for information on server-side configuration of Client Auto Update and a list of updatable ICA Clients.

UpdatesAllowed	Configures support for Auto Client Update. Specify On to allow updates or Off to disallow updates. The default value is On.
----------------	---

Client Device Mapping Parameters

Client device mapping parameters enable and disable client services such as sound support and client drive mapping.

COMAllowed	Enables or disables client COM port mapping. Specify Yes to enable, No to disable.
CPMAllowed	Enables or disables client printer mapping. Specify Yes to enable, No to disable.
CDMAllowed	Enables or disables client drive mapping. Specify Yes to enable, No to disable.
DisableSound	Enables or disables ICA Client sound support. Specify Off to disable sound. Leave out or specify On to enable.
ClientAudio	Like DisableSound, this parameter enables or disables ICA Client sound support. Specify Off to disable sound. Specify On to enable.
VSLAllowed	Enables or disables support for the Microsoft and Novell TCP stacks. Specify Yes to enable, No to disable.

Configuring NFuse Security



This chapter includes information on steps you can take to secure NFuse. A comprehensive security plan must include the protection of your data at all points in the application delivery process. This chapter describes NFuse security risks and recommendations for each of the major NFuse communication links:

- **Client Device — Web Server Communication.** Explains risks associated with passing NFuse data between Web browsers and Web servers and suggests strategies for protecting data in transit and data written on client devices.
- **Web Server — Citrix Server Communication.** Describes how to secure the authentication and published application information that passes between the NFuse Web server and your Citrix server farm.
- **ICA Client — Citrix Server Communication.** Explains risks associated with passing ICA session information between ICA Clients and Citrix servers and discusses implementation of NFuse and MetaFrame security features that protect such data.

Client Device — Web Server Communication

NFuse communication between client devices and the Web server consists of passing several different types of data. As the user identifies himself, browses applications, and eventually selects an application to execute, the Web browser and Web server pass user credentials, application set lists, and session initializing files. Specifically, this network traffic includes:

- **HTML form data.** NFuse Web sites use a standard HTML form to transmit user credentials from the Web browser to the Web server at user login time. Like all HTML forms, the NFuse form passes user information in clear text.
- **HTML cookies and pages.** After the user enters credentials in the NFuse login form, the Web server writes the credentials in a transient cookie on the client device. By default, this cookie contains user credentials in clear text. The browser retransmits the cookie to the Web server with each HTTP GET request; for example, when the user browses applications in folders or whenever the user switches between pages in an NFuse Web site.

The HTML pages sent from the Web server to the browser contain application sets. These pages list the applications available to the user.

- **ICA files.** When the user selects an application, the Web server sends an ICA file for that application to the browser. The ICA files produced by NFuse can contain user name and domain data in clear text and passwords encrypted using Basic encryption.

Risks

Attacks can exploit NFuse data as it crosses the network between the Web server and browser and as it is written on the client device itself:

- An attacker can intercept login form data, the credentials cookie, ICA files, and HTML pages in transit between the Web server and Web browser.
- Although the credentials cookie used by NFuse is transient and disappears when the user closes the Web browser, an attacker with access to the client device's Web browser can retrieve the cookie and steal plain text credential information.
- An attacker who can gain access to a Web browser's file cache can retrieve cached ICA files. Because ICA files contain user credentials, the attacker can then use these ICA files to initiate ICA sessions. As long as the user's password remains unchanged, the attacker can use the files repeatedly to access the Citrix server.

Recommendations

The following recommendations combine industry-standard security practices and Citrix-provided safeguards to protect data travelling between client devices and your Web server and data written to client devices.

Implement SSL-Capable Web Servers and Web Browsers

Securing the Web server to Web browser component of NFuse communication begins with implementing secure Web servers and Web browsers. Many secure Web servers rely upon SSL technology to secure Web traffic. *SSL* (Secure Sockets Layer) is an open, non-proprietary Web protocol that provides server authentication, data encryption, message integrity, and optional client authentication for a TCP/IP connection.

In a typical Web server to Web browser transaction, the Web browser first verifies the identity of the Web server by checking the Web server's server certificate against a list of trusted certificate authorities. After verification, the Web browser encrypts user page requests and then decrypts the documents returned by the Web server. At each end of the transaction, SSL message integrity checks ensure that the data has not been tampered with in transit.

In an NFuse deployment, SSL authentication and encryption creates a secure connection over which the user can pass credentials posted in the NFuse login form. Data sent from the Web server, including the credentials cookie, ICA files, and HTML application list pages, is equally secure.

To implement SSL on your network, you must have an SSL-capable Web server and SSL-capable Web browsers. The use of these products is transparent to NFuse. No NFuse-specific configuration of your Web servers or browsers must be completed. For information on configuring your Web server to support SSL, see your Web server's documentation.

Important Many SSL-capable Web servers use TCP/IP port 443 for HTTP communications. By default, the Citrix SSL Relay uses this port as well. If your Web server is also a Citrix server running the SSL Relay, make sure you configure either the Web server or SSL Relay to use an alternate port.

Encrypt Cookie Data

Among the example sites included in NFuse is one that encrypts the credential data placed in NFuse cookies. This example site demonstrates how to use Basic encryption scripted into Active Server Pages or JavaServer Pages files to encrypt the clear text user credential information entered in an NFuse login form. The encrypted cookie data can then be transmitted to the Web browser and retrieved by the Web server whenever it requires information from the cookie.

For information on cookie data encryption, see “Cookie Data Encryption” on page 191.

As an alternative to cookies, you can develop Web pages that use session variables on the Web server to track information. See your Active Server Pages or JavaServer Pages documentation for information on accessing session variables.

Use Ticketing

By default, previous versions of NFuse placed user credential information in the ICA files sent to client devices for ICA session initialization. An attacker able to intercept the file or retrieve the file from disk could repeatedly use the file to access a Citrix server. The use of NFuse's ticketing feature eliminates these dangers.

Note Support for ticketing is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems. See the end of this topic for information on creating ICA files that do not contain user credentials.

Ticketing provides authentication security by eliminating user credentials from the ICA files sent from the Web server to client devices. Tickets have a configurable expiration period and are valid for a single ICA session. After use, or after expiration, the ticket is invalid and cannot be used to access applications.

Ticketing uses Citrix servers to store credentials entered by users in NFuse login forms. When a user selects an application from the Web page, the NFuse Java objects on the Web server request from the farm a ticket for that user. The server farm generates a 30 character string that correlates the user to the user's credentials but does not contain the credentials themselves. The farm forwards this ticket to the Web server, which places the ticket in the ICA file sent to the client device. When the ICA Client uses the ticket to authenticate itself to the server farm, the server farm matches the ticket to the user's actual credentials and logs the user in.

Ticketing is a default component of the Web sites produced by the Web Site Wizard. When you create a site with ticketing enabled, the wizard places the [NFuse_Ticket] tag in the site's template ICA file. To implement ticketing in your site, make sure your template ICA file includes the [NFuse_Ticket] tag. For example:

```
<[NFuse_setSessionField NFuse_ContentType=application/x-ica]>

[WFClient]
Version=2
ClientName=[NFuse_ClientName]

[ApplicationServers]
[NFuse_AppName]=

[[NFuse_AppName]]
Address=[NFuse_IPV4Address]
InitialProgram=#[NFuse_AppName]
DesiredColor=[NFuse_WindowColors]
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0
AutologonAllowed=ON
[NFuse_Ticket]

[NFuse_IcaWindow]
```

Important Your template ICA file must also have the AutologonAllowed=ON parameter included.

When parsing a template ICA file that contains the [NFuse_Ticket] tag, NFuse replaces the tag with user name, domain, and password parameter/value pairs. The user name entry specifies the current user's user name in plain text. For the domain and password fields, NFuse specifies the ticket value; the password is set to the first 14 characters of the ticket and the domain to the last 16 characters of the ticket.

For example, when parsed the template ICA file above produces the following rendered ICA file:

```
[WFClient]
Version=2
ClientName=CrisplyCreper

[ApplicationServers]
Frame=

[Frame]
Address=10.8.7.62
InitialProgram=#Frame
DesiredColor=2
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0
AutologonAllowed=ON
Username=CliftonSantiago
Domain=\AB518FF5E8DE9F81
ClearPassword=F66CC862055897

DesiredHRES=640
DesiredVRES=480
```

The presence of the backslash (\) before the domain value alerts the Citrix server that the domain and password data comprises a ticket and should not be used for standard login. Instead, the Citrix server begins the process of retrieving the user's actual credentials stored in the farm.

To modify the ticket expiration period, edit the NFuse.properties file. By default, the properties file specifies a 200 second expiration period. To change this value, locate and edit the SessionField.NFuse_TicketTimeToLive=200 entry. After modifying the value, make sure you stop and restart your Web server. See "Configuring Web Server Extension Properties" on page 44 for more information.

You can implement credential security for those deployments that cannot use ticketing by removing user credential tags from template ICA files. Template ICA files that do not contain credentials pose no security risk but force the user to log into each application manually.

As an additional security precaution, you can prevent Web browsers from caching ICA files. Active Server Pages and JavaServer Pages include functions for preventing file caching that you can add to your Web sites. For example, in the file `launch.asp` in an Active Server Pages-based site produced by the Web Site Wizard, locate the line `Response.ContentType = "application/x-ica"`. Add the following two lines:

```
Response.ContentType = "application/x-ica"  
Response.CacheControl = "no-cache"  
Response.AddHeader "Pragma", "no-cache"  
Continue = True  
Response.Expires = 0  
While (Continue)
```

The `launch.asp` file is responsible for parsing your template ICA files. Placing these two entries in `launch.asp` causes the Web server to write parameters in your customized ICA files that alert most browser types to not cache the ICA file.

Web Server — Citrix Server Communication

Communication between the Web server and Citrix server in an NFuse deployment involves passing user credential and application set information between the NFuse Java objects on the Web server and the Citrix XML Service in the Citrix server farm. In a typical NFuse session, the Java objects pass credentials to the XML Service for user authentication and the XML Service returns application set information. The Web server and server farm use a TCP/IP connection and the NFuse XML protocol to pass the information.

Risks

The NFuse XML protocol uses clear text to exchange all data with the exception of passwords, which it passes using Basic encryption. The XML communication is vulnerable to the following attacks:

- An attacker can intercept the XML traffic and steal application set information and tickets. An attacker with the ability to crack Basic encryption can obtain user credentials as well.
- An attacker can impersonate the Citrix server and intercept authentication requests.

Recommendations

Citrix recommends implementing one of the following security measures for securing the XML traffic between your Web server and Citrix server farm:

- Use the Citrix SSL Relay as a security intermediary between the Web server and Citrix server farm.
- In deployments that do not support running the SSL Relay, run a Web server on your Citrix server.

Note Support for the Citrix SSL Relay is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems. See “Run a Web Server on Your Citrix Server” on page 172 for information on securing deployments that do not support the Citrix SSL Relay.

Use the Citrix SSL Relay

The Citrix SSL Relay is a Citrix security component that uses SSL to secure communication between NFuse Web servers and Citrix server farms. The SSL Relay provides Citrix server authentication, data encryption, and message integrity for a TCP/IP connection.

The SSL Relay operates as an intermediary in the communication between the Web server and Citrix server. When using the SSL Relay, the Web server first verifies the identity of the SSL Relay by checking the Relay's server certificate against a list of trusted certificate authorities. After this authentication, the Web server and SSL Relay negotiate an encryption method for the session. The Web server can then send all information requests in encrypted form to the SSL Relay. The SSL Relay decrypts the requests and passes them to the Citrix server. When returning the information to the Web server, the Citrix server sends all information through the SSL Relay server, which encrypts the data and forwards it to the Web server for decryption. Message integrity checks verify each communication has not been tampered with.

Configuring NFuse to Use the Citrix SSL Relay

The SSL Relay is a default component of MetaFrame XP. On MetaFrame 1.8 servers, you must install Citrix MetaFrame 1.8 Service Pack 2 to use the SSL Relay. In addition, each MetaFrame 1.8 server must have an installed and activated Feature Release 1 license. Make sure your Feature Release license is both installed and activated on each server.

On the Citrix server side, using the SSL Relay to secure NFuse communication requires installation of a server certificate on the SSL Relay server and verification of the SSL Relay server's configuration. For information on installing a server certificate and configuring the SSL Relay on MetaFrame XP servers, see the installation chapter of the *MetaFrame XP Administrator's Guide*. For MetaFrame 1.8 servers, see the *Feature Release 1 and Service Pack 2 Installation Guide for Citrix MetaFrame for Windows Version 1.8*. On either platform, you can also consult the application help in the Citrix SSL Relay Configuration Tool.

When configuring the SSL Relay, make sure your Relay server permits passing SSL traffic to the Citrix server you are using as the XML Service contact. By default, the SSL Relay forwards traffic only to the server on which it is installed. You can, however, configure the SSL Relay to forward traffic to other servers. If the SSL Relay in your deployment is on a machine other than the machine to which you want to send NFuse data, make sure the SSL Relay's server list contains the server to which you want to forward NFuse data.

On the Web server side, using the SSL Relay involves writing SSL Relay server information into your Web sites. You can cause an NFuse Web script to send all communications through an SSL Relay server by specifying SSL Relay server information in the `initialize()` method used on the `CitrixWireGateway` object. As an example, examine the following Active Server Pages code produced by the Web Site Wizard:

```
Set credentials = Server.CreateObject("com.citrix.nfuse.ClearTextCredentials")
credentials.initialize app.unUrlEncode(user), app.unUrlEncode(domain), app.unUrlEncode(password)

Set gateway = Server.CreateObject("com.citrix.nfuse.CitrixWireGateway")
gateway.initialize credentials, "myRelayServer" , 443, "Ssl", 0
```

The above `initialize()` method specifies that all communications will be directed to an SSL Relay server named *myRelayServer* on port 443. For information on the `CitrixWireGateway` object, see “`CitrixWireGateway`” on page 104.

The Web Site Wizard includes support for the SSL Relay and can place all necessary SSL Relay contact information into your pages. When creating a site that communicates through an SSL Relay server, the wizard prompts you to specify the SSL Relay server's name and the port it uses for SSL communication. By default, SSL Relay servers use port 443.

Adding Certificates to the Web Server Extension

As installed, the Web Server Extension contains root certificates for certain public certificate authorities supported by the SSL Relay. If you want to add support for other certificate authorities, you must add the certificate authority's root certificate to the Web Server Extension.

➤ To add a new root certificate to your Web Server Extension

1. Make sure the root certificate is in DER format.
2. Copy the root certificate to the following directory on your Web server:
 %SystemRoot%\keystore\cacerts by default on Windows Web servers.
 /keystore/cacerts by default on UNIX Web servers.

For information on certificates, see the installation chapter of the *MetaFrame Administrator's Guide* or the *Feature Release 1 and Service Pack 2 Installation Guide for Citrix MetaFrame for Windows Version 1.8*.

Run a Web Server on Your Citrix Server

For those deployments that do not support running the SSL Relay, the possibility of network attack can be eliminated by running a Web server on the Citrix server supplying the NFuse data. Hosting your NFuse Web sites on such a Web server routes all NFuse requests to the Citrix XML Service on the local host, thereby eliminating transmission of NFuse data across the network. Note, though, that the benefit of eliminating network transmission must be weighed against the risk of exploitation of the Web server.

In this deployment scenario, make sure your Web server and the Citrix XML Service operate on different TCP/IP ports. If you choose to use a non-default port for the Citrix XML Service, make sure you modify your Web pages to contact the local host on the non-default port.

Note On MetaFrame XP systems, the MetaFrame Setup routine lets you force the Citrix XML Service to share Internet Information Server's TCP/IP port instead of using a dedicated port. If you have enabled port sharing, the XML Service and the Web server use the same port by default.

At minimum, you can place both your Web server and Citrix server behind a firewall so that the communication between the two is not exposed to open Internet conditions. In this scenario, client devices must be able to communicate through the firewall to both the Web server and Citrix server. Your firewall must permit HTTP traffic (often over the standard HTTP port 80 or 443 if a secure Web server is in use) for client device to Web server communication. For ICA Client to Citrix server communication, the firewall must permit inbound ICA traffic on port 1494 and outbound traffic on a dynamically generated port above 1023. See your server documentation for information on using ICA with network firewalls.

For information on using NFuse with network address translation, see “NFuse’s Server Location Options” on page 153. This topic includes information on server location through firewalls.

ICA Client — Citrix Server Communication

NFuse communication between client devices and Citrix servers consists of passing several different types of ICA session data including initialization requests and ICA session information.

- **Initialization requests.** The first step in establishing an ICA session, called initialization, requires the ICA Client to request an ICA session and produce a list of ICA session configuration parameters that control various aspects of the ICA session such as the user to log in, the size of the window to draw, and the program to execute in the session.
- **ICA session information.** After session initialization, the ICA Client passes user keyboard and mouse input to the Citrix server as the user navigates the chosen application. In response, the Citrix server sends the ICA Client graphical updates.

Risks

To capture and interpret ICA Client to Citrix server network communications, an attacker must be able to crack the binary ICA protocol. An attacker with binary ICA protocol knowledge can:

- Intercept initialization request information sent from the ICA Client, including user credentials.
- Intercept ICA session information including text and mouse clicks entered by users and screen updates sent from the Citrix server.

Recommendations

Citrix recommends implementing the following security measures for securing the traffic between your ICA Clients and Citrix servers:

- Use RC5 encryption to encrypt initialization and session information. Support for RC5 encryption is included in MetaFrame XP, Feature Release 1 for MetaFrame 1.8, and SecureICA Services.
- Use ticketing to provide authentication security.

Note Support for RC5 encryption and ticketing are not available if your server farm is composed of MetaFrame for UNIX Operating Systems servers.

Use RC5 Encryption with Ticketing

RC5 encryption is a Citrix security component that encrypts ICA session information for ICA Clients connecting to Citrix MetaFrame for Windows servers. RC5 encryption uses the RC5 encryption algorithm from RSA Data Security, Inc. and the Diffie-Hellman key agreement algorithm with a 1024-bit key to generate RC5 keys. You can use RC5 encryption to apply 128-bit encryption to ICA session initialization including user authentication. Once the session is established, RC5 encryption supports 40-, 56-, and 128-bit session encryption that administrators can configure on a per-connection basis. Administrators can also enforce minimum encryption levels that allow connections only if the ICA Client supports the specified encryption level.

Several RC5-enabled ICA Clients are available on the ICA Client CD. Additional RC5-enabled ICA Clients are available for download on the Citrix download site. Please see your ICA Client documentation or the download site for a list of RC5-enabled ICA Clients.

Using RC5 encryption with NFuse is for the most part transparent. Once NFuse passes the ICA Client an ICA file for session initialization, NFuse has no further interaction with the ICA session established by the user. You must consider, however, the process by which the ICA file presents the Citrix server with user credentials.

In its most secure configuration, RC5 encryption uses 128-bit encryption to encrypt user authentication to the Citrix server. To establish this secure connection, the ICA Client and Citrix server must negotiate the connection prior to the ICA Client passing user credentials to the Citrix server for user log in. For this reason, Citrix servers require that ICA connections using RC5 encryption do not allow automatic login. (ICA connections using the least strong encryption level Basic do allow automatic login.)

RC5 encryption can, however, use automatic login with no security risk if your ICA files contain tickets instead of user credentials. For general information on configuring automatic login see “Configuring Authentication Over Encrypted Connections” on page 156. For general information on ticketing, see “Use Ticketing” on page 166. For information on using ticketing with RC5 encryption, see “Using Ticketing Over Encrypted Connections” on page 157.

Example Web Sites



This chapter contains information on the example Web sites included in NFuse. Example Web sites demonstrate implementations of various NFuse features and contain tutorial information that you can browse to learn about adding these features to your own custom Web sites or sites produced with the Web Site Wizard.

Note MetaFrame XP Setup offers administrators the option of installing NFuse during MetaFrame installation. This option installs the Web Server Extension on the server and places a single NFuse Web site in the Web server's document root directory but does not install the NFuse example Web sites described in this chapter. If you installed the Web Server Extension as part of MetaFrame XP Setup, you must install the full Web Server Extension from the NFuse CD-ROM or Web download file in order to use the NFuse example Web sites. See Chapter 3, "Configuring Your Web Server" on page 29 for more information.

This chapter is divided into the following areas of interest:

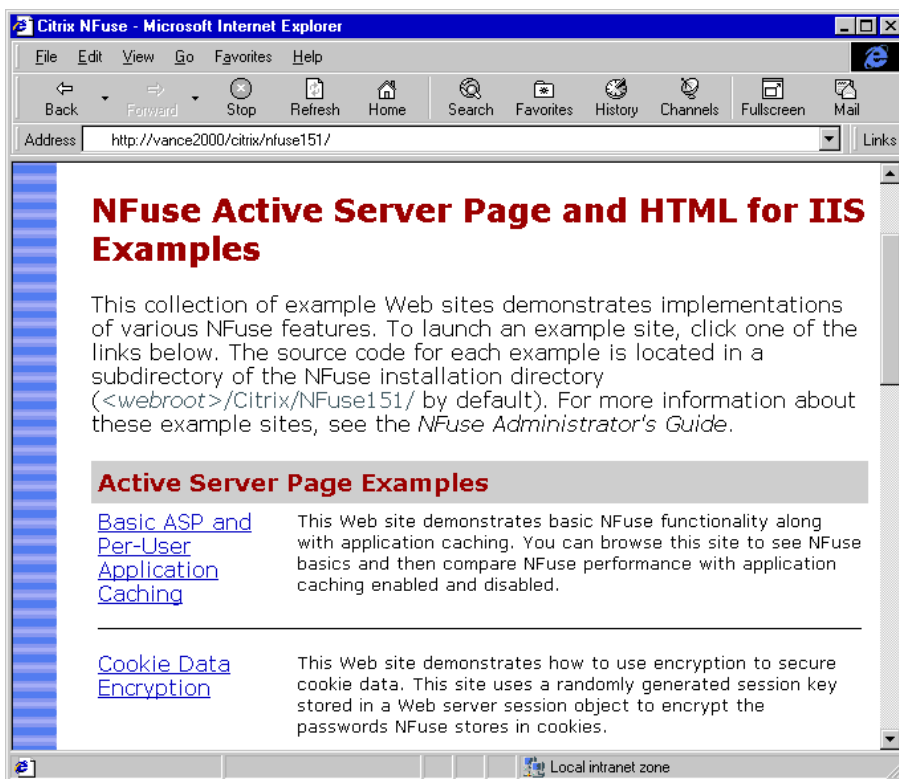
- **Improving NFuse Performance.** Includes information on several Web sites that cache published application information on an NFuse Web server and then use various filters to sort that data for presentation to users.
- **Securing NFuse.** Includes information on Web sites that demonstrate ways to secure Citrix data in an NFuse deployment. Describes using ticketing and encryption of NFuse cookie data to secure user credentials.
- **Improving Reliability and Usability.** Includes information on several sites that implement various improvements to user experience and reliability of NFuse. Describes using backup Citrix servers for published application information, displaying applications from multiple server farms in a single NFuse Web page, and using Web-based ICA Client installation.

Note that each example Web site described in this chapter demonstrates a single implementation of an NFuse feature. As an SDK, NFuse provides you with the flexibility to implement features in any way suited to your environment.

Before proceeding to topics on specific example sites, please see the next topic, “Getting Started,” for information on locating the example sites and browsing the example site launch page.

Getting Started

The Web Server Extension installation program places the example Web sites in the NFuse installation directory off your Web server's Web publishing root directory; for example `<webroot>/Citrix/NFuse151/`. Included in this directory is a default document that contains introductory information and links to the sites themselves. To load this document, point a Web browser at the URL for your NFuse installation; for example, `http://<ServerName>/Citrix/NFuse151`.



Note The default document shown above is the version installed on Internet Information Servers.

Use the default document as your starting point for examining the example Web sites. Click an example name to launch the site and execute its functionality.

The scripting model used in the example Web sites on your system depends upon your Web server's platform. When you install the Web Server Extension on Internet Information Server, the setup program installs example sites that use the Active Server Pages and Citrix's HTML for IIS models. On UNIX Web servers, the Web Server Extension installs sites that use the JavaServer Pages and Citrix HTML for Servlets models. For information on scripting models, see "Choosing a Layout Model" on page 50.

Improving NFuse Performance

You can optimize NFuse's main task, enumerating applications for each user, by implementing caching and filtering on your NFuse Web server. *Application caching* is a method of storing Citrix application objects in an application cache object on the NFuse Web server. By caching application objects on the Web server, NFuse eliminates the need to repeatedly query the Citrix server for the list of applications a user can access. NFuse uses the AppDataList and AppListCache objects to perform caching tasks. See "NFuse Java Object Reference" on page 103 for information on these objects.

Filtering is the process of sorting through the application cache for application information related to a specific user, user group, or Program Neighborhood folder. The following example sites each implement a different filtering method. You can implement various filtering methods to suit your environment.

Implementation Notes

Consider the following items when using the caching and filtering examples:

- For JavaServer Pages example caching sites, Citrix provides two versions of each example: one based upon JSP 1.0 and the other upon JSP 0.92. Some Web servers such as Netscape Enterprise Server 4.0 do not properly handle the JSP 1.0-based examples. Use the JSP 0.92 versions for these Web servers.
- When using cached application data, it is important that you implement an authentication check before granting a user access to the cache. For example, the example site that caches data on a per-user basis ("Basic ASP/JSP and Per-User Application Caching") generates a unique key based on the user's credentials and uses the key to map the user to the application data the user can access.

- To keep cached application information on the Web server current with actual application information maintained by the server farm, cached application objects persist for a finite time period (one hour by default). After this time, they are subject to removal. You can modify this behavior by editing the `CacheExpireTime` entry in the `NFuse.properties` file. See “Configuring Web Server Extension Properties” on page 44 for more information.

See “AppListCache” on page 137 for information on methods you can use when working with an application cache.

Basic ASP/JSP and Per-User Application Caching

This example site demonstrates basic NFuse functionality along with application caching. You can browse this site to see NFuse basics such as user authentication and application listing. After familiarizing yourself with these concepts, you can revisit the same pages with application caching enabled and experience the performance gains of using cached application information.

This example site’s implementation of caching uses a key to associate a user with a cached list of application objects. When the user logs in, the Web server retrieves the user’s entire application set from the Citrix server and creates a cache object to store the information. Along with the cache object, the Web server creates a key based upon the user’s credentials. Each time the Web server requires application information from the Citrix server, for example after a login or when the user browses a folder, the Web server retrieves the cache information associated with the user’s key. By default, cache objects expire after an hour. Within this period, the user can log in and out of the site repeatedly and browse applications quickly without the Web server having to contact the Citrix server for each application request.

Using the Web Site

To load this site, point a Web browser at the URL for your NFuse installation; for example, `http://<ServerName>/Citrix/NFuse151`. In the default document that appears, click **Basic ASP and Per-User Application Caching** on Internet Information Servers or **Basic JSP and Per-User Application Caching** on UNIX Web servers.

This example site correlates users to domains during application information gathering. When using this example site to access published applications hosted by a MetaFrame for UNIX Operating Systems server farm, users must provide domain information during login. Please see “Authentication and MetaFrame for UNIX Operating Systems” on page 56 for information on using a placeholder domain name to satisfy NFuse’s domain requirements.

If you are using User Principal Names (UPN), such as in an Active Directory deployment, when using the login form to log into the site, you can enter the entire UPN in the form's **Username** field or break the name into its name and domain components in the **Username** and **Domain** fields. For more information on using UPN with NFuse login forms, see "Using User Principal Names and Active Directory Support" on page 56.

Using this example site requires no configuration of NFuse or modification of files. The only requirement is that you log in as a user for whom applications have been published.

When browsing this site for the first time, click **Disable Cache** on the site's introductory page to examine basic functionality. After viewing your application set and browsing its folders, return to the site's introductory page and click **Enable Cache** to browse the site with caching turned on. Quick enumeration of applications is noticeable when you switch between pages in a view of your application set.

Implementing Per-User Application Caching

Caching functionality is contained in the file `applist_cache.asp` (or `applist_cache.jsp` on UNIX Web servers). Examine this file's use of the `AppListCache` and `AppDataList` objects. For information on these objects, see "AppListCache" on page 137 or "AppDataList" on page 135.

Application Caching and Filtering by Group

This site demonstrates the performance improvement you can achieve by using application filtering based upon user groups. This filtering method uses a specific application publishing strategy, group-based filtering, and caching of group application information on the Web server to improve application enumeration.

Note Support for filtering of applications based upon groups is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems.

To use this example, you must publish applications for entire user groups as opposed to publishing applications for individual users. For example, publish an application for the "Domain Users" group of a specific Windows NT domain as opposed to publishing the application for UserN from the domain X.

To retrieve applications for an entire user group, this site uses a special authentication method. Under standard conditions, the NFuse Web server sends the Citrix server an authentication request containing a specific user's credentials. In this example site, the NFuse Web server uses a user-to-group-mapping file named `groups.txt` to determine the user's group membership at login time. After determining group membership, the NFuse Web server sends the Citrix server an authentication request for the *user group* of which the user is a member.

Note This example site uses a Java object called `UserToGroupMap` to parse the mapping file and to create the association between the user and user group. This Java object is an unsupported object included in NFuse for demonstration purposes.

In response to an authentication request containing group credentials, the Citrix server prepares to send the application list. The NFuse Web server checks if it already has a cached list and if it does not, requests the Citrix server to send its list. By default, the NFuse Web server retains a cached list for an hour, after which the cache object is discarded. The next user to log in causes recreation of the cache object. If no cache object exists, the Citrix server returns a list of all applications published for the specified group. Otherwise, the Web server retrieves the list from the cache object for presentation to the user.

Using the Web Site

To load this site, point a Web browser at the URL for your NFuse installation; for example, `http://<ServerName>/Citrix/NFuse151`. In the default document that appears, click **Application Caching and Filtering by Group**.

Before proceeding, make sure you have applications published for specific user groups and that you have configured the mapping file `groups.txt`. To publish applications for a group, you may want to create a test group in one of your domains and create a test user who has membership in that group. Then publish applications for the test group.

To edit the user-to-group-mapping file, locate the file groups.txt in the example directory for this site:

```
; This file maintains the user-to-group mappings for this
; example Web site. This mapping list associates a user or users
; with a user group or groups. When authenticating a user to the
; server farm, the NFuse Web server checks this file to determine
; to which user group the user belongs.
;
; Below these comments, enter your user-to-group mappings. The
; format of each entry can be one of the following:
;
; domain \ username = group A, group B, group C
; * \ username      = group A, group B, group C
; domain \ *        = group A, group B, group C
; * \ *             = group A, group B, group C
;
; '*' is used as the wildcard for domain or username.
;
; For example, to associate a user named User1 from domain Domain1
; with the user groups "Domain Users" and "Domain Admins," add
; the following:
;
; Domain1 \ User1 = Domain Users, Domain Admins
;
; If you want to show all users in Domain1 the applications
; published for the group "Domain Users," add:
;
; Domain1 \ * = Domain Users
```

After the last commented line in the file, enter user-to-group mappings. For example, to associate a user named TestUser with the group Test Group in the domain MyDomain, enter: **MyDomain \ TestUser = Test Group**. Save the file and exit the text editor.

Note If you are using User Principal Names (UPN), such as in an Active Directory deployment, break the UPN into user and domain components when entering it into the mapping file. For example, for user *userx@domain.com* enter: **Domain.com \ Userx = Test Group**. When using the login form to log into the site, you can enter the entire UPN in the form's **Username** field or break the name into its name and domain components in the **Username** and **Domain** fields. For more information on using UPN with NFuse login forms, see "Using User Principal Names and Active Directory Support" on page 56.

Once you have published applications and edited the mapping file, you must log in to the site to proceed. NFuse presents you with the applications for the user group with which you are associated in the mapping file. As you browse the site, performance gains are noticeable when you switch between pages in a view of your application set. Notice application enumeration is quicker in this example site than in other example sites that do not use filtering and caching (such as the basic ASP/JSP site with caching disabled).

Implementing Filtering by Group

Filtering and caching functionality is contained in the file `applist.asp` (or `applist.jsp` on UNIX Web servers). This site uses the `GroupCredentials` object to provide group authentication. See “`GroupCredentials`” on page 110 for more information.

The site uses the `AppListCache` and `AppDataList` objects to implement caching. For information on these objects, see “`AppListCache`” on page 137 or “`AppDataList`” on page 135.

Application Caching and Filtering by Folder

This site demonstrates the performance improvement you can achieve by using application filtering based upon folders. This filtering method uses a specific application publishing strategy, caching of a server farm’s entire application list on the Web server, and filtering of that list to improve application enumeration.

To use this example, you must publish applications for specific user types in separate Program Neighborhood folders. For example, publish all applications for sales users in a “Sales” folder and all marketing applications in a “Marketing” folder.

To cache a farm’s entire application list on the NFuse Web server, this site uses a special authentication method. When a user logs in, the NFuse Web server sends the Citrix server an authentication request containing null credentials. In response to such a request, the Citrix server prepares to send the farm’s entire application list. The NFuse Web server checks if it already has a cached list and if it does not, requests the Citrix server to send its list. By default, the NFuse Web server retains a cached list for an hour, after which the cache object is discarded. The next user to log in causes recreation of the cache object.

The NFuse Web server then must filter the application list to show the current user only those applications the user needs. To filter the list, this site uses a user-to-folder-mapping file named `folders.txt`. This file maintains a list that maps users to specific Program Neighborhood folders. After checking the mapping list to determine a user’s folder association, the NFuse Web server displays the associated folder’s applications to the user.

Note This example site uses a Java object called `UserToGroupMap` to parse the mapping file and to create the association between the user and folder. This Java object is an unsupported object included in NFuse for demonstration purposes.

It is important to note that when the NFuse Web server determines a user-to-folder mapping, it displays all applications in the folder to the user. Whether the user can actually launch those applications depends on whether or not the user is a member of each application's configured user list specified at publishing time.

Using the Web Site

To load this site, point a Web browser at the URL for your NFuse installation; for example, `http://<ServerName>/Citrix/NFuse151`. In the default document that appears, click **Application Caching and Filtering by Folder**.

Important This example site correlates users to domains during application information gathering. When using this example site to access published applications hosted by a MetaFrame for UNIX Operating Systems server farm, users must provide domain information during login. Please see "Authentication and MetaFrame for UNIX Operating Systems" on page 56 for information on using a placeholder domain name to satisfy NFuse's domain requirements.

Before proceeding, make sure you have applications published in Program Neighborhood folders and that you have configured the user-to-folder-mapping file `groups.txt`. To publish applications in folders, you may want to create a test user in one of your domains. Then publish multiple applications for the test user. When publishing the applications, specify that each application be displayed in some test Program Neighborhood folder; for example, publish all applications for the user in a folder called Test Folder. If you are using MetaFrame for Windows servers, see the *MetaFrame Administrator's Guide* and the Citrix Management Console online help (MetaFrame XP) or Published Application Manager online help (MetaFrame 1.8) for information on application publishing. For information on configuring folders for applications published on MetaFrame for UNIX Operating Systems servers, consult the *Citrix XML Service for UNIX Operating Systems Administrator's Guide*.

To edit the mapping file, locate the file `folders.txt` in the example directory for this site:

```
; This file maintains the user-to-folder mappings for this
; example Web site. This mapping list associates a user or users
; with a Program Neighborhood folder. When filtering a server
; farm's entire application list, the NFuse Web server checks
; this file to determine which Program Neighborhood folder of
; applications to display for the current user.
;
; Below these comments, enter your user-to-folder mappings. The
; format of each entry can be one of the following:
;
; domain \ username = Program Neighborhood Folder
; * \ username      = Program Neighborhood Folder
; domain \ *        = Program Neighborhood Folder
; * \ *             = Program Neighborhood Folder
;
; '*' is used as a wildcard for domain or username.
;
; For example, to associate a user named User1 from domain Domain1
; with the Program Neighborhood folder Folder1, add the following:
;
; Domain1 \ User1 = Folder1
;
; If you publish the applications in a server farm in two Program
; Neighborhood folders, "\Marketing Folder" and "\Sales Folder," and
; you want all the users in the Marketing domain to access the
; applications in "\Marketing Folder" and the users in the Sales
; domain to access the applications in "\Sales Folder," add:
;
; Marketing \ * = \Marketing Folder
; Sales \ * = \Sales Folder
```

After the last commented line in the file, enter user-to-folder mappings. For example, to associate a user named `TestUser` from the domain `MyDomain` with a Program Neighborhood folder called `Test Folder`, enter: **`MyDomain \ TestUser = Test Folder`**. Save the file and exit the text editor.

Note If you are using User Principal Names (UPN), such as in an Active Directory deployment, break the UPN into user and domain components when entering it into the mapping file. For example, for user *userx@domain.com* enter: **Domain.com \ Userx = Test Folder**. When using the login form to log into the site, you can enter the entire UPN in the form's **Username** field or break the name into its name and domain components in the **Username** and **Domain** fields. For more information on using UPN with NFuse login forms, see "Using User Principal Names and Active Directory Support" on page 56.

If you are using this example site to access published applications hosted by a MetaFrame for UNIX Operating Systems server farm, make sure you specify either the placeholder domain name described earlier in this topic or a wildcard as the domain for each user. For example, if during login users specify a domain name of myDomain, your map file might contain the following entries:

```
myDomain \ PhilipTraum = \Engineering Folder
myDomain \ HorstSnarl = \Courseware Folder
```

Once you have published applications and configured the mapping file, you must log in to proceed. NFuse presents you with the folder of applications with which you are associated in the mapping file. As you browse the site, performance gains are noticeable when you switch between pages in a view of your application set. Notice application enumeration is quicker in this example site than in other example sites that do not use filtering and caching (such as the basic ASP/JSP site with caching disabled).

Implementing Filtering by Folder

Filtering and caching functionality is contained in the file `applist.asp` (or `applist.jsp` on UNIX Web servers). For information on using a null credential authentication request, see the description of the `initialize()` method in "ClearTextCredentials" on page 108.

The site uses the `AppListCache` and `AppDataList` objects to implement caching. For information on these objects, see "AppListCache" on page 137 or "AppDataList" on page 135.

Securing NFuse

You can enhance NFuse security by implementing ticketing and cookie data encryption. *Ticketing* provides authentication security by eliminating user credentials from the ICA files sent from the Web server to client devices. Tickets have a configurable expiration period and are valid for a single ICA session. After use, or after expiration, the ticket is invalid and cannot be used to access applications.

Cookie data encryption uses Basic encryption scripted into Active Server Pages or JavaServer Pages files to encrypt the clear text user credential information entered in an NFuse login form. The encrypted cookie data can then be transmitted to the Web browser and retrieved by the Web server whenever it requires information from the cookie.

Implementation Notes

Consider the following items when using the security examples:

- The ticketing expiration period is set by an entry in the NFuse.properties file. By default, the properties file specifies a 200 second expiration period. To change this value, locate the SessionField.NFuse_TicketTimeToLive=200 entry. After modifying the value, make sure you stop and restart your Web server. See “Configuring Web Server Extension Properties” on page 44 for more information.
- Implement ticketing and cookie data encryption as part of an overall security policy. See “Configuring NFuse Security” on page 163 for information on securing the entire NFuse system.

Ticketing

This example Web site demonstrates ticketing, an NFuse security feature. Ticketing secures user authentication by eliminating user credentials from the ICA files sent from the Web server to client devices. Tickets have a configurable expiration period and are valid for a single ICA session. After use, or after expiration, the ticket is invalid and cannot be used to access applications. You can use ICA files with tickets as an alternative to standard NFuse ICA files which contain user credentials encrypted using Basic encryption.

Note Support for ticketing is not available in the first release of the Citrix XML Service for MetaFrame for UNIX Operating Systems servers. See the Citrix download site for forthcoming updates to Citrix XML Service for MetaFrame for UNIX Operating Systems.

In an NFuse system, one of the tasks performed by the NFuse Web server is creating customized ICA files for users requesting applications. To gain access to applications, users must present these ICA files to the Citrix server farm at application login time. In previous versions of NFuse, the ICA file contained the user's user name, domain, and password information in the ICA file itself.

The presence of credentials in ICA files poses a security risk: attackers able to intercept an ICA file or retrieve an ICA file from a client device's browser cache or disk can mine the user credentials from the file and use those credentials repeatedly to log into Citrix servers. As long as the user's password remains valid, the attacker can reuse the ICA file to access the farm.

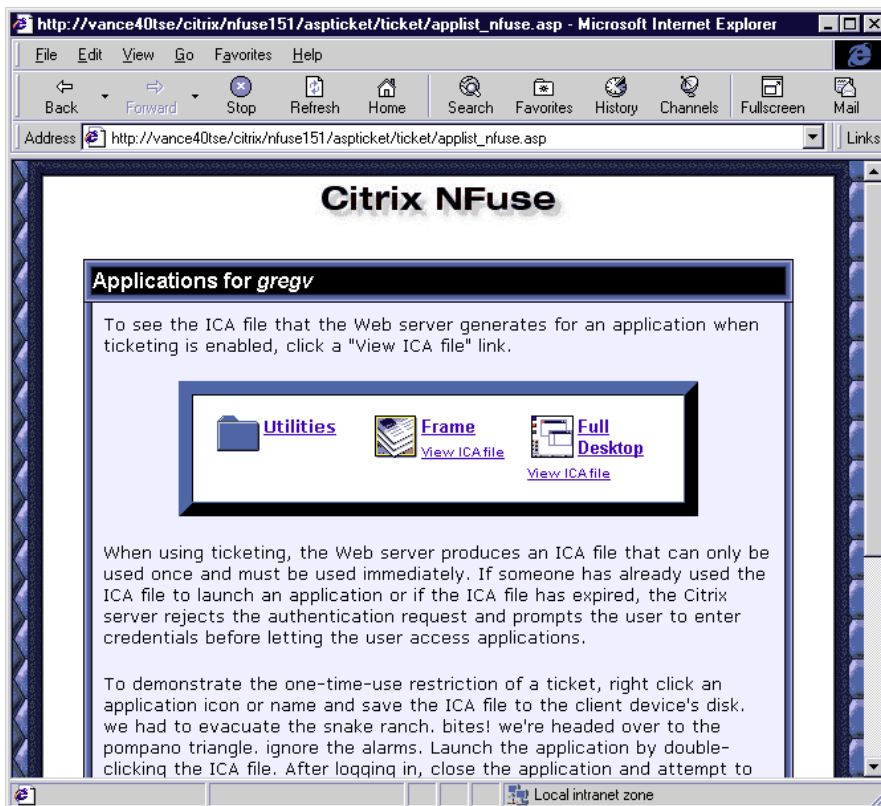
Ticketing removes credentials from ICA files by storing them instead in the server farm. When a user selects an application from an NFuse Web page, the NFuse Java objects on the Web server request from the farm a ticket for that user. The server farm generates a 30 character string that correlates the user to the user's credentials but does not contain the credentials themselves. The farm forwards this ticket to the Web server, which places the ticket in the ICA file sent to the client device. When the ICA Client uses the ticket to authenticate itself to the server farm, the server farm matches the ticket to the user's actual credentials and logs the user in.

In addition to removing credentials from ICA files, ticketing provides two further authentication safeguards: tickets are valid for a single ICA session and have a configurable expiration period. After authenticating a user, a ticket is no longer valid for additional authentications. The configurable expiration period limits the amount of time during which the ticket is valid and eliminates the possibility of an attacker retrieving a cached ICA file and using it indefinitely. A user who attempts authentication with a ticket that has already been used cannot log into the Citrix server farm without presenting actual user credentials.

Using the Web Site

To load this site, point a Web browser at the URL for your NFuse installation; for example, `http://<ServerName>/Citrix/NFuse151`. In the default document that appears, click **Ticketing**.

This example site lets you compare ICA files that contain tickets to those that include user credentials. To begin using the site, you must first log in. After logging in, the site presents you with the option of using ticketing or standard authentication. After you select one of these options, the site presents an application list page containing your published applications:



If you click **View ICA File**, the NFuse Web server produces an ICA file for the selected application and displays the file in your browser window. You can review the files to see the different credential information placed in ICA files for ticketing and standard authentication.

Implementing Ticketing

The actual code you must implement in your sites to support ticketing is minimal. A template ICA file that supports standard authentication contains the following five lines pertaining to authentication:

```
<[NFuse_IFSESSIONFIELDsessionfield="NFUSE_ENCRYPTIONLEVEL" value="basic"]>  
Username=[NFuse_User]  
Domain=[NFuse_Domain]  
Password=[NFuse_PasswordScrambled]  
<[/NFuse_IFSESSIONFIELD]>
```

In an ICA file that supports ticketing, the file contains the following in place of the five shown above:

```
AutoLogonAllowed=ON  
[NFuse_Ticket]
```

When the Web server processes such a template file, it replaces the [NFuse_Ticket] tag with the ticket number sent to it by the server farm.

You can use the Web Site Wizard to create sites that support ticketing or you can build ticketing into your own custom sites using the NFuse API. For information on using the Web Site Wizard with ticketing, see “Ticketing” on page 54. For information on ticketing methods in the NFuse API, see the description of the App object in “App” on page 119.

Cookie Data Encryption

This example Web site demonstrates how to use encryption to secure cookie data. NFuse stores the credentials that users enter in the NFuse login form in a client-side cookie. By default, these credentials are in clear text and are vulnerable to detection. Encrypting information in a cookie requires creation of a randomly generated session key, encryption of the data for placement in the cookie, and decryption of the data when the Web server requires the password value. This example site uses Active Server Pages/JavaServerPages code to create and store the key and to encrypt and decrypt the password.

This example site uses a 512 byte randomly generated session key to encrypt passwords. NFuse stores this key in an Active Server Pages or JavaServer Pages “session object” on the Web server. Upon receiving user credentials at login time, the Web site executes code that writes the credentials in the client-side cookie. Before placing the password in the cookie, the code retrieves the 512 byte key from the Web server’s session object and uses the key to encrypt the data.

Each time the Web site displays a list of applications, for example after initial login or when you browse a folder, the Web server must retrieve the credentials from the cookie and submit them to the Citrix server. Because the password is now in encrypted form in the cookie, the Web server executes code to decrypt the password before submitting it. This example uses code in the application list page along with the 512 byte key stored in the Web server's session object to handle the decryption. The decryption process also occurs when you launch an application; the site retrieves and decrypts the password for placement in an ICA file for the selected application.

Using the Web Site

To load this site, point a Web browser at the URL for your NFuse installation; for example, `http://<ServerName>/Citrix/NFuse151`. In the default document that appears, click **Cookie Data Encryption**.

This example site uses client-side JavaScript to present you with pop-up windows that display cookie data as it is encrypted and decrypted. Use these windows to examine the session key and encrypted passwords.

You do not have to modify or edit any example site files to use this site. To begin using the site you must log in.

Implementing Cookie Data Encryption

This site's implementation of cookie data encryption uses code contained in the following files:

- `login.asp` (`login.jsp` on UNIX Web servers). Creates the session key used to encrypt passwords.
- `applist_nfuse.asp` (`applist_nfuse.jsp` on UNIX Web servers). Encrypts the password and writes it in the cookie.
- `applist.asp` (`applist.jsp` on UNIX Web servers). Retrieves the cookie and decrypts the password when displaying application sets.
- `launch.asp` (`launch.jsp` on UNIX Web servers). Retrieves the cookie and decrypts the password when parsing template ICA files.

For a security overview, see “Configuring NFuse Security” on page 163.

Improving Reliability and Usability

The following example sites implement features that enhance NFuse's reliability and usability. The first site, called "Backup MetaFrame Servers," uses a list of Citrix servers to provide NFuse with alternate contacts for NFuse information should a primary contact server not be available. The "Multiple Server Farm Display" example demonstrates how to combine applications from multiple farms into a single Web page. The final example, "ICA Client Detection and Installation" demonstrates installation of ICA Clients through a client device's Web browser.

Backup MetaFrame Servers

This example site displays how to create a list of backup Citrix servers that the NFuse Web server can contact when generating a user's application set. Creating a backup server list ensures that users have access to their applications in the event of server failure. Should the default Citrix server fail to respond to an NFuse request, the NFuse Web server sends the request to each of the servers in the backup list until one responds. The application set list that the backup server sends to the Web server is identical to the one that would have been provided by a disabled default server had it responded.

Note This site's implementation of backup servers provides backup services during user login and any time a user requests an application listing (for example, when a user browses folders or refreshes applications). This site cannot contact a backup server during application launching.

Using the Web Site

To load this site, point a Web browser at the URL for your NFuse installation; for example, `http://<ServerName>/Citrix/NFuse151`. In the default document that appears, click **Backup MetaFrame Servers**.

This example uses a plain text file to store the backup server list. The file, `backupServer.txt`, is located along with the other files for this site in the example directory. Before continuing, edit the file to include the names of the Citrix

servers that you want to back up the default server. You must also specify the port on which each server is running the Citrix XML Service:

```
; This file stores the list of backup Citrix servers that NFuse
; contacts if the default server is unavailable. Add your backup
; servers in the order in which you want NFuse to contact them.
;
; Use the following format for your entries:
;
; serverName1:portNumber1
; servername2:portNumber2
;
; where serverName is the fully-qualified DNS name, IP address,
; or Windows NT server name of the Citrix server and portNumber
; is the port on which the server is running the Citrix XML
; Service. The port entry must be a number within a range of
; {1...65536}.
```

After the last commented line in the file, enter the names and port assignments for the servers you want to back up the default server. For example, if in the case of default server failure you want NFuse to contact a server named mySecondMFServer listening on port 8080, add the following line:

```
mySecondMFServer:8080
```

If you want a third Citrix server to provide backup support in case the default server and mySecondMFServer fail, add the following lines:

```
mySecondMFServer:8080
```

```
myThirdMFServer:800
```

For the purposes of this demonstration, disable your default Citrix server by stopping its Citrix XML Service. (To determine the name of your default server, locate the NFuse_CitrixServer entry in your NFuse.properties file.) Stopping the service causes the server to fail to respond to NFuse requests. When this server fails to respond, NFuse will contact the first server in your backup list. Remember to restart the service when you finish this demonstration.

Once you have configured your server list and disabled your default Citrix server, log in to the site to begin the demonstration.

Implementing Backup MetaFrame Servers

This site's implementation of backup MetaFrame servers uses code contained in the following files:

- `login.asp` (`login.jsp` on UNIX Web servers). Parses the backup server list file and writes the server list in the NFuse cookie.
- `backupServer.txt`. Contains the list of backup servers and their port assignments.
- `applist.asp` (`applist.jsp` on UNIX Web servers). Lists applications for users and contains code that calls a backup server should the default server fail during an application list request.
- `blank.asp` (`blank.jsp` on UNIX Web servers). Redirects application list information to `applist.asp` or `applist.jsp` when that information is retrieved from a backup server.

Multiple Server Farm Display

This example site displays how to use a single NFuse Web site to contact multiple Citrix server farms and to display their applications in a single Web page. You can use this example to contact multiple MetaFrame for Windows farms, MetaFrame for UNIX Operating Systems farms, or a combination of both server farm types.

This example is available in all four supported NFuse Web site models: Active Server Page and HTML for IIS versions on Internet Information Servers and JavaServer Page and HTML for Servlets versions on UNIX Web servers. Consider the following when examining these models:

- The Active Server Page and JavaServer Page versions use multiple logins. During authentication, these models present the user with a login form for each farm you want to display. Separate logins allow the site to query servers that have separate account authorities; for example, a MetaFrame for Windows farm and a MetaFrame for UNIX Operating Systems farm. Note that you can modify the site to use a single login if all farms share an account authority.
- The Active Server Page and JavaServer Page versions contact a list of farms you specify.
- The HTML for IIS and HTML for Servlets versions use a single login. These models present the user credentials entered in the single login form to each Citrix server farm. All farms queried must have a common account authority to successfully authenticate the user.

- The HTML for IIS and HTML for Servlets versions contact the default Citrix server (specified in the NFuse.properties file) and any additional servers you specify.
- When using MetaFrame for UNIX Operating Systems server farms, please see “Authentication and MetaFrame for UNIX Operating Systems” on page 56 for information on modifying the authentication process to work with UNIX servers.

Using the Web Site

To load this site, point a Web browser at the URL for your NFuse installation; for example, `http://<ServerName>/Citrix/NFuse151`. In the default document that appears, click **Multiple Server Farm Display**.

Configuration steps depend upon the example model you are using. Active Server Page and JavaServer Page sites use a plain text file to store the list of Citrix servers to contact. The file, `multiServer.txt`, is located along with the other files for this site in the example directory. Before continuing, edit the file to include the names of a Citrix server in each farm from which you want to retrieve applications. You must also specify the port on which each server is running the Citrix XML Service:

```
; This file stores the list of Citrix servers that NFuse
; contacts for application information. The list must contain
; a server from each farm from which you want to retrieve
; applications.
;
; Use the following format for your entries:
;
; serverName1:portNumber1
; servername2:portNumber2
;
; where serverName is the fully-qualified DNS name, IP address,
; or Windows NT server name of the Citrix server and portNumber
; is the port on which the server is running the Citrix XML
; Service. The port entry must be a number within a range of
; {1...65536}.
;
```

After the last commented line in the file, enter the names and port assignments for the servers you want to contact. For example, if you want to show all the applications from a farm represented by the server `myFirstMFServer` listening on port 8080 along with those from a farm represented by `mySecondMFServer` listening on port 800, add the following lines:

myFirstMFServer:8080

mySecondMFServer:800

The HTML for IIS and HTML for Servlets versions store server contact information in one of the sites' HTML documents. The file, `applist.htm`, is located along with the other files for this site in the example directory. Before continuing, edit the file to include the names of a Citrix server in each farm from which you want to retrieve applications. You must also specify the port on which each server is running the Citrix XML Service:

```
<!--Beginning Code: Enumerating another set of apps and folders from specified MetaFrame Server-->
<TABLE BORDER=0 CELLSPACING=8>
<!--specify the MetaFrame server after the "="-->
[NFuse_SetSessionField NFuse_CitrixServer=ENTER_CITRIX_SERVER_NAME_HERE]
<!--specify the MetaFrame server port after the "="-->
[NFuse_SetSessionField NFuse_CitrixServerPort=ENTER_CITRIX_SERVER_PORT_HERE]
```

Locate the text *ENTER_CITRIX_SERVER_NAME_HERE* and replace it with the name of a Citrix server in the farm you want to contact. Replace *ENTER_CITRIX_SERVER_PORT_HERE* with the port on which the server is running the Citrix XML Service.

By default, the HTML for IIS and HTML for Servlets versions contact the default server and the server you specify when editing `applist.htm`. If you want to add additional server farms, see the commented directions in `applist.htm`.

Once you have configured and saved your server list, log in to the site to begin the demonstration.

Implementing Multiple Server Farm Display

The Active Server Page and JavaServer Page examples use code contained in the following files to produce multiple farm display functionality:

- `getServer.asp` (`getServer.jsp` on UNIX Web servers). Parses the server list file.
- `multiServer.txt`. Contains the list of Citrix servers and their port assignments.
- `storeInfo.asp` (`storeInfo.jsp` on UNIX Web servers). Writes the server list information in the NFuse cookie.
- `login.asp` (`login.jsp` on UNIX Web servers). Prompts user for credentials and passes those credentials to `checkCredential.asp` or `checkCredential.jsp`.

- `checkCredential.asp` (`checkCredential.jsp` on UNIX Web servers). Verifies credentials and sends the user to the login page repeatedly until the credentials for all Citrix servers are entered.
- `applist.asp` (`applist.jsp` on UNIX Web servers). Retrieves application sets from the multiple farms.

The HTML for IIS and HTML for Servlets versions use a modified `applist.htm` to support multiple server farm display.

ICA Client Detection and Installation

The ICA Client Detection and Installation example site demonstrates the ICA Client deployment capabilities of NFuse. This example site implements Web-based ICA Client installation, an ICA Client deployment tool that you can use to deploy ICA Clients to any device that has a Web browser. When a client device user visits an NFuse Web site, the Web-based ICA Client installation code detects the device and Web browser types and prompts the user to install an appropriate ICA Client. In the case of 16- and 32-bit Windows devices, Web-based ICA Client installation can also detect the presence or absence of an installed ICA Client and prompt the user only if necessary.

Before using this site, make sure you copied the ICA Clients to your Web server during Web Server Extension installation. For more information, see “ICA Client Installation Files” on page 60. Use of this example requires no other configuration. Refer to “How Web-Based ICA Client Installation Works” on page 62 for step-by-step explanation of the functioning of this site.

Note By default, Web-based ICA Client installation offers 32-bit Windows client devices the ICA Win32 Web Client installation archive. See “ICA Win32 Client Installation Archives” on page 61 for information on modifying NFuse Web sites to use the ICA Win32 Program Neighborhood Client installation archive to install the ICA Win32 Client.

To load this site, point a Web browser at the URL for your NFuse installation; for example, `http://<ServerName>/Citrix/NFuse151`. In the default document that appears, click **ICA Client Detection and Installation**.

Index

A

- Active Directory 56, 181, 183, 187
- Active Server Pages 23
 - using the wizard to create 51
- ALTADDR 152, 154
- Apache Server
 - installing Web Server Extension 34
 - requirements 25
- API 22
- App object 119
- AppDataList object 135
- AppEnumerator object 113
- application programming interface 22
- application publishing 18
- application set 19
- AppListCache object 137
- AppSettings 130
- AppSettings object 130
- authenticating users 55
 - explicit login 55
 - guest login 55
 - with MetaFrame for UNIX Operating Systems 56

C

- caching
 - ICA files 169
- caching application data
 - AppDataList object 135
 - AppListCache object 137
 - configuring expiration of cache objects 46
 - folder filtering example Web site 184
 - group filtering example Web site 181
 - per-user example Web site 180
 - substitution field 79
 - to improve performance 179
- CDN 12
- Citrix Developer Network 12
- Citrix XML Service 19
- CitrixWireGateway object 104
- ClearTextCredentials object 108
- Client Auto Update 160
- client name 126, 146, 155
- codebase 53

- CTXALT 152, 154
- CTXCFG 149

D

- default Citrix server
 - overriding as communication link 45
- documentation 11
- reader response 13

E

- embedding applications 52, 53
- encryption 155
 - cookie data 166
- example Web sites
 - backup MetaFrame servers 193
 - basic ASP/JSP site 180
 - caching and filtering by folder 184
 - caching and filtering by group 181
 - encrypting data in cookies 191
 - how to find and use 178
 - ICA Client detection and installation 198
 - multiple farm display 195
 - per-user application caching 180
- explicit login 55

F

- filtering
 - by folder
 - example Web site 184
 - by group
 - example Web site 181
 - UNIX limitation 25
 - introduction to example filtering sites 179
 - per-user example Web site 180
- firewalls 152, 154, 173
- functional overview of NFuse 21

G

- GroupCredentials object 110
- guest login 55

I

- ICA Client device 20
- ICA Clients
 - ActiveX Control 52
 - ActiveX control
 - updating 53
 - Java 52
 - client name 155
 - configuring 65
 - launching and embedding capabilities 52
 - Macintosh (configuring) 66
 - name 146
 - Netscape Plug-in 52
 - required configuration 65
 - Web-based ICA Client installation
 - support in wizard 53
 - using to deploy ICA Clients 60
- ICA file
 - caching 169
 - introduction 71
 - parameters 147
 - reference 145
 - sections in an ICA file 146
 - structure 146
 - templates 71
 - using tickets in ICA files 157
- ICAPORT 149
- ICAPortNumber 149
- installing
 - Web Server Extension
 - on Apache, Netscape, and iPlanet 34
 - on IIS 31
 - Web Site Wizard 47
- Internet Explorer
 - using with the ICA Java Client 66
 - using with the ICA Macintosh Client 67
- Internet Information Server
 - installing Web Server Extension 31
 - requirements 25
- iPlanet Web Server
 - configuring the Web Server Extension 37
 - installing Web Server Extension 34
 - requirements 25

J

- Java objects 22
 - App 119
 - AppDataList 135
 - AppEnumerator 113
 - AppListCache 137
 - AppSettings 130
 - CitrixWireGateway 104
 - ClearTextCredentials 108
 - definition 19
 - GroupCredentials 110
 - TemplateParser 140
- JavaServer Pages 23
 - using the wizard to create 52

L

- launching applications 52, 53
- Linux
 - requirements 25
- local text echoing 160

M

- MetaFrame for UNIX Operating Systems
 - authenticating users 56
 - determining XML Service port 31, 36
 - requirements 23
 - role in NFuse 18
 - using filtering by folder 185
 - using multiple server farms 195
 - using per-user filtering 180
- MIME type
 - configuring for ICA Java Client 65
 - configuring for ICA Macintosh Client 66
- multiple subnets 152

N

- Netscape Enterprise Server
 - configuring the Web Server Extension 37
 - installing Web Server Extension 34
 - requirements 25
- Netscape Navigator
 - using with the ICA Java Client 66
 - using with the ICA Macintosh Client 66
- NFuse Java objects 19
- NFuse.properties
 - description and contents 44

O

override order of session fields 45, 102

P

performance (improving) 179

persistent caching 151

port (TCP/IP)

 changing the Web server port 46

 used by ICA Client 149

Program Neighborhood 18

proxy servers 158

publishing applications 18

R

requirements

 Citrix server 23

 ICA Client device 26

 Web server 25

S

secure Web servers 165

security

 cookies

 encrypting password example Web site 191

 encrypting password overview 166

 credentials

 protecting in cookies 166

 using tickets to protect credentials 166

 encryption

 cookie password 166

 ICA Client to Citrix server communication 174

 ICA file

 eliminating user credential exposure 166

 preventing caching 169

 using ticketing 166

 network communication

 between client device and Web server 164

 between ICA Client and Citrix server 173

 between Web server and Citrix server 169

 RC5 encryption 174

 SecureICA services 174

SSL

 adding certificates 172

 between Web server and Citrix server 170

 between Web server and Web browser 165

 configuring NFuse to use the SSL Relay 171

 configuring the SSL Relay 171

 ticketing 166, 174

 server location 152

 ICA-Client-side 153

 through firewalls 154

 Web-server-side 153

 servlet

 HTML for Servlets model 51

 NFuse's use of 23

 session fields

 introduction 73

 precedence 45, 102

 setting 98

 from a script 98

 from a template 99

 from a URL 100

 from the properties file 101

 in a cookie 100

 using 97

 SOCKS 158

 Solaris

 configuring the Web Server Extension 37

 requirements 25

 SpeedScreen 160

 SSL

 configuring the SSL Relay 171

 finding more information on the SSL Relay 11

 location of certificates on Web server 46

 methods used to configure 104

 secure Web servers 165

 SSL Relay UNIX limitation 25

 substitution tags 75

 using the SSL Relay 170

 using the wizard to configure SSL support 48

 subnets (using NFuse with multiple) 152

 substitution tag-based Web sites

 files included in 87

 substitution tags 23

 introduction 69

T

- TCP/IP port
 - changing the Web server port 46
 - used by ICA Client 149
- TcpBrowserAddress 152
- template ICA file 71
- TemplateParser object 140
- ticketing
 - configuring expiration period 46
 - configuring with encrypted connections 157
 - example Web site 188
 - methods used to create tickets 127
 - security overview 166
 - substitution tag 77
 - support in Web Site Wizard 54
 - UNIX limitation 25

U

- UDP 152
- updating ICA Clients 160
- UseAlternateAddress 152
- User Principal Name (UPN) 56, 181, 183, 187

W

- Web browsers
 - required configuration 65
- Web server
 - role in NFuse system 19
 - security 165
- Web Server Extension
 - configuring properties 44
 - files copied to Web server 34
 - installing on Apache, Netscape, and iPlanet 34
 - installing on IIS 31
- Web Site Wizard
 - authenticating users 55
 - configuring SSL support 48
 - creating scripted sites 51
 - creating substitution tag-based sites 50
 - embedding applications 53
 - installing 47
 - introduction 47
 - launching applications 52
 - overriding the default Citrix server 48
 - ticketing 54
 - Web-based ICA Client installation 53
- Web-based ICA Client installation
 - copying ICA Clients to your Web server 60
 - example Web site 198
 - how it works 62
 - introduction 60
 - support in wizard 53

X

- XML Service 19
 - determining port in use on Citrix server 30